*People's Democratic Republic of Algeria*
*Ministry of Higher Education and Scientific Research*
**University of Science and Technology Houari Boumedienne(USTHB)**
**Faculty of Electrical Engineering, Computer Science Department**

# Security in Mobile Ad hoc Networks: Detection and Isolation of Selfish Nodes

**Thesis to Obtain The Degree of Doctor in Computer Science**

**Submitted by:** Djamel DJENOURI

**Jury:**

Prof Mohamed BETTAZ, MESRS: **President**
Prof Mohamed OULD-KHAOUA, university of Glasgow, U.K: **Reviewer**
Prof Zaidi SAHNOUN, university of Constantine: **Reviewer**
Prof Amar AISSANI, USTHB: **Reviewer**
Prof Mohamed AHMED-NACER: USTHB: **Reviewer**
Prof Nadjib BADACHE, USTHB: **Thesis Supervisor**

April 2007

# Dedication

- To my parents
- my wife and lovely baby daughter
- my brothers
- my colleagues at CERIST, and all my friends. Especially, Lyes Khelladi, Mehdi Chelbabi, and Abdelouahid Derhab.

# Abstract

Ad hoc networking is a research field that attracts more and more attention amongst researchers. It includes a variety of topics involving many challenges. In this thesis, we deal with security problems, and focus on one related to the energy constraint of the nodes forming the ad hoc network, namely node selfish misbehavior or node non-cooperation. This problem threatens the service availability, one of the security requirement. It consists of a misbehavior in which the node, anxious about its battery shortage, drops packets originated from other nodes it is assumed to route, while using them as routers to transmit its own packets toward remote nodes. We first provide a general review of some security problems, along with the current solutions, then we survey the selfish misbehavior in a separate chapter. Before attempting to mitigate the selfishness problem we first treat its cause, by tackling the power management, and proposing a power aware-routing protocol. However, although the power-aware routing protocols, such as the one we propose, help improving the battery life time, they are far from eliminating this challenging problem. Therefore, a solution that detects and isolates selfish nodes is mandatory for self-organized ad hoc networks. We then propose a new solution to monitor, detect, and isolate such nodes [1].

[1]This project has been performed at the Center of Research on Scientific and Technical Information (CERIST), granted by the Algerian Ministry of Higher Eduction and Scientific Research

# Acknowledgment

First of all, I must thank ALLAH, who has helped me to pass all the steps of this project.

I would like to thank gratefully my supervisor Prof Nadjib Badache, for his guidance during all my research projects, and for encouraging me to keep working on ad hoc networks. I also thank him for his valuable advices and comments. Many thanks are due to the chief of our laboratory, Mrs Hassina Aliane, for her support and help, as well as my colleagues at the lab, especially Lyes Khelladi, Souad Benmezian, and Abdelouahid Derhab. I also wish to thank the folk of Heudiasyc lab, Prof Abdelmadjid Bouabdallah, for hosting me during my visit to compiegne university (UTC) at the beginning of this project, and for all the facilities he provided and advices he gave, as well as Immed Ramdani and Yacine Challal for their help during this visit, and the valuable discussions we got. Finally, I present my deep gratitude to my parents and my wife for being understanding, and for their encouragement and support.

# Contents

# List of Figures

# Introduction

Nowadays' wireless lightweight devices allow the creation of mobile networks, enabling users to be mobile and accessing to the network anytime and anywhere. A mobile ad hoc network (MANET) may be considered as a particular kind of wireless networks, free from any central or fixed infrastructure. In other words, in an ad hoc network all nodes are mobile devices that have to cooperate in order to replace the fixed and central infrastructure used in traditional networks. Many mobile devices used in MANET (such as notebooks, PDAs, sensors, etc.) are constrained by limitations in storage capacity, computational power (CPU), and particularly in energy supply provided by lightweight limited batteries. Furthermore, the wireless hertzian channels used in MANET that are limited in bandwidth and physical security, along with the frequent topology change caused by nodes mobility, make the design of protocols and applications for MANET challenging. Security is an important issue in computer networks, especially in MANET where no central administration is available for securing the network. It has brought many trends to the MANET arena, that are receiving more and more attention among researchers. The aim of this project is: i) First, to survey several security problems in MANET, as well as the current solutions proposed to overcome them while considering the MANET's features. Among these problems, we concentrate on an emerging one directly related to battery limitation, namely selfish misbehavior or non-cooperation of nodes. ii) Second, to consider the battery limitation problem and deal with the power awareness routing, by proposing new metrics and a new power-aware routing protocol. iii) Finally, to propose new solutions to detect and isolate selfish nodes. Selfish nodes in our context are those which do not cooperate to route packets for others, while soliciting these latter to transmit their own packets towards remote nodes. This misbehavior harms the quality of service, and moreover it threatens the service availability, one of the basic security requirements.

This thesis is organized as follows: The first chapter introduces the wireless mobile networks and particularly MANETs. It sketches their features and illustrates some basic problems, namely the routing and medium access. Different security problems will be presented in the second chapter, along with the current solutions. Classifications and discussion of these solutions will be provided. The problem of selfish misbehavior, to which we give more attention, will be presented separately in the

third chapter, where we discuss and classify the different solutions proposed thus far. The energy problem, which is the main cause of the selfish misbehavior, will be tackled in the forth chapter, by proposing a power-aware routing protocol. This chapter also includes a simulation study to assess the performance of the proposed protocol. Finally, our proposals for monitoring, judging and isolating selfish nodes are presented in the last chapter, along with mathematical analysis and simulation studies.

# Chapter 1

# Introduction to wireless and mobile ad hoc networks

## 1.1 Introduction

The emergence of handheld computers equipped with wireless communication devices, along with the rapid evolution in wireless communications enable creating new kind of computer networks, called mobile networks. These networks, aiming to provide users with an anytime and anywhere information access, allow users a free mobility while taking in charge their interconnections [Bad98].

Mobile ad hoc networks (MANETs) represent a special class of mobile networks, that attracts more and more attention amongst researchers. Their nature and features make the design of applications and protocols for them a complex and challenging task. In this chapter, we will present some basic concepts related to this kind of networks. We will first start with a classification of wireless networks. Afterwards, we will focus on the ad hoc class in the second section, which sketches their main features and applications. The routing problem will be tackled in the third section, followed by the medium access problem in the last one.

## 1.2 Wireless networks classes

Mobile networks, or generally speaking wireless networks, might be split up into two categories: infrastructured and infrastructureless networks. In the following we briefly present the first category, then we will give more attention to the second one.

### 1.2.1 Infrastructured networks

This type of networks involves fixed sites, that ensure the connection of mobile users, and thus constitute the basic network infrastructure. This class may be divided into

several classes, according their range (coverage area). Thereby, the mobility of users (we call nodes or hosts) is generally enabled within this range.

### Wireless Personal Area Networks (WPAN)

They are networks with a short range (around some tens of meters), relating wireless handheld computers and peripheral devices through a master station. Blue-tooth technology, also known as IEEE 802.15, is the main WPAN technology. It allows a theoretical throughput of 1Mbit/s.

### Wireless Local Area Networks (WLAN)

A WLAN permits to cover some hundreds of meters area, that is at the scale of a company's local network. This coverage is ensured by relating equipments in the coverage area through fixed stations termed access points. WIFI (WIreless FIdelity) is the standard of this kind of networks, founded on the IEEE 802.11 that will be introduced later. Depending on its version, WIFI offers a throughput ranging from 11 to 54Mbits/s, and a power range from 100 to 500 meters.

### Wireless Metropolitan Area Networks (WMAN)

Metropolitan wireless networks are based on the standard IEEE 802.16, and offers a throughput ranging from 1 to 10 Mbit/s for a power range around 4 to 10 km (covering a town or a city). This technology is largely used in the business arena by the telecommunication operators. A WMAN could be viewed as the interconnection of several WLANs, i.e. of their access points.

### Wireless Wide Area Networks (WWAN)

WWAN extends the previous class, by enabling a large coverage up to hundreds of kilometers. It may gather a variety of technologies. Cellular networks, like GSM, represent a typical example of WWAN. Fixed sites devoted to ensure the communication are called base stations. Each base station covers a *cell*, from which mobile devices can send and receive messages. Whereas, fixed sites are interconnected to each other by a wired network, often reliable and providing high throughput [DR92]. In this mode, a mobile device could be directly connected, at a given instant, to no more than one base station. It changes its base station each time it leaves a cell and moves to another. The mobile device (node) can communicate with other devices, as well the ones located in the same cell as those located in remote ones, through the base station to which it is directly connected.

### 1.2.2 Infrastructureless (ad hoc) networks

Mobile ad hoc networking concept tries to extend the mobility to all devices, and to make the deployment of a mobile network an operation free from any kind of fixed infrastructure or administration. Contrary to the previous class, in ad hoc networks mobile hosts themselves form, in an *ad hoc way*, the network infrastructure. Theoretically, there is no limitation on the ad hoc network's range. Therefore, it can include all the levels of the previous class, up to WWAN. Still, nowadays' technology is far from this scale, and all simulation and test studies currently published do not go beyond the WMAN level.

#### Ad hoc network representation

At a time t, an ad hoc network might be modeled by a graph $G_t = (V_t, E_t)$ where $V_t$ stands for nodes set, i.e. mobile devices, and $E_t$ represents existing connections between these nodes. If $(e = (u, v)) \in E_t$, then nodes $u$ and $v$ are able to directly transmit packets to each other at the time t. In other words, each node is in the power range of the other. We also say that the nodes are neighbors. If nodes use radios with different power ranges then links might be unidirectional, and thus the topology graph must be oriented. Nevertheless, almost all the research studies on ad hoc networks (except those dealing with communication problems in unidirectional ad hoc networks) suppose bidirectional links. We are interested in this thesis to ad hoc networks with bidirectional links. A node wishing to send packets toward another remote one has to use other intermediary nodes, that acts as routers to get over the specialized routers lack.

#### Ad hoc networks features

Mobile ad hoc networks are basically characterized by:

- Dynamic topology (Amorphous): Mobile hosts move freely and arbitrarily in the network terrain. As a result, the topology may change rapidly and in an unforeseeable way, see figure 1.1. Thus, continually tracking the ad hoc network's topology is unpractical. Basic protocols, such as routing, should not rely on the topology information.

- Energy contrast: Mobile hosts are supplied with autonomous limited resources (batteries). Contrary to infrastructured networks, in ad hoc networks there is no fixed infrastructure that can take advantage of unlimited energy resource. Further, the role of base stations and all fixed infrastructure must be taken in charge by mobile nodes themselves, which might led to frequent disconnections and batteries related problems. The energy is an essential issue, that has to be considered by all applications and protocols.

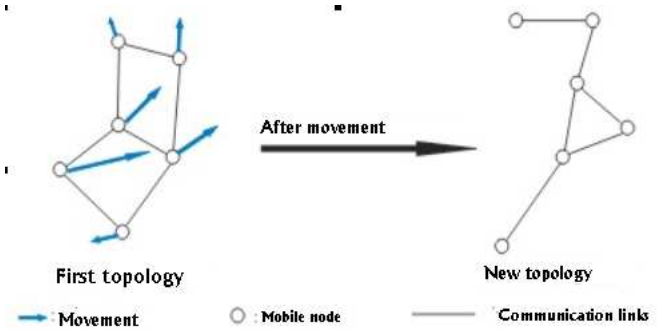


Figure 1.1: Topology change

- Limited bandwidth: An other important feature of wireless communication based networks is the use of a shared communication medium. This share makes the bandwidth devoted to hosts limited.
- Limited physical security: This feature is also due to the communication medium's nature, which is largely shared and easily accessible by any node. Moreover, security solutions proposed for traditional networks, which are centralized, are unapplicable in such a fully distributed system.
- Infrastructureless: Ad hoc networks are distinguished from the other mobile networks by the lack of any fixed preexistent infrastructure and any central administration. Therefore, mobile hosts are responsible of establishing and continually maintaining the network's connectivity. This represents a very big challenge.
- Terminals limitation: For mobility and portability constraint, devices (nodes) used in ad hoc networks, such as PDAs, notebooks, etc., are characterized, in addition to the energy limitation, by: moderate computing capacity, low memory capacity, and low storage capacity. Such devices need optimized algorithms and mechanisms, to perform computations and the different communication functions.

### Some applications of ad hoc networks

Ad hoc enabled applications include a huge spectrum. Generally speaking, ad hoc networks could be used in any application where the deployment of a wired infrastructure is too restricting, either because of the difficult deployment, or because the network's duration makes the wired infrastructure needless. Among these applications we quote:

- Military Applications: Military troops can take advantage of ad hoc networks. These networks allow them to move while communicating one to another in the battlefield, for keeping chiefs in touch with their soldiers.

- Rescue operations: These operations take place in disaster areas (e.g. fire, flooding, earthquake), where no wired infrastructure or base station may exist. If communication between rescue teams is required, the use of an ad hoc network is the unique solution.

- Conferences: Ad hoc networks are highly useful for a group of people wishing to create a temporary network. For instance, an ad hoc network connecting different portable devices (laptops, PDAs, notebooks) could be useful to propagate and share information between participants in a conference.

- Environment control: An ad hoc network could be deployed to collect a set of information of a given region. For example, a set of sensors may be deployed in a river where they form an ad hoc network, aiming to measure the water pollution.

- Home networks : Ad hoc networks could be used in home networks, where different equipments *directly* (with no master station) exchange information (sound, video, alarm) with each other. This can be viewed as an adhoc WPAN (Wireless Personal Area Network) application.

## 1.3 Routing in ad hoc networks

The routing is the method of dispatching data toward the appropriate destination through a network of connections. The routing problem for a network with fixed arcs and weights on these arcs consists of finding the optimal paths for packets in the network with respect to a given performance criterion.

As illustrated, the mobile ad hoc architecture is featured by the lack of a preexisted fixed infrastructure, in contrast to classical telecommunication networks (wired or cellular). To ensure the networks connectivity and overcome this lack and node mobility, each node is expected to cooperate and participate in routing, such to forward packets for nodes unable to directly reach their destination. Thus, each node acts as an application station and a router; it participates in a routing protocol allowing it to discover existent routes toward other nodes in the network. This routing protocol must consider all the ad hoc networks' features cited before. Particularly, it must take into account the frequent topology change caused by node mobility, as well as the limitation in energy supply.

Many earlier research studies on ad hoc networks have been devoted to designing basic routing protocols. More recent studies attempt either to secure those basic protocols, or to make them power-aware. Hereafter, we focus on basic protocols. We will tackle secure and power-aware protocols in the upcoming sections.

Following the manner of route creation and maintenance, routing protocols might be divided into two categories: proactive (table driven) protocols and reactive (on demand) protocols. Proactive protocols establish routes beforehand, basing on periodic exchange of route tables and/or topology information, whereas reactive ones seek routes on demand. Before presenting these two classes, we first introduce some basic concepts used by these protocols.

### 1.3.1 General concepts used by routing protocols

#### Global Positioning System (GPS)

It is a localization system composed of 24 satellites moving around the earth at an altitude of 10900 miles, making it possible for users supplied with GPS receivers to determine their geographical position with a precision between 10 to 100 meters. The satellites use mathematical computations to broadcast information that will be translated into geographical coordinates by the basic-receiver on earth.

#### Group

Decomposition into a set of groups (clusters) is one of the techniques used in ad hoc networks. Each cluster selects a node to be its head. Cluster-heads ensure routing of data and monitor the medium usage of their members. They have priority to access the channel, thus they have more chance to transmit than the other nodes.

#### Hierarchy

The group concept illustrated above could be extended to form hierarchal group levels. Cluster-heads in the level i become members in the level i+1, these new members get organized into a set of groups in the same way as in the lower level, and so on until reaching a level with solely one cluster-head.

#### Sequence number

It is a concept first proposed in DSDV [PB94]. It consists of labeling each route by a number initiated and monotonously increased by the route's destination. This way, a route with a higher sequence number is more fresh, an will be considered by intermediary nodes to update their routing tables.

#### Source Routing

It is a routing technique in which the transmitter (the source) of each packet sets the complete list of nodes by which the packet has to pass. In other words, the transmitter explicitly puts the route in the packet's header by identifying each hop with the address of the next node that has to receive the packet.

#### Direct Acyclic Graph (DAG)

An oriented graph is called a DAG for a destination vertex (node) $j$ if from every other node $i \neq j$, there is at least one route toward $j$. Protocols based on this method use special algorithms for the creation and the management of DAGs for each destination node.

#### Link reversal

It is a technique used by DAG-based routing protocols. It serves for reconstructing a DAG whenever a vertex loses its last route toward a destination because of a link failure. This is performed by reversing a set of links, i.e. changing their direction.

### 1.3.2 Proactive routing protocols

These protocols are based on the same philosophy as those used in wired traditional networks. They can be divided into two main subclasses, based on two different methods: Link state and vector distance. The two methods require a periodic update of routing information, that should be broadcast by the nodes.

In the link state protocol [MRR80], each node maintains its own vision of the network topology that includes the states of its links. To keep this vision up-to-date, each node periodically broadcasts, by flooding, link states of its neighbors to all nodes of the network. This is also performed when link states change. When a node receives link state information it updates its vision of the topology, and applies an algorithm of optimal paths computation in order to determine next hops. An example of the most common shortest path algorithms is the Djikstra algorithm. Note that a routing node computes the shortest distance from a given destination relying on the full image of the network formed by the most recent links of all routing nodes. That is, for a node to determine the next node toward a given destination, it has to receive the last link update messages, propagated by the network. OSPF (Open Shortest Path First) protocol is one of the most popular link state based protocols.

The basic distance vector algorithm [MW77] has been adopted for routing in wireless ad hoc networks, by considering each node as a routing node. In the distance

vector approach, each node broadcasts to its neighbors its vision of distances separating it from all the other nodes of the network. Relying on information received by all its neighbors, every routing node performs some computation to find the shortest route toward all destinations. The computation process is repeated whenever there is the minimum change regarding the distance separating two nodes, until a network stable state is reached. This technique is based on the Distributed Bellman-Ford (DBF) algorithm, which relies on the employment of update messages including a vector, each entry of which contains at least the distance toward a given destination. The approach of DBF is used by a variety of wired network routing protocols. A major performance problem of this algorithm is the high delay it requires to update hosts' routing tables, after a network partitioning, nodes failure, or when the network nodes number is huge. Other DBF problems are related to the absence of coordination between nodes for routing tables modification, that can be performed using out-of-date data, which results in the construction of routes including loops (the routing loop problem). The useless packets transfer that may take place with DBF is intolerable in mobile ad hoc networks, featured by limited bandwidth and moderate resources. Moreover, the frequent node mobility makes DBF convergence too long, which penalizes routing in such an environment. A lot of efforts has been undertaken to get over DBF problems, and makes it adaptable to the ad hoc context. One of them results in the DSDV (Destination Sequenced Distance Vector) protocol [PB94].

In link state based algorithms, the node convergence is less long than in DBF, which makes link state more suitable and employed by many modern networks. Nonetheless, the link state approach requires each node to maintain an up-to-date version of the whole network topology, which needs a large storage space and an overhead in the case of mobile and dynamic networks.

Proactive routing protocols gather ideas of the two previous approaches, and try to make them adaptable to mobile environment, by attempting to reduce or eliminate their shortcomings while considering the characteristics of the new environment. Table 1.1 illustrates some proactive protocols.

We remarked by simulation [BDD03, DDB06] that proactive protocols are not adaptable to node mobility, and that they are often less efficient than reactive protocols we present in the following.

### 1.3.3 Reactive routing protocols

As shown in the previous subsection, proactive routing protocols try to permanently maintain in every node of the network the best existing routes toward all the possible destination (that might represent all the nodes of the network). Routes still be safeguarded despite they are unused. The permanent safeguard of routes is ensured by a continuous update message exchange, resulting in a high cost especially in mobile

| **Protocol** | **Principles** |
|---|---|
| (DSDV Destination Sequenced Distance Vector) [PB94] | Distance Vector + sequence number |
| WRP (Wireless Routing Protocol) [SJ96] | Distance vector |
| GSR (Global State Routing) [CG98] | Link State + principle of DSDV |
| FSR (Fisheye State Routing) [PGC00] | improvement of GSR (by adding the fish-eye technique) |
| CGSR (Cluster-Gateway Switching Routing) [CWLG97] | Distance Vector + Groups |
| HSR (Hierarchical State Routing) [ICP+99] | Hierarchy |
| ZHLS (Zone-based Hierarchical Link State) [JNL99] | Link State + Hierarchy |
| DREAM Distance Routing Effect Algorithm for Mobility) [BCSW98] | Localization (GPS) |

Table 1.1: Some proactive protocols

networks.

Reactive routing protocols, called on demand routing, constitute the most recent protocols that are especially devoted for ad hoc networks. Almost all the protocols under evaluation by the Mobile Ad Hoc Networking Working Groupe of the IETF (Internet Engineering Task Force) [IET06] belong to this class of routing protocols. These protocols create and maintain routes depending on the need. When a node needs a route, it launches a global route discovery, in order to get specifiable unknown routing information. Many approaches could be applicable in the route discovery. The majority of algorithms used are based on the backward learning mechanism. The source node, searching a route toward the destination, broadcasts by flooding a request over the network. Upon receiving the request, intermediate nodes (of transit) try to teach the source the route. Once the destination is reached, it can send a reply using the route marked out by the request. This way, a route is established between the source and the destination nodes. This procedure may be reduced if an intermediary has already a route toward the destination. Once the route is computed, it must be safeguarded and updated whenever under usage. In most reactive protocols, as in proactive ones, an intermediate node that receives a packet to route finds out the next hop from its routing table. However, in the case of source routing protocols a node that receives a packet to forward does no more than retrieving the next hop from the header. This mechanism is used by some protocols, like DSR that will be presented in the next chapter.

| Protocole | Principles |
|---|---|
| DSR (Dynamic Source Routing) [DD96] | Source Routing |
| AODV (Ad hoc On-Demand Distance Vector) [PR99] | Combining DSR and DSDV (♯seq) |
| CBRP (Cluster Based Routing Protocol) [JJL99] | Groups |
| LMR (Lightweight Mobil Routing) [CE95] | DAG + Link Reversal |
| TORA (Temporally Ordered Routing Algorithm) [PC97] | DAG + Link Reversal + GPS, (derived from LMR) |
| ABR (Associativity Based Routing) [Toh96] | Associativity (links historic) |
| SSR (Signal Stability Routing) [DRWT97] | Signal powers of links |
| LAR (Located Aided Routing) [KV00] | DSR + GPS (For localization) |

Table 1.2: Some reactive protocols

On demand routing leads to a slowness due to routes seek, which reduces interactive application performance. This justify their absence in wired networks. But these protocols are more suitable for ad hoc networks than proactive ones [BDD03, SYQH04, DDB06]. Table 1.2 illustrates some reactive protocols.

## 1.4 Medium access control

Ensuring the sharing of the hertzian transmission medium (carrier) is the basic task of the medium access control (MAC) protocol. Contrary to the routing to which new protocols especially devoted for mobile ad hoc networks have been proposed, namely reactive protocols, in the MAC layer, wireless standard protocols have been adopted in this environment (with some modifications) [JLB04].

Wireless MAC protocols can be classified in different ways [Toh02]. According to the exchange mode, we distinguish synchronous protocols that use clocks and asynchronous ones. On the other hand, if we consider the communication initiator, we distinguish protocols in which the communication starts by the transmitter and others in which it starts by the receiver. The last classification takes into consideration the contention to acquire the channel, thus includes contention-based protocols and contention-free protocols that rely on central stations [Toh02]. The best protocols adaptable to the ad hoc environment and its features are the asynchronous competition-based ones.

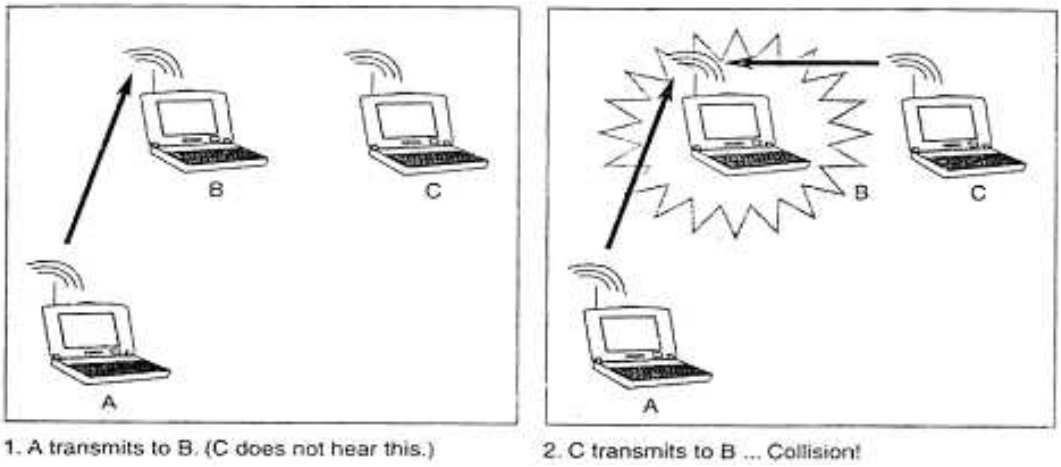


Figure 1.2: Hidden terminal problem

**Medium access problems in wireless networks:**
In the following we will present some wireless transmission problems having a major impact in the ad hoc environment, namely the hidden terminal and the exposed terminal problems.

**Hidden terminal problem:** Although contention-based protocols use no central station, they suffers from a great problem, known as the hidden terminal problem. In this kind of protocols nodes have to content to acquire the channel each time they wish transmit a packet. The protocol CSMA (Carrier Sens Multiple Access) [KT75] defines a basic strategy based on competition. In this strategy, when a node wishes to transmit it does not do so directly but it first senses (listens to) the channel. Thereby, it does not transmit its data unless it finds the channel free. If it finds it busy it backs off for a given duration. However, finding the channel free does not mean that it is free in the receiver's area. Figure 1.2 [Toh02] illustrates this phenomenon. Node A senses the channel and finds it free, then it sends its message to node B while node C is transmitting a message to B, which results in a collision at B. In this case, nodes C and A are called hidden to each other.

To avoid collision problems, all nodes neighbors to the receiver have to be informed that the medium is busy. For this purpose, a channel reservation mechanism is used. This mechanism manipulate control packets (of short size) of two types, RTS (Request To Send) and CTS (Clear To Send). When a node wishes to transmit data, it first sends an RTS to the next destination, this latter responses by a CTS to allow the transmission. Due to the broadcast nature of the wireless transmission medium, all nodes neighboring to the transmitter and the receiver node will be informed either by the RTS or CTS that the channel is occupied. This way, these nodes back off and

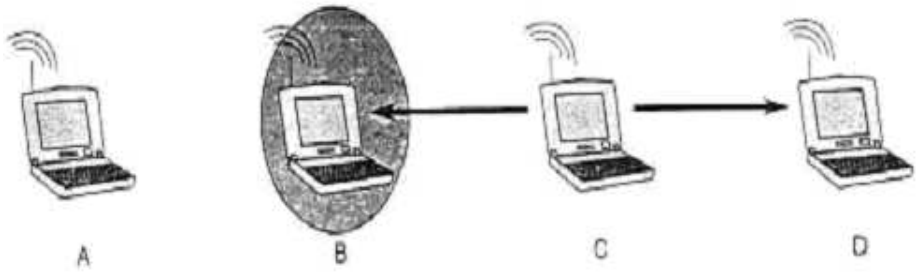


Figure 1.3: Exposed terminal problem

delayed their transmission to not disturb the ongoing communication and thus avoid collisions. The back-off duration differs from a protocol to another.

This mechanism prevents collisions between data packets, but does not eliminate collision neither between control packets nor between a control packet and a data packet. For instance, two nodes may transmit each an RTS in the same time to the same receiver. Further, if a mobile node joins a neighborhood after the RTS-CTS phase it may transmits an RTS while another node is transmitting a data packet. We realize that the RTS-CTS reservation mechanism mitigates collisions effects caused by the hidden terminal problem, but does not eliminate it. Note that the usage of acknowledgment packets (ACK) is essential to ensure reliability, since there is always possibility of collisions.

**Exposed terminal problem:** We can eventually consider the reverse problem of the hidden terminal called the exposed terminal. This one could be illustrated by the example of figure 1.3 [Toh02]. In this example we suppose that node B wishes to send data to A. It senses the medium and finds node C transmitting D a message. It consequently decides to back off, though its communication may take place with no problem. This causes a misuse of the bandwidth.

An exposed terminal (node B in this case) is thus a node located in the transmitter's (node C) power range and out of the receiver's (D) power range. This problem is of efficiency, its effects may be reduced by decreasing the back-off duration, but this would increase the hidden terminal effects. We point out that both the exposed terminal and the hidden terminal problems are far from being solved, and still represent a research trend.

The protocol IEEE 802.11 [S.G02], that is a standard in traditional wireless networks and the most adapted to the ad hoc environment, uses as well the CSMA sensing mechanism and the RTS-CTS reservation mechanism to avoid collisions, as ACK packets to ensure reliability. We have remarked through simulation studies that

it is generally the less influenced by node mobility [DMFL05] and the more scalable [DMF05], but its only shortcomings vs. the basic CSMA [KT75] is relatively the high delay.

# Chapter 2

# Security problems of mobile ad hoc networks and current solutions

## 2.1 Introduction

Because of the the features of mobile ad hoc networks, presented in the previous chapter, security is difficult to achieve in this environment. Earlier studies aimed at proposing protocols for some fundamental problems, such as routing, and tried to cope with the challenges imposed by the new environment. However, these protocols fully trust all nodes and do not consider the security aspect. They are thus vulnerable to attacks and misbehavior. More recent studies [DKB05] focused on security problems in MANETs, and proposed mechanisms to secure protocols and applications. This chapter surveys these studies. It presents and discusses several security problems, along with the currently proposed solutions at different network layers of MANETs. Security issues involved here include routing, medium access, key management, and intrusion detection systems (IDSs). Another network layer security problem, notably selfish misbehavior on data forwarding, will be presented with more details in the next chapter.

## 2.2 Generalities and basic concepts

### 2.2.1 Definitions

Hereafter, we briefly define some basic security concepts we will use later. Detailed descriptions of these concepts could be found in [Sta03]

#### Symmetric cryptography

It is a technique of cryptography in which a unique encryption/decryption key is used, called private key. That is, the message encrypted with a given key by a given

node has to be decrypted with the same key, hence the two corresponding entities need to share a common private key. In this scheme, we can find either one private key for each communicating couple of entities, or one private key shared by a group of entities.

#### Asymmetric cryptography

In this technique, each entity has a couple of keys: i) a public key, known by all the other corresponding entities, and used to ciphering (encrypting) messages, and ii) a private key which is kept secret to the entity, and used to decipher messages. Note that a message ciphered by a public key cannot be deciphered without the appropriate private key.

#### Digital signature

It is a string that associates a digital message to its origin, using asymmetric cryptography. Digital signatures are largely employed in computer security, as it allows to ensure non-repudiation that will be presented later.

#### Hash function

Hash functions take as input a message, and produce the so-called hash code (or finger print). The resulting code's size is constant independently from the size of the input message. More precisely, a hash function is a one-way function, allowing to get a short message clearly identifying the original long message. That is, it ensures that the hash code is unique for any message, and that the reverse operation (getting the original message from the hash code) is impossible. Message Authentication Code (MAC) stands for a special kind of hash functions, which employs symmetric cryptography. It might be considered as a hash function taking two different inputs: a message and a private key, to generate a fixed size output message, in such a way to make it impossible to produce the same chain without knowing the key.

#### Threshold cryptography

Threshold cryptography [Sha79] is used to enable n parties to share the capacity of conjointly perform cryptography operations. This is provided by dividing a private key into n different partial keys, in such a way to allow any k different parties with k different partial keys to perform the cryptographic operations.

### 2.2.2 Impact of MANET's features on security

As shown in the previous chapter, central servers, specialized hardware, and fixed routers are necessarily absent in MANET. The lack of such infrastructure precludes

the deployment of centralized host relationships. Instead, nodes uphold egalitarian relationships, which enforces security solutions to rely on a distributed cooperative scheme instead of a centralized one. Another feature that has a huge impact on security is the usage of wireless links, which renders ad hoc networks susceptible to attacks. Unlike wired networks, in which an adversary must gain physical access to the networks wires or pass through several lines of defense at fire-walls and gateways, attacks on a wireless ad hoc network can come from all directions and target any node. Hence, ad hoc networks will not have a clear line of defense, and every node must be prepared to defend against threats. Moreover, the MAC protocols used in ad hoc networks, such IEEE 802.11, rely on trusted cooperation in a neighborhood to ensure channel access, which leads to high vulnerability. The multi-hop nature also represents a serious vulnerability, as packets pass through different mobile nodes, possibly untrustworthy, before arriving at their final destination. Furthermore, any security solution must take into account the amorphous of the network topology, caused by nodes mobility. It also has to consider the nodes limitations, both in memory, computation, and particulary in power supply. This latter causes vulnerability to energy starvation attack (also known as sleep deprivation torture attack [SA99]), in which attackers target some nodes batteries to disconnect them. It also makes nodes anxious about their batteries and may lead them to behave selfishly in forwarding data.

### 2.2.3 Security requirements

The security services of ad hoc networks are not altogether different from those of other networks. The goal of these services is to protect information and resources from attacks and misbehavior. In dealing with network security, one faces the following requirements that an effective security architecture must ensure:

- Availability: Ensures that the desired network services are available whenever they are expected, in spite of the presence of attacks. Systems that ensure availability in MANETs seek to combat denial of service attacks, as well as node misbehavior such as node selfishness in packet forwarding.

- Authentication: Ensures that communication from one node to another is genuine. In other words, it ensures that a malicious node cannot masquerade as a trusted network node.

- Data confidentiality: Ensures that a given message cannot be understood by anyone other than its (their) desired recipient(s). Data confidentiality is typically enabled by applying symmetric or asymmetric data encryption.

- Integrity: Denotes the authenticity of data sent from one node to another. That is, it ensures that a message sent from node A to node B was not modified by any malicious node C during its transmission. If a robust confidentiality mechanism

is employed, ensuring data integrity may be as simple as adding one-way hashes [Sta03] before encrypting messages.

- Non-repudiation: In computer networks, non-repudiation is the ability to ensure that a node cannot deny the sending of a message that it originated. Digital signatures may be used to ensure this requirement.

### 2.2.4 Threats in MANET

We divide threats that can affect security in ad hoc networks into two classes: attacks and misbehavior.

#### Attacks

Attacks include any action that *intentionally* aims to cause any damage to the network. They can be divided according to their origin or their nature. An origin-based classification splits attacks into two categories, external and internal, whereas a nature-based classification splits them into passive attacks and active attacks.

- External attacks: This class includes attacks launched by a node that does not belong to the logical network, or is not allowed to access to it.

- Internal attacks: It Includes attacks launched by an internal compromised or malicious node. This is a more severe type of threat since the proposed defense toward external attacks is ineffective against internal nodes.

- Passive attacks: A passive attack is a continuous collection of information that might be used later when launching an active attack. For that, the attacker eavesdrops packets and analyzes them to pick up required information. Due to the nature of the wireless communication medium which is widely shared, it is easier for an attacker to launch such an attack in this environment than in wired environments. The security attribute that must be provided here is information confidentiality, which may be considered as a preventive solution. Nonetheless, due to the nature of the medium access that facilitates this attack, and the authorized employment of the promiscuous mode [1] used by some protocols, no detecting solution is currently available, and detecting this attack is a largely open research topic.

- Active attacks: Includes almost all other attacks launched by actively interacting with victims, such as: sleep deprivation torture, which targets the batteries; hijacking, in which the attacker takes control of a communication between two

[1] Note that this mode consists of capturing packets by a node that is not the appropriate destination

entities and masquerades as one of them; jamming, which causes channel unavailability by overusing it, attacks against routing protocols that we will see in the next section, etc. Most of these attacks result in a denial of service (DoS), which is a degradation or a complete halt in communication between nodes.

**Misbehavior**

We define misbehavior threats as an unauthorized behavior of an internal node that can result *unintentionally* in damage to other nodes, i.e., the aim of the node is not to launch an attack, but it may have other aims such as obtaining an unfair advantage compared with the other nodes. For instance, one may do not correctly execute the MAC protocol, with the intent of getting higher bandwidth, or it may refuse to forward packets for others to save its resources, while using their resources and asking them to forward its own packets. Up to now we have presented some basic concepts we will use later. In the following sections we will deal with the current research areas related to security in MANET, and we will discuss the existing problems and the proposed solutions.

## 2.3 Secure routing and MAC protocols

In this section, we discuss some security issues related to the lower layers. We first present a misbehavior related to the MAC protocol, then we will focus on the network layer and discuss the routing security issues.

### 2.3.1 Misbehavior in channel access

**Problematic**

We present a misbehaving activity that threatens one of the most important purposes of MAC protocols, namely the fairness in channel access. As illustrated in the previous chapter, wireless MAC protocols adopted in MANET [JLB04], such as IEEE 802.11, are fully distributed and contention-based. They use distributed contention resolution mechanisms for sharing the wireless channel. The contention resolution is typically based on cooperative mechanisms that ensure a reasonably fair share of the channel for all the participating nodes. In this environment, some selfish nodes in the network may misbehave by failing to adhere to the MAC protocol, with the intent of obtaining an unfair share of the channel. The presence of selfish nodes that deviate from the contention resolution protocol can reduce the throughput share received by conforming nodes.

The IEEE 802.11 MAC protocol [S.G02], which is a standard MAC protocol for wireless networks, has two mechanisms for contention resolution: a centralized mechanism called PCF (Point Coordination Function), and a fully distributed mechanism

called DCF (Distributed Coordination Function). PCF needs a centralized controller (such as a base station) and can only be used in infrastructure-based networks; thus, it is not to be considered in the ad hoc mode. In contrast, DCF is widely used in infrastructure-based wireless networks as well as in ad hoc wireless networks. It is used by almost all the wireless ad hoc MAC protocols. DCF uses the CSMA/CA (Carrier Sense Multiple Access/Collision Avoidance) option [2] for resolving contention among multiple nodes accessing the channel. A node (sender) with data to transmit on the channel selects a random back-off value from a range $[0; CW]$ and continuously senses the channel, where $CW$ (contention window) is a variable maintained by each node. While the channel is idle, the back-off counter is decremented by one after every time slot (a fixed interval of time), and the counter is frozen when the channel becomes busy. The node may access the channel when the back-off counter is decremented to zero. After the back-off counter is decremented to zero, the sender may reserve the channel for the duration of the data transfer using the RTS-CTS mechanism. After a successful RTS-CTS exchange, the sender transmits a DATA packet, which will be acknowledged by an ACK. If the nodes' data transmission is successful, the node resets its $CW$ to a minimum value ($CW_{min}$); otherwise, if the sender does not receive the CTS, then $CW$ is doubled, but it should not exceed a maximum value of $CW_{max}$. A misbehaving node may obtain more than its fair share of the bandwidth by:

- Selecting back-off values from a different distribution with smaller average back-off value than the distribution specified by DCF (e.g., by selecting back-off values from the range $[0, CW/4]$ instead of the range $[0, CW]$) [KV03]
- Using a different retransmission strategy that does not double the $CW$ value after collisions. Note that it is not beneficial for a selfish node to completely ignore the delay or to choose a very small constant period, since this may result in a very high collision rate, and thus the loss of the packets it sends.

Such selfish misbehavior can seriously degrade the throughput of well-behaved nodes. For instance, simulation results obtained by Kyasanur and Vaidya [KV03] show that for a network containing eight nodes sending packets to a common receiver with one of the eight nodes misbehaving by selecting back-off values from the range $[0, CW/4]$, the throughput of the other seven nodes is degraded by as much as 50 percent. To the best of our knowledge there is no published solution proposed (until 2004) to this complex problem, except the solution proposed by Kyasanur and Vaidya [KV03]. In the following, we present and discuss this solution.

### Solution

Due to the random selection of the back-off, it is hard to distinguish between the legitimate selection of small back-off values and a misbehaving selection. Hence,

[2] Note that this option employs the CSMA technique presented previously

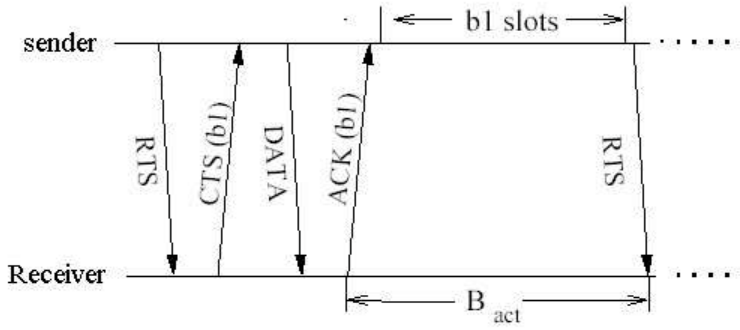


Figure 2.1: Receiver-Sender interaction

detecting a MAC layer misbehavior is a complicated problem. Kyasanur and Vaidya [KV03] have proposed a scheme to resolve this problem. The scheme consists of modifications to the IEEE 802.11 protocol that enable a receiver to identify sender misbehavior within a small observation interval. As illustrated in figure 2.1, instead of the sender, the receiver selects a random back-off value, b1, and appends it to the CTS and the ACK packets it transmits to the sender. The sender uses this assigned back-off value in the next transmission to the receiver. With these modifications, a receiver can identify senders deviating from the protocol by observing the number of idle slots between consecutive transmissions from the sender ($B_{act}$ in the example). If this observed number of idle slots is less than the assigned back-off, then the monitor realizes that the sender may have deviated from the protocol. The magnitude of observed deviations over a small history of received packets is used to diagnose sender misbehavior with high probability. The proposed scheme also attempts to negate any throughput advantage that the misbehaving nodes may obtain. To achieve this and discourage misbehavior, deviating senders are penalized, i.e., when the receiver perceives a sender to have waited for less than the assigned back-off, it adds a penalty to the next back-off assigned to that sender.

### Discussion

If we assume the receiver is a well behaving node, when the sender does not back-off for the duration specified by the penalty, it significantly increases the probability that it will be detected as a misbehaving node. Moreover, the punishment applied to the detected misbehaving node is a disincentive to such nodes to behave in this way. However, there are some problems with this solution:

- Deviations observed are not inevitably a misbehavior; they may be caused by the channel condition difference between the sender and the receiver, i.e. the

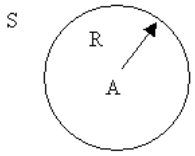


Figure 2.2: Nodes positions

hidden terminal problem. To illustrate this, we give the following example. Consider the situation of three nodes A, S, and R, where R is the receiver and S is the sender in our context, i.e., R is monitoring S after sending it a back-off. We assume R is within the power range of A, but S is not, viz. S and A are hidden to each other (figure 2.2). We also assume that A is transmitting packets. Hence, R's channel is busy, whereas S's channel is free. Consequently, S decreases its counter upon each slot unlike R, then the former will be able to access the channel when its back-off counter reaches 0, and this access will be considered by R as a deviation. If such a situation persists, R will register enough deviations to wrongly consider S to be misbehaving. So this solution may cause innocent nodes to be accused of misbehaving (false positives).

- We define a new misbehavior in channel access that we call cooperative misbehavior [DKB05]. We consider two nodes that wish to obtain unfairly high throughput to exchange packets, when they are in the same neighborhood (they are neighbors). The receiver may misbehave and assign unfair back-off values to the corresponding sender. The proposed scheme fails in this situation since it relies on the receivers trustworthiness.

### 2.3.2 Secure routing

The problem of all the basic MANET's routing protocols, presented previously, is that they trust all nodes and do not consider the security aspect. Therefore, they are highly vulnerable to attacks and misbehavior. It is very important to secure the routing protocol, as if the routing protocol can be subverted and messages can be altered in transit, then no amount of security on the data packets at the upper layers can mitigate threats. Recently, several secure MANET routing protocols have been proposed [HJP02, HPJ02, CM02, PH02, PSW+01, YNK01, HPJ03b, SDL+02, HJP03, PH02, ZA02]. Some of these solutions have been surveyed in [HP04]. In the following we deal with the security issues of routing protocols. After an overview of DSR [DD96] and AODV [CM02], two routing protocols involved in this section, we will present a classification of different attacks that threaten traditional MANET routing protocols, and we will discuss some recent proposed solutions.

### Overview of DSR AND AODV

In the following we give general descriptions of DSR and AODV, two protocols largely adopted by IETFs MANET working group [IET06]. An overview of these two protocols is essential, since the attacks presented later are analyzed in terms of these protocols. DSR (Dynamic Source Routing) [DD96] is a reactive protocol based on the source routing approach presented before. Remember that the principle of this approach is that the whole route is chosen by the source, and is put within each packet sent. Each node keeps in its cache the source routes learned. When it needs to send a packet, it first checks in its cache for the existence of such a route. If no entry to the appropriate destination is available in the cache, then the node launches a route discovery by broadcasting a request (RREQ) packet through the network. When receiving the (RREQ), a node seeks a route in its cache for the RREQs destination; finding such a route results in sending a route reply (RREP) packet to the source. However, if no appropriate route exists then the node adds its address to the RREQ and continues broadcasting. Route maintenance are ensured using route error RRER packets. When using a route, the failure on transmitting a packet through some link results in the initiation of a route error packet (RRER) from the detector of the failure (the transmitter over the link, also called the upstream node) toward the source. This latter then removes from its cache the failed route, as well as all routes including the failed link, and launches a new route request if necessary. Also, intermediate nodes update their cache by removing all routes including the failed link. Note that the detector of the failure could optionally salvage packets that are to be sent over the link if it has some other routes to their destination.

On the other hand, AODV (Ad hoc On-Demand Distance Vector) is a hop-by-hop routing protocol. When a node needs to send a data packet to a destination to which it has no route, it has to broadcast a RREQ to all its neighbors, then each neighbor does so until reaching the destination (or a node with a valid route to the destination), like in DSR. This node sends a RREP packet that travels the inverse path until reaching the source. Upon the reception of this reply, each intermediary updates its routing table. In this way, a route between the source and the destination is built. Unlike DSR, the source does not put the whole route within the outgoing packets; rather, the decision about the next hop is made separately after each hop. AODV assigns monotonically increasing sequence numbers to routes, which defines route freshness, as well as hop-count that defines route optimality. Route maintenance is ensured in the same way as DSR.

### Routing protocols attacks

The earlier proposed routing protocols for MANET are subject to many different types of attacks. Analogous exploits exist in wired networks, but can be more easily overcome by the existing powerful infrastructure. In this subsection, we present and

analyze several classes of attacks against ad hoc routing protocols. Without loss of generality, these attacks are discussed in terms of AODV and DSR, which are used as representatives of ad hoc on-demand protocols. Note that almost all traditional on-demand protocols have the same vulnerabilities. We think that table driven approach is unsuitable for MANET [DDB06], so it is excluded from the study.

**a) Attacks using modification:** Network traffic can be redirected and DoS attacks can be launched by modifying routing information [SDL+02], such as altering control message fields of data packets or forwarding routing messages with falsified values. We now detail several attacks that use modification.

**a.1 Redirection by modifying route sequence numbers:** Some routing protocols, such as AODV, instantiate and maintain routes by assigning monotonically increasing sequence numbers to routes. Consequently, any node may divert traffic through itself by claiming a route with a sequence number higher than the authentic value. We consider the example illustrated in figure 2.3 [SDL+02], and let us suppose that a malicious node M receives the RREQ originated from S for destination X after it is re-broadcast by B during a route discovery. M can redirect traffic toward itself by sending (unicasting) to B a RREP containing a much higher sequence number for X than the value last advertised by X. Eventually, the RREQ broadcast by B will reach a node with a valid route to X, and a valid RREP will be unicast back toward S. However, at that point B will have already received the false RREP from M. If the sequence number for X that M used in the false RREP is higher than the sequence number for X in the valid RREP, then B will drop the valid RREP, thinking that the valid route is stale. Consequently, all subsequent traffic destined for X that travels through B will be directed toward M. The situation will not be corrected until a legitimate RREQ and a legitimate RREP for X with a higher sequence number enter the network.

**a.2 Redirection by modifying hop-count:** Many traditional ad hoc routing protocols, such as AODV, use the hop-count field to determine the optimal path. Consequently, malicious nodes can increase the chances to be included in a newly created route by resetting the hop count field of the RREQ they forward to zero. Such an attack is most threatening when combined with spoofing, as detailed later. The redirection attack is possible even if the protocol uses other metric than the hops number. In this case all what the attacker has to do is to modify the field used to compute the metric, instead of the hop-count.

**a.3 Source routes modification:** As we have seen in the previous section, some routing protocols, like DSR, utilizes the source routing strategy. In which source nodes explicitly state routes in data packets. These routes lack any integrity checks, so alterations of source routes in packet headers can be easily performed by malicious

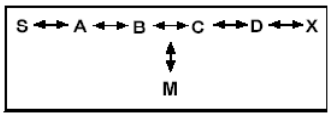


Figure 2.3: Example of an ad hoc network

nodes, resulting in denial-of-service (DoS) attacks. We illustrate this by the following example. Assume a path exists from S to X as shown in figure 2.3. Also assume that C and X are out of the power range of each other, as well as B and C, and that M is a malicious node attempting a denial-of-service attack. Suppose S wishes to communicate with X to which it has an unexpired route in its route cache. S transmits a data packet toward X with the source route (S,A,B,M,C,D,X) attached to the packet's header. When M receives the packet, it can alter the source route in the packet's header. For instance, it can delete D from the source route. As a result, when C receives the altered packet it attempts to forward it to X. Since X is out of C's power range, the packet will not reach X.

C will then consider that the link with X has been broken, and will send a RRER packet back to S via M. When M receives this packet, it will simply drop it. Therefore, S will still use the route through M, and this latter continue performing this way resulting in a denial of service attack against the routing service. This attack can also be used to cause sleep depravation (paragraph 2.2.2), since packets will be transmitted and *retransmitted* through compromised routes.

**a.4 Tunneling (wormhole):** Two remote nodes may collaborate to encapsulate and exchange messages between them along existing data routes, making impression as they are adjacent. This way, they may collaborate to falsely represent the length of available paths by encapsulating and tunneling between them legitimate routing messages generated by other nodes, preventing the intermediate nodes from correctly incrementing the metric used to measure path lengths (as hop count). For example, in figure 2.4 $M_1$ and $M_2$ are malicious nodes which use the path $(M_1, A, B, C, M_2)$ as a tunnel. When $M_1$ receives a RREQ from S, it encapsulates and tunnels it to $M_2$, then this latter normally forwards it. After $M_2$ gets the RREP from D, it tunnels it back to $M_1$. This latter does so to S resulting in the construction of a short wrong route $(S, M_1, M2, D)$, that may be considered as the most optimal one.

**a.5 Spoofing attacks:** Spoofing occurs when a node *misrepresents its identity* in the network, such as by altering its MAC or IP address in the outgoing packets. This attack can be readily combined with modification attacks. These two attacks (spoofing and modification) when combined with each other may result in serious mis-

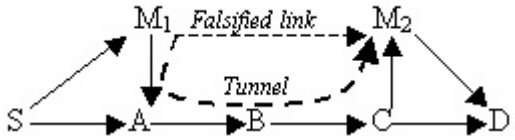


Figure 2.4: Tunnelling

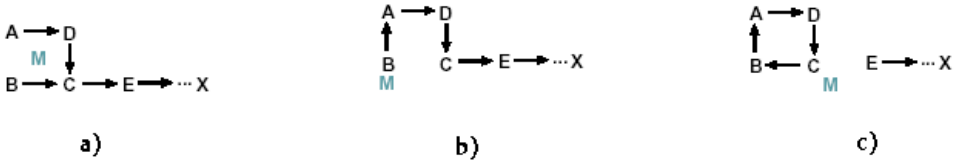


Figure 2.5: Route loop creation

information, such as creating route loops. We give an illustrative example [SDL+02] to show how this can be done in the context of AODV. Note that spoofing is also possible in DSR. However, since the protocol is based on the source routing, the source can easily avoid sending packets through routes including loops.

Assume a path exists between the five nodes illustrated in figure 2.5 (a) toward some remote destination X, as would follow after a route discovery. Also suppose that in addition to the links forming the route presented in figure 2.5 (a), B and A are neighbors, and M is a mobile node aiming a DoS attack. This latter can learn this topology by listening to the RREQ/RREP exchanges during route discovery. To start the attack, M changes its address to match the one of A, moves closer to B and out of the range of A. It then sends a RREP to B that contains a hop count to X that is less than the one sent by C, e.g., zero. B therefore changes its route to the destination, X, to go through A, as illustrated in figure 2.5 (b). M then changes its address to match Bs, moves closer to C and out of range of B, and then sends C a RREP with a hop-count to X lower than what was advertised by E. C then routes to X through B, as shown in figure 2.5 (c). At this point a loop is formed and X is unreachable from the four nodes.

**b. Attacks using fabrication** This class includes attacks based on the generation of false routing messages. Such attacks are difficult to detect.

**b.1 Falsifying Route Errors:** On-demand routing protocols, such as AODV and DSR, implement path maintenance to recover broken paths caused by nodes mobility. As illustrated before, when a link of an active route from node S to node D breaks down, the upstream node (predecessor) of the link sends S back a RRER

packet. If this latter has no other route to D and a route still be needed to this destination, it initiates a new route discovery. The vulnerability is that routing attacks can be launched by disseminating false RRER, resulting in valid routes destruction and extra overhead, which can cause DoS such and sleep deprivation.

**b.2 Broadcast falsified routes:** In DSR, a node can optionally update its routing table (*route cache*) relying on information in headers held by packets it forwards. Routes can also be learnt from promiscuously received packets. The vulnerability is that an attacker could easily exploit this method of learning routes and poison route caches of its neighbors, by broadcasting packets containing falsified routes.

**c. Rushing attacks:** Recently, Hu et al [HPJ03b] have defined a new attack called rushing attack. In almost all on demand routing protocols, to limit the route discovery overhead each node forwards only one RREQ originated from any route discovery, generally the first one received. This property can be exploited by *rushing* the forwarding of received RREQs.

For a route discovery, if the RREQs forwarded by the attacker are the first to reach each neighbor of the target, then any route obtained by this route discovery will inevitably include the attacker. That is, when a neighbor of the target receives the rushed RREQ from the attacker, it forwards that RREQ, and will not forward any further RREQ from this route discovery. As a result, the initiator will be unable to discover any usable routes (i.e. routes that do not include the attacker) containing at least two hops (three nodes). In general terms, an attacker that can forward RREQs more quickly than legitimate nodes can launch such an attack and include itself in all the discovered routes. The rushing attack can also be used against any protocol that predictably forwards any *particular* RREQ for each route discovery. The attacker does the same thing except that packets it sends must fit the appropriate feature [HPJ03b]. However, in the following we assume that nodes forward the first received RREQ.

How can an attacker rush RREQ packets? A malicious can use one or more of the following techniques:

- Remove MAC and/or network delays when forwarding packets: MAC and network layer protocols use delays on packets transmission to avoid collisions. Thus, an attacker may deny these delays to rush the request it forwards.

- Transmit RREQs in higher power: An attacker supplied with a powerful physical communication support can use a higher transmission power to forward RREQs, thereby covering a higher power range than other nodes, and further nodes will be reached in less hops. Unlike the other techniques, this one does not in general allow the attacker to include itself into a unique discovered route,

since it could not receive the RREP (except in the case where it has high sensitive receiver). However, it precludes discovering of valid routes.

- Employing wormhole technique with high quality route [HPJ03a]: two attackers can use a high quality tunnel to pass a RREQ packet among them, allowing it to reach its final destination before the other RREQs. This can be done when one node is closer to the source and the other is closer to the destination, and a path with *high quality* exists between the two nodes (eg, via a wired network). Note that a special high quality route is required for this attack, which is different from the previous tunnelling (wormhole) attack, that uses wireless multi-hop routes.

**d. Packet dropping (black-hole) attack:** To launch this attach, a malicious node, targeting a DoS attack either on a source or a destination, first includes itself in the route to be used to forward packets between the two nodes. This step could be performed using one of the previous attacks (such as rushing attack). After that, the malicious node simply drops (ignores) all data packets it receives to forward, which results in a DoS. This attack is very similar to selfish misbehavior that will be presented later, and has the same consequences. However, the only difference is that selfish nodes might drop both data and control packets, unlike the malicious ones that are motivated to only drop data packets. Hence, solutions against selfishness (that will be detailed later) are more general and directly applicable to counter the black-hole attack.

Table 2.1 [DKB05] provides a summary of DSR and AODV vulnerabilities to the attacks presented in this section. As shown, AODV is vulnerable to sequence number and hop-count modification attacks, since it is based on distance vector and uses sequence numbers. On the other hand, DSR which is based on the source routing and uses promiscuous route learning, is sensitive to source route modification, and to falsified routes broadcast as well. Both protocols are vulnerable to tunneling, spoofing, falsified route error packets fabrication, rushing, and black-hole attacks.

### Solutions

**a. Authentication during all routing phases:** This solution consists of using authentication techniques during all routing phases, to exclude attackers and unauthorized nodes from the participation in the routing. Most of the proposed solutions belonging to this class modify existing routing protocols to build authentication-based ones [SDL+02, CM02, HPJ02, YNK01]. Since they use digital signatures, these solutions rely on a Certificate Authority (CA). This reliance on a fixed server renders the solution centralized and less flexible. To overcome this problem (reliance on central authority), distributed key management solutions (that will be presented after) could

| Attack | AODV | DSR |
|---|---|---|
| **Attacks using modification** | | |
| modifying route sequence numbers | yes | no |
| modifying hop counts | yes | no |
| modifying source route | no | yes |
| Tunneling | yes | yes |
| **Spoofing attacks** | yes | yes |
| **Attacks using fabrication** | | |
| Falsifying Route Errors | yes | yes |
| Broadcast falsified routes | no | yes |
| **Rushing attacks** | yes | yes |
| **Black-hole** | yes | yes |

Table 2.1: Vulnerabilities of AODV and DSR

be used. But note that these solutions are far from being efficient enough. The major advantage of this approach is that it excludes external unauthorized nodes from participating in the routing. Therefore, all the attacks presented previously are prevented when launched by an external node. Moreover, some of the attacks launched by an authorized node can be beaten, as illustrated in table 2.2.

**b. Trust level metric:** Yi et al. [YNK01] define a new metric called trust value that governs the routing protocol behavior. This metric is to be embedded into control packets to mirror the minimum trust value required by the sender. Thus, a node that receives any packet can neither process it nor forward it unless it provides the required trust level presented in the packet. This way, the authors design SAR (Security-Aware Routing), a protocol derived from AODV and based on the hierarchical trust values metric and authentication. In SAR, this metric is also used as a criterion to select routes when many routes satisfying the required trust value are available. To define nodes' trust values, the authors address the example of military context, where trust level matches to the node's owner rank. But in the general context, where there is no hierarchy in the network, defining the nodes' trust values is problematic. This technique is based on authentication and requires a hierarchal key sharing. The advantage of this solution on the previous one is that it prevents attacks from an internal node on a higher trust level.

**c. Secure neighbor verification:** It consists of a three round authenticated message exchange between two nodes before each one claims the other as neighbor. If this exchange fails, then the well-behaving node ignore the other, and does not handle packets sent by it. This solution beats the illegal use of high power range to launch the rushing attacks. Since the sender using higher powers cannot receive the packet from further nodes, then it will not be able to perform the neighbor detection

process, and will be ignored [HPJ03b]. The major drawback of this solution is the important overhead when the mobility increases.

**d. Randomize message forwarding:** This technique is proposed by Yi et al. [HPJ03b] to minimize the chance that a rushing adversary can dominate all returned routes. In traditional RREQ forwarding, the receiving node immediately forwards the first RREQ and discards all subsequent RREQ. Using this new scheme, a node first collects a number of RREQs, and selects a RREQ at *random* to forward. There are thus two parameters related to this technique: first, the number of RREQ packets to be collected, and second, the algorithm by which timeouts are chosen.

We think [DKB05] the drawback of this solution is that it increases the delay of the route discovery, since each node must wait for a timeout or at least for receiving a given number of packets before forwarding the RREQ. Moreover, the random selection prevents the discovery of optimal routes. Route optimality may be defined as hops number, energy efficiency, or according to some other metric, but it is not random.

**e. Onion routing** Awerbuch et al. [AHNRR02] propose the use of an efficient asymmetric encryption strategy to protect and ensure anonymity for source routes when employing a source routing protocol. This strategy consists of encrypting a discovered source route during the route discovery in an *onion-like* form [SGR97], and transmitting data packets using this onion-encrypted route. During the route reply (respectively request broadcasting) phase, each node adds its address to the next (respectively previous) portion of the discovered route, and encrypts the outcome using the public key of the previous node (respectively its own public key). This way, each node will be able to only read the next hop when data packets are transmitted, and not any other. The onion encryption of a discovered source route $(n_0, n_1, ....., n_k)$ is performed during the reply phase as follows:
$n_k$ ID is encrypted with $P_{n_{k-1}}$ (the public key of $n_{k-1}$), the result is denoted by $[n_k]_{P_{n_{k-1}}}$, at $n_{k-1}$ this outcome is concatenated to $n_{k-1}$ ID and encrypted with $P_{n_{k-2}}$: $[n_{k-1}, [n_k]_{P_{n_{k-1}}}]_{P_{n_{k-2}}}$, and so on till reaching the source's successor $(n_1)$. The outcome of all these operation is the following onion encrypted source route:
$[n_1, .....[n_{k-1}, [n_k]_{P_{n_{k-1}}}]_{P_{n_{k-2}}}.....]_{P_{n_0}}$
This encrypted route will be used to route each data packet. Node $n_0$ decrypts the route and gets $n_1$ address to which it transmits the packet, the remaining part is encrypted with $_{P_{n_1}}$ and cannot be deciphered by $n_0$. $n_1$ does the same thing and route the packet, and so on until reaching the final destination.

**Example:** Assume a discovered source route (B,C,D), which connects A to D, is to be used by A to transmit a data packet. The onion-encrypted sequence of this route is: $[B, [C, [D]_{P_C}]_{P_B}]_{P_A}$. When decrypting the route with its own private key,

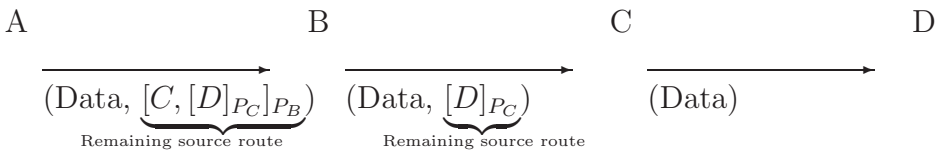


Figure 2.6: Example of forwarding a packet using Onion-encryption

| Solutions | Attacks prevented | Drawbacks |
|---|---|---|
| **Authentication during all phases** | - all external attacks, and the following internal attacks<br>- spoofing<br>- redirection by modifying rout sequence number | - requires certificate authority or key sharing mechanism |
| **Trust level metric** | - all attacks prevented by authentication<br>- all attacks on higher trust level nodes | - requires certificate authority or key sharing mechanism<br>- difficulty to define trust level |
| **Secure neighbor verification** | - all attacks prevented by authentication<br>- rushing | - requires certificate authority or key sharing mechanism<br>- important overhead when mobility increases |
| **Randomize message forwarding** | - rushing | - latency |
| **Onion encryption** | - all external attacks, and the following internal attacks<br>- spoofing<br>- DoS by modifying source route | - requires certificate authority or key sharing mechanism<br>- high computational cost |

Table 2.2: Solutions to secure routing protocols

node A retrieves B's address to which it transmits the packet. The other addresses (C, D) are hidden to A, and could not be deducted since they are asymmetrically encrypted (assuming the asymmetric encryption mechanism is robust). Identically, B (respectively C) gets c's (respectively D's) address using its own private key to which it forwards the packet. Figure 2.6 illustrates this example.

This mechanism ensures that each node is just able to identify its successor, where the rest of the route is kept anonymous. Consequently, DoS by modifying source route attack is prevented. When combined with authentication, this mechanism is powerful and efficient, but it suffers from the high computation cost.

For each solution presented previously, table 2.2 summarizes the preventable attacks and the shortcomings. As illustrated, all the solutions overcome all external attacks, and some internal attacks. Still, almost all the solutions require a certificate authority or a key sharing mechanism, which is problematic in MANETs, as we will see later. Randomized message forwarding is the only solution that is not based on such a requirement, but the problem with this approach is the important latency it introduces. Each solution has other drawbacks, as discussed previously. Note that no solution prevents the black-hole attack when performed by an internal node. However, solutions against selfishness, we will tackle later, do so.

## 2.4 Key management

As we have seen in the previous sections, most of the solutions proposed for securing routing rely on cryptography, and assume the existence of an underlying mechanism for providing and managing keys. Many secure applications and services also use cryptography and rely on this assumption. However, because of the lack of any central infrastructure or administration, key management is problematic in MANET.

There are basically two kinds of key infrastructure. The first one involves the private key infrastructure, which establish common private keys used for symmetric cryptography, like symmetric group keys largely used for securing group communications [3] [CH03, dPML+03, LPH03, YD02, MAH00]. The second kind is the public key infrastructure, which provides a couple of keys (public/private) used for asymmetric cryptography, like in digital signature. Providing such an infrastructure in MANET is challenging, due to their infrastructureless nature. Indeed, the role of this infrastructure should be spread out to all mobile nodes (or a subset of them), which form the key infrastructure. Therefore, the MANET's key management system should neither trust nor rely on any *fixed* Certificate Authority (CA), but should be distributed and self-organized.

### 2.4.1 Private key infrastructure

The private key management protocols have been classified into two classes [STW98]: key distribution protocols, which are centralized and based on a third trusted party, and key agreement protocols which are distributed. The suitable class for our environment is certainly the second one. Hereafter, we focus our discussion on some solutions belonging to this class. We will deal with some novel solutions specially devoted to MANET, but also with some previous ones which were either directly adopted in MANET or used to build new sophisticated solutions.

#### Diffie-Hellman two-party agreement (DH)

This basic protocol proposed in a landmark paper [DH76] allows two nodes to build a common key. The principle of this protocol is simple: the two involved nodes, $M_1$ and $M_2$, send one another a partial key to be used for the common key computation. $M_1$ generates a random number $r_1$ ($1 \leq r_1 \leq p$), and sends $\alpha^{r_1}$ to $M_2$, such that $\alpha$ and $p$ are constants known by each node. On the other hand, $M_2$ generates a random number $r_2$, and sends $\alpha^{r_2}$ to $M_1$. Thereby, each node could compute the common key which is $\alpha^{r_1 \times r_2}$. This solution is based on the discrete logarithmic arithmetic, and also relies on the agreement on the parameters $\alpha$ and $p$ between the two nodes. Although

[3] Public keys also might be used in group communications, especial when using digital signature

it is simple and limited to two nodes common key establishment, this protocol was used to design more sophisticated protocols, as we will see in the following.

### General Diffie-Hellman (GDH)

Steiner et al. [STW96] proposed a n-party generalization of the basic two-party DH protocol (described before). The new protocol consists of $n$ rounds, allowing $n$ nodes to establish a common key. In the first $n-1$ rounds, contributions are collected from each node. In the first round, $M_1$ generates $r_1$ and computes $\alpha^{r_1}$, that it sends to $M_2$. In the second step $M_2$ generates $r_2$, computes $\alpha^{r_2}$ and sends it to $M_3$, along with $\alpha^{r_1}$ and $\alpha^{r_1 \times r_2}$. This latter sends to $M_4$ (after making the required computations) the third round partial factors, i.e. $\alpha^{r_1 \times r_2}$, $\alpha^{r_1 \times r_3}$, $\alpha^{r_2 \times r_3}$, as well as the third round partial key $\alpha^{r_1 \times r_2 \times r_3}$, and so on for each $M_i$ $(i < n)$.

Upon the $(n-1)^{th}$ round, the collector node $M_n$ receives the $(n-1)^{th}$ round partial factors, and the $(n-1)^{th}$ round partial key, then generates its random number and computes the final key $K$ [4]. In the last round, node $M_n$ sends each $M_i$ the appropriate $(n)^{th}$ round partial factor, i.e. $\alpha^{(\prod_{j=1}^{n} r_j)/r_i}$. Consequently, each node uses its random number to compute the common key $K$. Note that partial factors are used to avoid sending the final resulted key during the last round. Also note that the $(n-1)^{th}$ round requires $n-1$ operations (sending the partial factor to each node), which makes the computational complexity of the solution $O(2 \times (n-1))$ as shown in table 2.3. Figure 2.7 is an illustrative example of this protocol.

Even though it uses a collector, this solution is contributory, since each node contributes in the key computation with the random number it generates. Nevertheless, the major drawback of this solution is the important overhead, due to the message size rising from round to round [5]. This can also cause a problem of scalability.

### Password authentication based key exchange

Another approach proposed for distributed systems and largely adopted for MANET is password authentication. The aim of this technique is to build a strong private key from a weak share. To this end, Bellovin and Merrit [BM92] have proposed a protocol called "Encrypted Key Exchange" (EKE), in which two parties could drive a strong key from a weak shared password. Unlike the password, the constructed key is not vulnerable to dictionary attack [6], which makes it usable for long period. The authors assumed that the involved nodes, $M_1$ and $M_2$, share a weak secret password P, and use a public function as well as a one way function($f(.,.)$). $M_1$ is also assumed

[4] $K = \alpha^{(\prod_{j=1}^{n} r_j)}$

[5] Particulary, the partial factors

[6] Dictionary attack consists of employing a dictionary to automatically and continuously guest words attempting to break a password

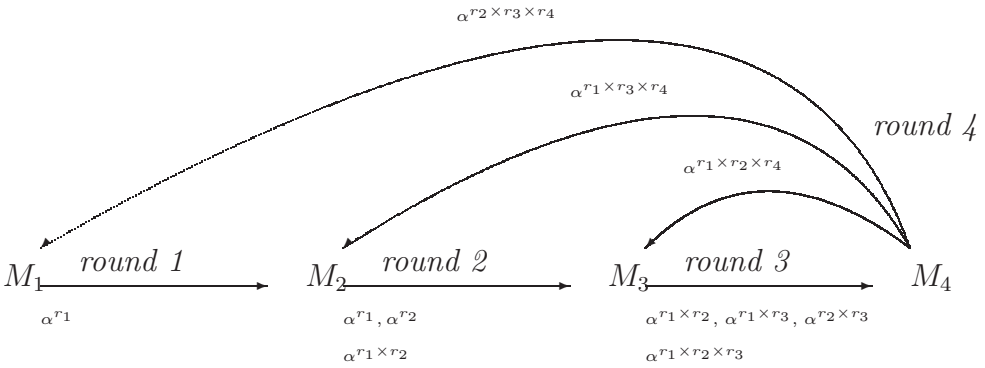


Figure 2.7: Example of General Diffie-Hellman with 4 nodes

to possess a random key pair, respectively for encryption and decryption. The two nodes establish a strong key after a 5 rounds message exchange.

During these rounds, each node $M_i$ generates a random string ($S_{M_i}$) it confidentially exchange with the other. Using the public function and the random keys, $M_1$ ensures authentication of $M_2$ in the third step, i.e. $M_1$ ensures that $M_2$ is a node that knows p. This latter ensures that $M_1$ knows p upon the forth step, using the previous functions and a random key it generates during the first steps. At that point, each node computes the common key:
$K = f(S_{M_2}, S_{M_2})$
More details of the algorithm and its steps can be found in [BM92].

One obvious proposed way to extend this protocol to multi-party is to elect a leader, and to execute the previous two-party protocol between each node and the leader. At the end of the protocol run, each node shares a key with the leader. An additional step will be required for this latter to pick a common key, and to distribute it to all the parties using the pairwise session keys it shares with them. Overall, the computation complexity of this n-party solution is $O(6 \times (n-1))$; $5 \times (n-1)$ for the execution of the 5 steps two-party protocol between each node and the leader, and $(n-1)$ for the additional step.

One disadvantage of this solution is its overhead, since each step consists of one-to-many (from $M_n$ to each $M_i$) or many-to-one (from each $M_i$ to $M_n$) message exchange, which is very expensive in a multi-hop environment. Another drawback of this solution is that the common session key choice is not contributory but unilateral, viz. unlike the general Diffie-Hellman protocol described before, the leader selects by itself the common key, thus the key is not inevitably composed of contributions of other parties. The only contribution of these nodes is for the pairwise session keys construction. To mitigate this problem Asokan and Ginzboorg [AP00] have proposed a

contributory multi-party version, that we will present hereafter.

### Contributory password authentication

Asokan and Ginzboorg [AP00] have proposed a contributory extension of the previous solution. The new protocol also relies on a collector node, and is composed of the following four steps:

1. The collector $M_n$ sends to all nodes its public key $E$ [7] encrypted with the shared secret $P$.
2. Each node $M_i$ sends $M_n$ two random quantities $S_i$ and $R_i$, double-encrypted with $E$ and $P$. At this point, $M_n$ can ensure that each party knows P.
3. $M_n$ sends each $M_i$ the set $\{S_j, j = 1..n\}$ encrypted with $R_i$.
4. Finally, each $M_i$ computes the private key $K = f(S_1, ....., S_n)$, where $f()$ is an n-input one-way function. Each $M_i$ sends $K(S_i, H(S1, S2, .....Sn))$ to $M_n$ for key confirmation, such that $H()$ is a public hash function.

In this solution, each node $M_i$ participates in the construction of $K$ with $S_i$, which makes the solution contributory. This solution requires only 4 steps instead of 6, which decreases its latency and improves its computational complexity compared with the previous one (as shown in table 2.3).
However, the two password authentication-based solutions require a pre-share of a secret, which is not always possible in dynamic networks.

### Hyper-cube based approach

Another extension of Diffie-Hellman (DH) protocol to n-party adopted for MANET have been proposed by Becker and Wille [BW98] [8]. In this fully distributed approach, nodes are arranged in a $d$ dimensional hypercube, such that $d$ fulfils the following condition: $2^d$= nodes number.

To clarify the approach we first describe it for $d = 2$, i.e nodes number n = 4, then we will generalize this description. For four nodes $M_{1..4}$, the protocol runs two rounds (steps). In the first one $M_1$ and $M_2$ engage a two-party DH protocol and calculate $S_{1,2} = \alpha^{r_1 \times r_2}$. Similarly and simultaneously, $M_3$ and $M_4$ engage a two-party DH protocol and calculate $S_{3,4} = \alpha^{r_3 \times r_4}$. In the next round, $M_1$ and $M_3$ participate in a new two-party DH protocol using respectively $S_{1,2}$ and $S_{3,4}$ as the random number r.

[7] This key is just temporary used for the private key establishment, it can be generated randomly by the node.

[8] Note that this approach has been proposed for group key, but it can be generalized to general private key

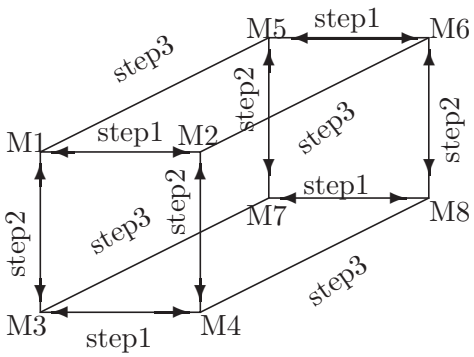


Figure 2.8: Example of the Hyper cube protocol

$M_2$ and $M_4$ do the same thing simultaneously. The result at the end of the protocol is the computation of $K = \alpha^{r_1 \times r_2 \times r_3 \times r_4}$.

This approach can be easily generalized for $d > 2$ as follows: In each dimension j of the n d-dimensional hyper-cube, each node has one neighboring partner, with which it performs a two-party DH during the $j^{th}$ round using the outcome of the previous round ($(j-1)^{th}$), as described before. Upon the $d^{th}$ round, each node will share the same key with the others.

Figure 2.8 summarizes the protocol for $d = 3$. The protocol execution takes only $d$ rounds which is $\log_2(n)$, whereas GDH execution requires $n$ rounds. This decreases the computation complexity from $O(2 \times (n-1))$ to $O(2 \times \log_2(n))$ as shown in table 2.3. Moreover, the overhead exchanged by each couple of nodes is limited. It is fixed and unaffected by the steps number, unlike GDH where the overhead increases from a step to another. Thanks to these advantages, the hyper-cube based solution might be more scalable than GDH. Nevertheless, the price is that all the nodes are involved in each step.

This solution needs the number of nodes $n$ to have the form: $n = 2^d$, otherwise (when $2^d < n < 2^{d+1}$) it does not function. The protocol 'octopus' [BW98] solves this problem as follows: The first $2^d$ nodes play the role of central controllers. The remaining $n - 2^d$ nodes are distributed among the central controllers in such a way that there will be no more than one node associated to each controller. In a first additional round, controllers simultaneously execute two-party DH with their associate nodes. Afterwards, controllers engage in a $d$ round hyper-cube protocol run using the information collected in the first round, resulting in a common key sharing. In a last additional round, the key is distributed to the associated nodes. Another hyper-cube approach problem remains untreated with 'octopus', namely the nodes arrangement in the hyper-cub, viz. the construction of partners in each dimension,

which is equivalent to nodes ordering. If this ordering is predefined and static, then the overhead becomes important, and the message exchange between two partners would not cost inevitably one unit as expected, because partners could not remain neighbors during all the protocol executions (due to nodes mobility). On the other hand, establishing nodes ordering dynamically requires important overhead when node mobility increases.

**Some hybrid solutions**

Asokan and Ginzboorg [AP00] proposed a hybrid solution where they add the password authentication principle to GDH. It is indeed a generalization of the password authentication application to the two-party DH (generalization of authenticated two-party DH), that we describe hereafter:

Assume that the two nodes A and B share a password $P$. The execution of authenticated two-party DH between these nodes consists of 4 steps. In the first one, A *encrypts* its contribution ($\alpha^{r_1}$) with $P$ and sends the outcome to B. This latter do the same thing with its contribution ($\alpha^{r_2}$) and computes the common key $K = \alpha^{r_1 \times r_2}$, and then it sends A its contribution encrypted with $p$ along with another random number $C_b$ (it generates) encrypted with $K$. In the third step, A retrieves $\alpha^{r_2}$ (after decryption), computes K that it uses to get $C_b$, and generates a random number $C_a$. Afterwards, it encrypts both $C_a$ and $C_b$ with $K$ and sends them to B. Getting back $C_b$ ensures A's authentication for B. This latter does the same thing and sends back $C_a$ encrypted with K in the final step. At that point, A will be convinced about B's authentication.

The authors proposed to use this basic protocol with GDH as follows: After collecting the $(n-1)^{th}$ round GDH partial key ($\alpha^{r_1 \times \ldots \times r_{n-1}}$), node $M_{n-1}$ sends each node this key, then each node sends $M_n$ another item $C_i$ (computed from the partial key and a random number) ciphered with $p$. This latter sends back to each $M_i$ a combination [9] of $C_i$ and its own contribution $\alpha^{r_n}$, enabling each node to compute the final key $K = \alpha^{r_1 \times \ldots \times r_n}$. The protocol takes $4 \times (n-1)$ steps: $2 \times (n-1)$ to compute and distribute the $(n-1)^{th}$ GDH partial key, $(n-1)$ to send to $M_n$ items, and $(n-1)$ to send combinations to each $M_i$. Note that the overhead of the $(n-1)^{th}$ GDH partial key computation steps has been decreased, since partial factors are not transmitted during these steps. The same authors proposed another simple hybridation, by adding the same principle (password authentication) to the hyper-cube based key agreement. The idea is merely to use the basic authenticated two-party DH in each step. Consequently, the hyper-cube based key agreement computation complexity is multiplied by 2 in this hybrid solution.

[9] More details about this combination computation and the protocol steps can be found in [AP00]

| Solutions | Distribution | Based on secret share | Contributory | Computational complexity |
|---|---|---|---|---|
| **GDH** | Partially (uses collectors) | No | Yes | $2 \times (n-1)$ |
| **n-party Pwd authentication** | Partially | Yes | No | $6 \times (n-1)$ |
| **contributory n-party Pwd authentication** | Partially | Yes | Yes | $4 \times (n-1)$ |
| **Hyper-cube** | Totaly | No | Yes | $2 \times \log(n)$ |
| **Octopus** | Almost totaly | No | Yes | $2 \times (\log(n)+1)$ |
| **Hyper-cube + Pwd** | Totaly | Yes | Yes | $4 \times \log(n)$ |
| **GDH+Pwd** | Partially | Yes | Yes | $4 \times (n-1)$ |
| **Cluster-based** | Partially (only between leaders) | No | Yes | |

Table 2.3: Private Key management solutions

Another hybrid solution was proposed in [LWF02], where Li et al. suggest to combine centralized and key agreement approaches. The main idea is to cluster nodes in groups according to their physical position, and to associate a leader (connector) to each group. The key agreement protocol is executed between these connectors, which distribute the resultant key to members of their groups. This solution is general and lacks of detailed precisions on the key agreement protocol to use, which is indeed problematic. Moreover, the groups should be reconstructed when nodes change their positions, which requires important overhead and extra computation steps when the mobility increases.

Table 2.3 summarizes some features of the n-party solutions discussed in this section with respect to different issues. The first one is the distribution. Some solutions are totally distributed, and all nodes participate to the execution of the protocol; others are partially distributed since the protocol execution is restricted to a subset of nodes. GDH, hyper-cube based, octopus, and cluster-based solutions do not require any pre-sharing of secret; the others require a pre-sharing of some password. Except the standard n-party password-based, all the other solutions are contributory, i.e. the key is not chosen unilaterally by one node but different nodes contribute for its selection. Computation complexity of each solution discussed in this section is given in the table. Note that the computational complexity of the cluster-based hybrid protocol depends on the key agreement protocol to use, which was not specified.

### Rekeying

An important issue related to private key, especially in *dynamic* group communication (relying on symmetric group key), is *rekeying*. The group key must be revoked and redistributed to all the remaining nodes in a secure, reliable, and timely fashion whenever group membership changes, particulary when a node leaves the network.

This problem has been extensively studied in the wired networks context and several protocols have been proposed. Rekeying schemes have been categorized into two classes, *stateful* and *stateless* [ZXJ04]. The stateful class includes several protocols based on the use of logical key trees, such as LKH [WGL98] and ELK [PST01]. In these protocols, keys are encrypted by the key server before distribution. The key server uses key-encryption keys (keys used to encrypt keys) that were transmitted to members during previous rekeying operations to encrypt the keys to be transmitted in the current rekeying operation. Thus, a member have to receive all the key encryption keys of interest in all the previous rekeying operations, otherwise it will not be able to decrypt the new key.

Stateless group rekeying protocols [NNL02, LNS03, SMF+02] form the second class of rekeying protocols. In these protocols, a legitimate user only needs to receive the key of interest in the current rekeying operation to decode the current group key. The stateless feature makes these protocols very attractive for dynamic ad hoc networks. However, all these protocols have much higher communication overhead than the stateful protocols. The protocols of both classes are centralized and not directly applicable to ad hoc networks.

Thus far, only few works have been devoted to rekeying in ad hoc networks. Lazos and Poovendran [LP03] have proposed a new solution by adapting the LKH scheme for ad hoc networks, but the proposed solution suffers from a high communication and computation overhead [ZXJ04]. Another novel solution devoted to MANET has been proposed by Kaya et al. [KLNY03]. This solution uses public-key techniques and is somewhat expensive in both communication and computation. An interesting protocol with regard to the overhead is GKMPAN [ZXJ04]. However, this protocol provides only *partial statelessness*. Further, the simulation study performed to assess GKMPAN [ZXJ04] has solely considered a static network, thus the satisfactory results only reflect a network with no node mobility. We think mobility is a very important feature of MANET, so an efficient rekeying scheme should consider this issue, and should require low cost for distributing the new key in a mobile environment.

Rekeying remains an open research topic, and proposing a fully stateless and efficient protocol that takes into account MANET's features, especially the infrastructureless and node mobility, is a challenging problem.

### Group communication and private keys

Group communication technologies have proven their importance in different fields of our daily life, such as education, entertainment, and other industries. E-learning and video conferences are examples of applications that belong to these fields.

Group communication, both one-to-many or many-to-many, has become increasingly important in ad hoc networks. In MANET group communication, issues differ from those in traditional networks because of the features discussed before, notably the variable and unpredictable nature of the wireless medium, the resource limitation, the infrastructureless nature, and the node mobility.

Multicast routing in ad hoc networks can be classified into tree-based and mesh-based approaches [DMM03]. In tree-based multicast routing protocols data packets are forwarded on a single path to a given receiver. The union of the paths to all receivers forms the multicast-tree, which may be sender specific or common to all senders in a multicast session. Examples of tree-based multicast routing protocols for MANET are: Bandwidth Efficient Multicast Routing Protocol [OKS99], Multicast Ad Hoc on Demand Distance Vector Routing (MAODV) [RP99], Multicast Core Extraction Distributed Ad Hoc Routing (MCEDAR) [SSB99], and Adaptive Demand-Driven Multicast Routing (ADMR) [JJ01]. Typically these approaches include mechanisms such as local repair of the distribution tree or backup paths in order to compensate for the frequent topological changes. In mesh-based approaches there may be multiple paths to each receiver. This redundancy provides increased protection against topological changes. Examples of mesh-based multicast routing protocols for mobile ad-hoc networks are: On Demand Multicast Routing Protocol (ODMRP) [LGC99], Core-Assisted Mesh Protocol (CAMP) [GLAM99], Neighbor Supporting Ad hoc Multicast Routing Protocol (NSMRP) [LK00], and Dynamic Core Based Multicast Routing Protocol (DCMP) [LN02].

A private group key infrastructure is essential to secure such multicast communication, regarding both routing and data information. Moreover, in this type of communication the group membership is dynamic, i.e. nodes leave and join the group continuously. An efficient rekeying mechanism is therefore mandatory to ensure a robust multicast system. Providing efficient group communication is one of the main issues in MANET. The nature and the features of ad hoc networks make this issue even more challenging. While these networks are rapidly gaining popularity, there is a strong need to develop efficient strategies to support group communication. Particulary, it is essential to suggest new efficient solutions for private key distribution and revocation, that cope with MANET's features.

### 2.4.2 Public key infrastructure

Thus far, we have presented some solutions related to the private key management. In this part, we will treat the public key management problem. In a public key infrastructure, each node has a public/private key pair. Public keys can be distributed to other nodes, while private keys should be kept confidential to individual nodes. This kind of keys is essential for any service or application that employs asymmet-

ric cryptography, like many of the techniques described in 2.3.2, which use digital signature. In traditional public key infrastructure, there is a trusted entity called certification authority (CA) that distributes nodes' public keys in *certificates*. The CA has a public/private key pair. The private key is used to sign certificates binding public keys the CA provides for nodes, while the public key is used by nodes to check the certificates authentication.

However, it is problematic in MANET to establish a key management service using a *single* CA. A standard approach to improve the service availability is replication, but a naive replication of the CA makes the service more vulnerable, since compromising any single replica that possesses the service private key could lead to collapse of the entire system. To solve this problem, recent solutions propose to distribute the trust over a set of nodes by letting them share the key management responsibility. In the following, we present these solutions.

**First general solution**

Zhou and Haas [ZH99] are the first who have treated the public key management in MANET. They have proposed the use of threshold cryptography [Sha79] in order to distribute the trust over a set of nodes. In their approach, the key management service with a $(n, k+1)$ configuration $(n \geq 3k+1)$ consists of $n$ special nodes, called servers. Each server stores public keys of all the nodes in the network, and knows the public keys of the other servers. Thus, servers can establish secure links among them. The authors assume that the adversary cannot compromise more than $k$ servers in any period of time with a certain duration.

The proposed $(n, k+1)$ threshold cryptography scheme allows n parties to share the ability to perform a cryptographic operation (e.g., creating a digital signature), so that any $k+1$ parties can perform this operation jointly, whereas it is infeasible for at most $k$ parties to do so. Therefore, the service private key $k$ is divided into $n$ shares, assigning one share to each server. For the service to sign a certificate, each server generates a partial signature for the certificate using its private key share and submits the partial signature to a *combiner* (any server). With $k+1$ correct partial signatures, the combiner will be able to compute the signature for the certificate. To make sure that a compromised combiner cannot prevent a signature from being computed, the authors suggest to use $k+1$ servers as combiners. This ensures that at least one combiner is correct and is able to compute the signature, since the maximum number of compromised servers is assumed not to exceed $k$. A combiner can verify the validity of a computed signature using the service public key. In case a verification fails, the combiner tries another set of $k+1$ partial signatures. This process continues until the combiner constructs the correct signature.

To make the solution fault tolerant (especially to tolerate mobile adversaries [OY91]), share refreshing has been proposed [ZH99]. This technique consists of rebuilding new sharing of the same service private key $SK$, and is performed as follows: First, each server randomly generates $(s_{i1}, s_{i2}, ..., s_{in})$, an (n, k+1) sharing of 0. Then, every sub-share $s_{ij}$ is distributed to server j through a secure link. When server j gets the sub-shares $s_{1j}, s_{2j}, ..., s_{nj}$, it just sums and adds them to its old shares $s_j$ to get its new share $s'_j$ [10].

This first solution is very general, it lacks precision on how to select servers. An important drawback of this solution we remark [DKB05] is that the communication pattern of the proposed protocol between a collector and $k+1$ or more servers is one-to-many-to-one (*manycast*), viz. a collector asking for an authentic certificate of some node needs to contact at least $k+1$ servers and to receive at least $k+1$ replies, which requires important overhead and long delay. Another drawback is related to the parameter $k$; the authors suggest that this parameter should reflect the maximum number of the nodes that can be simultaneously compromised by a malicious, but no precisions on how determine $k$'s value have been illustrated.
On the other hand, this solution tackles the problem of fault tolerance, especially against mobile adversaries [OY91], and proposes the efficient technique of share refreshing.

**MOCA**

Another solution that employs threshold cryptography (subsection 2.2.1) to distribute CA functionality over specially *selected* nodes is MOCA [YK03]. The choice of these nodes or MOCAs (MObile Certificate Authorities) is based on their security and their physical characteristics. That is, mobile nodes may be heterogeneous in many respects, especially in terms of their security. In this case, the most secure nodes would be selected as MOCAs. To illustrate this heterogeneity, the authors gave the example of a battlefield scenario, where nodes are soldiers that have different ranks. According to their ranks, nodes would be equipped with computers that are different in power, capabilities, transmission ranges, levels of physical security, and so on. In such a case, the most secure and powerful nodes (of the most ranked soldiers) should be selected as MOCAs.

In the proposed protocol (MOCA certification Protocol or MP), a client that requires a certification service sends a Certification Request (CREQ) packet, then any MOCA that receives a CREQ accordingly responds with a Certification Reply (CREP) packet containing its *partial signature*. The client waits a fixed period of time for $k$ such CREPs. When the client collects $k$ valid CREPs, it can construct the

[10] $s'_j = s_j + \sum_{i=1}^{n} s_{ij}$ is a new share of: $SK + \sum_{i=1}^{n} 0 = SK$

*full required signature* and the certification request succeeds. Hence, unlike the previous solution, in this one the requestor itself acts as a combiner. If too few CREPs are received, the client's CREQ timer expires and the certification request fails. On failure, the client can retry or proceed without the certification service. The CREQ and CREP messages are similar to route request (RREQ) and route reply (RREP) messages of on-demand routing protocols.

The shape of a MOCA framework is determined by the total number of nodes in the network, the number of MOCAs, and the threshold value for secret reconstruction (number $k$ of MOCAs required to issue certificates). The total number of nodes in the network (M), that can change dynamically over time, is not a tunable parameter. The number of MOCAs (n) is determined by the characteristics of nodes in the network, such as physical security or processing capability, and it is also not tunable. Given $M$ and $n$, the last parameter $k$ which is the threshold for secret recovery, is indeed a tunable parameter for the proposed solution. Once $k$ chosen and the system deployed, it is expensive to change $k$. Therefore, it is important to understand the effects of varying $k$ on a given system. $k$ can be chosen between 1 (a single MOCA) and $n$ (all MOCAs). Setting $k$ to a high value has the effect of making the system more secure against possible adversaries. Since $k$ is the number of MOCAs an adversary needs to compromise to collapse the system. But at the same time, a high value can cause huge communication overhead for clients since any client needs to contact at least $k$ MOCAs to get a certificate. Therefore, the threshold $k$ should be chosen to balance the two conflicting requirements, and it is clear that no one value will fit all systems.

It is possible that an ad hoc network does not have enough heterogeneity among its nodes, which may make it difficult even impossible to choose MOCAs based on this heterogeneity assumption. In such cases the authors suggest to choose *randomly* a subset of nodes as MOCAs. We think this selection might be inefficient, and a rigorous selection strategy that fulfills a defined criterium, like decreasing the overhead or the latency, is required. This still represents an open research area. Note that like the previous solution, MOCA also suffers from the important overhead and long delay.

**Certificate chain based solution**

Capkun et al. [CBH03, HBC01] have proposed a fully distributed self-organizing public-key management system in which nodes generate their keys, and then issue, store, and distribute public-key certificates. This system is similar to PGP (Pretty Good Privacy) [Zim95] in the sense that public-key certificate of each node is issued by the node itself. However, in order to remove the reliance on on-line servers (which are clearly incompatible with MANET), the proposed system does not rely on certificate directories [Zim95] for the distribution of certificates. Instead, certificates are stored and distributed by the nodes, and each node maintains a local certificate *repository*

that contains a limited number of certificates selected by the node according to an appropriate algorithm.

When node $u$ wants to verify the *authenticity* of $v$'s public key, the two nodes ($u$ and $v$) merge their local certificate repositories, then $u$ tries to find an appropriate certificate *chain* from $u$ to $v$ in the merged repository. An algorithm for the construction of local certificate repositories has been proposed [CBH03]. The basic operations of this public-key management scheme are:
**Creation of public keys:** The public key and the corresponding private key of each node is created locally by the node itself. Note that no details about keys creation has been fixed in the two previous solutions.
**Issuing public-key certificates:** If a node $u$ believes that a given public key $K_v$ belongs to a given node $v$, then $u$ can issue (using its own signature) a public-key certificate regarding $K_v$. There may be many reasons for $u$ to believe that $K_v$ belongs to $v$. For instance, $u$ may receive $K_v$ on a secure channel that is associated with $v$, or someone trusted by $u$ claims that $K_v$ belongs to $v$.
**Storage of certificates:** Certificates issued in the system are stored by nodes in a fully decentralized way. Every node maintains a local certificate repository that has two parts: First, each node stores the certificates that it issued. Second, each node stores a set of additional certificates issued by other nodes, which are selected according to an appropriate algorithm given in [CBH03]. This additional set of certificates is obtained from other nodes, thus some underlying routing mechanisms are assumed to exist.
**Key authentication:** When a node $u$ wants to obtain the authentic public key $K_v$ of another node $v$, it asks other nodes (possibly $v$ itself) for $K_v$. In order to verify the authenticity of the received key, the requested node provides $u$ with its local certificate repository, then $u$ merges the received repository with its own repository and tries to find an appropriate *certificate chain* from $K_u$ to $K_v$ in the merged repository.

Unlike the two previous solutions, this one is fully distributed and more suitable for MANET environment. It also gives more precisions on the issuing of public keys. Moreover, this solution does not use threshold cryptography, and it decreases both the overhead and the latency, since a requestor needs only to ask one node for the certificate instead of asking $k$ different servers.

However, we should note that this approach provides only *probabilistic* guarantees, viz. it does not ensure to a requestor node to get a certificate, since no node saves keys of all the other nodes. On the other hand, in the previous solutions collectors maintain keys of all nodes, so these solutions are deterministic and guarantee providing correct certificate if no adversary can simultaneously compromise enough servers to jeopardize the solution [11], provided that there is no partition in the network preventing nodes

[11] $k + 1$ servers for the first solution and $K$ for the second

| Solutions | Distribution | Based on threshold crypto | Collector when using Threshold crypto | Overhead | Latency | Guaranty |
|---|---|---|---|---|---|---|
| **First solution** | partially | yes | any server | important | important | deterministic if no partition |
| **MOCA** | partially | yes | the requestor | important | important | deterministic if no partition |
| **Certificate chain based** | fully | no | | moderate | short | probabilistic |

Table 2.4: Public key management solutions

to contact the servers. All these features are recapitulated in table 2.4.

## 2.5 Intrusion Detection Systems (IDSs)

An intrusion may be defined as: "any set of actions that attempt to compromise the integrity, confidentiality, or availability of a resource" [RGAM90], or "Any unauthorized or unwanted activity on a system or a network" [Lee03]. An IDS may be defined as: "A system that tries to detect and alert on attempted intrusions into a system or a network" [Lee03].
The history of security research has taught us a valuable lesson, no matter how many intrusion prevention measures are inserted in a network, there are always some weaknesses in the systems that one could exploit to break in [ZLH03]. These weaknesses include design and programming errors, and various social engineering penetration techniques as well. Hence, Intrusion prevention measures (proactive solutions) cannot eliminate attacks, and they must be fortified with IDSs. An IDS presents a second wall of defense and is essential for any high-survivability network.

The primary assumptions of intrusion detection are:

- User and program activities are observable, for example via system auditing mechanisms, and more importantly,
- normal and intrusion activities have distinct behavior.

Therefore, intrusion detection involves capturing *audit data* and reasoning about the evidence in the data to determine whether the system is under attack. There are mainly two classes of IDSs:

- Anomaly detection: These IDSs consider activities that deviate significantly from the established normal usage profiles as anomalies, i.e. possible intrusions, where *normal* patterns are defined beforehand. For instance, the normal profile of a user may contain the averaged frequencies of some system commands used in his login sessions. If for a session that is being monitored the frequencies are

significantly lower or higher, then an anomaly alarm will be raised. The main advantage of anomaly detection is that it does not require prior knowledge of intrusions and can thus detect new intrusions. The main disadvantage is that it might be unable to describe what the attack is, and might have high false positive rate. An example of such kind of IDS in traditional networks is IDES [LTG+92]

- Misuse detection (signature-based): they rely on the use of specifically known patterns of well-known unauthorized behavior and attacks to match and identify *known* intrusions. For instance, a rule for the "guessing password attack" can be "there are more than four failed login attempts within two minutes" [ZLH03]. The main advantage of this technique is that it can accurately and efficiently detect instances of known attacks. Its main drawback is that it lacks the ability to detect the truly innovative (i.e. newly invented) attacks, whose patterns are unknown. IDIOT [IKP95] and STAT [SH95] are examples of signature-based IDSs in traditional networks.

Another classification of traditional IDSs is based on the type of the audit data used, it includes:

- Network-based IDS: Normally, an IDS of this category runs at the gateway of a network where it captures and examines packets that go through the network hardware interface.

- Host-based IDS: Relies on operating system audit data to monitor and analyze the events generated by programs or users on a host.

An intrusion detection model has two components: *The features* (attributes or measures), and *the modeling algorithm*. Defining a set of predictive features that accurately capture the representative behaviors of intrusive or normal activities is the most important step in building an effective intrusion detection model, and it can be independent of the design of the modeling algorithm. This latter is a rule-based pattern matching that uses the features to identify intrusions.

### 2.5.1 Problems of traditional IDSs

The vast difference between traditional networks and MANET makes it very difficult to directly apply traditional intrusion detection techniques to MANET. The most important difference is the infrastructureless of MANET. In other words, unlike wired networks where traffic monitoring is usually done at switches, routers and gateways, *a MANET does not have such traffic concentration points* where the IDS can collect audit data for the entire network.
The second big difference consists of MANET's limitation. IDS modules when implemented within MANET nodes should take these limitations into account and should

not consume too much resources (battery, CPU, bandwidth, etc.).

Another big problem with MANET is the lack of a clear separation between normalcy and anomaly. For instance, a node that sends out false routing information could be merely someone that has temporarily out-of-date routing information, due to volatile physical movement. Intrusion detection may find it increasingly difficult to distinguish false alarms from real intrusions.

Anjum et al. [ASS03] have investigated by simulation the ability of various MANET routing protocols to facilitate intrusion detection to a traditional host signature based IDS. More precisely, they have measured the detection rate of some attacks whose signatures are assumed to be know by the IDSs modules, which have been assumed to perform *independently*. The results show that the detection rates are dramatic, especially when node mobility increases. This illustrates the inefficiency of traditional host-based IDSs when directly applied to MANET. We also realize that nodes cooperation is mandatory.

### 2.5.2 Novel solutions

Recently some IDSs have been proposed for MANET. They are all distributed, host-based, anomaly-based, and cooperative. The cooperation, however, may be fully and equally distributed among nodes, or it may be based on a hierarchical nodes organization. The IDS design differs from a solution to another, as we will see in the following paragraphs.

#### Correlation-based IDS

In [HFLY03], a correlation and learning based approach for constructing anomaly detection models for MANET routing protocols has been proposed. The authors claimed that a strong feature correlation exists in normal behavior, and that such correlation can be used to detect deviations caused by abnormal (or intrusive) activities. To explore such a correlation, the authors have developed a cross-feature analysis anomaly detection approach, and suggest to compute a *classifier* to each defined feature. Each classifier is *learned* from a *set of training data*, and helps computing the most likely value of the corresponding feature given the values of the other features (conditional probability).

The original anomaly detection problem, i.e. whether a record (a set of observed values of the features) is normal or not, is solved as follows: Given a record, first apply each classifier to compute the probability of the appropriate feature given the values of the others, then compute the average probability and compare it with a *predefined* threshold. If the average probability is lower than the threshold an alarm is raised. To clarify this approach, we give the following simple illustrative example

[HFLY03]:

**Example:** Consider a simple ad hoc network with two nodes. Packets can only be sent from one end to the other if they are within each other's transmission range. The following three features are defined.

1. Is the other node *reachable*?
2. Is there any packet *delivered* during the last five seconds?
3. is there any packet *cached* for delivery during the last five seconds?

For simplicity, all features are assumed to take binary values, i.e. either true or false. All events enumerated in table 2.5 are normal, whereas all the other combinations not enumerated in this table are abnormal. For instance, $delivered = true$ while $reachable = false$ is an abnormal situation, since no delivery is possible if the corresponding node is unreachable. Therefore, $(false, true, true)$ and $(false, true, false)$ are abnormal events (records).

An illustrative classifier can be used in this example that works for each feature $f_i$ as follows:

- If only one record is normal where the two other features have been assigned with a particular set of values, the record is selected as the predicted one with the associated probability of 1, i.e. probability of $v_i$ (the possible value of $f_i$) given the other values is 1. For instance, when a packet is delivered and cached during the past five seconds ($delivered = cached = true$) the corresponding node is inevitably reachable, thus the probability of $recheable = true$ given $delivered = cached = true$ is 1.
- If both records are normal, the associated probability of each one is 0.5. For example, when no packet is neither delivered nor cached during the past five seconds ($delivered = cached = false$), the corresponding node could be either reachable or unreachable. The probability of $recheable = true$ (respectively $recheable = false$) given $delivered = cached = false$ is thus 0.5.
- If both are abnormal, the associated probability of each one is 0.5.

Table 2.6 shows conditional probabilities and their averages for each possible record; the features reachable, delivered and cached are denoted respectively by r, d, c. As it can be realized from tables 2.5 and 2.6, a threshold of 0.5 allows to correctly classify all the events.

In the proposed IDS, this table is not pre-computed and saved by nodes, but each line is computed when the appropriate record is observed during the monitoring period (five seconds in this example). For instance, when the record (false, true, true)

| Reachable | Delivered | Cached |
|---|---|---|
| true | true | true |
| true | false | false |
| false | false | true |
| false | false | false |

Table 2.5: Normal events

| Reachable | Delivered | Cached | Class | p(r/d,c) | p(d/r,c) | p(c/r,d) | Average |
|---|---|---|---|---|---|---|---|
| true | true | true | normal | 1 | 1 | 1 | 1 |
| true | true | false | abnormal | 0.5 | 0 | 0 | 0.17 |
| true | false | true | abnormal | 0 | 0 | 0 | 0 |
| true | false | false | normal | 0.5 | 1 | 1 | 0.83 |
| false | true | true | abnormal | 0 | 0 | 0.5 | 0.17 |
| false | true | false | abnormal | 0.5 | 0 | 0.5 | 0.33 |
| false | false | true | normal | 1 | 1 | 0.5 | 0.83 |
| false | false | false | normal | 0.5 | 1 | 0.5 | 0.67 |

Table 2.6: Probabilities table

is observed, $p(r = false/d = true, c = true)$, $p(d = true/r = false, c = true)$ and $p(c = true/r = false, d = true)$ will be computed (using the classifier), whose values are respectively: 0, 0, 0.5. These probabilities will be averaged and compared with the threshold, then an alarm is raised since the average value (0.17) is bellow the threshold.

This simple example with two nodes has been given just for illustration. In their study, however, the authors defined a total of 141 more complicated features, which capture the basic view of the network topology and the routing operations, as well as traffic patterns. During their simulation, the authors used trace data of normal runs for training the anomaly detection models, the set of classifiers and the threshold are computed in this phase. Afterwards, they run a set of various attacks such as: packets drooping (black hole), routing loops creation, sleep deprivation etc., on AODV routing protocol (presented in 2.3.2), and collect the trace data for evaluating the model. First, an anomaly detection model computes features and detects anomalies locally on each node. As soon as there is one detector (on one node) that identifies an abnormal event, this latter is counted as an anomaly alert (true detection or false alarm) for the network. The results show that the model has good performance on detecting anomalies. To provide more accurate information about the attack, its type and the attacker whenever possible, cooperation among nodes is mandatory.

In their subsequent work [Lee03], the authors have proposed a cooperative IDS based on the previous anomaly detection model, where they have defined rules to be activated whenever some anomaly is observed. The rules have been defined for a given set of attacks and are based on traffic analyzes in promiscuous mode. When an alert is provided by the anomaly detection model, the monitoring module (either of the same node or of another one, according to the defined rules related to the anomaly)

is activated to determine the attack. The simulation study shows satisfactory results on the attack type detection and the false alarm rate, especially regarding packets dropping and sleep deprivation (using flooding) attacks.

The satisfactory simulation results reflect the application of the model computed from a sample network to the same one. We think the feature correlation may change from an application to another, and from a network to another. In this case, the learning phase is required each time at the initialization of the network. This requirement might be inappropriate for dynamic ad hoc networks. Therefore, more investigations on different configurations are required, which can represent a research topic [DKB05].

### Cluster-based IDSs

In the previous cooperative solution, each node participate in the monitoring to detect type of attacks. However, when the threat level is low, it might be inefficient to keep each MANET's node always monitoring, since MANET's nodes typically have limited battery power. Instead, Hung and Lee [Lee03] have suggested that *periodically* each cluster of neighboring nodes *randomly* and *fairly* elects a clusterhead, to lead IDS functions for the entire neighborhood (cluster). It instructs the other members on how the feature computation is to take place.

For this purpose, the authors have proposed a clusterhead assignment algorithm that aims at fairly electing a clusterhead, in such a way that every node is a member of at least one cluster. A cluster is defined as a group of nodes that are in the same neighborhood. That is, each node of the cluster is in the one-hop vicinity of each other. As a special case, a node that cannot be reached by anyone else forms a single node cluster. The proposed algorithm ensures randomness in election decision, and also implements *periodical* re-election to guarantee equal service time for every node. The algorithm also guarantees that no node can manipulate the selection process to increase (or decrease) the chance for it (or another node) to be selected. Detailed description of this algorithm can be found in [Lee03]. After clustering, two detection schemes that define how and where features are computed and transmitted have been proposed:

- LFSS (Local Feature Set Scheme): In this scheme, feature computation is still done locally on each node. But anomaly and attack detections are performed by the clusterhead. For each feature sampling period, a randomly selected cluster member (which can be the clusterhead itself) is requested to transmit its whole features set to the clusterhead.

- CLFSS (Clusterhead-Assisted Local Feature Set Scheme): In order to reduce the burdens on cluster members and the traffic overhead, the clusterhead can

help compute some of the features, more specifically traffic-related features. In this scheme, the clusterhead overhears incoming and outgoing traffic on all members of the cluster. The corresponding traffic-related features are computed in the same way as a local node does in LFSS, as if the cluster is a large node as a whole. Nevertheless, the members (more precisely, one member at a time) are still responsible for computing and transmitting other features (traffic-unrelated features) to the clusterhead, like in LFSS. The simulation results shows that CLFSS outperforms LFSS.

We think that this approach (cluster-based) might cause important overhead for clusterheads reconstruction when node mobility increases [DKB05], since clusters are totally depending on the network topology. Moreover, no comparison between the previous approach and this one has been made. Does the cluster-based approach really provide improvement? this remains an open question.

### Agent-based IDSs

**1. Cooperative agent-based IDS:** Zhang et al. [ZLH03] have proposed a novel agent-based architecture in which every node equally participates in intrusion detection and response. Each node is responsible for detecting signs of intrusion (anomalies) locally and independently, but neighboring nodes can collaboratively investigate in a broader range.

In the systems aspect, individual IDS agents are placed on each node. Each IDS agent runs independently and monitors local activities, including user and system activities, as well as communication activities within the radio range. It detects intrusion from local traces and initiates response. If anomaly is detected in the local data but the evidence is inconclusive, then neighboring IDS agents will cooperatively participate in global intrusion detection actions. These individual IDS agent collectively form the IDS system to defend the MANET. The internal of an IDS agent can be conceptually structured into six pieces.

- The data collection module: It is responsible for gathering audit data from various sources, including the system and user activities within the mobile node, the communication activities of this node, and the observable communication activities within the node's neighborhood. Therefore, multiple data collection modules can coexist in one IDS agent.
- The local detection engine: It uses data collected by the data collection module to detect any local anomaly.
- Cooperative detection engine: It is used by the detection methods that need broader data sets or require collaborations among IDS agents
- The local response module: It triggers local actions in the mobile node.

- The global response module: It coordinates actions among neighboring nodes.
- The secure communication module: It provides a high-confidence communication channel among IDS agents.

This framework is very general and can be used to design an IDS that includes different layers, provided that the anomaly detection model defines features and captures attacks of these layers. For example, at the MAC level the following features could be considered: The total number of channel requests during a given period $s$, the number of nodes making a request etc. At the application layer, the features could include: the total number of requests to the same service, the number of different services requested, the average duration of a service etc. [ZLH03].

The authors extended their work by dealing with the network layer and proposing an anomaly detection approach based on classifiers and features correlation (as the first presented solution), which also requires a training phase and has the same drawbacks.

**2. Clustered agent-based IDS:** Kachirski and Guha [KG03] have proposed an agent-based IDS, consisting of three kinds of agents: monitoring, decision making, and action. Monitoring agents are responsible for both network (packet level) and host (user and system levels) monitoring [12]. Decision agents make decision about the intrusion detection evidence, while action agents take action upon an intrusion is detected by the intrusion detection agents. A clustering algorithm has been proposed, which elects clusterheads in charge of holding decision making and *network* monitoring agents; note that the other kinds of agents are present in every node. An important input parameter for this algorithm is the maximum number of hops a clusterhead is located from any other member. Fixing this parameter to one means that each node will have at least one neighboring node hosting a network monitoring agent (clusterhead), which is idem to the clustering algorithm of the second solution. This will provide more accuracy but requires important overhead. To preserve nodes resources, two-hop cluster scheme has been proposed, in which the maximum number of hops between any node and the clusterhead is two. For the same purpose, the size of queues used for packets monitoring has been *limited*, which allows to limit the IDS overhead. Unlike the previous agent-based solution, the decision making of this one is not totaly cooperative but *independent*. That is, not all nodes are responsible for making decision about IDS evidence, but only those hosting decision making agents (clusterheads). These agents use information provided from the local monitoring agents, as well as those regarding network traffic provided by the network monitoring agents. They collaborate to make decisions. Upon detecting an intrusion, the action

[12] We can thus distinguish two subcategories: network monitoring agents and host monitoring agents

| Solutions | Cooperation | Correlation-based | Cluster-based | Agent-based | Drawbacks |
|---|---|---|---|---|---|
| **Correlation-based** | totally | yes | no | no | - all nodes have to monitor network traffic<br>- requires a learning phase |
| **Cluster-based** | partially | yes | yes | no | - overhead for clusters reconstruction<br>- requires a learning phase |
| **Cooperative agent-based** | totally | yes | no | yes | - all nodes have to monitor network traffic<br>- requires a learning phase |
| **Clustered agent-based** | partially | no | yes | yes | - overhead for clusters reconstruction, but less than the second solution<br>- it is very general, and lacks of specifications about the anomaly detection model |

Table 2.7: Main features of the presented IDSs

agent hosted by the intruder node will be informed.

The anomaly detection model has not been specified but left as a future work. The author's perspectives also include a novel cooperative detection algorithm, and investigations of possible attacks on the proposed IDS as well.

Table 2.7 recapitulates the main features and drawbacks of the solutions presented in this section. Note that *correlation-based IDS* is the *cooperative* one (not the basic one) of [Lee03], presented previously. Both the first and the third solutions are *totaly* cooperative, i.e. all nodes equally participate in monitoring and intrusion decision. On the other hand, cluster-based IDSs are partially cooperative, viz not all nodes participate in network monitoring and intrusion decision but only clusterheads are responsible for these tasks. These solutions have overhead for cluster reconstruction, especially when node mobility increases. The last solution decreases this overhead by employing the two-hop cluster scheme. In the totally cooperative solutions every node has to monitor the whole traffic in its vicinity, which might be inefficient when the threat level is low. Except the forth solution which is very general and lacks specification about the anomaly detection model to use, all the others are correlation-based, and thus might require a learning phase, which is inappropriate for dynamic MANET.

## 2.6 Conclusion

In this chapter we have studied different MANET security issues, and we have shown that the special features of this new environment make it more vulnerable to threats, and that solutions developed for standard networks are often either unsuitable or not directly applicable in this environment. We dealt with several problems related to

different network layers. For the network layer we presented different types of attacks on routing protocols, and then we classified and discussed some proposed techniques to mitigate these attacks. Almost all these techniques use public key encryption and thus require certificate authority (CA) for key management, which is also problematic in MANET. Regarding the MAC layer, which has not received enough attention in the literature, we presented the selfishness on channel access misbehavior, which breaks the fairness and greatly affects the network efficiency. The only solution proposed in the literature was presented and discussed. In our discussion we illustrated how this solution may wrongly accuse well-behaving nodes, and how it is unable to detect what we called cooperative misbehavior. This problem also represents a fertile field of research.

As for the application layer, we studied the key management problem, that can also be considered as an underlying mechanism for securing lower protocols such as routing (as shown earlier). Several solutions for private key and public key management techniques were analyzed. We showed that it is a challenge to provide an efficient contributory private key solution where all nodes participate in key construction with minimum overhead and computation. In the public key infrastructure, however, we think issuing public keys is not a great problem and the PGP method (each node locally issues its own keys) can be directly adapted, as proposed in [CBH03]. The problem in this type of key infrastructure is providing authentic and efficient key distribution. By authentic we mean that when a requestor asks for a public key of another node, the protocol should ensure that the right key is being provided, and that no adversary can successfully provide the requestor with a falsified one. On the other hand, efficiency means moderate overhead and latency. Intrusion Detection Systems (lDSs), which are essential when preventive measures fail, were presented. As we saw, MANET features raise the complexity of this problem, creating a wide research area. All of the MANET IDSs we presented are host-based, anomaly-based, and fully or partially cooperative. In networks with a low threat level, it might be irrelevant to keep all nodes monitoring the traffic and equally sharing IDS tasks. Cluster-based solutions suggest the division of nodes into cluster, thereby only clusterheads will be responsible for these tasks. But the cluster construction requires overhead, especially when the nodes mobility increases. More investigation into the efficiency of this approach (cluster-based) is required. Furthermore, the lack of clear separation between normal state and anomaly when designing the anomaly model is a great problem in MANET that the conception of IDSs faces. Consequently, in practice a learning phase will be required each time at the initialization of the network, which might be inappropriate for dynamic ad hoc networks. More investigation into this issue is required.

Security in ad hoc networks remains an interesting research field that includes many topics. We will give more attention to an emergent security problem caused

by the battery limitation, namely selfish misbehavior on packet forwarding. This problem and the solutions proposed in literature will be surveyed in the following chapter.

# Chapter 3

# Selfish misbehavior in mobile ad hoc networks

## 3.1 Introduction

As illustrated earlier, due to the infrastructureless nature of MANET packets sent between distant nodes are expected to be relayed by intermediate ones, which act as routers and provide the forwarding service. The forwarding service is closely related to the routing. It consists in *correctly* relaying the received packets from node to node until reaching their final destination, following routes selected and maintained by the routing protocol. These services (routing and data forwarding) together are at the core of the network layer. After dealing with secure routing in the second chapter, we now tackle another network-layer security problem related to data forwarding.

The nature of MANET makes cooperation among nodes essential for the system to be operational. In some MANET's applications where all nodes belong to a single authority (in the application layer point of view) and have a common goal, e.g. soldiers in a military unit during a battlefield or rescuers in a rescue team during a rescue operation, nodes are cooperative by nature. However, in many civilian applications, such as networks of cars and provision of communication facilities in remote areas, nodes typically do not belong to a single authority and do not pursue a common goal. In such networks, forwarding packets for other nodes is not in the direct interest of anyone, so there is no good reason to trust nodes and assume that they always cooperate. Indeed, nodes try to preserve their resources, and particulary their batteries. To take this constraint in charge many power-aware routing protocols have been proposed (as it will be shown in the next chapter), but all these solutions do not eliminate the problem due to the complex nature of the network. Moreover, they rely on the nodes cooperation and well-behaving, and thus are as vulnerable as tradition routing protocols. As a result, users that are permanently anxious about their limited batteries might behave *selfishly*. A selfish node regarding the packet forward-

ing process is the one that takes advantage of the distributed forwarding service and asks others to forward its own packets, but would not correctly participate in this service. This misbehavior represents a potential danger that threatens the quality of service, as well as one of the most important network security requirements, namely the availability.

In this chapter we survey the problem of selfishness on packet forwarding in MANET, and we sketch the solutions currently proposed to mitigate this problem, focusing on the features, the advantages, and the drawbacks of each one. We also classify them according to their principles and features. This chapter is organized as follows: The next motivating section illustrates the causes and the effects of selfishness in MANET. In the third section, the current reactive solutions are presented. First, four basic monitoring solutions are reviewed, respectively: End-to-end ACKs, Watchdog and Pathrater, Activity-based Overhearing, and Probing. Followed by four reputation-based solutions: Signed Token, CORE, CONFIDANT, and Friends and Foes. Section 4 is devoted to preventive solutions, where we present: two economic based approaches (Nuglets and SPRITE), data dispersal, and the game theory based approach. Finally, section 5 concludes this state-of-the-art.

## 3.2 Selfishness: causes and effects

Most of one node's energy is likely to be devoted to forward packets for others. The simulation study of [BH03] shows that when the average number of hops from a source to a destination is around 5, then almost 80% of the transmission energy will be devoted to packet forwarding. This makes each node unwilling to forward packets not of direct interest to it, and motivates it to behave selfishly. Selfish behavior may lead to serious problems when performed by many nodes in the network, such as throughput degradation, latency increase, and network partition that threatens the availability of services, one of the security requirements.

Marti et al. [MGLB00] show by simulation that if 10% to 40% among the nodes misbehave on data forwarding, then the average throughput degrades by 16% to 32%. Further, Buttyan and Hubaux [BH01] show that large networks are more affected regarding throughput degradation. Another simulation study [MM02a] has been completely devoted to analyze the effects of selfishness on the network performance, in terms of throughput and delay. In this study, node selfish misbehavior is divided into three types: i) the first selfishness consists in dropping data packets but not control packets, ii) the second consists in dropping both kinds of packets, iii) whereas in the third one, two energy thresholds T1 and T2 (T1>T2), are used. In this type, the node behaves cooperatively until its energy state decreases and reaches T1, then when it is between T1 and T2 the node follows the first type of selfishness, and finally it follows the second type when its energy state fails under T2. The main

results of the simulations based on the first two types are:

- The network performance decreases with the increase of selfish nodes rate
- The performance degradation caused by the selfishness of type 1 is more important than that caused by the selfishness of type 2. The reason is that the first type causes the inclusion of selfish nodes in routes, and because these nodes will not forward data packets many of them will be dropped and lost when sent through these routes (including selfish nodes). This problem does not exist in the second selfishness type, where selfish nodes exclude themselves from participating in routing by dropping requests they receive.

The simulations based on the third selfishness type show an impressive result; the performance increases with the nodes mobility up-to a given speed (13 m/s). The authors argue this result as follows: When mobility is low, all nodes located in the central area of the network stay there and consume more energy than peripheral ones (nodes that have few neighbors), since central nodes are within much more routes than the peripherals. Hence, central nodes energy states reach T1 very quickly, resulting in a rapid selfish behavior triggering. On the contrary, when node mobility increases the location of a node changes from a central to a peripheral position and vice-versa with a high rate, such that the energy consumption will be equally distributed among nodes. Consequently, the selfish behavior is delayed, which improves the performance.

The previous studies show how the selfish behavior of nodes with respect to the packet forwarding threatens the MANET's system. Resolving this problem is therefore mandatory to ensure the well-functioning of the system. In the following, we present and discus the current solutions proposed in literature. We try to give general descriptions and classifications of these solutions, and to provide discussions focusing on their advantages and drawbacks.

## 3.3 Reactive solutions

Here we present and discus reactive solutions that aim at detecting selfish misbehavior on packet forwarding when it appears in the network. As we will see, the detection may be limited to the route including the selfish node, or may give deeper information and identify the selfish. Upon the detection of a selfish, routing through this node will be avoided. More stringent solutions suggest to punish these misbehaved nodes by excluding them from the service, and some of them allow the redemption and the reintegration of punished nodes.

We split reactive solutions up into two main classes, monitoring and reputation-based solutions. The monitoring class includes basic approaches that focus on the monitoring phase and suggest techniques to control the forwarding process, while

reputation-based solutions are more sophisticated and propose mechanisms to isolate the nodes detected as selfish. Still, these solutions (reputation-based) incorporate a monitor component that use some of the monitoring approach or other.

### 3.3.1 Monitoring based

We will present here four monitoring approaches, two of them are based on the promiscuous mode monitoring, while the others rely on the employment of acknowledgments (ACKs). As we will see, the advantage of the promiscuous monitoring compared with ACKs employment is that the first one requires no overhead for monitoring, and allow to monitor both directed and broadcast packets (packets sent to one neighbor and to all neighbors respectively). However, the promiscuous mode monitoring has many troubles regarding the accuracy on detections, especially when employing the power control technique as we will see later.

#### End-to-end ACKs

This mechanism consists of monitoring the reliability of routes by acknowledging packets in an end-to-end manner, to render the routing protocol reliable (like TCP). That is, the destination node acknowledges the successfully received packets by sending a feedback to the source. A successful reception implies that the corresponding route is operational, while a failure in the ACK reception after a timeout may be considered as an indication that the route is either broken, compromised, or includes selfish nodes. For each route the routing protocol maintains a rating reflecting the route reliability, which is updated each time a piece of data (a set of data packets) is transmitted across the route as follows: It is increased for each successful reception (when the source receives the ACK of that piece), and decreased for each failed piece (when a timeout expires without receiving an ACK). When the path rating of a given route decreases below a defined *threshold*, assumed to be high enough to overcome the losses due to collisions, this route will not be used anymore. Moreover, the routing protocol may rely on this rating as a metric and choose the most reliable routes.

The ACKs must be signed to ensure no-repudiation, otherwise a selfish node may misbehave by not forwarding packets and sending back a *falsified ACK* to the source without being detected. Note that it is beneficial to a selfish to perform like this, since an ACK sending costs much less than a piece of data packets. The signature of the ACKs requires an end-to-end security association between the source and the destination. i.e. the source must have the public key of the destination, which requires a public key distribution mechanism to be ensured (section 2.4.2).

The major problem of this technique is the lack of the misbehaving node detection. This technique may detect routes containing misbehaving or malicious nodes,

and those which are broken, but without any further information regarding the node causing the loss of packets. However, this technique helps to avoid sending packets through unreliable routes, and it can be combined with other more sophisticated techniques. It is used in SMTP [PH03] along with another technique, namely data dispersal which will be presented later, as well as in [AHNRR02] combined with probing which will also be illustrated after. Note that this mechanism is also used in [CGM05], where the authors propose a *cross-layer* mechanism that exploits TCP ACKs instead of adding explicit ACKs at the network layer, which reduces the overhead. This mechanism, however, is not combined with any other detective technique in this solution, since this later aims only at avoiding unreliable routes.

### Watchdog and pathrater

As far as we know, Marti et al. are the first who treated the problem of nodes misbehavior in MANET. In [MGLB00], they define two techniques which they call *watchdog* and *pathrater*. The first one is a monitoring technique that identifies misbehaving nodes, whereas the second helps routing protocols to avoid transmitting packets through these nodes. These techniques are used along with DSR [DD96] to built a misbehavior mitigating routing protocol.

**Watchdog:** It is a basic technique on which many further solutions rely. It aims to detect misbehaving nodes that do not forward packets, by monitoring neighbors in the promiscuous mode. We explain the watchdog's principle through the following example. Suppose node S sends packets to D using a route including (possibly amongst others) respectively three intermediate nodes: A, B, and C. When A transmits a packet to B to forward to C, A can check whether B forwards each packet by analyzing packets it overhears during a given timeout. If A overhears a packet it is monitoring during the fixed timeout then it validates its forwarding, otherwise it raises a rating regarding B, and will judge that B is misbehaving and notify S as soon as the rate exceeds a given threshold. This monitoring is generalized for each pair of hops in the source route.

The watchdog is able to detect misbehaving nodes in many cases, and requires no overhead as long as nodes well-behave. It allows to monitor all packets regardless whether they are directed or broadcast. Further, the watchdog technique can be considered as a passive acknowledgment in each hop when implemented over a MAC protocol that does not uses explicit ACKs. Nonetheless, the watchdog fails to detect the misbehavior in cases of collisions, partial collusion, and power control employment as illustrated hereafter. After a collision at C, B can circumvent retransmitting the packet without being detected by A. B can also circumvent the watchdog by partially dropping packets, viz. at low rate than the configured accusa-

tion threshold. The watchdog fails when two successive nodes collude to conceal the misbehavior of each other. That is, B can collude with C, by not reporting to A when C misbehaves. Further, the watchdog technique may cause false detections when the configured threshold fails [1], and especially when the monitored node uses the power control technique to preserve its power. This technique, largely used by the current power-aware routing protocol presented later, consists in using adaptive transmission powers according to the distance separating the transmitter and the receiver, instead of using a fixed full-power. In our example, when C is closer to B than A and B employs the power control, A cannot overhear forwarding of B and may accuse it wrongly.

**Pathrater:** The pathrater combines knowledge of misbehaving nodes with link reliability, to pick the route that is most likely to be reliable. For every other node it knows about in the network, each node maintains a rating that reflects its movement stability and routing activity. More the node is active and *successfully* forwards packets, more its rating increases [2]. Each node computes a path metric by averaging the ratings along the path, such that when there are multiple paths to the same destination the path with the highest metric would be chosen. Note that this differs from the standard DSR which chooses the shortest path in the route cache. The watchdog informs the pathrater about any misbehavior detection, to exclude routes containing any node known as misbehaving. We point out that since the pathrater depends on knowing the exact path a packet has to traverse, it must be implemented with a source routing protocol.

A serious problem with this solution (watchdog and pathrater) is that it does not punish the detected misbehaving nodes. Upon the detection of a misbehavior, the detector informs the source node, thereby the rating regarding the misbehaving is updated. Despite this rating update ensures that transmissions through the misbehaving node is avoided, no measure is taken against this node. We conclude that this technique by itself does not prevent nodes from misbehaving.

### Activity-Based Overhearing (ABO)

In [KKWS04] the authors propose the termed *Activity-Based Overhearing*, which is a generalization of the watchdog. In this technique, a node constantly monitors in the promiscuous mode the traffic activity of all its neighbors, and oversees the forwarding of each packet whose next forwarder is also in its neighborhood. This can increase the number of observations and improve the watchdog efficiency. It also mitigates the collusion problem, as illustrated in the following example:

[1] the number of packets lost due to mobility and channel condition exceeds the configured threshold

[2] The full algorithm of rating assignment is available in [MGLB00]

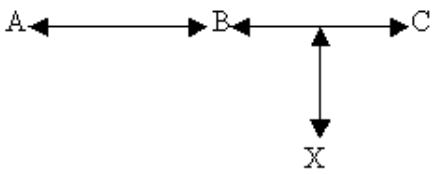


Figure 3.1: Example of network topology

Consider again the previous example of three aligned nodes A, B, and C such that A monitors B's forwarding toward C. As we have seen previously, B could collude with C and do not report its droppings, the watchdog fails behind this collusive misbehavior. Assume another node X in the neighborhood of both B and C, figure 3.1. In this case, when employing ABO, X could overhears B's forwarding and detect afterward C's dropping. Nonetheless, this general technique suffers from all the other problems of the watchdog, especially the one related to the power control technique as it relies on the promiscuous mode monitoring.

### Probing

We have seen that the end-to-end ACK approach allows to monitor routes and to detect unreliable ones containing misbehaving or failed nodes, but fails to detect the appropriate nodes responsible of the unreliability. The other monitoring solutions we have seen, however, directly monitor nodes. The probing approach could be viewed as a combination of route and node monitoring. This approach consists in simply incorporate commands into data packets to acknowledge them. These commands are called probes and intended for selected nodes. Probes are launched when a route that contains a selfish or a malicious node is detected (but not the ID of that node). Awerbuch et al. are the first ones who used this mechanism. The protocol they propose in [AHNRR02] is based on the end-to-end feedbacks to monitor routes, thus requires the destination to return an ACK (acknowledgment) to the source for every successfully received data packet. The source keeps track of the number of recent losses (ACKs not received over a window of recent packets). If the number of recent losses violates the acceptable threshold, the protocol registers a fault between the source and the destination and starts a dichotomic search on the path, in order to identify the faulty link. The end-to-end ACK employed could be considered as the route monitoring phase, and the dichotomic search as the node monitoring on suspicious routes. The source controls the search by specifying a list of intermediate nodes on the future data packets. Identifiers of these nodes are onion-encrypted [SGR97] (see 2.3.2). Each node in the list, in addition to the destination, must send an ACK for the packet. These nodes are called probed nodes. The list of probes defines a set

of non-overlapping intervals that cover the whole path, where each interval covers the sub-path between the two consecutive probes that form its endpoints. When a failure is detected on an interval, the interval is divided into two by inserting a new probe. This new probe is added to the list of probes appended to future data packets. The process of sub-division continues until a fault is detected on an interval that corresponds to a single link, as shown in figure 3.2. In this example node $I_2$ is assumed a selfish node that drops all packets, including those containing probing, and it is also supposed not replying to probing commands. As illustrated, $I_2$ will be detected after 4 probes (when the previous assumptions are held).

This solution suffers from many drawbacks. In addition to the high cost of onion-encryption and the communication overhead, there is no reliable detection of the dropper. A selfish node could analyze each packet it receives before deciding either to forward this packet or not. This way, when it gets a probe packet it would notice that a probing is under way, and would consequently choose to cooperate and forward packets for a limited time until the probe is over. In the context of the previous example, node $I_2$ could forward packets including probing and reply to the probing sent to it. This way no failure probing will take place [3], thus this probing will not be able to detect the selfish node in this case.

In [KKWS04], the authors propose an enhanced probing approach called *iterative probing*, which differs from the previous solution in the fact that each command is addressed to one node instead of a set of nodes. Therefore, the command contains one encrypted node ID added to a special field in data packets. If a data packet includes no probing command then the field will contain a random number, such that a recipient cannot distinguish data packets including probing from regular data packets, unless it is the destination of the probing command.

The solution suffers from the problem of important overhead, since it uses both end-to-end ACK and probing commands (when a misbehaving appears), both generate packets that travels multi-hop. This solution is also unreliable. It allows to detect the link containing the selfish node but cannot distinguish which of the two nodes forming the link is actually the misbehaving one, since there is no knowledge of the selfish node behavior upon the reception of a probing (either it sends back the ACK or not). To mitigate this problem Kargl et al. proposed the *unambiguous probing*. The principle of this mechanism is simple and can be summarized as follows: Assume after an iterative probing a link $(X_i, X_{i+1})$ will be detected. To determine which one of the two suspicious nodes is the guilty (the selfish), the source node asks the node $X_{i-1}$ to check if it can *overhear* the forwarding of $X_i$. If so then $X_{i+1}$ is the guilty, otherwise the guilty is $X_i$. This mechanism (unambiguous probing) suffers from the watchdog's problems, as it relies on the promiscuous monitoring at the predecessor of

[3] No faulty interval will be detected

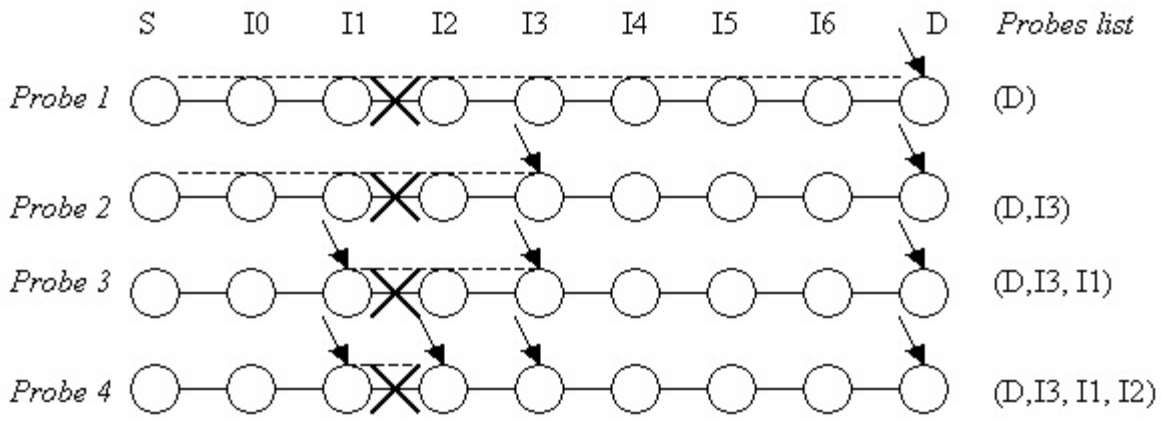


Figure 3.2: Example of the basic probing

the suspicious link. Note that probing was proposed and is applicable only to directed packets.

### 3.3.2 Reputation-based

The reputation of a given community member can be defined as the amount of trust granted by the other members regarding its well-behavior on a given function, according to their experience with it. Members that helpfully contribute to the community life get good reputation among community's members, while others who refuse to cooperate are badly reputed and gradually excluded from the community. In our context, the reputation of a node is the trustworthiness the other ones grant to it regarding its cooperation and participation in forwarding packets. Our definition is large such that including both solutions that asses nodes' reputation by real values or boolean values (well-behaving vs. misbehaving), provided that they punish bad reputed nodes.

Reactive reputation-based solutions are more elaborate than the previous monitoring solutions, and deal with the post-detection issues. Still, to detect selfish nodes they simply incorporate approaches proposed by those basic monitoring solutions. Each node keeps track of each other's reputation according to the behavior it observes, and the reputation information may be exchanged between nodes to help each other inferring the accurate values. There is a trade-off between efficiency in using available information and robustness against misinformation. If ratings made by others are naively considered, the reputation system can be vulnerable to false accusations or false praise. However, if only one's own experience is considered, the potential of learning from experiences made by others goes unused, which decreases the efficiency. In the following, we present four solutions belong to this general class

of reputation.

### Signed token

In [YML02], the authors describe a unified network layer solution, based on the approach of mutually according admission in neighborhood using signed tokens. It aims at protecting both the routing and the data forwarding. Threshold cryptography-based signature [Sha79] and the watchdog technique [MGLB00] are at the core of this solution. The solution is structured around four closely interacted components: i) *neighbor verification* that describes how to verify whether each node in the network is well-behaving or selfish, ii) *security enhanced routing protocol* which enhances AODV [CM02] and extends to the termed AODV-S that explicitly incorporates the security information in routing, iii) *neighbor monitoring* that is based on the watchdog to describe how to monitor the behavior of each node in the network, and how to detect packet droppers, iv) and finally v)*the intrusion reaction* which describes how to alert the network and isolate the misbehaving, and serves as a bridge between neighbor verification and neighbor monitoring.

Nodes in a neighborhood mutually accord participation admissions, and nodes without *up-to-date* admissions are excluded from any network service. Each node has a token issued by its local neighbors allowing it to participate in the network operations, which implements the concept of participation admission. The token has a period of expiration, whose value depends on how long the holder node has been behaving well (its *reputation*). This latter renews (updates) the token before its expiration. Nodes in a neighborhood collaboratively monitor each other to detect any misbehavior.

The solution employs asymmetric cryptography. There is a global key pair SK/PK (Secret Key and Public Key), such as each token carried by a node is signed with SK and broadcast periodically in the hello message to ask for a new validation. Note that the solution uses a hello protocol [4]. PK is known by all nodes, but none has the SK. Indeed, each node has a partial key which is as a part of SK and participates by providing a *partial signature of order K*, thereby $K$ different partial signatures are sufficient to provide the right signature (the principle of threshold cryptography presented in 2.2.1). To decide whether to provide a partial signed token for the requestor or not, the requestor's historical behavior is considered, which is drawn according to information collected using the promiscuous monitoring, and detections of neighbors as well. Once a node is detected as selfish, the detector informs its neighbors, and

[4]The hello protocol consists of periodic broadcast of a special packet called *hello* or *beacon* to inform nodes in a neighborhood about the presence of the broadcaster, which allows each node to be aware about its neighbors.

the selfish node is isolated as soon as $K$ different nodes detect it. Isolating a node in a neighborhood is achieved by not providing it with tokens.

**Discussion:** Although the authors do not evoke the notion of reputation, we categorize this solution in the reputation-based class since each node is accorded and denied services in its neighborhood according to its past behavior. The reputation value of each node could be simply considered boolean, i.e. well-behaving or selfish. Well-behaving nodes will be served and accorded tokens, while misbehaving ones will be isolated.

Since the detected misbehaving are isolated and excluded from any network service, the lack of a punishment mechanism against detected misbehaving nodes problem of the previous basic solutions is resolved. However, this solution has many disadvantages. First, all the watchdog's problems described previously remain untreated, since the neighbor monitoring component completely relies on it. The second disadvantage of this solution is that it prevents a node which has less than $K$ neighbors from communicating, and poses a critical issue on the choice of the parameter (threshold) $K$ for the sharing of the secret key. The choice of low $K$ weakens the key (It will be more breakable), whereas the choice of high values requires high connectivity which is not always ensured in MANET. Finally, we point out that the solution uses the notion of token expiration, and requires a timestamping mechanism which is problematic in distributed systems. This issue was not treated by the authors.

### CORE

Michiardi and Molva [MM02a] suggested a generic reputation-based mechanism termed CORE, supposed to be easily integrated with any network function. Unlike the previous solution this one gives more rigorous definitions to the notion of reputation, and defines three types of reputations: i) *subjective reputation* that is calculated directly from a node observations, and gives more relevance to the past observations in order to minimize the influence of sporadic misbehavior in recent observations, ii) *indirect reputation*, which is calculated basing on the information (observations) provided from other nodes, and iii) *functional reputation* that combines the subjective and indirect reputation. Each node maintains the three reputations for each other in a reputation table that is updated in two different situations; during the request phase of a given function, and during the reply phase corresponding to the result of the function execution. In the first phase, only subjective reputation related to misbehavior are updated (relying on negative information provided from the monitor component). Whereas, in the second phase only indirect reputations are updated *positively*. That is a reply message containing a list of all the entities that *correctly* behaved is supposed to be transmitted back to the source node at the end of the

function execution, so that the indirect reputations of these well-behaving nodes are *increased*. CORE is implemented with DSR, and uses the watchdog for monitoring and collecting direct observations, thus both directed and broadcasted packets could be monitored. It can be applied to packet forwarding function, both on data and route request packets. For the route discovery function, the aim is to detect misbehaving nodes that do not participate in this function and do not forward route request packets. During the request phase of the route discovery, the negative rating factor of the next provider may be observed by the requestor's watchdog, like in [MGLB00], while the identity of the nodes that participate in the function are reported to the initiator during the reply phase. The routing service will be denied to route requests issued from nodes classified as misbehaving, i.e. nodes whose functional reputation values become negative ($< 0$).

Similarly, the CORE scheme can be used to monitor the data packet forwarding function during the first step (negative rating observation). But as opposed to the route discovery function, data packet forwarding function does not include separate operations that can be qualified as request and reply phases, which harden the indirect reputation updates. However, the authors propose to add end-to-end ACKs, the transfer of which can be considered as the reply phase.

**Discussion:** The signed token mechanism [YML02] problem of preventing nodes with less than $K$ neighbors to communicate described previously does not exist in this solution. Also, in contrast to the previous solution observations are propagated beyond neighborhoods. However, only the positive observations (of well-behaving) are so propagated but not the negative ones. The purpose is to provide robustness for the solution and prevent the vulnerability of rumors propagation which can cause DoS (Denial of Service) attacks. We think this reduces the potential of learning from observations made by others, and can decrease the efficiency of misbehavior detections in the network. Contrary to the previous solution where the isolation is performed collectively by all nodes in neighborhoods, the isolation in CORE is performed *unilaterally* by each node basing merely on its own view of the behavior of the others. This could represent a potential threat of possible false accusations, as when an isolator does not forward packets for another node *unilaterally* isolated, other neighboring nodes (that are not isolating the appropriate node) would consider this as illegal behavior. Further, the solution does not allow redemption after detection, as when a node is excluded by another node it will not be asked to execute the service for this detector and will never be able to redeem and increase its reputation with it. If the nodes exchange their own experiences with each other (their views of reputations and not only observations), such a *redemption* would be possible. Moreover, note that all the watchdog's drawbacks related to detections are present with this solution, since the solution relies on the watchdog mechanism for monitoring.

**CONFIDANT**

It is another reputation-based solution, proposed in [BLB02]. It consists of four components present in each node. The first one is the *monitor*, which is very similar to the watchdog [MGLB00]. It registers the deviations from the normal behavior and calls the reputation system as soon as a given misbehavior occurs. The *trust manager* is the second component, which deals with the incoming and the outgoing ALARM messages. ALARM messages are sent by the trust manager of a node to warn others of misbehaving nodes, i.e. the protocol is based on negative information propagation. Outgoing ALARMS are generated by the node itself according to its experience observations, or after a misbehavior report reception. The recipients of these ALARM messages are so-called *friends*, which are considered to be configured on a user-to-user basis way. Incoming alarms, originate from either outside friends or other nodes, are checked for trustworthiness before triggering a reaction. The trust manager uses a filtering of incoming ALARM messages according to the *trust level* of the reporting node. To define trust levels, first a *general* mechanism similar to the trust management used in PGP (Pretty Good Privacy) [Zim95] for key validation and certification has been proposed. In their recent work [BLB04], the authors propose a modified Bayesian [O.B85, Dav00] mechanism that gives less importance to past observations than recent ones [5], and allows redemption. The third component of CONFIDANT is the *Reputation System* that manages the node's view on reputations of the others. Each node reputation is represented by a rating that changes according to a rate function, assigning different weights to the types of behavior detection, i.e. the greatest weight for own experience, a smaller weight for observations in the neighborhood, and the smallest one to reported experience. The rationale for this weighting scheme is that nodes trust their own experiences and observations more than those of other nodes. Once the rating of a node exceeds a configured threshold, the *path manager* is called for action. This latter is the last component, it is responsible for punishing the misbehaving nodes by not relaying any packet for them, as well as deleting paths containing misbehaving nodes and path re-ranking according to nodes trustworthiness.

**Discussion:** Unlike the previous reputation-based solution (CORE), with CONFIDANT reliable negative information are propagated beyond the neighborhood. To mitigate the vulnerability to DoS (Denial of Service) attacks by propagating rumors, the trust manager is proposed along with the rate function that assigns different weights to the types of behavior detections, in such a way to give more importance to local observations when computing the reputation rating. Moreover, The path manager component clarifies punishments against detected misbehaving. The simulation results [BLB02] performed by the authors using GloMoSim [ZBG98] show a significant improvement in term of goodput compared to the standard DSR (with which CONFI-

[5] Contrary to CORE that gives *more* importance to past observations

DANT has been implemented). Nevertheless, like the previous solution the isolation is performed independently by the path manager of each node. Recall that this could represent a potential threat of possible false accusations, as when an isolator does not forward packets for another node unilaterally isolated, other neighboring nodes would observe that and consider it as illegal behavior when they are not isolating the appropriate requestor. Also, all the watchdog's drawbacks presented previously remain untreated in this solution, since the monitor component fully relies on this technique. Finally, note that in the recent solution employing the Bayesian approach, the authors guest that the views of nodes' reputation are periodically exchanged with each other, which causes an important overhead.

### Friends and Foes

Contrary to the previous solution (CONFIDANT), friends and foes [MR03] gives as much importance to the past observations as to the present ones. Thus, it uses a long lived memory. In this solution, nodes are permitted to publicly claim that they are unwilling to forward packets to some nodes, as each node maintains basically three sets: a set of *friends* to which it is willing to provide services, a set of *foes* to which it is unwilling to provide services, and finally a set of nodes known to act as if it is their foe (they do not provide services for it) named set of *selfish*. These three sets are *periodically* broadcast in the neighborhood. Each node also maintains other variables for its neighbors, especially its view of their friends and foes, which are updated according to its experience and to the messages it receives periodically from its neighbors. When a node is asked to forward a packet it does so only when the asker is a friend, and count accordingly a credit for this friend. Also, every node chooses routes such that the next forwarder is its friend, then monitors the forwarding using the watchdog technique. It deletes a credit for the monitored node if this latter is perceived to correctly forward the packet, and puts it in the selfish set as soon as the number of packets it drops exceeds a given threshold.

The solution allows redemption and permits to a selfish node to be reintegrated by broadcasting a special packet (called SelfState) acknowledging that it has behaved selfishly with the appropriate nodes. To prevent abusing this mechanism, the selfish is first charged with penalties; it must broadcast two SelfState packets to consume additional energy, and the maximum value of its credit (the maximum number of packets it can send without providing forwarding services) is decreased by all neighbors. In addition to data packets, the authors propose the use of this solution to secure DSR control packets against selfish dropping.

**Discussion:** This solution defines a robust method of redemption that allows selfish nodes reintegration, while preventing these latter from abusing the proposed tolerance. Nonetheless, it suffers from some problems that we illustrated hereafter.

First, this solution has all the watchdog problems on which it relies for monitoring. The second problem is related to the overhead. The authors argued that the solution does not cause important overhead because control packets of each node are merely sent in its neighborhood. However, these packets are broadcast periodically which could be significant in networks with high connectivity. Further, since each node only keeps information about its current neighbors and the information of nodes leaving its neighborhood are arisen, a mobile selfish can easily circumvent and would never be detected. Finally, note that the solution is integrated with DSR, and is used to secure DSR's control packets from dropping. However, a basic principle of the solution is that each forwarder chooses the next one among its friends. Therefore, routing is made hop-by-hop and the solution is not applicable to a source routing protocol as DSR. Indeed, any reactive hop-by-hop routing protocol could be integrated with this solution, such as AODV [CM02].

## 3.4 Preventive techniques

Thus far, we have presented reactive solutions that aim at detecting selfish misbehavior on packet forwarding when it appears in the network. Another class of solutions includes approaches that *proactively* try to mitigate the misbehavior or its effects, either by motivating nodes to cooperate or by taking measures to prevent packets from being dropped before sending them. This is helpful to reduce the problem, but does not eliminate it permanently. Hence, a reactive solution which detects such a misbehavior remains essential. In this section we present three approaches we classify as preventive: i) economic-based solutions, ii) data dispersal, and iii) game theory based solutions.

### 3.4.1 Economic-based

In the following we present two economic-based solutions, inspired by some economic principles which they project on to the packet forwarding in MANET.

#### Nuglets

Buttyan and Hubaux propose an economic-based approach [BH01] stimulating nodes to cooperate for packet forwarding in MANET, which they model and analyze in [BH03]. They introduce what they call *virtual currency* or *nuglets*, along with mechanisms for charging the service usage. The basic idea of this technique is that nodes which utilize a service must pay for it (in nuglets) to the provider ones. This makes nuglets essential for utilizing the network, and renders each node interested in increasing its stock of nuglets by providing services for other nodes. Besides stimulating for the provision of services, this mechanism can also force nodes to make a moderate usage of the network services, since they become charged. Nuglets are represented

by counters at nodes, each one's value corresponds to the wealth of the holder. In order to prevent a node from illegitimately increasing its own counter, this latter is maintained by a trusted and tamper-resistant hardware module, termed *security module*. Only this module can directly perform operations on the counter. Nuglets loaded in a packet are protected from illegitimate modification and detachment from its original packet by cryptography mechanisms. The physical and data link layers (where the security module is built) are assumed to be robustly protected, such that users cannot modify them. Further, the neighborhood of a node is assumed not to change very fast, so as to make it feasible for a node to keep track of its neighbors by running a hello protocol. Besides discovering its neighbors, the security module uses the hello protocol (like the signed token described before) to establish and maintain security associations with the security modules of the neighboring nodes.

As for packet forwarding charging, the authors suggest three models: *packet pursue model (PPM)*, *packet trade model (PTM)*, and a *hybrid* one. In the first model the source is charged. It estimates the required nuglets on each hop and put the total number estimated of nuglets in the packet, then each forwarder acquires the required nuglets from the packet. The required nuglets charged by a forwarder may depend on many things, such as the amount of energy used for the forwarding operation, the current battery status of the forwarder, and its current nuglets number. If a packet has not enough nuglets to be forwarded then it is *discarded*. The advantage of this model is that it may deter nodes from sending useless data and overloading the network. However, the drawback is that it is difficult to estimate the total nuglets number that are required to reach a given destination. If the source under-estimates this number then the packet will be discarded and the source loses its investment in this packet, whereas an overestimation causes a wasting of the precious nuglets. On the other hand, in the PTM approach the packet does not carry nuglets, but it is traded for nuglets by intermediate nodes on each hop. Each intermediary *buys* it from the previous one for some nuglets (except the first intermediary that receives the packet for free from the source), and sells it to the next one (or to the destination) for more nuglets. This way, each intermediary that provides a service by forwarding the packet increases its number of nuglets, and the total cost of forwarding the packet is covered by the final destination. In contrast to the previous model, in this one the source has not to know in advance the number of nuglets required to deliver a packet. Furthermore, letting the destination pay for the packet forwarding makes this approach applicable in the case of multicast packets. However, a serious disadvantage is that this approach does not deter nodes from overloading the network. Another disadvantage is of overhead, since a price negotiation is required on each hop for each packet.

The two models can be combined in the following way: the source loads the packet with some nuglets before sending it, the packet is handled according to the PPM until

it runs out of nuglets, then it is handled according to the PTM until the destination buys it. This hybrid model gets over the packet loss problem of PPM.

**Discussion:** Nuglet is a new economic-based approach that motivates and obliges nodes to cooperate and forward packets for each other, because when a node behaves selfishly it will be unable to send its own packets. Moreover, this solution allows the nodes redemption, since a node which is unable to send its own packets because it runs out of nuglets is not excluded from being asked to participate in the data forwarding service and earning nuglets. But this approach suffers from some disadvantages; If a well-behaved node is not asked to route enough packets then it cannot send enough packets, and will be unfairly excluded. A node may be excluded from the routing process because of its position (it has few neighbors and belongs to just few routes) or because of the communication patterns of its neighbors (they have no communications with nodes to which it has routes). Furthermore, this technique does not prevent a node with enough nuglets from misbehaving, especially if it has not enough packets to send. Another issue related to this technique is that its robustness totaly relies on the famous assumed tamper-resistent hardware, but no detail on such a hardware was provided.

### SPRITE

Zhong et al. [ZCY03] propose another economic-based solution termed SPRITE, in which each node has a *virtual credit* maintained and continuously updated by a central authority called Credit Clearance Service (CCS). The principle is simple; when a node sends its own messages (as a source) it loses credits, and gains credits when it forwards messages for other nodes. To implement this, each forwarder is assumed reporting to the CCS for each message it forwards a *receipt*, a signed small message derived from the original one. This reporting is assumed to be performed whenever the node switches to a fast connection with a backup power. When the CCS gets reports related to a receipt, it charges the source of the message and compensates the intermediate nodes. The credit that an intermediary receives depends on whether its forwarding has been successful, and wether the message has reached its final destination. A forwarding is considered successful if the next node on the route [6] reports a valid receipt. Signing receipts prevents nodes from forging them, so none can report a receipt without really receiving a message. However, as soon as a node receives a message, it can easily reports the receipt without forwarding the message. The compensation strategy takes this problem into account, and prevents reporters that provide receipts of messages which does not reached the finale destination (messages not reported by the destination) from earning credits. The authors provide a

[6]Note that SPRITE is implemented with a source routing protocol (DSR), and the receipt contains the source route of the appropriate message

modeling and a formal proof of the solution, which shows that the solution is cheat-proof (under a set of conditions). That is, truth telling (reporting receipt only when forwarding a message, and not denying any forwarding) is the optimal strategy for every node. The proof also illustrates that the solution is collusion-resistant. Further, the solution was extended with little modifications to broadcast control packets (like route request of the routing protocol), for which the CCS computes a tree based on receipts it receives before updating credits. This way, redundancy is avoided.

**Discussion:** Like Nuglets, SPRITE is an economic-based strategy that motivates nodes to collaborate. However, the major advantage of SPRITE is that it does not require any tamper-resistant hardware. Also, virtual money in this solution are considered as credits and are not held in packets, contrary to Nuglets. Consequently, the strategy of charging the source is efficient for SPRITE, since the problem of packet dropping due to virtual money lack presents with the packet pursue model of Nuglets (see the previous subsection) does not exist here. Remember that the source-charging strategy has the advantage of preventing nodes from sending useless data that overload the network, and makes them rational when using the network services. Further, the proposed compensation strategy overcomes collusion (on falsely reporting receipt), providing that the destination well-behaves. Nevertheless, the elimination of the tamper-resistant dependency was ensured by using a *central* authority (CCS) that manages credits, which makes the solution centralized, and thus introduces another drawback. Distributing the CCS is mandatory for this solution to be applicable in MANET, basically featured by the total decentralization. Another disadvantage of this solution is that it assumes the cost of reporting a receipt to be negligible, and requires the reporting to be performed when the node switches to a fast connection and gets backup power, which is not always possible in MANET.

### 3.4.2 Data dispersal

This scheme is based on Rabin's algorithm [Rab89] and takes advantage of the existence of multiple routes from a source to a destination, to increase the reliability when transmitting packets. It consists in adding *redundancy* to the message to be sent, then the message and the redundancy are divided into a number of *pieces* and dispersed on the available routes, so that even a *partial* reception can lead to the successful reconstruction of the message at the receiver. Note that node-disjoint routes ensure more efficiency. This technique can overcome partial packets loss, that can occur due to misbehavior on some used routes.

This approach is based on a mathematical framework. To illustrate it let us assume that the message which has to be sent is a set of m streams, where a stream is a set of bits (eg: character or integer) which can be considered as a data unit. We also suppose that the source of the message has N different routes to the destination.

Let $A$ be a $N$ random M-vectors, i.e. $A$ is a matrix of $N$ lines and $M$ columns, such as each line can be viewed as a vector of M elements (streams) and each vector is constructed randomly (the elements are random numbers).

First, the message is divided into $L$ sequences, each of $M$ streams. If $M$ does not divide $m$ then extra redundancy is added to the message. The result can be modeled by the following matrix: $B = (S_1|S_2|.....|S_L)$, where each $S_i$ is a column vector of size $M$.
Consider the following matrix $W = A \times B$, such that $\times$ denotes the matrix product. W's length is $N$ lines and $L$ columns, hence each line vector is a *piece* that can be sent on a different route. $M$ pieces among the $N$ transmitted are necessary to reconstruct B at the reception. The reconstruction of $B$ is performed using the following formula:

$B = A'^{-1} \times W'$
such that $W'$ is a matrix formed from $M$ lines of $W$ ($M$ pieces among the well received ones), $A'$ is the corresponding matrix formed from $A$, and $A'^{-1}$ is the reverse matrix of $A'$. Note that lines used to form $W$ and the corresponding ones in $A'$ are not inevitably adjacent.

**Example:** To understand the principle of this approach we give the following illustrative example:
Assume the message to be sent (data) consists of 6 streams, i.e. $m = 6$. That is: $data = m_1m_2m_3m_4m_5m_6$.
Also assume $N = 4$ (the number of available routes), and let us set $M = 2$ (the number of pieces required to construct $data$). Thus $L = m/M = 3$, and

$$B = \begin{vmatrix} m_1 & m_3 & m_5 \\ m_2 & m_4 & m_6 \end{vmatrix}$$

Suppose:

$$A = \begin{vmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 0 \\ 1 & 1 \end{vmatrix}$$

Thus: $$W = A \times B = \begin{vmatrix} m_1 & m_3 & m_5 \\ m_1 + m_2 & m_3 + m_4 & m_5 + m_6 \\ m_1 & m_3 & m_5 \\ m_1 + m_2 & m_3 + m_4 & m_5 + m_6 \end{vmatrix}$$

Each line of this matrix consists of a piece to be transmitted on a different route, such that the reception of any two pieces among these four pieces at the destination enables the reconstruction of $B$, hence the message $data$.
e.g: If we use the first and the last pieces then:

$$A' = \begin{vmatrix} 1 & 0 \\ 1 & 1 \end{vmatrix}, A'^{-1} = \begin{vmatrix} 1 & 0 \\ -1 & 1 \end{vmatrix}, W' = \begin{vmatrix} m_1 & m_3 & m_5 \\ m_1 + m_2 & m_3 + m_4 & m_5 + m_6 \end{vmatrix}$$

Using these matrices $B$ will be computed as $A'^{-1} \times W'$. Note that the matrix $A$ must be shared by the source and the destination.

**Discussion:** The ratio $N/M$ or the *redundancy factor* is a crucial parameter for this solution. Increasing this ratio ensures more reliability, since few number of pieces among the overall sent pieces would be required to reconstruct $B$, but high values of this ratio cause important overhead. On the other hand, decreasing the redundancy factor reduces the overhead, but gives less reliability. The choice of this parameter is therefore a trade-off issue. It should strike a balance between reliability and overhead.

Even though this mechanism does not prevent nodes from misbehaving and does not motivate nodes to cooperate, unlike the previous ones, it is helpful to reduce the selfish misbehavior effects on the communication reliability, and can be combined with a reactive solution. In [PH03] the authors propose SMTP, a solution that uses this mechanism. However, this solution has the end-to-end feedback technique drawbacks presented previously, since it relies on it.

### 3.4.3 Game theory based

In this approach the forwarding process is viewed as a game, where nodes have to continually decide whether to forward or not to forward packets. The purpose of this approach consists of defining strategies to ensure fairness to all nodes. Since users may be selfish, there is no guarantee that they will follow a particular strategy unless they are convinced that they cannot do better by following some other strategy. In game theory terms a strategy which constitutes a *nash equilibrium* [Mye91] needs to be identified. Nash equilibrium can be defined as a strategy profile having the property that no player can benefit from *unilaterally* deviating from the strategy [SNFR03]. In other words, it is a feature which ensures that if a cheat player tries to deviate from the strategy whereas all the others follow it, the cheat cannot reach more benefits than the others.

Some solutions based on this approach have been proposed, such as [SNFR03, WL06, ZLLY05]. For instance, in [SNFR03] nodes are distributed among classes according to their energy constraints and their expectation of lifetime. The source node asks intermediate ones to relay packets before sending them, then each node has to decide whether to accept or reject forwarding packets for this source. If one node refuses to forward packets then it returns a negative ACK back to the source, and consequently the session is blocked. Otherwise, the request is forwarded until reaching the last router (destination's predecessor) which sends a positive ACK back to the source. A node that has relayed *much* more traffic than the amount that has

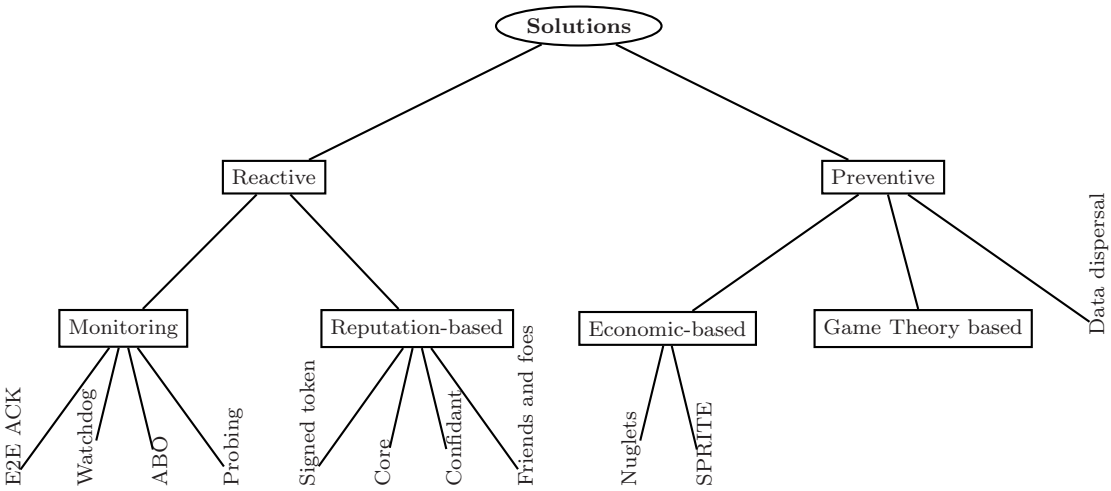


Figure 3.3: Classification of the presented solutions

been relied for it (according to a defined factor) refuses to participate in the session. A node that has relayed more traffic than a defined amount also rejects the participation. In other cases, the node agrees to forward packets.

It has been proved that the proposed algorithm leads to a nash equilibrium. That is, if all nodes accurately execute the algorithm then any *individual* deviation from a node will not allow it to reach a greater throughput than the so-called pareto optimal value, reached by all well-behaved nodes in the nash equilibrium. This solution, as well as all the ones belonging to this class, trust the ACKs of intermediate nodes. Indeed, a selfish node may agree to participate in a session and to forward packets, in order to give impression that it executes accurately the protocol, but actually would not forward packets when it receives them. The approach needs to be combined with a reactive monitoring solution for resolving this problem.

## 3.5 Conclusion

In this chapter we have presented and discussed the different current proposals which deal with the selfish misbehavior on packet forwarding in MANET, focusing on their features and drawbacks. We have first classified them into two main classes: reactive and preventive solutions. Reactive (or detective) solutions aim at *actively* detecting the misbehavior when it appears, while preventive ones try to either proactively prevent any misbehavior by motivating and forcing nodes to cooperate, or to take precautions to avoid packets from being lost before sending them. The history of networks security has taught us a valuable lesson, no matter how many preventive measures are inserted in a network, there are always some weaknesses in the systems that one could exploit to break in. Like IDSs (presented in chapter 2) used against

| Solutions | Features | Drawbacks |
|---|---|---|
| **End-to-end ACKs [PH03]** | - The destination sends back ACKs to the source<br>- Detects routes that includes misbehaving | - Does not detect the appropriate node<br>- Important overhead<br>- Lack of punishment |
| **Watchdog [MGLB00]** | - Promiscuous mode usage<br>- Detects misbehaving in many cases<br>- No overhead when there is no misbehaving | - Fails to detect the misbehaving in the following cases:<br>• after a collusion<br>• collusive misbehavior<br>• when the monitored control its transmission power<br>- Causes faults detection when using adaptable powers<br>- Lack of punishment |
| **ABO [KKWS04]** | - Generalization of the watchdog<br>- Provides more traffic to monitor<br>- Mitigates collusive misbehavior<br>- More efficiency when the watchdog is operational | Watchdog's drawbacks |
| **Dichotomic probing [AHNRR02]** | - Incorporates commands into data packets<br>- Requires end-to-end feedback employment<br>- Uses onion encryption | - Tamperable |
| **Iterative probing [KKWS04]** | - Incorporates commands into data packets<br>- Requires end-to-end feedback employment<br>- Uses asymmetric encryption<br>- Detects the link containing the misbehaving node | - Fails to detect the appropriate misbehavior |
| **Unambiguous probing [KKWS04]** | - Iterative probing + watchdog employment on the detected link upstream node | - watchdog's detection drawbacks |
| **Signed Token [YML02]** | - Nodes in neighborhood accord to each other participation admission<br>- Uses threshold cryptography<br>- Based on boolean reputation | - Unfairly excludes nodes with less than the predefined threshold (K) from the service<br>- Watchdog's detection drawbacks |
| **CORE [MM02b]** | - Based on continuous reputation<br>- Only positive impressions are propagated | - Unilateral isolation<br>- No propagation of misbehaving detection<br>- Watchdog's detection drawbacks |
| **CONFIDANT [BLB02]** | - Based on continuous reputation<br>- Built on negative impression | - Unilateral isolation<br>- Watchdog's detection drawbacks |
| **Friends and foes [MR03]** | - Each node splits up the others into 3 sets, and<br>- it periodically broadcast its view in its neighborhood<br>- Routing and forwarding decisions are based on these sets<br>- Defines a robust redemption technique | - Watchdog's detection problems<br>- Very important overhead<br>- Mobile misbehaving problem |
| **Nuglets [BH01]** | - Nodes that use the service pay nodes that offer it<br>- Motivates nodes to cooperate<br>- Forces nodes to make moderate usage of the service<br>- Allows node redemption | - Unfairly prevents nodes not asked to forward packets from sending their own<br>- Does not prevent nodes that do not need Nuglets to misbehave<br>- Requires a tamper-resistant hardware |
| **SPRITE [ZCY03]** | - Based on the credit principle (generalization of Nuglets)<br>- Motivates nodes to cooperate<br>- Does not request tamper-resistant hardware<br>- Overcomes collusion | - Centralized<br>- Needs a fast connection use for reporting |
| **Game theory based** [SNFR03, WL06, ZLLY05] | - Based on the Nash equilibrium strategy<br>- Motivate nodes to cooperate | - Totally trust intermediate nodes ACKs |
| **Data dispersal [PH03]** | - Based on Rabin's algorithm<br>- Requires multi-path routing<br>- Reduces the misbehaving affects and increases the reliability | - Does neither detect nor prevent misbehavior of nodes |

Table 3.1: Features and drawbacks of the current solutions

attacks and intrusions, reactive solutions are essential to beat the selfishness.

We have divided the reactive solutions into two subclasses, monitoring solutions and reputation-based solutions. Monitoring solutions consist of basic techniques for controlling packet forwarding. In this chapter four techniques belonging to this category have been illustrated: two among them are based on the promiscuous monitoring, namely the watchdog and the activity based overhearing (ABO), and two on acknowledgment; end-to-end ACK and probing. The advantage of the promiscuous monitoring solutions is that they require no overhead when nodes well-behave, and they allow to easily monitor both directed and broadcast packets. Further, the second one (ABO) mitigates the collusion problem as shown before. However, these solutions fail to detect selfish nodes and may cause wrong accusations in many cases, especially when employing the power control technique. The first technique using acknowledgment is end-to-end ACK, which consists in making the routing protocol reliable like TCP. Despite this solution causes an important overhead, and does not detect selfish nodes but only routes including such nodes, it helps routing packets around unreliable routes, and may be combined with a more sophisticated solution like SMTP [PH03], [AHNRR02] and [CGM05]. In this latter, the overhead issue was treated, and the TCP ACKs were exploited through a cross-layer approach to reduce it. Probing is the last monitoring technique, it uses the end-to-end ACK to monitor routes, and improves it by adding a dichotomic probing phase to detect the appropriate selfish nodes whenever a route becomes suspicious. As illustrated, the first solution based on this technique [AHNRR02] is just unreliable. Iterative probing [KKWS04] is more effective but allows to merely detect the link including the selfish node and has high overhead. Unambiguous probing [KKWS04] deals with the node detection issue, by suggesting to utilize the promiscuous monitoring at the predecessor of the suspicious link. This would have inevitably the watchdog's (promiscuous monitoring) problems. The major lack related to these monitoring solutions is the post-detection issues, i.e. punishment and selfish nodes knowledge (experience) exchange between nodes. Reputation solutions are more elaborate and particulary deal with these issues. All the reputation-based solutions involved in this chapter utilize the watchdog for the monitoring phase, thus inherit all its drawbacks. We realize that proposing a solution basing on a more reliable monitoring approach represents an open research topic.

In Signed token [YML02], the first reputation-based solution presented, as well as in the last one (friends and foes [MR03]) a node's reputation might be viewed as a boolean (selfish vs well-behaving). Contrary to the two others, that provide more rigorous definitions. The signed token uses threshold cryptography involving a parameter $K$, and nodes in each neighborhood cooperate to provide participation admission to each other. The major drawback of this solution is preventing nodes with less than $K$ neighbors to communicate. Consequently, the parameter $K$ is critical and poses a trade-off issue between operability and robustness. This problem does

not exist with CORE [MM02b]. In this latter observations are propagated beyond the neighborhood, but only the positive ones. Not propagating negative observations would prevent the vulnerability of propagating rumors aiming DoS attacks, but this way the experience of others gets unused. On contrast, CONFIDANT [BLB02] propagates negative observations beyond the neighborhood, while considering the rumors problem and taking measures to mitigate it at the trust manager component. Further, CONFIDANT with its modified Bayesian approach for reputation gives less and less importance to past observations, which allows redemption contrary to CORE that gives more importance to past observations. Nonetheless, in both CORE and CONFIDANT the isolation is performed unilaterally by each node, which might result in false accusation. As when a node isolates unilaterally another and denies forwarding packets for it (punish it), other neighbors would consider its behavior illegal. The problem is more serious with CORE, since in CONFIDANT nodes' experience are exchanged, and it allows redemption. Note that this problem does not exist with signed token. As for friends and foes [MR03], it gives as much importance to the past observations as to the present ones, but defines a robust method for redemption. However, this solution suffers from the important overhead it might cause, particularly in high connected networks, since it relies on periodic broadcast of control messages whose sizes are proportional to the number of neighbors. It also suffers from the mobile selfish problem, as each node only keeps information about its current neighbors.

Regarding preventive techniques, four solutions have been presented: two amongst them are based on economic approaches; Nuglets [BH01] and SPRITE [ZCY03]. The former assumes a tamper-resistent hardware that manage virtual money or the so-called nuglets, whereas the latter eliminates this assumption, but introduces a central authority (CCS) which is unappropriate for MANET. Distributing this CCS could represent a research trend. Data dispersal is another technique that mitigates packets loss. Even though it does neither prevent selfishness nor motivate cooperation, it could be helpful when combined with a reactive solution like in SMTP. The last preventive solution we presented is relying on the game theory approach. We noted that the major drawback of this solution (and of this subclass in general) is that it totally trusts nodes ACKs, hence it needs to be combined with a reactive monitoring technique. Finally, Figure 3.3 illustrates our classification, where table 3.1 sketches the main features of all the solutions involved in this manuscript chapter.

# Chapter 4

# Power-Aware Routing in MANET

## 4.1 Introduction

As shown previously, nodes in MANET are supplied with lightweight autonomous limited batteries. Although this autonomy allows a free mobility, it makes nodes concerned about their resources, and may motivate them to behave selfishly and deny forwarding packets for others (as illustrated in the previous chapter). Before attempting to solve the selfishness problem, we first deal with its causes and try to tackle the problem of power-awareness at the routing layer.

The energy consumed by the wireless radio is very important, especially for small devices. Earlier researches on minimizing the energy consumption of these devices was concentrated on the hardware level. However, an important gain may be obtained in the software and communication protocol level, especially for routing in ad hoc networks, where packets follow multi-hop routes. Recent studies [DB02a] show that power efficient routing protocols for ad hoc networks may be designed from existing traditional reactive protocols, presented in the first chapter, by adding new mechanisms and metrics instead of employing the min-hop full-power strategy.

First, battery-aware metrics have been proposed [SWR98]. This kind of metrics consists of considering the battery states of nodes along the route, thereby selecting routes basing on the node's battery freshness. Many routing protocols based on this strategy has been designed [MY05, CR00]. Another promising approach proposed to reduce the energy consumption is the transmission power control [DB02a], which consists in using adaptive transmission powers according to the distance separating the transmitter and the receiver, instead of using a fixed full-power. This mechanism has been largely used to define power-aware metrics and power-aware routing protocols [JV05, DBB02]. Nonetheless, no protocol consider the node battery state, contrary to the ones based on battery-aware metrics, that are battery state centric but do not consider the transmission power control mechanism employment. In this chapter we

deal with this issue, and propose new metrics that strike a balance between the two approaches. Basing on these metrics we propose a power-aware DSR-based routing protocol [DB04, DB06b], that aims at ensuring a long life to the batteries of all nodes. The rest of this chapter is organized as follows, after a brief representation of the two approaches, we will present our metrics, followed by our DSR-based routing protocol. Finally, we will asses the performance of our protocol through a simulation study.

## 4.2 Current Route Selection Strategies

### 4.2.1 Routes containing the freshest batteries

The aim of this strategy is to select routes that contain the freshest batteries. For this purpose, a cost representing the battery state is associated with each node, and the route that minimizes the total sum of these costs is considered as the most optimal. In [SWR98], a definition of the cost function proportional to the battery voltage has been proposed. The cost of node i is given by:
$Fi(zi) = 1/(1 - g(zi))$.
Where $zi$ denotes the measured voltage (that gives a good indication of the energy used thus far), and $g(zi)$ $(0 \leq g(zi) \leq 1)$ is the normalized power (in percentage) consumed for the voltage $zi$. For a given value of $zi$, $g(zi)$ can be obtained from the discharge curve that features the battery. An example of a discharge curve is illustrated in figure 4.1 [SWR98]. It represents the voltage vs the consumed power ratio (the power consumed / the initial capacity) of the Lithium-Ion battery, largely used in mobile devices. Saving this information (the curve) allows the node to obtain the consumed power ratio by the battery by checking the battery voltage. Many routing protocols based on this strategy has been proposed, such as [MY05, CR00]. All these protocols, however, rely on the employment of the fixed full-power to transmit packets.

### 4.2.2 Minimizing the total required power

In the standard ad hoc routing protocols packets are sent using the full power, such that the whole node's power range is covered. In fact, data packets are not propagated, but they are sent to a single node (since these protocols are uni-cast), often located at a distance lower than the power range. Instead of using the full power, it is more efficient to use a dynamic power according to the distance between the sender and the receiver. This principle is known as the transmission power control [DB02a, KML04]. Figure 4.2 [DB02b] shows the energy savings when using the power control technique (for one transmission) according to the distance between the sender and the receiver for a data packet of size 512 bytes. We clearly see how the technique is gainful, particularly when the two nodes are close to each other.

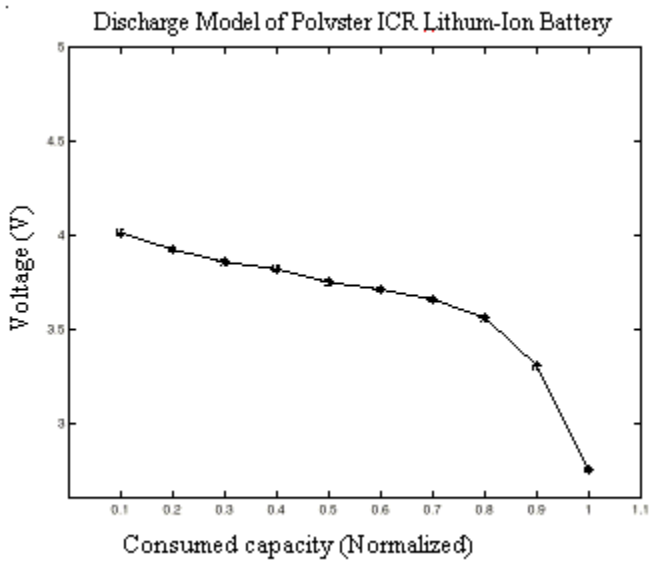


Figure 4.1: Polystor ICR_18650 lithium-ion battery discharge curve

The purpose of the routing protocols designed around this strategy is to minimize the total required power when sending each packet. When using the power control technique, a cost may be associated to each link as the power required on it. Thereby, the optimal route when using this strategy is the one that minimizes the sum of these costs, instead of the shortest one. As the required power on each hop does not correspond linearly to the distance ($d$), but it is proportional to $d^{\alpha}$, where $\alpha > 2$, min-power routes are not inevitably the shortest ones. i.e. long routes could be more optimal, with regard to this criterion, than shortest ones. The selection of long routes may affect a little bit another QoS (Quality of Service) parameter, namely the delay.

A variety of routing protocols using the power control have been proposed [DBB02, BM02, GCNB03, OH04, JV05]. They are all derived from well-known reactive protocols, and either minimize the total power required [DBB02, BM02, GCNB03], or strike a balance between the total power required and the shortness to take into account the latency [OH04, JV05]. But none considers the node's battery state.

### 4.2.3 Discussion

The major drawback of the battery-aware strategy is that it does not consider the link cost at all. When batteries states are close to each other, using this strategy results in the selection of the shortest routes, regardless of their costs, which causes

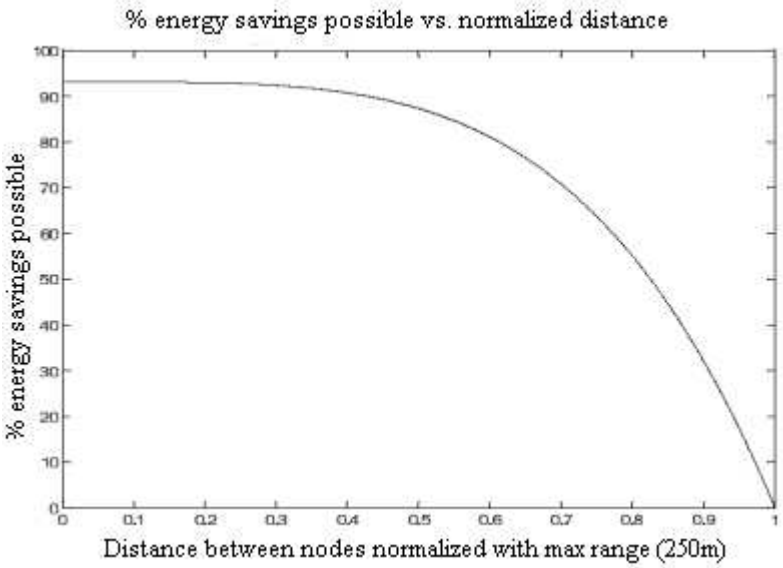


Figure 4.2: Energy gain vs. normalized distance (/250m)

important waste of energy (even when employing the power control technique, which was not used by the battery-aware routing protocols). This strategy minimizes the difference in energy consumption between nodes, but does not ensure long life to the batteries. In other words, it ensures that nodes remain alive together, but not for the longest period.

On the other side, the problem with the power control strategy is that it does not take into account the battery state, which might result in an overuse of a subset of nodes as shown in the following example:
We consider the stationary network represented in figure 4.3, where weights represent the powers required on the links. We assume there is a session between node $S$ and node $D0$, and an other between $S$ and $D1$. We assume that $P0 + P2 < P3 + P4$. When using this strategy, node *I0* will be used to route packets both for *D0* and *D1*, while node *I1* will not be used at all. This way, node *I0* may lose its battery capacity before the others, then a soon network partition can take place (node *D0* will not be reachable anymore). This partition would be avoided if node *I1* was used.

## 4.3 New power-aware balancing Strategy

### 4.3.1 Metrics

As illustrated in the previous section, each of the two power-aware strategies has a drawback. We think that both the link cost and the battery state should be taken into

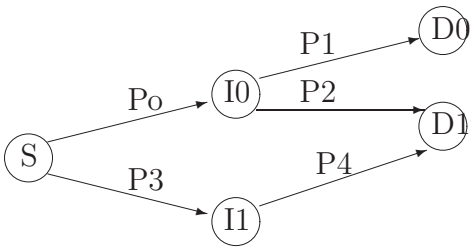


Figure 4.3: Stationary Network

account when selecting routes. Therefore, a trade-off between the battery freshness and the required power minimization should be overcome when selecting routes. For this purpose, we define new metrics on which the routing is based. First, we assume without loss of generality that the network is stationary, and we try to resolve the problem in a centralized manner at a given time t.
Let $(S, L, P, E)$ be a quadruple such as:
$S$: vertices set
$L$: arcs set, $L = \{(si, sj)/si \in S \text{ and } sj \in S\}$. Such that $(S, L)$ is the graph representing the network topology at the time t.
$P$: a function which associates to each element $(s_i, s_j)$ of $L$ a value in $\mathbb{R}$ (the real numbers set), called the *weight* of $(s_i, s_j)$
$E$: a function which associates to each element s from S a value in $\mathbb{R}$, called the *state* of s.
On a given route $C = I_0, I_1, \cdots, I_{n-1}$ we propose the following metric:

$$M_c = \sum_{i=0}^{n-2} \alpha * E(I_i) + \beta * P(I_i, I_{i+1}) \ldots\ldots (4.1)$$

The aim is to choose the route $C$ which minimizes $M_c$, where:
$\alpha$ is the states *rate*, $\beta$ is the links *rate* $(\alpha, \beta \in [0, 1])$, such that $\alpha + \beta = 1$.
The function $E$ should reflect the battery state, while the function $P$ should reflect transmission power costs over links.
Let $E$ be the following function:
$E(s_i) = 1/(1 - eng(s_i)) \ldots\ldots (4.2)$
Where $energ(si) \in [0, 1]$ is the rate of the energy consumed by node $s_i$. This function is monotonous in the interval $[0, 1]$, i.e. more the node's consumption rate increases, more its state value increases.
For $P$, we propose the following function:
$P(s_i, s_j) = 1/(1 - PowRate(s_i, s_j)) \ldots\ldots (4.3)$
Such that $PowRate(s_i, s_j) = \frac{Pow(s_i, s_j)}{MaxPow(s_i)}$

Where: $Pow(s_i, s_j) \in [0, 1]$ is the required power over the link $(s_i, s_j)$, $MaxPow(s_i)$ is the full power of the node $s_i$ (the power that allows it to cover its power range). Note that $P$ increases with $PowerRate$.

When using this metric ($M_c$), more $\alpha$ increases more the routes containing fresh batteries get favored, and more it decreases more routes that minimize the total transmission energy cost are favored. Hence, we propose the following strategy: More the difference between nodes' batteries increases, more $\alpha$ should be increased, and more nodes' batteries become closer, more it should be decreased. This way, batteries states are constantly well-balanced, while taking advantage of the power control technique and considering the total power when transmitting data packets. We define the difference between batteries states at the time t by:
$EngDiff = \max_S(eng(s_i)) - \min_S(eng(s_i))$

It is the difference between the maximum rate of the energy consumed by the nodes and the minimum one, note that $Engdiff \in [0, 1]$.
It remains to find an increasing function for $\alpha$: $\alpha(EngDiff) : [0, 1] \longrightarrow [0, 1]$.
We propose the following functions, where $\alpha 0$ is an initial value:
$F1 = (1 - \alpha 0) * EngDiff + \alpha 0$
$F2 = \sqrt{(1 - \alpha 0)^2 * EngDiff + \alpha 0}$
The two functions are increasing in the interval $[0, 1]$, but they differ from each other in the way of increasing. The latter is more increasing in this interval than the first one. What is the way of increasing that gives more efficiency ? We will investigate this question later in our simulation study.

Using (4.1), (4.2), and(4.3), we rewrite the metric $Optc$ representing the optimality of the route $C = I_0, I_1, \ldots, I_{n-1}$ connecting node $I_0$ to node $I_{n-1}$ as follows:

$$Opt_c = \sum_{i=0}^{n-2} \frac{\alpha}{1 - eng(I_i)} + \frac{1-\alpha}{1-Pow(I_i,I_{i+1})/MaxPow(I_i)} \ldots\ldots (4.4)$$

Where: $\alpha$ could be represented either by $F1$ or $F2$

The optimal path corresponds to the minimum value of $Opt_c$. At time t, $Pow(s_i, s_j)$ value is known for each link, the value of $eng(s_i)$ is known for each node, thus both $EngDiff$ and $\alpha$ can be computed. When using this metric, the optimal path connecting any pair of nodes at the instant t can be computed (using Djikstra algorithm for instance). In time, values of $eng(s_i)$ change, then the computation of parameters and routes must consequently be done again. Using this approach, node $I_0$ of the previous example will not be used all along the sessions between node $s$ and nodes $D_0$ and $D_1$. Because when $I_0$ starts consuming its battery capacity for routing packets, $E(I_0)$ increases, and the route $(s, I_0, D_1)$ becomes less optimal than $(s, I_1, D_1)$,

hence the last one will be used for routing packets to $D1$. This way, a soon network partition is avoided.

Thus far, we have proposed new metrics that have been explained using a centralized solution. It is obvious that in practice no node can have an accurate view of the network topology. Still, these metrics can be employed by any distributed routing protocol, as we will see later.

### 4.3.2 New data dispersal technique

Many reactive routing protocols, such as DSR [DD96], are multi-routes, i.e. after a route discovery many routes may be found. But in DSR, just one route (considered to be the optimal) among the available ones is chosen. All the other routes are considered only as alternatives, and they are not used until the optimal one fails.

We think it is beneficial to take advantage of the multi-routes existence, and disperse the data load over a maximum number of nodes. Moreover, route optimality should be considered, in such a way to transmit more packets on the most optimal routes.

Let *nbrpaths* be the number of routes available that relay a source node $s$ to a destination $d$. We assume that these routes are ordered according to their optimality (route 0 is the most optimal one with respect to our metric). We propose to transmit on each route i the following number of successive packets:

$$nbrpaquets_i = \ 2^{nbrpaths-i} \quad \ldots\ldots (4.5)$$

Thereby, node $s$ sends the first $2^{nbrpaths}$ packets on the first route (the most optimal route), and the following $2^{nbrpaths-1}$ (as much as half the previous number) on the second one, and so on till using the last route on which it sends two packets. After this round, the first route will be reused to send the next $2^{nbrpaths}$ packets (starting another round), and so on till all the packets are transmitted. This way all the routes will be used, and the number of packets sent on each route i is twice as much as the number of packets sent on the next route (i+1) in each round.

## 4.4 New DSR-based power-aware routing

The protocol we propose is derived from DSR, which is a distributed reactive protocol based on the source routing concept (as illustrated in the second chapter). In this section we first illustrate the main modifications we have added to DSR, then we will present our protocol in more details and discuss the operations we have added.

### 4.4.1 Main Modifications

We mainly add to DSR the following:

- Implementation of the power control technique: The required powers are computed on each link during the route discovery procedure, as in [DBB02]. On each link, the transmitter of a RREQ puts the transmission power in the RREQ's header, then the receiver retrieves this value, that it uses along with the reception power it gets from its radio and the appropriate environment propagation model to compute the required power.

- Consider energy state: for this purpose, we assume that each node is able to determine its consumed energy rate. This can be achieved by implementing a bottom level mechanism allowing nodes to determine their battery voltage. When getting its battery voltage, the node can deduce its energy consumed rate from the battery discharge curve [SWR98, Gol97]. Moreover, each node should have a view on energy states of the other nodes. To ensure this, we suggest to piggyback the energy state information into control packets, especially into reply packets and E_state packets that we present hereafter.

- Adding E_state packet: each node has a global view of the energy states of the others. To get a better view (especially about neighbors), we propose that each node broadcasts in its neighborhood a special packet called E_state, which carries the current energy state of the sender. This small packet is sent each time the energy consumed rate changes with a certain rate. For instance, if we fix this rate to 25%, each node sends three packets; the first up on consuming 25% of the total battery capacity, the second up on 50%, and the last one up on 75%. This adds an overhead of complexity $O(3*n)$ packets transmission during the whole network life, where n is the nodes number. Generally speaking, if the rate is $1/x$ then the cost is $(x-1)*n$ packets transmission. We note that it may be interesting to increase this rate when batteries capacities decrease, because the energy state information become more and more important when the batteries lose their capacities. Although increasing this rate provides fresh information, it has the drawback of requiring more overhead.

- Routes are stored in each node's cache according to our metric $Opt_c$ presented previously (formula 4.4).

- Packets format modification: In order to implement our technique and metrics, we add some modifications to packets format. Mainly, each field of the record part (source route) should contain, in addition to the IP address of a node i, the current energy state of the node i ($E_i$), and the power to use on the link relaying it to the next hope $P(i, i+1)$.

- Adding the possibility to discover more routes: when a node receives a request that it has already received, contrary to DSR, it may continue relaying it if this request has followed a different route from the one followed by the first received request. This depends on the ALL_ROUTES flag that we add to the request. Although this technique allows discovering much more routes than the standard DSR, it results in more overhead, especially when the connectivity increases. We will test the performance of this technique in the next section.

- Route maintenance modification: mobility may make a power used on a link insufficient without causing the link failure. This occurs when nodes forming the link move away from each other, while keeping the distance between them less than their power range. Therefore, a mechanism allowing to discover the new suitable power is required. We suggest to proceed as follows: When a packet fails to pass through a link, and the MAC layer protocol of the sender turns it back to the routing protocol, the last protocol tries to resend the packet using the full-power, by paging it back to what we call **a directed request**, which is a special request packet that differs from the ordinary request in the fact that it is sent to a unique node (the upstream node of the appropriate link), instead of being propagated. Moreover, the sender updates its variables in order to temporally avoid sending packets via this link. If the upstream receives the packet, then it replies with a maintenance reply packet. Otherwise, after a given timeout the sender's MAC protocol turns back the directed request to its routing protocol, then this latter detects the link failure.

- Using data dispersal: to distribute the routing load over a maximum number of nodes, and to benefit from the available routes in cache, we propose to use our data dispersal technique presented previously.

### 4.4.2 Protocol description and verification

In the following we describe the algorithm executed by each node, and we analyze and verify the operations added to DSR, relying on the correctness and the stability of DSR. Our protocol can be summarized by the following events:

***Having packets to send***

This local event is triggered when a node has one or more data packets to send to any destination. If the node has any routes to the destination then it can send the packets, using our data dispersal technique previously presented. However, if the node has no route to the appropriate destination then it launches a route request (if no request to the destination has recently been launched). When the node finds no route to the destination, it reacts exactly as in DSR, i.e. re-launches the request after a timeout. The only difference appears when the node has more than one route to the destination. In this case it considers all these route, contrary to DSR where just the

shortest one is used. Also, our protocol considers route optimality (according to our metric) when selecting the number of packets to be transmitted on each route. Note that dispersing data packets causes no problem, since data in packet-based networks (to which ad hoc networks belong) usually follow different routes and the upper protocols (TCP/UDP) reorder them.

***Receiving a route request packet (RREQ)***

When a node receives a request packet, and if it is a mater of a directed request, then it computes the new power required and sends a reply. Otherwise, if the request is an ordinary one and if the node is not its final destination, then it computes the required power on the previous link, adds this computed value along with its current battery state to the RREQ, and continues its broadcast as in DSR. If the node is the RREQ's destination it sends back a RREP exactly like in DSR, it just adds to each field of the source route discovered the power-aware information computed and collected along the route discovery process. The directed request does not exist in the standard DSR. We have added it in order to maintain unbroken links, on which the power used becomes insufficient because of node mobility. This operation does not affect the protocol correctness: when sending a directed RREQ, if the destination still be in the sender's power range then it replies and provides the new required power, thereby the forwarding continues like in DSR. On the other hand, if the link is broken then the sender will not receive any reply, and after a timeout it will react for the link broken detection like in DSR. However, the impact of these operations is the possible rise of the latency, that will be investigated later in the simulation. The power-aware information addition into RREQs increases the overhead, but provides important energy gain later when transmitting data packets. In the simulation study we present in the next section, we will check whether the gain is more important than the cost.

***Receiving a route reply packet (RREP)***

When a node receives a maintenance reply packet, it updates its cache, re-sends awaiting packets for this maintenance, and sends a route error back to each node which is using the maintained link. Otherwise, if the reply type is not for maintenance, the receiver reacts as in DSR. Re-sending awaiting packets is analogous to salvaging packets after a link break in the standard DSR. The only difference is that when salvaging after a link break the packets will be transmitted on a new discovered route, but in our case they will be transmitted on the same route using a new up-to-date power on the current hop. Sending a route error packet to nodes currently using the maintained link has the purpose of informing these nodes about the actual required power on the link in question, which would be useful at these nodes for routes selection. The only impact of this procedure is the overhear increase.

***Receiving a data packet to forward***
This event is triggered whenever a node receives a data packet for which it is not its final destination but an intermediate router. If the next hop of the routing header was recently broken then the node sends an error packet to the source. If the link is waiting for maintenance then the node puts the packet in a waiting queue. Otherwise, if the link is supposed to be reliable, the router continues forwarding the packet like in DSR, but using the power specified in the source header instead of using the full-power. The only considerable difference from DSR is when the link is waiting for maintenance. In this case, packets are delayed. All the possible subsequent operations in this case have been already discussed.

***Receiving a link error message***
It is a local event triggered by the MAC protocol (IEEE 802.11 enables it) when the node fails to relay a packet to its next hop. If the packet returned is a directed request, then the node detects a link failure. Otherwise, it supposes that the power used over the link is not sufficient anymore, and tries to maintain this link by sending a directed request.

***Receiving a route error packet (RRER)***
Error packets can be launched either because of a link failure or because the power required on a link changes, according to the cost field value (a field we add to the RRER packet). If the cost field value is $\infty$, then it is a matter of a link failure and the node reacts as in DSR. Otherwise, it just updates its cache by taking into consideration the new power value, and it normally continues forwarding the RRER packet.

***Receiving E_state***
When a node receives an E_state packet it updates the parameters used to compute routes, and its cache as well.

## 4.5 Simulation study

To evaluate the performance of our protocol we have driven a simulation study using GloMoSim [ZBG98], to which we have added many extensions, such as the implementation of our protocol. In this study, we derived from our protocol four versions, by varying the parameter ALL_ROUTES, and the function $\alpha$, then we compare them to the standard DSR available in GloMoSim from which our protocol has been derived. As it will be illustrated later in the next section, our protocol shows important improvement in power consumption while keeping acceptable delays. We termed our protocol DSRPA as DSR Power-Aware, and its versions as follows:
DSRPA0: ALL_ROUTES parameter is disabled, and $\alpha = F1$
DSRPA1: ALL_ROUTES parameter is enabled, and $\alpha = F1$
DSRPA2: ALL_ROUTES parameter is disabled, and $\alpha = F2$

| # nodes | 10 |
|---|---|
| Simulation Time | 900S |
| Terrain | 330m * 250m |
| Power range | 150 m |
| Mobility pattern | Random way point |
| Propagation pattern | Free space |
| MAC protocol | IEEE802.11 |
| Application traffic generation | CBR |
| Replaying from cache | No |
| Minimum time separating 2 requests | 10 mS |
| Time separating 2 transmissions of a request | 500 mS |
| E_state sending fraction | 10% |
| Link failure detection timeout | 1S |

Table 4.1: Simulation set up

DSRPA3: ALL_ROUTES parameter is enabled, and $\alpha = F2$

### 4.5.1 Simulation environment

We simulated a network of 10 nodes moving around in a 330 * 250 $m^2$ area during 900 seconds. Each node has a power range of 150 m, and moves according to the random way point model [MHJ98]. Table 4.1 summarizes the simulation set up.

### 4.5.2 Metrics of comparison

Our purpose is to minimize the energy consumption and maximize the battery life time of **all** nodes, so that the network partition will be avoided as long as possible. Intuitively, this means maximizing the average battery life time and minimizing the battery life time difference between nodes. A protocol is considered more power-efficient than another if it causes; less energy consumption, more average battery life time, and less battery life time difference. By introducing new routing metrics, our protocol adds more communication and computation overhead, thus more energy consumption. The measurements of the energy consumption allow us to check whether the energy gain provided from our protocol is higher than the energy consumption caused by its overhead. Moreover, this overhead may increase the latency. To investigate this impact, the end to end delay is also included as a metric of comparison in our simulation. Our simulation study includes the following metrics of comparison:

**Consumed Energy**

The energy computation in GloMoSim is based on NCR Wavelan radio model. Node i's consumed energy ($PC_i$) is computed using the following formula:

$$PC_i = \sum_{Rx} TD \times (RRR - RSR) + \sum_{Tx} TD \times (RTR \times p/p_{max} - RSR) + RSR \times (R_{off} - R_{on}) \ldots\ldots (4.6)$$

Where:
$TD$ = packet size $/bandwidth + ST$
$ST$ = 192 micro second
$RTR(RadioTransmissionRate) = 3/second$, $RRR(RadioReceptionRate) = 1.48/second$, $RSR(RadioSleepRate) = 0.18/second$
$R_{on}$: the radio turning on time (simulation start time)
$R_{off}$: the radio turning off time (simulation end time)
$P$: the transmission power
$p_{max}$: the maximum full power, it is 72.321 mWhr in our simulation, which is the power allowing to cover the 150m power range in the free space model.
Tx: The set of transmitted packets
Rx: The set of received packets

In other words, $PC_i$ is the sum of: the power consumed for receiving all packets, the one consumed for transmitting all packets, and the power consumed during the time when the radio is in the sleep mode. Since the last part of formula 4.6 ($RSR \times (R_{off} - R_{on})$) is equivalent to the power consumed in the sleep mode for the *whole* simulation time, the equivalent consumed energy in the sleep mode during receptions and transmissions has been subtracted in the first two sums ($\sum_{Rx} TD \times RSR$ and $\sum_{Tx} TD \times RSR$). Thus, the formula 4.6 can be rewritten into:

$$PC_i = \sum_{Rx} TD \times RRR + \sum_{Tx} TD \times RTR \times p/pmax + RSR \times (R_{off} - R_{on}) - \sum_{Tx} TD \times RSR - \sum_{Rx} TD \times RSR \ldots\ldots (4.7)$$

The first two sums represent together the energy consumed for communication, the former represents the total energy consumed in reception, whereas the latter is the total energy consumed in transmission. The remainder of the formula is the energy consumed by the radio when it is in the sleep mode, which is unaffected by the routing protocol and is always the same, hence we do not consider it.

Our first metric of comparison is the consumed energy for communication, it is the two first sums of the previous formula averaged to make an average metric on the number of nodes (m), it is given by:

$$E_{com} = \sum_{i=1}^{m} \frac{\sum_{Rx} TD \times RRR + \sum_{Tx} TD \times RTR \times p/p_{max}}{m} \ldots\ldots (4.8)$$

This metric can be divided into two parts; the reception average energy, and the transmission average energy, they are respectively given by:

$$E_{RX} = \sum_{i=1}^{m} \sum_{Rx} \frac{TD * RRR}{m} \ldots\ldots (4.9)$$

$$E_{TX} = \sum_{i=1}^{m} \sum_{Tx} \frac{TD * RTR * p/p_{max}}{m} \ldots\ldots (4.10)$$

**Average battery life time**

It is the average battery discharge time. That is, the average time when nodes' batteries are discharged, it is given by:

$$BL_{avg} = \sum_{i=1}^{m} BL_i/m \ldots\ldots (4.11)$$

$BL_i$: is the discharge time of nodes i's battery, we also call it the battery life time of node i.

**Battery life time difference**

It is the difference between the maximum battery life time and the minimum one, formally speaking:

$$BL_{dif} = \max_{i=1..m}(BL_i) - \min_{i=1..m}(BL_i) \ldots\ldots (4.12)$$

The performance regarding this metric will be achieved by minimizing it. Thus, a protocol is more energy efficient than another if it increases the average battery life time and decreases this metric.

**End to end delay**

This well-known metric is the average time separating the data packets transmission from source nodes and their arriving at destinations. If we note this metric by *delay*, then:

$$delay = \sum_{i \in pr} \frac{delay_i}{\parallel pr \parallel}$$

such that: $pr$ is the set of packets received by all the destination nodes, $\parallel pr \parallel$ is the number of the received packets and, $delay_i$ is the transfer delay of the packet i, where:

$delay_i$ = packet i arrival time - packet i transmission time.

We will investigate the impact of our protocol's computation and communication overhead on this metric.

### 4.5.3 Simulation stages

We evaluate our protocol's performance by comparing it to DSR implemented in GloMoSim in different network loads, battery charges, and mobility situations. In order to investigate the mobility impact on our solution, we use along our simulation three kinds of mobility: mob0 which is a stationary situation (no mobility), mob1 which is a medium mobility, the nodes average speed is 0.5 m/s resulting in an average link change [DDB06] of 8.00, and finally the high mobility mob2, resulted from an average speed 1m/s which is important vis-a-vis the simulation scenarios, since it causes an average link change of 14.00. The network load is generated by CBR sessions between remote nodes, that last for the whole simulation time. The packet size is maintained to 1 Kb all along our simulation, and we vary the sender's CBR load (throughput) by changing the time separating two transmissions, the loads used are 1kb/s, 2kb/s, 3 kb/s, 4kb/s and 5 kb/s. We assume that all the nodes have the same inial battery charge. During the performance evaluation vs. the load, it is fixed to 300mWhr when measuring the energy and the end to end delay (this value avoids any discharge during the 10 minutes of simulation), and this parameter is fixed to 160 mWhr when measuring the other metrics to cause discharges during the simulation time. During the measurements vs. the battery capacity, the batteries charge is changed from 40 mWhr to 200mWhr. Note that these values are far from the real batteries charges [RVW02] that require much more simulation time, thus very powerful equipment. However, these last measurements (vs. the battery charge) illustrate the impact of the battery charge on the protocol performance. All the metrics presented in the next sections are measured in the different situations of mobility, network load, and battery capacity, requiring as much as 225 scenarios (executions).

## 4.6 Simulation results

### 4.6.1 Consumed energy

The histograms A, C, and E of figure 4.4 show the DSR consumed energy and those of each DSRPA versions v.s the load in the different mobilities (mob0, mob1, and mob2). The other diagrams; B, D and F represent the energy consumption of DSR and that of the best version of our protocol, they clearly illustrate the difference between DSR and our protocol. We remark from the diagram A that all the versions of DSRPA consume too less energy than DSR, and they have almost the same consumption. But for mobility Mob2, the versions that use ALL_ROUTES option (DSRPA 1 and DSRPA3) consumes slightly more energy than the others. However, all the versions of DSRPA outperform DSR in all situations. From the diagrams B, D and F, we can see that the difference between DSR and DSRPA decreases with the mobility, but it increases with the traffic load even for the high mobility (Mob2). We conclude that

DSRPA ensures an important gain in the energy consumption, especially when the network load increases. In the following, we will investigate this gain by analyzing the two parts of this metric (transmission and reception energy).

**Transmission energy**

Diagrams representing the transmission energy (figure 4.5) have almost the same forms as the previous ones. However, we remarked some differences:

- The difference between DSR and DSRPA is more important especially for mob2
- The versions that use ALL_ROUTES option do not consume more energy for transmissions than the others.

**Reception energy**

We remark that the plots in diagrams A and B of figure 4.6 have almost the same forms as the corresponding ones of the previous figure. Nevertheless, for the mobility Mob 2, and contrary to the previous plots, we remark that before the debit 4Kb/s DSR consumes slightly less energy than DSRPA, and DSRPA becomes a little bit more efficient for the highest load (5Kb/s). On the one hand, unlike DSRPA, DSR uses the promiscuous mode, thereby nodes receive all packets they overhear when they are in the sleep mode, which explains the DSRPA's gain. On the other hand, mobility causes more overhead for maintaining routes in DSRPA that uses adaptable powers. As we have said, node mobility may renders powers used on links insufficient without breaking these links down, which causes extra maintenance only to DSRPA but not to DSR. This overhead for maintenance explains the difference in favor of DSR in the first part of the diagram C. We point out that because the extra maintenance overhead has been largely compensated by the gain provided from the power control technique and from the metric used to select routes, this outperforming of DSR was not observed neither in the transmission energy part nor in the total communication energy. The histograms are not represented, since they have quite the same forms as those of figure 4.4. The minor extra consumption of versions that use the ALL_ROUTES option is due to the wide propagation of requests.

### 4.6.2 Average Battery life time

On diagrams A, C, and E of figure 4.7 we present the average battery life time vs. the load. The same metric is represented vs. the battery initial charge on the other diagrams. The histograms representing the DSRPA's versions are omitted since all these versions have all but the same values. We can clearly see that DSRPA is always more efficient than DSR, and that the difference between DSR and DSRPA increases with the network load and the battery charge, but it slightly decreases with the mobility. The DSRPA's gain is due to the energy gain analyzed previously.

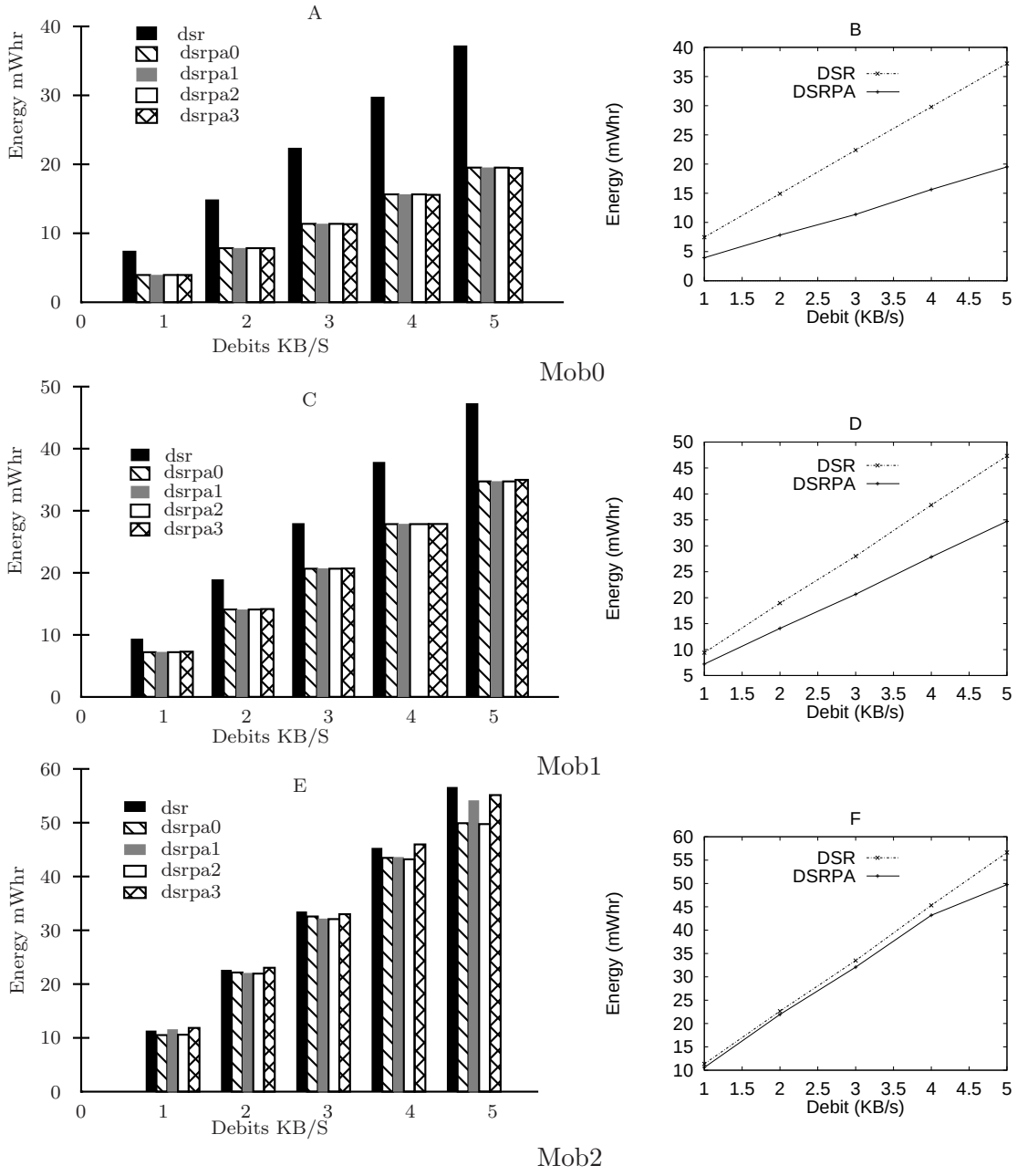


Figure 4.4: Consumed communication energy

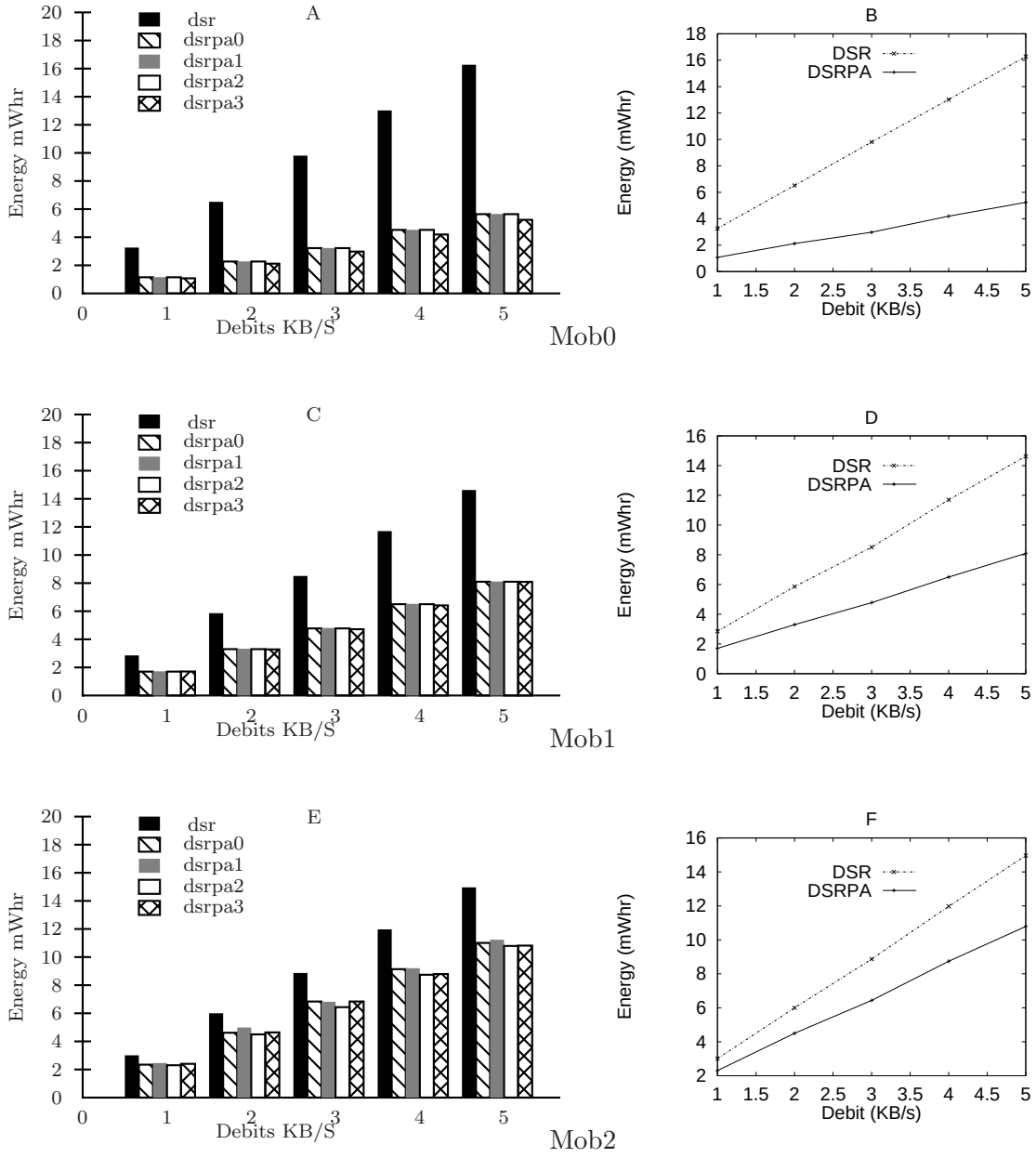


Figure 4.5: Consumed transmission energy

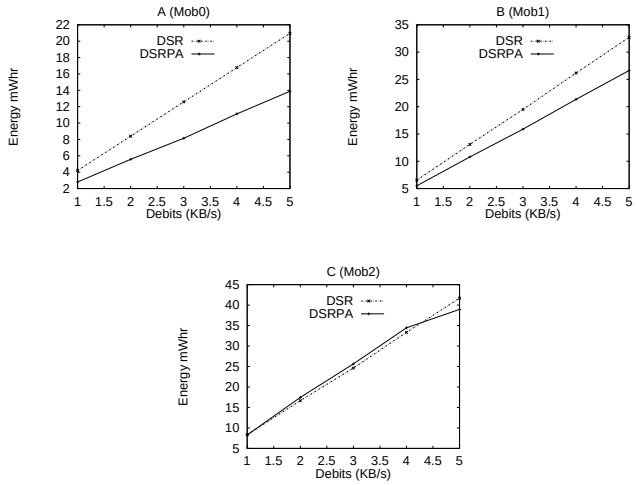


Figure 4.6: Consumed Reception energy

### 4.6.3 Battery life time difference

Diagrams B, D and F of figure 4.8 illustrate how the difference between DSR and DSRPA with respect to the battery life time difference is important. It increases with the load and also with the mobility, contrary to the other metrics. This performance is mainly due to the battery-awareness of our metrics, and also to the employment of a dispersing technique that disperses the network charge over the available routes. From diagrams A, C and E, we can see that DSRPA3 has often the best performance (the minimum difference), the other versions are close to each other. Hence, the function F2 and the ALL_ROUTES option minimize the battery life time difference while keeping the average battery life time good enough. But note that the choice of the ALL_ROUTES option must depend on the network connectivity. When the connectivity is high, this option causes a lot of requests broadcasting resulting in more energy consumption.

Figure 4.9 shows the difference between DSRPA and DSR on the battery life time difference vs. the initial battery charge. The plot A is monotony increasing upon the charge 80mWhr, where B is increasing up to the charge 120, then it becomes stable. The last one shows an important increasing from charges 80 to 160, and it is nearly stable out of this interval. Generally speaking, the difference between DSRPA and DSR with respect to the battery life time difference increases with the initial battery charge, and it is very important for the high charges (160mWhr an 200mWhr) compared with the low ones (40 mWhr and 80 mWhr).

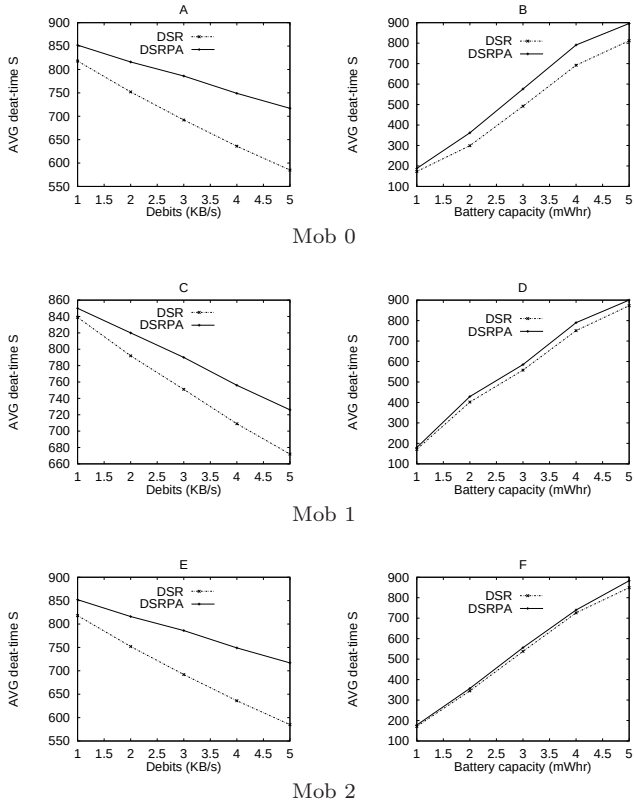


Figure 4.7: Average life time

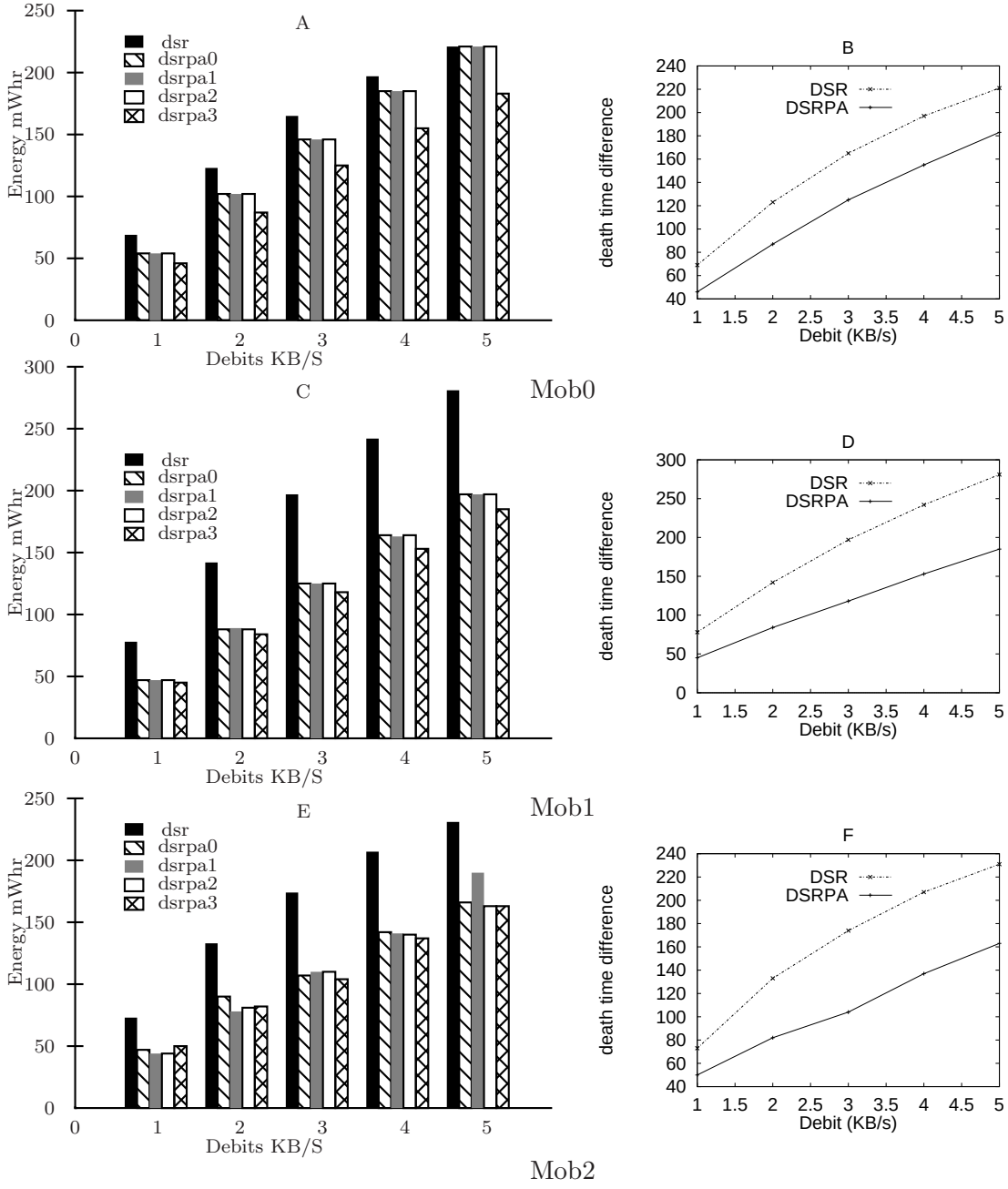


Figure 4.8: Battery life time difference vs the load

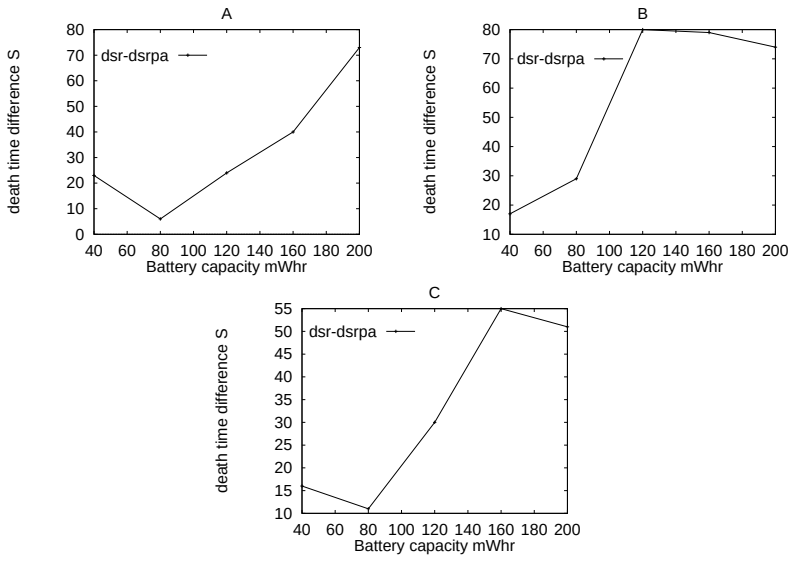


Figure 4.9: Battery life time difference vs Battery charge

### 4.6.4 End to end delay

Figure 4.10 shows the end to end delay of data packets vs. the network load in different mobilities. We remarked that all the versions of our protocol have always almost the same values, thus we omit histograms representing these versions.

As illustrated in plots of diagrams A, B, and C of figures 4.10, representing respectively mobilities mob0, mob1, and mob2, the delay of our protocol is too close to that of DSR in all situations. Nevertheless, the delay in DSR is a bit lower than that of DSRPA. This is mainly due to the computation and communication overhead added by our protocol to exchange power-aware information. However, the difference is minor, and overall it is unaffected by the mobility or the load increase.

## 4.7 Conclusion

One of the promising techniques proposed in literature to reduce the energy consumption in ad hoc networks is the power transmission control. When using this technique, the obvious metric for route selection is to minimize the total power required, but this choice may cause an overuse of a subset of nodes, resulting in a network partition. To avoid such a problem, battery states should be considered when selecting routes.

In this chapter, we have defined new power-aware metrics and a dispersal technique that can be implemented with any routing protocol. The proposed metrics

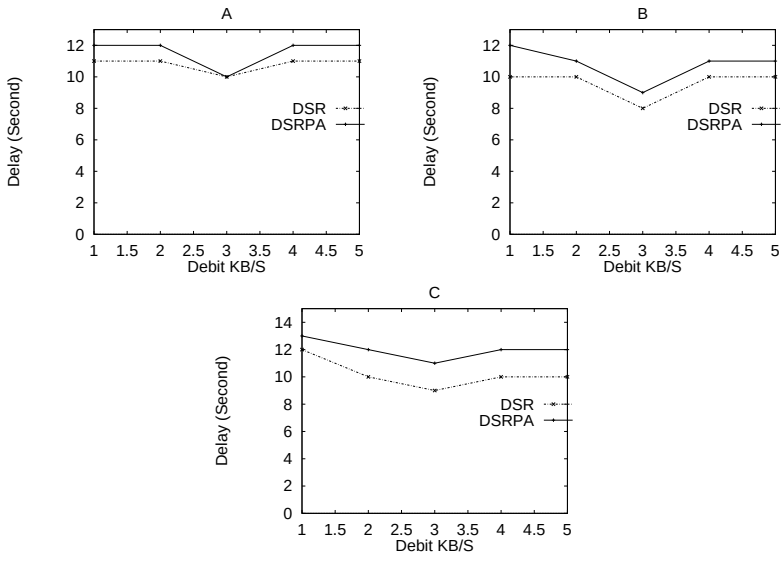


Figure 4.10: End to end delay vs Load

represent both powers required on links and the states of nodes' batteries, while the dispersing technique aims at distributing the network data load over as much nodes as possible while considering their optimality. Basing on these metrics and techniques we have presented a new power-aware DSR-based routing protocol, whose purpose is to allow nodes to stay alive together as long as possible.

The performance of our protocol has been evaluated by simulation in different situations of mobility and network load. We have defined two kinds of comparison metrics, the first represents the energy consumption, where the second represents the battery life time. The last kind includes two metrics: the average battery life time, and the battery life time difference. This latter is note related to the consumed energy at all. We have also measured the end to end delay, and investigated the impact of the overhead introduced by our protocol's operations on this important QoS (Quality of Service) metric. An efficient protocol must both increase the average battery life time (by decreasing the consumed energy), and decrease the battery life time difference, without affecting the end to end delay.

The results of our simulation show important improvements compared with the standard DSR, especially for high load situations, while keeping the end to end delay too close to the one of DSR. The second function (F2) for the parameter $\alpha$ shows a bit more efficiency, as well as the ALL_ROUTES parameter. But as for the last one, it must be treated carefully in high connected networks.

The proposed protocol relies on the cooperation and the well-behaving of nodes. However, in some applications, notably the self-organized ad hoc networks applications where nodes do not belong to a single authority and do not pursue common goal, a node has no direct interest to consume its energy on routing packets for other nodes, and may tend to be *selfish* (as illustrated before). In the following chapter we will propose a solution to this emergent problem.

# Chapter 5

# New solution to get over selfish misbehavior

## 5.1 Introduction

As we have seen in the third chapter, all the current reputation-based detective solutions employ the promiscuous monitoring of the watchdog in their monitoring module, and thus inherit all its drawbacks, especially the ones related to the power control use. In our proposal, we first define a novel approach of monitoring that overcomes some watchdog's shortcomings, and which is notably operational when employing the power-control technique. We then gradually decrease the overhead it engenders [DBar], to make it appropriate for the MANET environment. After the monitoring component, we will complete the solution by proposing methods for detecting and isolating selfish nodes [DB06c], which are basically based on our new monitoring technique. We mathematically analyze [DBar, DB06a] our solution and asses its performance by simulation.

## 5.2 Monitoring

In this section we define a new approach and a gradual solution [DBar] to mitigate the problems of the watchdog. Particularly, we aim at suggesting a monitoring solution operational when employing the power control technique, but with a reasonable cost (overhead).

Like the watchdog, in our approach each node of the source route monitors the forwarding of each packet it sends. Hence, it works with a source routing protocol. To explain the concepts, we use the same example of two hops illustrated in the chapter 3 when introducing the watchdog, then the concepts will be smoothly generated to the whole route exactly like the watchdog, as the same process is repeated on each

couple of hops until reaching the final destination.

Suppose that A sends packets to B and monitors its forwarding to C. We define a new kind of feedbacks we call *two-hop ACK*, an ACK that travels two hops. In our context, node C acknowledges packets sent from A by sending this latter via B a two-hop ACK. We suppose that the sending of a two-hop ACK by C after receiving a packet is a non violable operation. This can be ensured by implementing this simple function at a tamper resistant module (e.g. hardware). All the other functions of our solution, however, do not need such a requirement. Contrary to C, node B has no interest of not forwarding a two-hop ACK after it correctly forwards the data packet. Nonetheless, it could escape from the monitor without being detected by simply sending A a *falsified* two-hop ACK. Note that performing in this way is power economic for B, since sending a short packet like an ACK consumes too less energy than a data packet. To overcome this vulnerability, we use an asymmetric cryptography based strategy as follows:
Node A generates a random number and encrypts it with C's public key (PK), then appends it in the packet's header. When C receives the packet it retrieves the number, decrypts it using its secret key (SK), encrypts it using A's PK, and puts it in a two-hop ACK it sends back to A via B. When A receives the ACK it decrypts the random number and checks whether it matches with the one it has generated, to validate the forwarding of B regarding the appropriate packet. However, if B does not forward the packet A will not receive the two-hop ACK, and it will be able to detect this misbehavior after a timeout. This strategy requires a key distribution mechanisms enabling a security association between each pair of nodes. For this purpose, any of the ad hoc adaptable solutions presented in the second chapter (2.4.2) can be used, notably the certificate chain based one [CBH03]. Note that the same keys could be employed for other security purposes at other layers.

Also note that the encryption/decryption operations have minor impact, since they are applied merely on the random number and not on the whole packet holding it. We avoided the use of digital signatures in order to avoid useless packet hash computation. As we will see later, acknowledging each packet by a two-hop ACK requires an important communication and computation overhead. We will gradually decrease this overhead, firstly by employing the ordinary MAC ACKs, and secondly by defining a random requesting strategy.

### 5.2.1 Solution 1

Our first solution implements the two-hop ACK, but with no optimization. As we will see, this solution is efficient but engenders too high overhead.

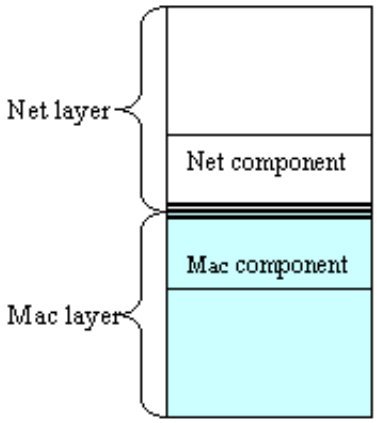


Figure 5.1: Solution 1 architecture

**Solution description**

Hereafter we describe the algorithm executed at the monitor of each node i. We use the following notations:
• $R_{Key}$: encrypting R with Key
• $R^{key}$: decrypting R with Key
• $P_X$: the public key of node X
• $S_X$: the secrete key of node X.

Node i keeps a buffer called Wait2HopsACK, each entry of which corresponds to a packet whose forwarding is being monitored by i. The entry contains the address of the forwarder node, as well as the random number generated by i for the packet. The solution is composed of two parts; the first one is located at the network layer and can be viewed as a sub-layer at the bottom of this layer, whereas the second one is located at the MAC layer and is a sub layer at the top of this latter (figure 5.1 illustrates this architecture). This cross-layer design allows to put the critical operations at the lower layer, which rushes them and ensures robustness, while keeping all the other operations of monitoring at the network layer with the routing protocol. This cross-layer also enables the overhead reduction in the second solution.

Each node, except the destination, is monitored by the previous node. To monitor its successor, each node i adds to each packet it receives from the routing protocol a random number it generates, encrypted with the PK of the successor's successor in the source route, along with i's address. Afterwards, it maintains the generated random number as well as the monitored node (i's successor) address in an entry within Wait2HopsACK. When a packet is received from another node X, i's MAC component automatically generates and sends X back a two-hop ACK, after encrypt-

ing and decrypting again the random number as described previously. The Network layer component removes the appropriate entry upon the reception of the two-hop ACK, and as a timeout is associated to each entry, the lack of the two-hop ACK after the timeout results in recording a packet dropping regarding the appropriate forwarder node. Note that the timeout should be fixed to optimally cover the required time for receiving the two-hop ACK related to the packet forwarding. Algorithms 1 and 2 describe respectively the network component and the MAC component of this solution.

**Algorithm 1** Network layer component of solution 1

***When receive a packet D from the routing protocol to send to node X (X either the next hop or the destination and i is either the source or a forwarding node):***

**if** (X $\neq$ D's destination) **then**
R = a generated random number
Y = X's successor in the source route
append $(R_{P_Y}, i)$ to D's header
add(R,X) to the buffer Wait2HopsACK
**end if**
send D to X

***When receive a packet D from the MAC protocol sent by X:***

**if** (X $\neq$ D's source) **then**
remove the random number generated by X's predecessor from the header along with the corresponding node address
**end if**
send the packet to the network layer protocol

***When receive a two-hop ACK packet TwoHopsACK from the MAC layer component***

$R' = TwoHopsACK.Rand^{S_I}$
**if** $(R', TwoHopsACK.sender) \in Wait2HopsACK$ **then**
remove $(R', TwoHopsACK.sender)$ from Wait2HopsACK
**end if**

***When a timeout of a Wait2HopsACK entry (R,X) is exceeded***

notice a packet dropping regarding node X

### Discussion and correctness proof

Unlike the watchdog, this solution does not rely on the promiscuous mode usage, and is consequently operational regardless of the employment of the power-control technique. If channels are reliable such as no packet can be lost due to channel conditions

**Algorithm 2** MAC layer component of solution 1

***when receive a packet D sent by X from the MAC protocol***
  **if** (X $\neq$ D's source) **then**
    Y = X's predecessor in the source route
    Get the random number R generated by y
    $R' = R^{S_I}$
    $R'' = R'_{P_Y}$
    construct a two-hop ACK packet TwoHopsACK
    TwoHopsACK.Rand= $R''$
    TwoHopsACK.sender=I
    TwoHopsACK.dest=Y
    send two-hop ACK to X
  **end if**
  pass the packet up to the network component
***when receive a two-hop ACK packet TwoHopsACK***
  **if** (TwoHopsACK.dest $\neq$ i) **then**
    TwoHopsACK.sender=i
    forward TwoHopsACK to TwoHopsACK.dest
  **else**
    pass the packet up to the network layer component
  **end if**

or nodes mobility, this solution allows to detect any drop and consequently any misbehavior as it will be proved later. In practice, however, packets may be lost due to channel conditions or mobility of nodes and not inevitably because of a misbehavior performed by B. This is known as unintentional dropping. For this reason, A should not accuse directly B but records ratings and uses a threshold before judging B. We will deal with the judgment issue in the next section.

Unlike the end to end ACK (described in the third chapter), this new mechanism allows to detect the misbehaving node and not just the route containing such a node. Moreover, our solution enables the employment of the power-control technique and gets rid of many shortcomings of the watchdog, as the forwarding of B will not be validated at A until C really receives the packet and sends back the two-hop ACK, unlike the watchdog where the validation is only related to B's first transmission. When a collusion appears at C, B should retransmit the packet, otherwise A will not validate its forwarding, contrary to the watchdog where B could deny retransmitting the packet without being detected. Also, remember that when A is closer to B than C and when A uses the watchdog to monitor B, this latter could save its energy and makes the transmission power strong enough to be overheard by A but less than the required one to reach C. This vulnerability is avoided in our solution, and B must retransmit the packet to make its forwarding valid. Nevertheless, this initial solution requires an important overhead, as we will see later.

Now, we will prove the correctness of our solution. Our proof [DB05b] basically relies on the assumption that the links are totally reliable. That is, no packet can be unintentionally dropped. The proof model we use is constructed using petri nets, on which the proof is preformed by applying some linear algebra concepts. Hereafter we give the basic linear algebra theorems we use in the proof, followed by the model and the prof. Definitions of the mathematical concepts used here can be found in the appendix.

**Theorem 1.** ***(Marking reachability [G.W83]):*** *A necessary condition of the marking M reachability from $M_0$ is that:*
*$M - M_0$ must be orthogonal to ($\perp$) all the solutions of $C^t.X = 0$, where $^t$ denotes the transpose matrix, and C the incidence matrix (see definition 2 in the appendix) formally speaking:*
$M \in R(< Net, M_0 >) \Rightarrow \forall x \in \{X \setminus C^t X = 0\}, M - M_0 \perp x.$
*We point out that:* $M - M_0 \perp x \Leftrightarrow (M - M_0)^t x = 0.$

**Theorem 2.** ***Host state using linear algebra [G.W83]):*** *This theorem exploits a linear algebra concept to build a sufficient condition regarding the host state. It is as follows:*
*If a marked petri net admits a norm for a marking M, then M is a host state of this*

*net.*

**Theorem 3** ([G.W83]). *Let* $f \in N^{np}$ *and* $E = \{t \in T / \Delta(t, f) > 0\}$, *if* $E = \emptyset$ *or* $f^t M < min_{t \in E}(\sigma(t, f))$ *then*
$\forall M' \in R(Net, M), \forall p \in \| f \|, M'(p) \leq \frac{f^t M}{f(p)}$.
*Such that np is the number of place in the petri net, the other functions and notations used in this theorem are described in the appendix.*

**The model:**
As we have seen before, in our approach each node monitors the next hop forwarding of each packet it sends, i.e. node A sends packets to B and monitors its forwarding to C, A may be either the source or an intermediate node. This concept is generalized along the path from the source to the destination. To prove that the protocol does what it has to do (detects the packets dropped), we have just to prove that when the monitoring node A sends n packets to B, then if B drops m out of these n packets, A validates **exactly** n-m forwarding, thereby it detects the m packets dropped. We model our protocol by the following petri net: $Net_0$=(P,T,I,O,$M_0$)
P=$\{P_0, P_1, ....., P_{10}\}$
$P_0$: buffer of packets to send by A
$P_1$: buffer of packets dropped at B
$P_2$: buffer of packets that are being monitored by A, i.e. tokens in this place represents entries in the A's Wait2HopsACK buffer
$P_3$: buffer of packets whose forwarding has been validated, i.e. tokens in this place represent entries removed from the A's Wait2HopsACK buffer
$P_4$: buffer of valid two-hop ACKs sent (forwarded) from B to A, which have not been treated by A
$P_5$: buffer of packets sent from A to B, not already received by B (at the network layer)
$P_6$: buffer of packets received by B, not already treated
$P_7$: buffer of packets forwarded by B to C, not already received by C
$P_8$: buffer of packets received by C, not already treated
$P_9$: buffer of a two-hop ACK packets sent by C, not already received by B
$P_{10}$: buffer of two-hop ACK packets received by B

T=$\{T_0, T_1, ....., T_8\}$
$T_0$: node A sends a packet to B
$T_1$: B receives a packet from A
$T_2$: A validates a packet forwarding and remove the appropriate entry from its Wait2HopsACK buffer
$T_3$: B forwards a packet
$T_4$: B drops a packet
$T_5$: C receives a packet from B
$T_6$: C sends a two-hop ACK packet

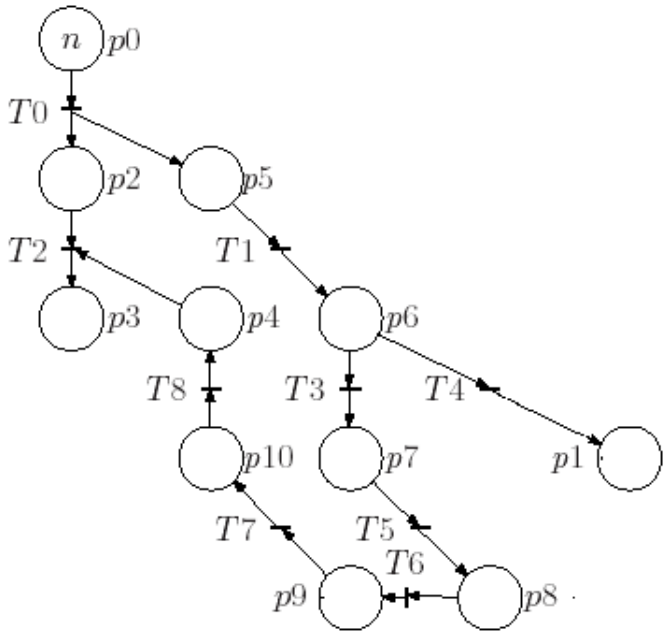


Figure 5.2: The initial petri net

$T_7$: B receives a two-hop ACK packet
$T_8$: B forwards a two-hop ACK packet

I and O are represented by the following matrices:

$$I = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad O = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$M_0$ is represented by the following vector: $\begin{bmatrix} n & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^t$.
Figure 5.2 illustrates this petri net.

**Theorem 4.** *Let n be the number of packets sent by A, and m the number of packets*

*dropped by B, then our protocol ensures the following:*
$\forall n \in \mathbb{N}, \forall m \in \mathbb{N}, m \leq n$, *A validates exactly n-m packets forwarding.*

Since each packet is monitored by A and is associated with a timeout, supposed to be great enough to the required time for receiving the two hops ACK related to the packet forwarding, if the packet forwarding is not validated (the packet has not been removed from the buffer up to a timeout), then it will be supposed dropped and will cause the B's rating increase. Hence, validating exactly n-m packets is equivalent to detect m B's dropping. Thereby, this theorem shows the protocol correctness. We bring the problem to our petri net model, and we propose the following lemma:

**Lemma 1.** $\forall n \in \mathbb{N}$, $M_f = \begin{bmatrix} 0 & m & m & n-m & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^t, m \leq n$, *is a sink host state to* $(Net0, M_0)$.

If so, we realize that our system terminates at $M_f$. In other words, whatever the sequence of transitions fired, $M_f$ will be reached (host state property), and no transition will be enabled from this marking (sink state property).
The semantic behind this, according to our model, is that when B drops m packets out of n, A will *inevitably* validate *exactly* n-m forwarding of B. This because $M_f$ reaching means: the number of packets dropped by B ($p_1$'s tokens) is m, the number of packets whose forwarding is validated by A ($p_3$'s tokens) is n-m, and no other validation will take place since $M_f$ is a sink state. We have just to prove this lemma to conclude the previous theorem.

**Net reduction** Now we reduce our initial petri net $Net_0$ using substitution. $p_5$ is a substitutable place, since it fulfills all the substitutable place conditions, we take $H = \{T_0\}$ (the input transition of $p_5$), $F = \{T_1\}$ (the output transition of $p_5$), and m=1.

To the resulting net we do the same thing with $p_7$, and so on for $p_8, p_9, p_{10}$. The resulting net of this five substitution sequence is obtainable from $Net_0$ by removing the places $p_5, p_7, p_8, p_9, p_{10}$ and the transitions $T_1, T_4, T_5, T_6, T_7, T_8$, and adding a transition $R(T_4)$ relating $p_6$ to $p_4$ such that $I(p_4, R(T_4)) = O(P_6, R(T_4)) = 1$. This last resulting net is also reducible, since $P_6$ is substitutable. The only difference in this substitution is that the set F includes two elements i.e. $F = \{T_3, R(T_4)\}$, whereas H contains just one element ($T_0$), the final un-reducible net is $Net_r(P_r, T_r, I_r, O_r, M_{r_0})$:
$P_r = \{P_0, P_1, P_2, P_3, P_4\}$ (a subset of P)
$T_r = T'_0, T'_1, T_2$.
Note that $T'_0, T'_1$ are different from $T_0, T_1$, for writing simplicity, we note them in the following respectively $T_0$ and $T_1$.

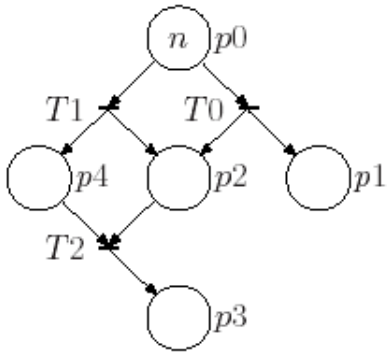


Figure 5.3: The reduced petri net

$$I_r = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, O_r = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}, M_{r_0} = \begin{bmatrix} n & 0 & 0 & 0 & 0 \end{bmatrix}^t$$

Figure 5.3 illustrates this net.

**Theorem 5.** ***(equivalence of host state existence when reducing [G.W83]):*** *the host state property is reducible by substitution.*

In other words, if $Net_2$ is the reduced petri net of $Net_1$, obtained by substituting the place p, then $M_{f1}$ is a host state of $Net_1$, iff $M_{f2}$, which is obtained from $M_{f1}$ by removing the entry related to p, is $Net_2$'s host state. We can generalize this theorem to $Net_n$ obtained by more than one substitution using transitivity. The same process applies with the sink sate property.cv

It results from the theorem 5 and the reduction of the initial network, that the lemma 1 can be reducible to the following one.

**Lemma 2.** $\forall n \in \mathbb{N}, M_f = \begin{bmatrix} 0 & m & m & n-m & 0 \end{bmatrix}^t, m \leq n$, *is a sink host state to* $(Net_r, M_{r_0})$.

In the following we prove this lemma.

It is obvious that $M_f$ is a sink state, since it enables no transition. It remains to prove that it is a host state. We exploit theorem 2 and try to find a norm.

**The norm:**

**Lemma 3.** $\forall(M = \begin{bmatrix} X_0 & X_1 & X_2 & X_3 & X_4 \end{bmatrix}^t) \in \Psi, X_1 \leq n,$
*such that* $\Psi = R(< Net_r, M_{0r} >)$

**Proof:** For $f = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \end{bmatrix}^t$: $\Delta(T_0, f) = 0, \Delta(T_1, f) = \Delta(T_2, f) = -1$, hence: $E = \emptyset$ (see theorem 3).
It results from the theorem 3: $\forall M \in \Psi, \forall p \in \| f \|, M(p) \leq (\frac{f^t M_0}{f(p)} = n/f(p))$, thus:
$(M(1) = X_1) \leq n$.

This lemma ensures that $n - X_1 \geq 0$, this term will be used in the following norm we propose:

$V : (\Psi \subset \mathbb{N}^5) \rightarrow \mathbb{N}$
$V(M) = X_0 + 3X_4 + [|\log(\frac{X_0 + \max(X_1, X_2) + 1}{\min(X_1, X_2) + 1})|]_{sup} + [|\log(\frac{X_2 + \min(n - X_1, X_3) + 1}{X_0 + X_2 + \max(n - X_1, X_3) + 1})|]_{sup}$

Such that $|x|$ denotes the absolute value of x, and $[x]_{sup}$ stands for the upper integer part defined by:
$\forall r \in R, [r]_{sup} = m \in N$ such that $m \geq r$ and $m - 1 < r$.

This function is $\mathbb{R} \rightarrow \mathbb{N}$. Therefore, each part of the sum V ($Term_i$ $i = 0..3$) is within $\mathbb{N}$, even though the functions used (log, /) give values in $\mathbb{R}$. Using the previous lemma we can verify that the functions used are well defined in $\Psi$, therefore they are applications from $\Psi$ to $\mathbb{N}$, and so is the norm V. In the purpose of proving lemma 2, we suggest the following lemma:

**Lemma 4.** *V is a norm for* $M_f$ *in* $Net_r$

To prove this we have to prove that V fulfills the two conditions of definition 5 (appendix).

**The first condition:** $V(M) = 0 \Leftrightarrow M = M_f \ldots\ldots (con1)$
remember that $M_f = \begin{bmatrix} 0 & m & m & n - m & 0 \end{bmatrix}^t, m \leq n$

i) $M = M_f \Rightarrow V(M) = 0$ : this implication is obvious, we need just to compute $V(M_f)$, we will find that it equals to 0.

ii) $V(M) = 0 \Rightarrow M = M_f$ :
Since V is the sum of four terms in $\mathbb{N}$, V(M)=0 $\Leftrightarrow \forall i \in 0, 1, 2, 3$ $Term_i = 0$

$$V(M) = 0 \Rightarrow \begin{cases} Term_0 = 0 \Leftrightarrow X0 = 0 & \& \\ Term_1 = 0 \Leftrightarrow X4 = 0 & \& \\ Term_2 = 0 \Leftrightarrow \log(\frac{X_0 + \max(X_1, X_2) + 1}{\min(X_1, X_2) + 1}) = 0 & \& \\ Term_3 = 0 \Leftrightarrow \log(\frac{X_2 + \min(n - X_1, X_3) + 1}{X_0 + X_2 + \max(n - X_1, X_3) + 1}) = 0 \end{cases}$$

$$V(M)=0 \Rightarrow \begin{cases} X0=0 & \& \\ X4=0 & \& \\ \frac{\max(X_1,X_2)+1}{\min(X_1,X_2)+1}=1 \Leftrightarrow \max(X_1,X_2)+1=\min(X_1,X_2)+1 & \& \\ \frac{X_2+\min(n-X_1,X_3)+1}{X_2+\max(n-X_1,X_3)+1}=1 \Leftrightarrow X_2+\min(n-X_1,X_3)+1=X_2+ \\ \max(n-X_1,X_3)+1 \end{cases}$$

$$V(M)=0 \Rightarrow \begin{cases} X0=0 & \& \\ X4=0 & \& \\ \max(X_1,X_2)=\min(X_1,X_2) \Leftrightarrow X_1=X_2 & \& \\ min(n-X_1,X_3)=\max(n-X_1,X_3) \Leftrightarrow X_3=n-X_1 \end{cases}$$

V(M)=0 $\Rightarrow$ M= $\left[\begin{array}{ccccc} 0 & m & m & n-m & 0 \end{array}\right]^t, m \leq n \Rightarrow M=M_f \square$

**The second condition:**

$\mathbf{V(M) \neq 0 \Rightarrow \exists M' \in R(< Net, M >)}$ **such that** $\mathbf{V(M') < V(M)} \ldots\ldots (con2)$

**case 1:** $\mathbf{X_0 \neq 0}$

In this case $T_0$ is enabled, we will prove that it leads to a marking $M'$ which holds the condition.

Let us consider $M=\left[\begin{array}{ccccc} X_0 & X_1 & X_2 & X_3 & X_4 \end{array}\right]^t \in R(< Net_r, M_{0_r} >)$, and $M(T_0 > M_{T0}$, then:

$M_{T0}=\left[\begin{array}{ccccc} X_0-1 & X_1+1 & X_2+1 & X_3 & X_4 \end{array}\right]^t$

$V(M) \neq 0$ is verified since $X0 \neq 0$, we have to verify that $V(M_{T0}) < V(M)$

$V(M_{T0})=\underbrace{X_0-1}_{\overline{Term_0}}+\underbrace{3X4}_{\overline{Term_1}}+\overline{Term_2}+\overline{Term_3}$

We have: $\overline{Term_0}=Term_0-1 \Rightarrow \overline{Term_0} < Term_0$, and $\overline{Term_1}=Term_1$

Now, we try to prove that $\overline{Term_2} \leq Term_2$ and $\overline{Term_3} \leq Term_3$ to realize that $V(M_{T0}) < V(M)$

$\overline{Term_2}=[|\log(\frac{X_0-1+\max(X_1+1,X_2+1)+1}{\min(X_1+1,X_2+1)+1})|]_{sup}=[|\log(\frac{X_0-1+\max(X_1,X_2)+1+1}{\min(X_1,X_2)+2})|]_{sup}=$
$[|\log(\frac{X_0+\max(X_1,X_2)+1}{\min(X_1,X_2)+2})|]_{sup}$

Since: $1 \leq \frac{X_0+\max(X_1,X_2)+1}{\min(X_1,X_2)+2}) \leq (\frac{X_0+\max(X_1,X_2)+1}{\min(X_1,X_2)+1}$, we realize that $\overline{Term_2} \leq Term_2$, because log is an increasing function in the interval $[1,+\infty[$, as well as the absolute value and upper integer part functions.

It remains the last terms; $Term_3$ and $\overline{Term_3}$.

$\overline{Term_3}=[|\log(\frac{X_2+1+\min(n-X_1-1,X_3)+1}{X_0+X_2+\max(n-X_1-1,X_3)+1})|]_{sup}$

In one hand, we have:

$\min(n-X_1-1,X_3) \geq \min(n-X_1,X_3)-1 \Rightarrow$

$X_2+1+\min(n-X_1-1,X_3)+1 \geq X_2+\min(n-X_1,X_3)+1 \ldots\ldots (5.1)$

On the other hand: $X_0+X_2+\max(n-X_1,X_3)+1 \geq X_0+X_2+\max(n-X_1-$

$1, X_3) + 1 \ldots\ldots (5.2)$

From 5.1 and 5.2 it results: $\frac{X_2+1+\min(n-X_1-1,X_3)+1}{X_0+X_2+\max(n-X_1-1,X_3)+1} \geq \frac{X_2+\min(n-X_1,X_3)+1}{X_0+X_2+\max(n-X_1,X_3)+1}$
Since: $\frac{X_2+1+\min(n-X_1-1,X_3)+1}{X_0+X_2+\max(n-X_1-1,X_3)+1} \leq 1$, and $|log(x)|$ is a decreasing function on $[0,1]$, we realize that $\overline{Term_3} \leq Term_3$

We have proved that: $\overline{Term_2} \leq Term_2$, $\overline{Term_3} \leq Term_3$, $\overline{Term_0} < Term_0$, and $\overline{Term_1} = Term_1$. Hence, $V(M_{T0}) < V(M)$. It results that then the condition cond2 is fulfilled (we have just to take $M' = M_{T0}$)

**case 2: $\mathbf{X_0 = 0}$**
in this case both $T_0$ and $T1$ are disabled.

**case 2.1: when $\mathbf{T_2}$ is enabled ($X_2 > 0, X_4 > 0$)**
As in case 1, $V(M) \neq 0$ is verified. We will perform in the same way and try to prove that $V(M_{T2}) < V(M)$, such that $M(T_2 > M_{T2}$.
$M_{T2} = \begin{bmatrix} 0 & X_1 & X_2 - 1 & X_3 + 1 & X_4 - 1 \end{bmatrix}^t$
$V(M_{T2}) = 3(X_4 - 1) + [|\log(\frac{\max(X_1,X_2-1)+1}{\min(X_1,X_2-1)+1})|]_{sup} + [|\log(\frac{X_2-1+\min(n-X_1,X_3+1)+1}{X_2-1+\max(n-X_1,X_3+1)+1})|]_{sup}$
$= \underbrace{-1}_{\overline{Term_0}} + \underbrace{3X4}_{\overline{Term_1}} + \underbrace{[|\log(\frac{\max(X_1, X_2 - 1) + 1}{\min(X_1, X_2 - 1) + 1})|]_{sup} - 1}_{\overline{Term_2}} +$
$\underbrace{[|\log(\frac{X_2 - 1 + \min(n - X_1, X_3 + 1) + 1}{+X_2 - 1 + \max(n - X_1, X_3 + 1) + 1})|]_{sup} - 1}_{\overline{Term_3}}$
As in case 1, we have $\overline{Term_0} < Term_0$ (since $Term_0 = 0$), and $\overline{Term_1} = Term_1$, so we have just to prove that $\overline{Term_2} \leq Term_2$ and $\overline{Term_3} \leq Term_3$

$\mathbf{\overline{Term_2} \leq Term_2}$ :
using the property: $\log(a/b) = \log(a) - \log(b)$, we will have:
$Term_2 = [|\log(\max(X_1, X_2) + 1) - \log(\min(X_1, X_2) + 1)|]_{sup}$
$\overline{Term_2} = [|\log(\max(X_1, X_2 - 1) + 1) - \log(\min(X_1, X_2 - 1) + 1)|]_{sup} - 1$
$\max(X_1, X_2 - 1) \leq \max(X_1, X_2) \Rightarrow \max(X_1, X_2 - 1) + 1 \leq \max(X_1, X_2) + 1$
$\Rightarrow \log(\max(X_1, X_2 - 1) + 1) \leq log(\max(X_1, X_2) + 1) \ldots\ldots (5.3)$
The last implication is justified by the fact that $\max(X_1, X_2 - 1) + 1 \geq 1$ and the function log is increasing in the interval $[1, +\infty]$
$\min(X_1, X_2 - 1) \geq \min(X_1, X_2) - 1 \Rightarrow \min(X_1, X_2 - 1) + 1 \geq \min(X_1, X_2) \Rightarrow \log(\min(X_1, X_2 - 1) + 1) \geq \log(\min(X_1, X_2)) \Rightarrow \log(\min(X_1, X_2 - 1) + 1) + 1 \geq \log(\min(X_1, X_2)) + 1 \ldots\ldots (5.4)$
As $\log(x) + 1 \geq \log(x + 1)$, $\log(\min(X_1, X_2)) + 1 \geq \log(\min(X_1, X_2) + 1)$, thus:

$(5.4) \Rightarrow \log(\min(X_1, X_2 - 1) + 1) + 1 \geq \log(\min(X_1, X_2) + 1)$
$\Rightarrow -\log(\min(X_1, X_2 - 1) + 1) \leq -\log(\min(X_1, X_2) + 1) + 1 \ldots\ldots (5.5)$

$(5.3 \text{ and } 5.5) \Rightarrow \log(\max(X_1, X_2-1)+1)-\log(\min(X_1, X_2-1)+1) \leq \log(\max(X_1, X_2)+1) - \log(\min(X_1, X_2) + 1) + 1$
$\Rightarrow [|\log(\max(X_1, X_2 - 1) + 1) - \log(\min(X_1, X_2 - 1) + 1)|]_{sup} \leq [|\log(\max(X_1, X_2) + 1) - \log(\min(X_1, X_2) + 1)|]_{sup} + 1$
$\Rightarrow [|\log(\max(X_1, X_2-1)+1)-\log(\underline{\min}(X_1, X_2-1)+1)|]_{sup}-1 \leq [|\log(\max(X_1, X_2)+1) - \log(\min(X_1, X_2) + 1)|]_{sup} \Rightarrow \overline{Term_2} \leq Term_2$
In the same way, $\overline{Term_3} \leq Term_3$ can be proved.
As in case 1, the condition con2 is fulfilled, (the existence of $M'$ is justified by $M' = M_{T2}$).

**case 2.2: when $\mathbf{T_2}$ is disabled**

We will gradually prove that the unique form of the marquin reachable from $M_{0_r}$ and representing such a case in the same time is $M_f$. Let M be The marquin representing case 2.2, $M = \begin{bmatrix} 0 & X_1 & X_2 & X_3 & X_4 \end{bmatrix}^t \in R(< Net_r, M_{0_r} >)$

i) We prove $\mathbf{X4 = 0}$
$T_2$ is disabled $\Rightarrow (X_2 = 0$ or $X_4 = 0)$.
We will prove $(X_2 = 0 \Rightarrow X_4 = 0)$ to conclude that always $X_4 = 0$ in this case (case 2.2), because when $X_2 \neq 0$, $X_4$ must be 0 to disable $T_2$.
We prove this using the reductio ad absurdum. Assume $X_2 = 0$ and $X_4 \neq 0$. That is: $M = \begin{bmatrix} 0 & X_1 & 0 & X_3 & X_4 \end{bmatrix}^t \in R(< Net_r, M_{0_r} >), X_4 \neq 0$
According to theorem 1, $\forall v \in \{{C_r}^t Y = 0\}, (M - M_{0_r}) \perp v$

$$ {C_r}^t Y = 0 \Leftrightarrow \begin{bmatrix} -1 & -1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & -1 \\ 0 & 0 & 1 \\ 0 & 1 & -1 \end{bmatrix}^t \times \begin{bmatrix} Y_0 \\ Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \end{bmatrix} = 0 $$

$$ \Leftrightarrow \begin{bmatrix} -1 & 1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 & 1 \\ 0 & 0 & -1 & 1 & -1 \end{bmatrix} \times \begin{bmatrix} Y_0 \\ Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \end{bmatrix} = 0 $$

$$ \Leftrightarrow \left\{ \begin{array}{ll} -Y_0 + Y_1 + Y_2 = 0 & \& \\ -Y_0 + Y_2 + Y_4 = 0 & \& \\ -Y_2 + Y_3 - Y_4 = 0 & \end{array} \right\} \ldots\ldots\ldots \text{(sys1)} $$

$(M - M_{0_r}) \perp Y \Leftrightarrow (M - M_{0_r})^t Y = 0$
$\Leftrightarrow -nY_0 + X_1Y_1 + X_3Y_3 + X_4Y_4 + = 0 \ldots\ldots (5.6)$
a) when $X_3 \neq n$, we remark that $\Omega_0 = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \end{bmatrix}^t \in \{{C_r}^t Y = 0\}$ (it is a

solution to sys1)
$\Omega_0$ fulfills (5.6) $\Rightarrow X_3 = n$, which represents a contradiction
b) when $X_3 = n$, we remark that $\Omega_1 = \begin{bmatrix} 2 & 1 & 1 & 2 & 1 \end{bmatrix}^t \in \{C_r{}^t Y = 0\}$ (it is a solution to sys1)
$\Omega_1$ fulfills (5.6) $\Rightarrow X_1 + X_4 = 0 \Rightarrow X_1 = -X_4 \Rightarrow X_1 < 0$ (since $X_4 \neq 0$) which represents a contradiction.
Whatever the values of $X_3$ and n, ($X_4 \neq 0$ and $X_2 = 0$) leads to contradictions, thereby: $X_2 = 0 \Rightarrow X_4 = 0$, hence, $X_4 = 0$.
$M = \begin{bmatrix} 0 & X_1 & X_2 & X_3 & 0 \end{bmatrix}^t$

ii) $\mathbf{X_1 = X_2}$
$(M - M_{0_r}) \perp Y \Leftrightarrow (M - M_{0_r})^t Y = 0 \Leftrightarrow -nY_0 + X_1Y_1 + X_2Y_2 + X_3Y_3 = 0 \ldots\ldots (5.7)$
$\Omega_2 = \begin{bmatrix} 0 & 1 & -1 & 0 & 1 \end{bmatrix}^t \in \{C_r{}^t Y = 0\}$ (it is a solution to sys1)
$\Omega_2$ fulfills (5.7) $\Rightarrow X_1 = X_2$
M= $\begin{bmatrix} 0 & X & X & X_3 & 0 \end{bmatrix}^t$

iii) $\mathbf{X_3 = n - X}$
$(M - M_{0_r}) \perp Y \Leftrightarrow (M - M_{0_r})^t Y = 0 \Leftrightarrow -nY_0 + XY_1 + XY_2 + X_3Y_3 = 0 \ldots\ldots (5.8)$
$\Omega_3 = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \end{bmatrix}^t \in \{C_r{}^t Y = 0\}$ (it is a solution to sys1)
$\Omega_3$ fulfills (5.8) $\Rightarrow X_3 = n - X$
We realize that M= $\begin{bmatrix} 0 & X & X & n-X & 0 \end{bmatrix}^t$ in this case (case 2.2)
Therefore, $V(M) = 0$, which means that $V(M) \neq 0$ is false, consequently the condition con2 is satisfied $\square$.
Hence we have proved that V is a norm to $M_f$ (lemma 4), and we realize the correctness of lemma 2, lemma 1 and the theorem 4.

**Communication and computation complexity**

We have proved that if channels are reliable and no packet will be lost, then our first monitoring solution detected accurately all the packets dropped. Moreover, it enables the power-control employment, contrary to the watchdog. In the following, we will analyze its overhead, while keeping the same assumption of reliability.

Since the two-hop ACK monitoring is performed on each couple of hops in the route, its overhead clearly increases with the route length, as well as with the number of packets to be monitored. Thus, we consider monitoring n packets on an $h$ hops route, where each node misbehaves with a probability $\theta$. This latter parameter ($\theta$) affects the number of packets to be monitored on each hop as we will see in the following.

Mathematically, we mean by misbehaving with a parameter $\theta$ that the behavior of the monitored node for each packet follows a Bernoulli distribution with a parameter $\theta$ (the probability of dropping). Monitoring n packets could be considered as the

repetition of the previous operation (monitoring one packet) $n$ times. The number of packets dropped is then expressed by a Binomial distribution with a parameter (mathematical expectation) $n \times \theta$.

We first deal with the communication overhead, which is the number of two-hop ACK transmissions (the only kind of control packet the solution adds). This number is as much as twice the number of two-hop ACK requested, since each packet travels two hops and involves two separate transmissions in this first solution. The number of two-hop ACK requested on the $i^{th}$ hop (which is a random variable) is simply the number of data packets arrived at this hop (as each packet is monitored in this first solution), we denote by $Na_i$. The total number of two-hop ACKs requested (which is a sum of random variables) is:

$\sum_{i=0}^{h-2} Na_i \ldots\ldots (5.9)$.

The number of of two-hop ACK transmissions $Np_0$ is twice the previous one:

$$Np_0 = 2 \times \sum_{i=0}^{h-2} Na_i.$$

In the following we will compute the mathematical expectation of this random variable:

$$E(Np_0) = 2 \times \sum_{i=0}^{h-2} E(Na_i).$$

On the first hop $E(Na) = n$, since the source transmits all its packets. On the second one it decreases by the number of packets dropped by the first forwarder, i.e. it decreases by $n\theta$ and becomes $n(1-\theta)$. On the third one this latter decreases by $n(1-\theta)\theta$ and becomes $n(1-\theta)^2$, and so on unti the last hop involving a monitoring (h-2, h-1), on which the number of ACK requests is $n(1-\theta)^{h-2}$. As a result:

$$E(Np_0) = 2 \times \sum_{i=0}^{h-2} n(1-\theta)^i = 2 \times n \times \sum_{i=0}^{h-2} (1-\theta)^i$$

As the last sum is a finite geometric series (when $\theta \neq 0$) of $h-1$ terms, with a first term equals to 1 and ratio of $(1-\theta)$, we get:

$$E(Np_0) = \begin{cases} O(2 \times \frac{n}{\theta}(1-(1-\theta)^{h-1})) & when \quad \theta > 0 \\ O(2 \times (h-1) \times n) & when \quad \theta = 0 \end{cases} \ldots\ldots (5.10)$$

Note that the watchdog has no communication overhead, as it uses no control packets for monitoring.

Now we deal with the computation overhead, which is the number of steps (at the network layer level) for the execution of the monitoring distributed algorithm. As it relies on the promiscuous mode monitoring, the watchdog includes only the forwarding of the packets, as the overhearing is performed simultaneously (in the same step) on each hop with the forwarding on the next hop. Hence, Each forwarding is considered as one step at the network layer. Note that it might actually needs more steps at the MAC layer, e.g: when using IEEE 802.11, one forwarding may involve 4 steps (RTS/CTS/DATA/ACK), and more when collisions occur (see the first chapter). However, we avoid these considerations and make our analysis at the network layer. The total steps $SW$ is simply the total number of packet forwarded, which is the number of packets arrived at each hop (including the last one). It is the number expressed by formula (5.9) plus the number of packets forwarded at this hop. That is:

$$SW = \sum_{i=0}^{h-1} Na_i.$$

Identically to $E(Np_0)$, the mathematical expectation $E(SW)$ can be computed. It is:

$$E(SW) = \begin{cases} O(\frac{n}{\theta}(1-(1-\theta)^h)) & when \quad \theta > 0 \\ O((h) \times n) & when \quad \theta = 0 \end{cases} \quad \ldots\ldots (5.11)$$

On the other hand, our first solution needs, in addition to the previous steps of forwarding, the transmissions of the two-hop ACKs. The number of the latter is simply $Np_0$. So the totally number of our first solution's steps $SS0$ is the sum $SW + Np_0$. Its expectation ($E(SS0) = E(SW) + E(Np_0)$) can be computed from (5.10) and (5.11). It is:

$$E(SS0) = \begin{cases} O(3 \times \frac{n}{\theta}(1-(1-\theta)^h)) & when \quad \theta > 0 \\ O(3 \times (h) \times n) & when \quad \theta = 0 \end{cases} \quad \ldots\ldots (5.12)$$

Note that when a packet is transmitted before the previous one reachs its destination, some steps could be performed simultaneously when $h > 3$, both in the watchdog and our solution. However, we considered the worst case in our analysis, in which steps are performed sequentially. It typically takes place when each packet is not sent until the previous one reaches its destination. Both the computation and the communication overhead of our first solution are important. In the following, we try to gradually decrease this cost.

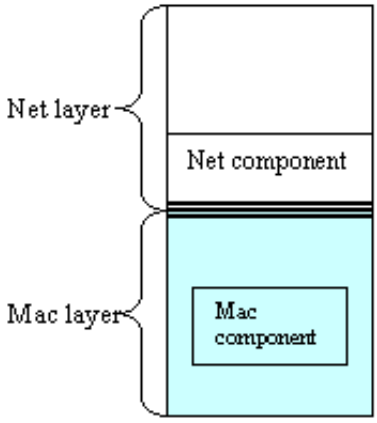


Figure 5.4: Solution 2 architecture

### 5.2.2 Solution 2

#### Solution description

To decrease the overhead we propose that node C employs the MAC ordinary ACK, where it integrates the two-hop ACK to acknowledge the forwarding of B. Consequently, the two-hop ACK forwarding costs only one extra transmission from B to A, instead of two. The MAC header and the MAC ACK packet should include the two-hop ACK information. A little extension is then required. We extend the MAC ACK packet with the following fields:
• TowHopsACK: A flag indicating if the ACK contains a two-hop ACK.
• Rand: Contains the random number
• SecondDest: The destination of the two-hop ACK.
We also add the following fields to the packet's MAC header:
• TwoHopsACK: A flag indicating if the packet requires a two-hop ACK
• Rand: Contains the random number
• TwoHopsSrc: The node to which a two-hop ACK sending is required.
There is no change required to the network component, but the MAC component needs to be changed. The packets provided by the physical layer have to be catched and treated by our protocol before sending any ACK back. Hence, in the scheme of this solution the MAC component can be viewed as a component within the MAC protocol instead of a sub layer over it, like shown in figure 5.4.

Upon the reception of a packet sent by X, node i encapsulates the information required to acknowledge X forwarding at X's predecessor in the ordinary ACK it sends back. When X receives this ordinary ACK it cheeks the TwoHopsACK flag,

and then realizes that the packet contains a two-hop ACK. It consequently constructs a two-hop ACK and sends it back to its predecessor. The MAC component of this latter passes the two-hop ACK to the network component to validate the forwarding of X and removes the appropriate entry from the $wait2hopsACK$ buffer. Algorithm 3 describes the MAC component of this solution.

**Discussion**

The only difference between the previous solution and this one is that the two-hop ACK packets are piggybacked to the ordinary MAC ACK. The modeling of the solution remains the same, as well as the correctness proof, since our model does not make any distinction between the transmission on a separate packet or within the MAC ACK. On the other hand, the communication complexity of the previous solution is reduced and divided by two, since half of the two-hop ACK separate transmissions are eliminated in this solution, i.e. the communication complexity is:

$$E(Np_1) = E(Np_0)/2 = \begin{cases} O(\frac{n}{\theta}(1-(1-\theta)^{h-1})) & when \quad \theta > 0 \\ O((h-1) \times n) & when \quad \theta = 0 \end{cases} \quad \ldots\ldots (5.13)$$

As for the computation complexity, the number of steps required for transmitting the two-hop ACKs is divided by two. The computation complexity of this solution is thus:

$$E(SS1) = \begin{cases} O(2 \times \frac{n}{\theta}(1-(1-\theta)^{h})) & when \quad \theta > 0 \\ O(2 \times (h) \times n) & when \quad \theta = 0 \end{cases} \quad \ldots\ldots (5.14)$$

### 5.2.3 Solution 3

**Solution description**

Although the previous optimization (solution 2) reduces the overhead, this latter remains high, especially when nodes well-behave ($\theta = 0$). We think the requesting of two-hop ACK should not be performed for each packet. Indeed, it should depend on the monitored node behavior, and be reduced as long as the node correctly forwards packets. For this purpose, we suggest to *randomize* the ACK ask [DOMB05], i.e. node A does not ask C an ACK for each packet but upon sending a packet to forward it *randomly* decides whether it asks an ACK or not with a coefficient (probability) $p$, then conceals this decision in the packet. A simple way to conceal the decision is to exploit the random number. For instance, when the node decides to ask an ACK, it selects an even number, and an odd number when it decides not to ask the ACK [1]. This *random* selection strategy prevents the monitored node from deducing which

[1] A necessary condition to use this is that the encryption algorithm is asymmetric with respect to the parity, i.e. the cypher of an even (resp an odd) number is not always an even or odd number

**Algorithm 3** MAC component of solution 2

***When receive a packet D sent by X***

**if** (D.MACHeader.TwoHopsACK == true) **then**
$R = D.MACHeader.Rand^{S_I}$
$R' = R_{P_{D.MACHeader.TwoHopsSrc}}$
construct an ACK packet ACKpack
ACKpack.TwoHopsACK = true
ACKpack.Rand= $R'$
ACKpack.SecondDest=D.MACHeader.TwoHopsSrc
the other fields have to be filed out by the MAC protocol
send the ACK to X
**else**
send an ordinary ACK if the packet requires an ACK
**end if**
pass the packet up to the network component after doing the handling required by the MAC protocol (moving the MAC header, frames defragmentation,..etc)

***When receive packet D from the network layer***

Do the required processing required by the MAC protocol (fragmentation, making the MAC header,..etc)
**if** (D's source $\neq$ i) **then**
D.MACHeader.TwoHopsACK = true
D.MACHeader.Rand= the random number generated and encrypted by the network component
D.MACHeader.TwoHopsSrc = i's predecessor
**else**
D.TwoHopsACK = false
**end if**
forward the packet

***When receive an ACK packet ACKpack***

**if** (ACKpack.TwoHopsACK ==true) **then**
construct a two-hop ACK packet TwoHopsACK
TwoHopsACK.Rand= ACKpack.Rand
TwoHopsACK.dest= ACKpack.SecondDest
TwoHopsACK.sender=I
send two-hop ACK to ACKpack.SecondDest
**end if**
do the handling required by the MAC protocol

***When receive a two-hop ACK packet TwoHopsACK***

pass the packet up to the network layer component

packets contain ACK requests. Note that getting such information allows a misbehaving to drop packets with no requests without being detected. When the node decides not to ask an ACK it directly validates the forwarding, whereas when it decides to ask an ACK it waits for it during a timeout like in the ordinary two-hop ACK.

A coefficient $p$ is maintained for each monitored node, and continuously updated as follows:
It is set to 1 (the initial value representing zero-trust) when a timeout exceeds without receiving the requested ACK, and to the trust value $P_{trust}$ when the requested ACK is received. This way, more trust is given to well-behaving nodes, and the ACK requesting is enforced after a lack of one ACK which allows to achieve all by the same performance in misbehaving detections (true positives) like the ordinary two-hop ACK as we will see later. Algorithms 4 and 5 illustrate our third solution.

**Algorithm 4** Network layer component of solution 3

***When receive a packet D from the routing protocol to send to node X (X either the next hop or the destination and i is either the source or a forwarding node):***

```
if (X ≠ D's destination) then
  Decide whether to require an ACK with probability P_X
  if (Two-hop ACK requirement set) then
    R = generate an even random number
  else
    R = generate an odd random number
  end if
  Y = X's successor in the source route
  append (R_{P_Y}, i) to D's header
  if (Two-hop ACK requirement set) then
    add(R,X) to the buffer Wait2HopsACK
  end if
end if
send D to X
```

***The handling of all the other events is identical to algorithm 1***

Unlike the first and the second solutions, this one does not detect exactly all the packet dropped. Thus, the previous petri nets modeling and proof are inappropriate. Hereafter, we first model this last solution, then we will mathematically compare its performance and cost vs. the previous one [DB06a].

**Algorithm 5** MAC component of solution 3

***When receive a packet D sent by X***

**if** (D.MACHeader.TwoHopsACK == true) **then**
  $R = D.MACHeader.Rand^{S_I}$
  **if** ($R$ mod 2 = 0) **then**
    construct an ACK packet ACKpack
    ACKpack.TwoHopsACK = true
    $R' = R_{P_{D.MACHeader.TwoHopsSrc}}$
    ACKpack.Rand= $R'$
    ACKpack.SecondDest=D.MACHeader.TwoHopsSrc
    the other fields have to be filed out by the MAC protocol
    send the ACK to X
  **else**
    send an ordinary ACK if the packet requires an ACK
  **end if**
**else**
  send an ordinary ACK if the packet requires an ACK
**end if**
pass the packet up to the network component after doing the handling required by the MAC protocol (moving the MAC header, frames defragmentation,..etc)

***The handling of all the other events is identical to algorithm 3***

**Modeling and analysis**

The main difference between the random two-hop ACK and the previous solutions is that the former uses a parameter $p$, the value of which changes from a packet to another. The previous solutions could be viewed as the random two-hop ACK with $p$ always fixed to 1. In the following, we try to model the change of $p$ at some monitor node during the monitoring of $n$ packets. As before, we assume the monitored node misbehaves with a probability $\theta$ (the behavior is independent from a packet to another). In our model, we consider the state $X_i$ as the value of $p$ upon monitoring the packet i. $X_i$ $i = 1...n$ consist of n random variables. The possible values of $p$ in the algorithm, thus of $X_i$ in the model, are 1 and $p_{trust}$. We denote them respectively $E_1$ and $E_2$. For a given value of $X_{i-1}$, the value of $X_i$ depends solely on $\theta$ (the probability of misbehaving), and not on the previous values of $X$, i.e.:

$P[X_i = xi|X_{i-1} = x_{i-1}, X_{i-2} = x_{i-2}, ....., X_0 = x_0] = P[X_i = xi|X_{i-1} = x_{i-1}]$, $\forall 1 \leq i \leq n, \forall x_i \in \{1, p_{trust}\}$

Therefore, $X_i$ $i = 1...n$ form a Markov chain. The transitions probabilities are:

- $P[X_i = E_1|X_{i-1} = E_1] = \theta$. If $p$ is fixed to 1 it remains so iff the monitored packet is dropped.
- $P[X_i = E_2|X_{i-1} = E_1] = 1 - \theta$. The complement of the previous probability
- $P[X_i = E_1|X_{i-1} = E_2] = \theta p_{trust}$. If $p$ is fixed to $p_{trust}$ for the packet $(i-1)$, it will be set to 1 iff that packet is dropped and detected. The probability of detection is the probability of asking an ACK which is $p_{trust}$, and the events dropping the $i^{th} packet$ and requesting ACK for the $(i-1)^{th}$ packet are independent.
- $P[X_i = E_2|X_{i-1} = E_2] = 1 - \theta p_{trust}$, the complement of the previous probability

These probabilities are independent of n, thus the chain is homogenous. Its transition matrix is:

$$T = \begin{bmatrix} \theta & 1-\theta \\ \theta p_{trust} & 1-\theta p_{trust} \end{bmatrix}$$

Figure 5.5 represent the transition graph of this chain.

$P_{trust}$ is strictly positive, and different from 1 in our last solution. When $0 < theta < 1$ the chain is irreducible and aperiodic, as all the elements of T become strictly positive [BGdMT98]. Therefore, it admits a stationary distribution [BGdMT98]. In the following we assume $\theta$ fulfills the previous condition ($0 < theta < 1$), then we will deal separately with the particular cases of $\theta = 0$, and $\theta = 1$.

In the stationary distribution, we note:
$P_1$: the probability to be in $E_1$.

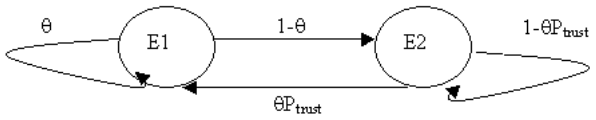


Figure 5.5: Markov chain transition graph

$P_2$: the probability to be in $E_2$.
The vector $\Pi = [P_1 P_2]$ can be obtained from the resolution of [BGdMT98]:

$$\begin{cases} \Pi \times T = \Pi \\ \sum_i \Pi_i = 1 \end{cases}$$

By resolving this system, we get:
$P_1 = \frac{\theta p_{trus}}{1-\theta(1-p_{trust})}$
$P_2 = \frac{1-\theta}{1-\theta(1-p_{trust})}$

This means that $p$ of our algorithm will be fixed to 1 with probability $P1$, and to $P_{trust}$ with probability $P2$. Its expectation is then:
$E(p) = 1 \times P_1 + p_{trust} \times p_2 = \frac{p_{trust}}{1-\theta(1-p_{trust})} \ldots\ldots (5.15)$.

**Detection:** We recall that in the operation of monitoring one packet, the monitored node misbehaves (drop the packet) with a probability $\theta$. Hence, its behavior for each packet follows a Bernoulli distribution with a parameter (mathematical expectation) $\theta$. Monitoring n packets could be considered as simply the repetition of the previous operation (monitoring one packet) $n$ times. Therefore, as illustrated before, the number of packets dropped ($pdr$) for n packets is a random variable that is the sum of n random variables following a Bernoulli distribution with parameter $\theta$. Thus, it follows a Binomial distribution with expectation:
$E(pdr) = \theta \times n \ldots\ldots (5.16)$

Let us note the number of packets dropped and detected by the third solution by $pd$. Our purpose in the following is to assess this number, i.e. compute its mathematical expectation $E(pd)$ in the stationary distribution, as well as the detection ratio $DR = E(pd)/E(pdr)$, which reflects the efficiency of the random asking strategy in detection.

The number of packets detected by the random strategy ($pd$) also follows a Binomial distribution, since it is the result of repeating a Bernoulli operation $n$ times with parameter $\theta \times p$, but the only difference from the continuous requesting (ordinary two-hop ACK) is that in the random strategy $p$ is not constant. We have:

$$E(pd) = \sum_{i=1}^{n} \theta E(p) = \theta \times n \times E(p) = \theta \times n \times \frac{p_{trust}}{1-\theta(1-p_{trust})} \ldots\ldots (5.17)$$

- From (5.16) and (5.17) we get:

$$DR = (E(pd)/E(pdr)) = \frac{p_{trust}}{1-\theta(1-p_{trust})} \ldots\ldots (5.18)\ .$$

**Overhead:** Remember that the number of two-hop ACK requested on the $i^{th}$ hop in the ordinary two-hop ACK monitoring is simply the number of data packets arrived at this hop, denoted by $Na_i$. In the second solution, the total number of two-hop ACKs requested $Np_1$ (which also represents the number of two-hop ACK transmissions) can be expressed by:

$Np_1 = \sum_{i=0}^{h-2} Na_i$, thus:

$$E(Np_1) = \sum_{i=0}^{h-2} E(Na_i) \ldots\ldots (5.19)$$

We will use this latter formula as both $Np_1$ and $Na_i$ are random variables.

As for the last (random) solution the number of two-hop ACKs requested (on each hop) can be expressed by: $Na_i \times P$, which is a random variable resulting from the multiplication of two random variables.
The total number of two-hop ACKs requested in this solution $Np$ is:

$Np = \sum_{i=0}^{h-2} PNa_i$, hence:

$$E(Np) = \sum_{i=0}^{h-2} E(p) \times E(Na_i) = E(p) \sum_{i=0}^{h-2} \times E(Na_i) = E(p)E(Np_1) \ldots\ldots (5.20).$$

From (5.19) and (5.20) we obtain the communication overhead reduction factor ($RF$):

$$RF = E(Np_1)/E(Np) = 1/E(p)$$

$$RF = \frac{1-\theta(1-P_{trust})}{P_{trust}} \ldots\ldots (5.21)$$

**Particular cases of $\theta$:**

i) $\theta = 0$.

- The $DR$ is meaningless in this case, since there is no dropping.

- $RF$: In this case, $p_i = p_{trust}$, $\forall i > 1$.
  $Np_1 = (h-1) \times n$ according to (5.13).

  $Na_i = n \forall i$, as there is no dropping

  In the random solution, as $p = 1$ for the first packet (on each hop), and $p = p_{trust}$ for the others, we have:

  $$Np = \sum_{i=0}^{h-2} np_{trust} + (1 - p_{trust})$$

  $Np = p_{trust} \times (h-1) \times n + (1-p_{trust})(h-1) = O(p_{trust} \times (h-1) \times n) = p_{trust} \times Np_1$

  We realize: $RF = 1/P_{trust}$

  Note that the previous expression of $RF$ equals $1/p_{trust}$ when $\theta = 0$. So, $RF$ is identical.

ii) $\theta = 1$.

In this case, $p_i = 1$, $\forall i$. The random solution is simply equivalent to the continuous requesting. So both, $DR = 1$ and $RF = 1$.

Also note that both the previous expressions of $RF$ and $DR$ equals 1 when $\theta = 1$. Consequently, the previous expressions of $DR$ (formula 5.18) and $RF$ (formula 5.21) could be generalized to represent these two parameters for all possible values of $\theta$, i.e. $\forall \theta \in [0, 1]$ for $RF$, and $\forall \theta \in ]0, 1]$ for $DR$, as $DR$ has no sense when $\theta = 0$.

Regarding the computation complexity, the number of steps required for transmitting the two-hop ACKs is divided by $RF$. The computation complexity of this solution is thus:

$$E(SS2) = E(SW) + (1/RF) \times E(Np1) =$$
$$\begin{cases} O((1 + 1/RF) \times \frac{n}{\theta}(1 - (1-\theta)^h)) & when \quad \theta > 0 \\ O((1 + 1/RF) \times (h) \times n) & when \quad \theta = 0 \end{cases} \quad \ldots\ldots (5.22)$$

Figures 5.6 and 5.7 respectively show different values of the detection ratio ($DR$) and the reduction factor ($RF$) according to $\theta$ for some usual values of $p_{trust}$. We realize from the figures that $P_{trust} = 0.5$ strikes a balance between efficiency (detection ratio) and cost (reduction factor). It decreases the complexity overhead as much as half (when nodes well-behave), while keeping the detection ratio good enough (always $\geq 0.5$). Contrary to $P_{trust} = 0.25$ that has too low values of detection ratio for low and average misbehaving, and to $P_{trust} = 0.75$ that has too low values of reduction factor. Thus, we fix $P_{trust} = 0.5$ later in our simulation study.

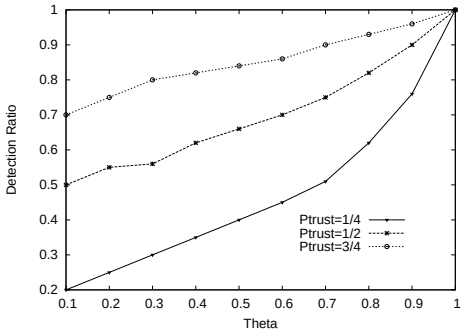


Figure 5.6: Detection Ratio

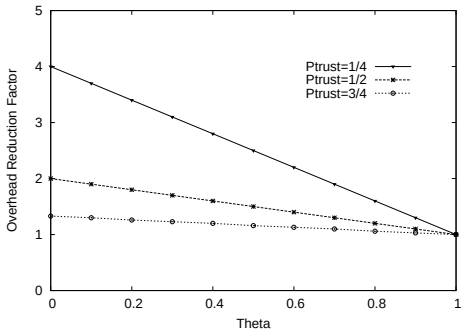


Figure 5.7: Overhead Reduction Factor

**Continues misbehavior:** The most common case treated in literature, and also the most rational selfish behavior, is the continuous misbehaving, either from the beginning or upon the node's battery state decreases below a defined level. This could be modeled by the previous Markov chain by replacing the parameter $\theta$ with 1.

$$T = \begin{bmatrix} 1 & 0 \\ p_{trust} & 1 - p_{trust} \end{bmatrix}$$

The worst case is when the monitored node starts misbehaving when $P = P_{trust}$, i.e. the initial state in our model is E2. We define the probability of convergence after n steps (n packets) $P_{conv}(n)$ as the probability of asking an ACK after sending n packets from the beginning of misbehaving. It is the probability of converging to the continuous asking, because upon asking an ACK the random solution will require an ACK for each subsequent packet in this case, in which the monitored drop all packets. In the model, this probability is the one of the passage from E2 to E1 after n steps. The discussion is independent from the admission of a stationary distribution. According to the model and [BGdMT98]:

$P_{conv}(n) = T^n_{2,1}$

**Lemma 5.** *$\forall n > 0$ we have:* $T^n = \begin{bmatrix} 1 & 0 \\ 1 - (1 - p_{trust})^n & (1 - p_{trust})^n \end{bmatrix}$

**proof:** by recurrence on n.
For $n = 1$, $T^1 = T$ fulfills the lemma.
Assume the equality for $n - 1$ (recurrent assumption), and we will prove it for n. Hence by assumption:

$$T^{n-1} = \begin{bmatrix} 1 & 0 \\ 1 - (1 - p_{trust})^{n-1} & (1 - p_{trust})^{n-1} \end{bmatrix}$$

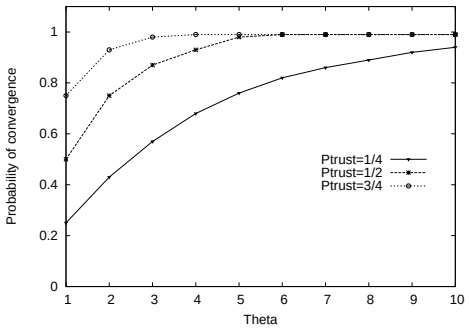


Figure 5.8: Probability of Convergence

On the other hand: $T^n = T^{n-1} \times T$. By computing this matrix product we get:
$T^n_{1,1} = 1$
$T^n_{1,2} = 0$
$T^n_{2,1} = 1 - (1 - p_{trus})^{n-1} + p_{trust}(1 - p_{trust})^{n-1}$
$T^n_{2,2} = (1 - p_{trus})^n$

We have just to simplify the writing of the element $T^n_{2,1}$, as all the others fulfills the lemma.

$T^n_{2,1} = 1 - ((1 - p_{trus})^{n-1}(1 - p_{trust})) = 1 - (1 - p_{trus})^n ... \square$

Consequently, the probability of convergence is:

$P_{conv}(n) = 1 - (1 - p_{trus})^n$

Figure 5.8 illustrates the convergence probability.

Note that if the node starts misbehaving when $P = 1$, the initial state is E1, $P_{conv}(n)$ would be simply the probability of remaining at E1 after n steps, which is equals to 1, i.e. $P_{conv}(n) = 1 \forall n$.

Also, we remark that $P_{trust} = 0.5$ has a very good values of $P_{conv}$.

### Broadcast packets (RREQ)

Thus far, we proposed a monitoring solution for *directed packets* and gradually improved it. We mean by directed packets those sent to one recipient. This category includes booth data packets and many control packets (RRER, RREP). However, the two-hop ACK approach is inefficient and unrealistic with broadcast packets, such as RREQs. The reason is that a broadcast packet is sent to all neighboring nodes, and requiring an ACK from all the two-hop neighbors for each neighbor is impractical at all.

We suggest for this kind to use another simple approach. Each node monitors each RREQ it forwards or launches as a source. The monitoring starts from the reception of the RREQ (or its launch if the node is the source) and ends after a timeout from its retransmission. For each RREQ, the transmitter monitors all its neighbors. It should either receive (or overhear) the RREQ or a RREP from them, except from which it received the RREQ if the node is not the source. If none of these packets is received from a neighbor X, then the monitor notices a packet dropping for X.

This solution requires no communication overhead for monitoring, but needs nodes to be neighborhood-aware. Neighborhood-awareness can be achieved by employing beacons either at the MAC or at the routing protocol.

**Simulation study**

In our mathematical study we assumed that channels are reliable, and no packet would be lost. This assumption, however, is hard to achieve in the ad hoc environment, featured by the the mobility of nodes and collisions of packets due to the hidden terminal problem (see chapter 1), which might cause loss of packets.

To assess the performance of the proposed monitoring protocol in more realistic conditions, we have driven a GloMoSim-based [ZBG98] simulation study we present hereafter [DB05a]. We have simulated a network of 50 nodes, located in an area of $1500 \times 1000m^2$ where they move following the random way-point model with an average speed 1m/s, and a pause-time of 2s, for 900 seconds of simulation time. To generate traffic we have used three CBR sessions between three pairs of remote nodes, each consists of continually sending a 512 bytes data packet each second. On each hop, each data packet is transmitted using a controlled power according to the distance between the transmitter and the receiver. This simulation setup engendered many collisions and packets loss.

In this subsection we compare two versions of our monitoring protocol, 2HopACK (solution 2) and Random 2HopACK (solution 3), as well as the watchdog (WD) with regard to the misbehaving detection rate, the false detection rate (rate of false accusations as misbehaving) and the number of two-hop ACK packets (which represents the overhead). We measured these metrics vs. the misbehaving nodes rate, which represents the rate of nodes that continuously misbehave and drop data packets they are asked to relay.

Each point of the plots presented hereafter has been obtained by averaging five measurements with different seeds. Note that we implemented our protocol with DSR for this simulation, like WD. Also note that WD requires no kind of ACK, so the last metric (number of two-hop ACK) concerns merely our protocol's versions. For judg-

ing a node as a selfish, we use in this step (the monitoring approach assessment) a threshold of packets number, fixed (both in WD and our protocol) to 100 packets. We will deal later with this issue of threshold and judgment, and propose a more rigorous solution.

The first version of our protocol requires an ACK for each packet, while the second one uses the efficient technique of randomizing the ACK requests that reduces the overhead, especially when the misbehaving nodes rate is low as shown in figure 5.9. The decrease of the packets number with the misbehaving nodes increase in this figure for both protocols can be argued by the fact that the misbehaving increase causes more and more packets dropping during their routing, then decreases the number of packets to be monitored. The cost of this overhead decrease is a minor loss in detection efficiency, as shown in figure 5.10. But, we can clearly see in the same figure that both versions have better detection than WD. Figure 5.11 illustrates how our protocol (the two versions) decreases hugely the false detection rate compared with WD. We should point out that this metric (false detection) and the misbehaving rate are not monotonously depending. Finally, the small difference (before the ratio 30%) between our protocol and WD regarding power consumption represented in figure 5.12, is basically due to the overhead. We can also see on the same figure that the random version reduces a little bit the power consumption. Note that when the misbehaving rate is higher than 30%, we remarked the data reception ratio of the two versions of our protocol was slight below the one of the WD (due to collisions caused by two-hop ACK)[2]. Since we used CBR above UDP, the packets lost are not retransmitted. Therefore, we argue the difference between our protocol and WD when the misbehaving ratio is above 30% by the fact that these packets (lost in our protocol but not WD) are relayed until the destination in WD resulting in more consumption.

Overall, our protocol clearly outperforms the WD, with a minor cost in energy consumption. However, regarding the overhead the first version of our protocol has a considerable cost. The random version decreases hugely this cost while keeping almost the same efficiency on detection. Thus, we use it as the monitoring protocol component in our general solution. Also, we remark that despite of improving the true detection and decreasing the false one compared with WD when employing the power-control technique, there still be possibility of false detections in a realistic environment. So, a judgment policy should be defined in such a way to mitigate false accusations. In the upcoming sections we will deal with this issue, as well as the selfish nodes punishment.

[2]We remarked that the reception ratio for misbehaving ratio below 30% was almost the same for the two versions of our protocol and WD

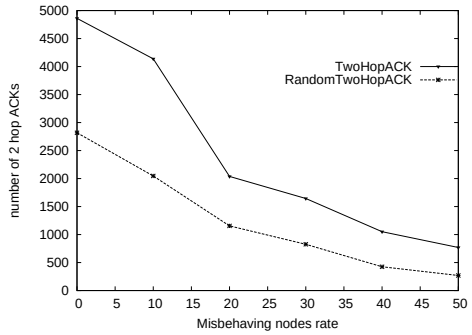


Figure 5.9: Number of two-hop ACK vs. Misbehaving rate

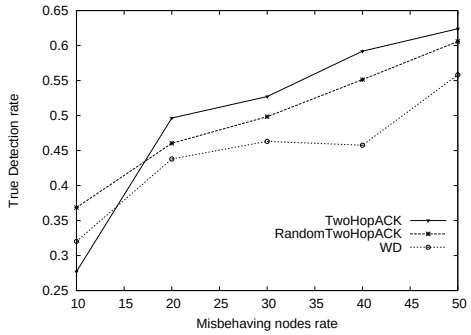


Figure 5.10: True detection vs. Misbehaving rate

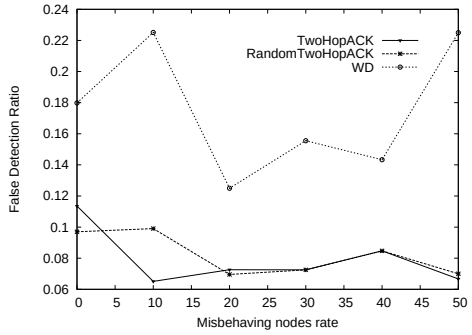


Figure 5.11: False detection vs. Misbehaving rate

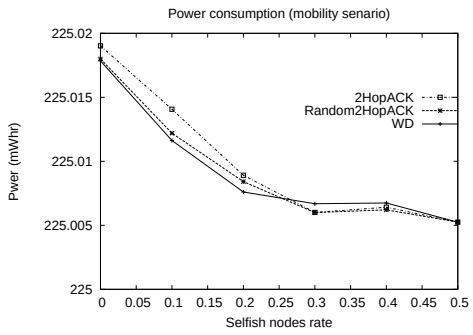


Figure 5.12: Power consumption vs. Misbehaving rate

## 5.3 Detection and Isolation

### 5.3.1 Local Detection

#### Bayesian approach for judgment

Our monitoring solution (random two-hop ACK) allows to confirm the correct forwarding of packets. Though, when a monitoring node notices that some packet has been dropped over a link it should not directly accuse the monitored one as misbehaving, since this dropping could be caused by collisions or the mobility. Indeed, a threshold of tolerance should be fixed. In this subsection, we propose a Bayesian approach allowing nodes to decide about the behavior of each other. In this approach, well-behaving of nodes improves their reputation, whereas intentional or unintentional packet dropping decreases it.

The Bayesian approach [Dav00] is a mathematical estimation method that consists of estimating a parameter the observations of which follow a Bernoulli distribution by a Beta distribution. It has the advantage of not needing a memory. That is, only

the latest observations are safeguarded, and not all the observations. The Bayesian approach for nodes reputation regarding packet forwarding in MANET has already been used by Buchegger and Le-Boudec [BLB04], but their solution requires periodic transmissions of huge control packets (as shown in the third chapter).

Since misbehaving is usually exception rather than the norm, information exchange in our solution is limited to negative impressions. It is simpler and engenders no overhead when nodes well-behave. Each node i thinks that each other node j misbehaves with a probability $\theta$, which is a random variable estimated by a Beta distribution $Beta(a, b)$. Initially, with no prior information, $\theta$ is assumed uniform in [0,1], which is idem to Beta(1,1). As observations (that follow a Bernoulli distribution with parameter $\theta$ as previously illustrated) are made, a and b are updated as follows:
$a = a + u$, $b = b + 1 - u$
where u=1 if the observation consists of a dropping, and 0 otherwise. Remember that a dropping in our solution is a lack of a required two-hop ACK (the monitor does not receive the ACK after a timeout), and that if the monitor node does not ask a two-hop ACK then the observation is considered as non-dropping. After as many observations as the decision could be made ($\theta$ could be approximated by the mathematical expectation $E(Beta(a, b))$), j will be judged. This is denoted by the decision (or stationary) point, while the number of observations is expressed by a+b. Upon reaching this point, j will be accused as misbehavior as soon as: $E(Beta(a, b)) > E_{max}$. Note that: $E(Beta(a, b)) = a/(a + b)$.

$E_{max}$ could be fixed to 0.5, or for more efficiency, it should be estimated empirically for each network as follows:

1. Make simulations with no misbehaving for different scenarios that estimate the network, and then compute E at each node.

2. Find out the maximum value in all scenarios from the decision point then consider it as $E_{max}$

In mathematical estimation methods, the decision (stationary) point is the one upon which the difference between two subsequent observations could be negligible. One usual choice is that fulfilling the following condition:
$Var(Beta(a, b)) < \epsilon$.
Such that $Var$ is the mathematical variance, and $\epsilon$ is a very small positive. Note that:
$Var(Beta(a, b)) = \frac{a \times b}{(a+b+1) \times (a+b)^2}$

In Buchegger's approach [BLB04], every node periodically broadcasts in its neighborhood its view of $\theta$ regarding all the other nodes. Nodes use this information (known as second hand information) to update their own opinion on the behavior of

the others. To decide about the acceptance of the provided information, each node performs complicated tests on the trustworthiness of the provider. The problem with this proactive solution is that it causes an important overhead, even if nodes well-behave. Our approach is rather reactive, thus no such information are exchanged. Indeed, each node performs monitoring separately and informs the others as soon as a misbehaving is *approved*, as we will see later.

**Control packets**

As for routing control packets we do not use the previous Bayesian-based approach, because it requires many observations before making any decision. Contrary to data packets, this requirement is not appropriate for control packets, since there are few packets of of such kind compared with the first one. Further, dropping control packets like RREQs (Route REQuests) and RREP (Route REPly) should not be tolerated, as it completely excludes selfish nodes from routes. Therefore, we should be more sever in judgment regarding this kind of packets.

We simply use a fixed packets number threshold, i.e. when a node observes that another node X drops more than a given *sever* threshold number of control packets it judges X as misbehaving.
Note that the strategy of dropping up to the tolerable threshold is not efficient and safe for a misbehaving, since it cannot know whether and how much the monitor will notice false observations because of channel conditions and node mobility. Hereafter we illustrate the actions to be taken when some node makes a negative judgment about another.

### 5.3.2 Misbehaving approval and isolation

After dealing with the monitoring and local judgment issues, we now tackle the post-detection phase, and propose methods allowing a detector node to confirm its detection, and to isolate the detected node.

Isolating a misbehaving node means:

- do not route packets through it, to avoid losing them
- do not forward packets for it, to punish it

A node X that judges some other node Y as misbehaving should not isolate it unilaterally, but must ensure its isolation by all nodes. This is because when X unilaterally isolates Y, the others could consider X as misbehaving when they realize that it does not forward packets for Y. In social life, a person that accuses another for a crime must show proofs. One possible way to prove the accusation is to get

witnesses against the accused person.

Identically, we suggest a testimony-based protocol (both for data and control packets) to isolate a detected node. Upon a detection, the detector informs nodes in its neighborhood about the dropper (the accused), and asks for witnesses by broadcasting a WREQ (Witness REQuest) packet. It also puts the detected node ID in a special set we call *suspicious set*. Each node receiving the WREQ investigates the issue as follows:

### Directed packets

If the packet for which the investigation is launched is a directed packet, i.e. sent to one recipient, then this latter immediately sends a *signed* WREP (Witness REPly) packet to the accuser in the following two cases:

- if its suspicious set includes the accused node
- if the accused node's misbehaving expectation is close to $E_{max}$ or the number of control packets considered dropped is close to the configured maximum threshold

Otherwise, when it has not enough experience with the accused node (Y), and if Y is its neighbor then it asks the successor of Y whether it has received packets forwarded from this latter, by sending an ACREQ (ACcusation REQuest) packet using a route that does not include it. But first, in order to avoid false accusations, the investigator should ensure that the accuser (node X) has really sent a packet to the accused node (Y) to be forwarded to the appropriate successor. One possible way to do this is to check whether such a packet has been recently overheard, using the promiscuous mode. The node also should check whether Y has sent X an ACK *just after* overhearing the data, to ensure that the former has really received the packet and that the latter is not impressing it (as it will be illustrated later). Note that unlike the watchdog, the information provided from the promiscuous mode in our solution are not used for the monitoring, but only for witnessing, aiming at improving efficiency of detections.

If Y's successor has not recently received any packet *forwarded* from Y, it sends a *signed* ACREP (ACcusation REPly) packet to the investigator, then this latter testifies for the accusation and sends the accuser node a signed WREP (Witness REPly) packet.

### Broadcast packets (RREQ)

In this case the node, if it is a neighbor of Y, merely checks whether it has recently received (respectively overhear) either any RREQ *forwarded* from Y, or a RREP orig-

inated from it. To do this, each node keeps the RREQs and RREPs it receives in a buffer for a short time.
If neither RREQ nor RREP have been received, then it testifies for the accusation and sends the accuser node a signed WREP (Witness REPly) packet. But must first ensure that X has really recently sent out a RREQ, by checking in its buffer.

When the detector collects $k$ validations from its neighbors, with at least one provided by direct experience (without asking the successor of Y), it broadcasts in the network an accusation packet (AC) containing signatures of all validating nodes. The requirement of at least one direct witness will be argued later. Each node receiving such a valid accusation isolates the guilty. Otherwise, if the detector fails to collect $k$ validations then it does not isolate the detected node, but keeps it in the suspicious set. Algorithms 6 and 7 illustrate our detection and isolation approaches.

**Algorithm 6** Algorithm executed by a node i, describing the detection approach

***When receive a notification u from the monitor regarding node j (u=1 if dropping and 0 otherwise):***

**if** (the notification is about a data packet) **then**
  $a_j = a_j + u$
  $b_j = b_j + 1 - u$
  $\theta_j = a/(a+b)$
  **if** (Decision point reached) **then**
    **if** $(\theta_j > E_{max})$ **then**
      Put j in the suspicious set
      Launch WREQ against j
    **end if**
  **end if**
**else** {the notification is about a control packet}
  $nbrcontroldrp_j = nbrcontroldrp_j + 1$
  **if** $(nbrcontroldrp_j > threshold)$ **then**
    Put j in the suspicious set
    Launch WREQ against j
  **end if**
**end if**

### 5.3.3 Analysis and discussion

#### Detection

We have proposed a Bayesian approach allowing nodes to judge one another, where each node estimates each other's misbehavior with a probability that follows a Beta(a,b)

**Algorithm 7** Algorithm executed by a node i, describing the isolation approach

***When receive a WREQ sent by X against j:***

**if** (The WREQ is about a directed packet) **then**
  **if** ($j \in$ *the suspicious set* or $\theta_j \approx E_{max}$ or $nbrcontroldrp_j \approx threshold$) **then**
    send a direct signed WREP to X
  **else**
    **if** (j is a neighbor of i) **then**
      **if** (a packet from X to j was overheard as well as the ACK) **then**
        send ACREQ toward j's successor using a route that does not include j
      **end if**
    **end if**
  **end if**
**else** {The WREQ is about a broadcast packet}
  **if** (neither RREQ nor RREP has been received from j) **then**
    send a direct signed WREP to X
  **end if**
**end if**

***When receive a ACREQ sent by Y against j where X is the previous hop:***

**if** (no packet has been recently forwarded from j including X as the previous hop) **then**
  send Y a ACREP
**end if**

***When receive a ACREP regarding X accusation:***

send X a signed undirect WREP

***When receive a WREP sent by X against j:***

**if** (WREP.type = direct) **then**
  $nbrdirectwit = nbrdirectwit + 1$
**else**
  $nbrundirectwit = nbrundirectwit + 1$
**end if**
**if** ($nbrdirectwit + nbrundirectwit = k$ and $nbrdirectwit > 0$) **then**
  broadcast AC to isolate j
**end if**

distribution, whose parameters (a,b) are updated as observations are made. As illustrated, when enough observations with regard to a given monitored node are collected such that the judgment point is reached, the monitoring node will accuse the monitored one as soon as the estimated probability ($E(Beta(a,b))$) exceeds the configured maximum tolerance, i.e. $E(Beta(a,b)) > E_{max}$.

$E(Beta(a,b)) > E_{max} \longleftrightarrow \frac{a}{a+b} > E_{max} \longleftrightarrow a > \frac{b \times E_{max}}{1-E_{max}}$:

This latter ($\frac{b \times E_{max}}{1-E_{max}}$) represents the tolerable threshold, i.e. the number of packets our solution tolerate their dropping. This tolerable threshold is proportional to b, the number of packets forwarded. The more a node forwards packets, the more its tolerable threshold increases. Forwarding packets after unintentional or intentional droppings that do not result in accusation would decrease $E$, which allows redemption before accusation. This redemption could not be possible when setting the tolerable threshold to a fixed number of packets.

Note that due to the random monitoring strategy some packets could be considered forwarded by the monitor while being indeed dropped. In other words the random strategy, solution 3, does not detect all packets dropped. Thus, the maximum number of packets a node could drop without being judged as misbehaving is the sum of the tolerable threshold and the packets not detected by the monitor. This latter is the total number of packets monitored (a+b or n) minus the number of packets detected (E(pd) computed previously). Formally speaking:

$MaxDrop = (a + b - E(pd)) + \frac{bE_{max}}{1-E_{max}}$

By replacing $a + b$ with $n$, $b$ with $(1 - \theta)n$ (as $\theta = a/(a + b) = (n - b)/n$), and E(pd) with its expression in (17), we get:

$MaxDrop = n \times (1 - \frac{\theta p_{trust}}{1-\theta(1-p_{trust})} + \frac{(1-\theta)E_{max}}{1-E_{max}}) \ldots\ldots (5.23)$

Still, remember that the strategy of dropping up to the tolerable threshold is not efficient and safe for a misbehaving, since it cannot know whether and how much the monitor will notice false observations because of channel conditions and the mobility.

Regarding control packets, misbehaving on their forwarding is more crucial. It allows a selfish node to exclude itself from routes, such as to get no data packet to forward. Further, the number of control packets is generally minor compared with data packets. All these renders the previous Bayesian approach ineffective with this kind of packets. Indeed, we have proposed to use a fixed packets number threshold. The strategy of dropping up to the tolerable threshold is inefficient for the same reason cited before (of data packets).

### Approval and Isolation

To mitigate false detections and false accusations (rumors) vulnerability, we have proposed a witness-based protocol. In this protocol, a node that detects and accuses another as misbehaving must approve its accusation before taking any measure against it. It should not isolate the assumed misbehaving unilaterally, because this could result in false detections against it. However, it could avoid routing its own packets through this node in all cases [3]. Upon the detection of a misbehaving, the detector launches locally in its neighborhood a call for witness, using a broadcast control packet that costs only one transmission. Neighbors that consider the accused node as suspicious, or those whose misbehaving estimations are close to the tolerable threshold (respectively which did not receive a RREQ if the accusation is for RREQ) testify against it by sending the requestor a signed reply packet. Those which have not enough experience with the accused node investigate this accusation and ask the successor of this latter whether it has recently received packets from it. But first, they ensure that the accuser really sent the packet to the accused to forward to the claimed successor. To do this they must be neighbors of the accused node, otherwise they do not testify. The following example illustrates the investigation: Assume three aligned nodes, A, B and C, and another node D in A's range as shown in figure 5.13. When A accuses B not forwarding packets to C and sends a call for witness, D investigates the issue. But before asking C it ensures that A has really sent the packet and B has received it, by checking the data and ACK packets overheard. This is because D could not ensure that B has received the data packet by merely overhearing it. For instance, if D is closer to A than B, A (attempting a DoS attack against B) could send the packet using a power strong enough to be overhead by D, but not by B. Requiring the ACK [4] reception from B *just after* the data ensures that B has really received the data from A. To do this, D simply safeguards the overheard packets (their headers) during a short period. This way, a node that asks the accused node's successor has no doubt that the accused node has received a data packet to forward to the successor in question. Any collision at D prevents it from testifying, but has no effect on false detections.

Upon the reception of the ACREQ, the asked node (C) replies with a signed ACREP packet if it has not received any packet from B. A coincidental collision at C at that moment, however, would result in a false reply if A is attempting a DoS attack, then in a false testimony. Nevertheless, the requirement of at least one direct witness (testifying from its direct experience) mitigates wrong accusation caused by this kind of false testimonies. The signature of the packets prevents their spoofing, thus no node could testify using the ID of another.

[3] regardless of whether the proofs are gotten or not

[4] The source of this ACK should be authenticated at the MAC layer

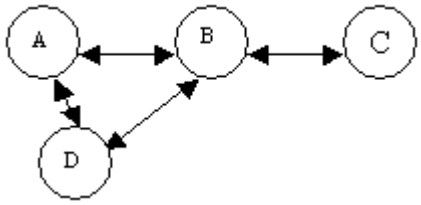


Figure 5.13: Example of connections

The accuser node has to collect $k$ different signatures to approve its accusation. Theoretically, $k-1$ is the maximum number of misbehaving nodes that could exists at any time. In practice, however, it is hard to determine such a number, so it should be fixed to strike a balance between efficiency and robustness. Setting $k$ to a high value increases the robustness of the protocol against false detections and rumors, but decreases its efficiency regarding true detections. On the other hand, a low value of $k$ allows high detections, but opens the vulnerability of rumors and increases the unintentional false detections (false positives), since $k$ nodes could collude to accuse maliciously any node, respectively wrongly accuse it. This issue related to $k$ will be investigated in the next section.

Once the accuser collects $k$ valid signatures, it broadcasts an accusation packet including all signatures through the network (by flooding) to isolate the guilty. This broadcast is costly, but it is not performed until a node is detected and approved as misbehaving. Except for the monitoring, our solution requires no overhead as long as nodes well-behave, as no opinions are exchanged periodically. This makes our solution reactive, unlike the other reputation-based solutions presented previously.

### 5.3.4 Simulation study

Hereafter, we asses the witness-based isolation method, and we study the impact of the parameter $k$ ( number of witnesses required) on our solution. To allow nodes to achieve the decision point we have raised the simulation time to 1500 seconds, and the number of CBR sessions to 23 as well. Still, all the other parameters have been maintained as described previously (when assessing the monitor approach). We compare two versions, the first with one witness required for the isolation and the second with two, respectively denoted by one_witness and two_witness. Each of which uses the random two-hop ACK for monitoring with $p_{trust}$ fixed to 0.5 (as argued previously), and the Bayesian approach for accusation with $E_{max}$ fixed to 0.5 and $\epsilon$ to 0.05. The comparison is performed regarding true and false isolation rate, i.e. the rate of nodes correctly (respectively wrongly) *isolated*, in different misbehaving rates. Note that misbehaving nodes and CBR sessions have been fixed in such a way to judge

(correctly or wrongly) all the monitored nodes before the end of the simulation. That means for each monitored node at least one of the nodes monitoring it reaches the decision point, but no matter whether it judges it correctly or not.

As illustrated in figure 5.14 and 5.15 two_witness considerably improves (decreases) the false positive rate, especially when the misbehaving rate exceeds 10% but losses a little bit on the true positive rate compared with the one_witness version. False detections in our scenarios are due to the mobility and packet collisions that increase unintentional dropping, thus the likelihood of false accusations. The second version has unacceptable values with respect of this metric, particulary when the misbehaving rate is low. Two-witness mitigates this shortcoming, and also cuts down the vulnerability of collusive false accusation attack compared with one-witness, since more than two nodes have to collude to isolate a node. We point out that false detections for high misbehaving rates are low because more nodes misbehave more packets are likely to be dropped earlier through their route, which reduces the number of observations, thus the number of droppers reaching the threshold.

The parameter $k$ could be increased to make the solution less tolerant on false detection and false accusation attacks, but should depend on the connectivity of nodes to not lose efficiency on detections. In networks with low connectivity, it should not be increased lots, because this would prevent nodes from finding witnesses, and consequently decreases the efficiency in detection. In our scenarios, we remarked that fixing $k$ to 3 was not efficient at all.

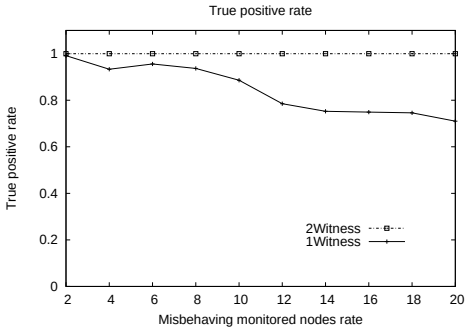


Figure 5.14: True detection vs. Misbehaving rate

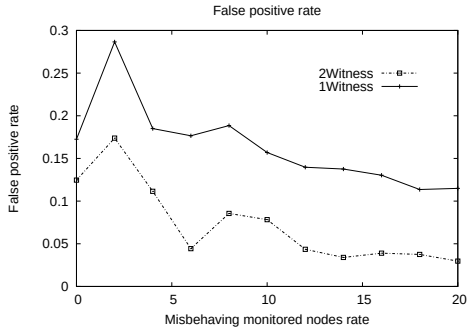


Figure 5.15: False detection vs. Misbehaving rate

## 5.4 Architecture of the general modular solution

After proposing solutions for monitoring, judging, and isolating selfish nodes, now we gather all these proposals in a generale modular solution. As illustrated in figure

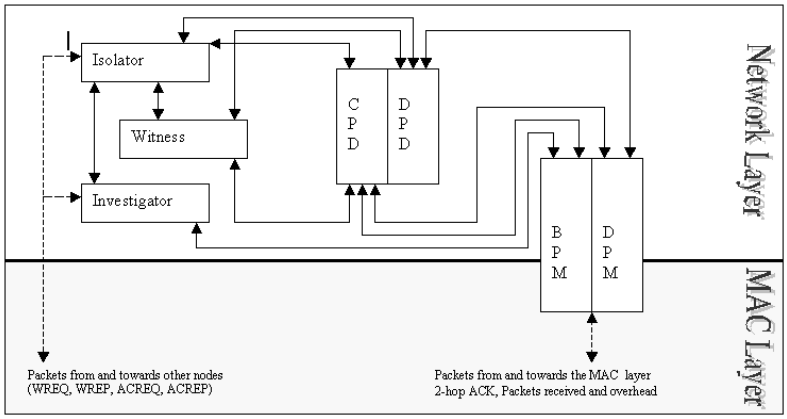


Figure 5.16: General solution framework

5.16, we divided the solution into five modules: The first one is the monitor, that implements the random two-hop ACK monitoring technique and the monitoring approach for broadcast packets. The second module is the detector which is in charge of detecting the misbehaving of the monitored nodes by implementing the Bayesian approach, and using the information provided by the previous module. The isolator is our third module, it implements the witness-based approach for detection approval, in cooperation with the investigator. This fourth module investigates accusations before witnessing when the node has not enough experience with the accused. Finally, the witness is the last module, it responses to testimony requests of the isolator. These modules in each node work together and cooperate in order to monitor the forwarding of each successor and make accurate judgment about it. As soon as a monitored node is considered misbehaving, the detector proceed to a global isolation against the detected.

### 5.4.1 Monitor

This module monitors the forwarding of both directed and broadcast packets. Since we have proposed a separate solution for each kind of packets, this module is composed of two submodules, namely directed packets monitor and broadcast packets monitor. Also note that contrary to all the other modules located in the network layer this one is designed across the network and the MAC layers, as it implements the monitoring solution which relies on this cross-layer design.

#### Directed Packets Monitor (DPM)

This module deals with packets directed to one recipient, this kind includes data packets, route reply (RREP), and route error (RERR), whereas broadcast packets

are left for the next submodule. It implements the random two-hop ACK solution, and informs the detector about all the detected packets dropped, as well as the ones forwarded (or considered forwarded when not requesting two-hop ACKs).

#### Broadcast Packets Monitor (BPM)

This submodule deals with the broadcast packets (RREQ), and implements the solution we proposed for this kind of packets. It informs the detector module about all negative observations (packets dropped), as well as the positive ones.

The BPM keeps track of the packets received and overheard in the promiscuous mode at the MAC level, and safeguards them during a short time for usage by the investigator module. The packets capturing needs to be performed at the MAC layer since both data and MAC ACKs are involved, which explains the cross-layer design of BMP.

### 5.4.2 Detector

The detector is in charge of locally judging the behavior of the monitored nodes, using the information provided by the monitor. Also, this module is composed of two part. The first one focuses on data packet, while the second on routing control packets.

#### Data Packets Detector (DPD)

The DPD implements the Bayesian approach enabling nodes to decide about the behavior of each other. The notifications it uses are provided from DPM. And as soon as a node is judged as misbehaving, the DPD informs the isolator that tries to isolate the misbehaving all over the network.

#### Control Packets Detector (CPD)

This submodule deals with control packets, for which the Bayesian approach is ineffective (as argued before). As depicted in figure 5.16 CPD receives notifications from both DPM and BPM, because it deals with both directed (RREP and RERR) and broadcast (RREQ) packets. When the CPD observes that some node drops more than a given *sever* threshold number of packets, it judges it as misbehaving and informs the isolator.

### 5.4.3 Isolator

When the isolator of some node X receives a notification about a misbehavior judgment of other node Y from the detector module (DPD or CPD) it first informs the witness module (which puts Y in the suspicious set), then tries to get proofs allowing

it to isolate Y in the whole network using the protocol we proposed previously. That is, it broadcasts in its neighborhood a WREQ. The isolator of each node receiving the WREQ interrogates the witness module, if this latter witnesses against Y then the isolator immediately sends a *signed* WREP (Witness REPly) packet to node X's isolator. Otherwise it informs the investigator, which tries to investigate the issue and to provide the isolator with an indirect testimony. Note that the isolator informs the investigator about the nature of the last packet causing the accusation. When the isolator of node X collects $k$ validations from its neighbors with at least one provided by direct experience (without investigation), it broadcasts in the network an accusation packet (AC) containing signatures of all validating nodes.

### 5.4.4 Witness

This module is responsible for providing testimonies against suspicious nodes. As we mentioned, when the witness receives a notification about a misbehaving node from the local isolator it puts its ID in the suspicious set. When interrogated by the isolator (after this later receives a WREQ from another isolator), the witness testifies against the accused if the conditions of direct testimony (5.3.2) are held.

### 5.4.5 Investigator

The investigator is used when the witness does not provide a direct testimony, due to lack of enough experience with the accused node. Upon demand from the isolator, the investigator investigates an accusation according to the nature of the packet for which the investigation is, following the protocol described in 5.3. It uses information collected by BPM.

## 5.5 Conclusion

In this chapter we have proposed solutions to monitor, detect, and isolate selfish nodes that do not forward packets in mobile ad hoc networks. For monitoring, we proposed the efficient technique of two-hop ACK and reduced its cost firstly by taking advantage of the cross-layer design and exploiting the MAC layer ACK, and secondly by suggesting the random requesting. The analysis and simulation results show that the random two-hop ACK is all but as efficient as the ordinary two-hop ACK in high true and low false detection, while hugely reducing the overhead. For local judgment we have proposed a Bayesian approach, that allows redemption before making decisions, and decreases false accusations due to channel conditions and node mobility. However, we have been less tolerant with control packets whose dropping is crucial. Compared with the trust manager of [BLB04], our solution does not use any periodic packet exchange, thus requires no overhead as long as nodes well-behave. Once a node is judged locally as misbehaving by some other node, this latter must

approve its detection to ensure the isolation by all nodes. For this end, we proposed a testimony-based protocol, that enforces the detector to collect at least $k$ witnesses before isolating the detected node. Fixing $k$ is a trade-off problem, high values mitigate rumors aiming DoS attack as well false detections (especially for control packets with which we have been more sever) but reduces the efficiency on detection, contrary to low values. In our simulation, the protocol with two witnesses showed considerable improvement regarding false accusation while keeping the true detection good enough. This parameter could be risen to ensure more robustness, but should depend on the connectivity to keep efficiency. For instance, fixing $k$ to 3 is not efficient in our scenarios at all, as it reduces dramatically the true positives.

In our simulation, the nodes misbehaving was for data packets. As a perspective, we plane to make more investigations with misbehaving on control packets, thereby evaluating (analytically and by simulation) the solutions we proposed for control packets. Making a rigorous definition for the threshold of dropped control packets number also represents a perspective of our solution.

# General conclusion and perspectives

In this thesis we dealt with security in mobile ad hoc networks, particularly with an emergent problem caused by the energy limitation of mobile devices used in this kind of networks. We first surveyed several security problems related to different network layers, and classified and discussed a variety of solutions proposed thus far. These problems range from misbehaving on channel access, to secure routing, to key management ( both public key and private key infrastructures) and intrusion detection systems (IDSs). We showed that due to the specificities of ad hoc networks, each field involves several challenges and research trends. As the ad hoc network is totally infrastructureless, every node is expected to share its resources and forward packets originated from other nodes. Due to its battery limitation, a node might misbehave by dropping packets of others while using their resources to get its own packets relied toward remote nodes. We gave more attention to this emergent security problem that does not exist in traditional networks, termed nodes selfishness on packet forwarding, which threatens the availability. We first tried to reduce the anxiety of nodes about the battery lifetime, and we dealt with power-awareness in routing protocol. We proposed a DSR-based power-aware routing protocol that consider both the transmission power minimization when employing the high efficient technique of transmission power control, and the node battery state to fairly distribute the information load over routers. The simulation results showed that our protocol improves the battery lifetime, and especially the difference between nodes with respect to this performance criterion. Still, the problem of energy is far from being eliminated, and nodes remain concerned about their batteries. Therefore, a security solution to tackle the nodes misbehaving on packet forwarding is mandatory.

We surveyed the solutions that treat nodes selfish misbehavior separately in the third section. We notably realized from this study that almost all the detective solutions employ the watchdog in their monitoring technique. Indeed, this later suffers from many drawbacks, especially when using the power-control technique. To enable the employment of such a power-efficient technique with no effect on monitoring, we proposed in the last chapter a new monitoring solution, whose cost has been gradually decreased. We mathematically analyzed the efficiency of the last solution (random

two-hop ACK), and its cost vs. our first solution, whose correctness has been proved (with some assumptions). The comparative study illustrates how the random two-hop ACK decreases both the communication and computation overhead while keeping the efficiency good enough. The simulation study confirm these results in a more realistic scenarios with nodes mobility and packets collisions, and shows how our protocol clearly outperforms the watchdog with respect to the efficiency in detection. We completed the solution by dealing with the judgment and isolation issues. Regarding local judgment, we proposed a Bayesian approach that allows redemption before making decisions, and decreases false accusations due to channel conditions and node mobility. Our Bayesian approach does not use any periodic packets exchange, thus requires no overhead as long as nodes well-behave. Once a node is judged locally as misbehaving by some other node, this latter must approve its detection to ensure the misbehaving isolation by all nodes. This is to be achieved by executing the testimony-based protocol we proposed, which enforces the detector to collect at least $k$ witnesses before isolating the detected node. Fixing $k$ is a trade-off problem, high values mitigates rumors aiming DoS attacks as well as false detections (especially for control packets with which we have been more sever) but reduces the efficiency on detections, contrary to low values. Our simulation results show how that fixing $k$ to 2 provided better results than the ones with $k = 1$ in our scenarios. However, we should point out that the value of $k$ should depend on the network connectivity and configuration, as there is no value that is fit for all situations.

The selfish misbehavior is not only limited in data packets. A selfish node might simply drop control packets during the route discovery procedure to exclude itself from routes, and prevent getting any packet to forward. For broadcast (RREQ) packets we proposed another monitoring technique, since the two-hop ACK is not effective with this kind of packets. Moreover, the Bayesian approach is also ineffective with all kinds of control packets. Thus, we have been less tolerant with this kind of packets whose dropping is crucial. Investigating the control packet dropping misbehavior and the performance of our solution represents a perspective of this work. Also, note that our protocol could be integrated with any source routing protocol, including the secure ones presented in chapter 2. This would make them robust against the black-hole attack (also presented in chapter 2), in addition to securing them against selfish misbehavior. Moreover, it can be used along with the power control technique. However, note that like all the current power-aware protocols, ours (presented in chapter 3) relies on the information provided from nodes to compute route metrics and select routes, and completely trusts such information. This makes the direct integration of our solution with a power-aware routing protocol vulnerable, as a node could simply in this case claim a low battery state and/or a high transmission power to exclude itself from routes. This problem cannot be solved by simply applying some solutions proposed for secure routing. Because although the secure routing strategies could ensure authenticity, and prevent the construction of wrong routes, none of

them prevent a node from claiming false information regarding itself. Integrating the solution with a power-aware protocol (like the one we proposed in chapter two) remains in our perspectives.

# Bibliography


[AHNRR02] Baruch Awerbuch, David Holmer, Cristina Nita-Rotaru, and Herbert Rubens. An on-demand secure routing protocol resilient to byzantine failures. In *ACM Workshop on Wireless Security (WiSe'02)*, Atlanta, Georgia, USA, September 2002.

[AP00] N. Asokan and P.Ginzboorg. Key agreement in ad hoc networks. *Computer Communications*, 23(17):1627–1637, 2000.

[ASS03] Farooq Anjum, Dhanant Subhadrabandhu, and Saswati Sarkar. Signature based intrusion detection for wireless ad-hoc networks: A comparative study of various routing protocols. In *Vehicular Technology Conference, Wireless Security Symposium*, Orlando, Florida, USA, October 2003.

[Bad98] Nadjib Badache. La mobilite dans les systemes repartis. *Technique et science informatiques*, 17(8):969–997, Janvier 1998.

[BCSW98] S. Basagni, I. Chlamtac, V. R. Syrotiuk, and B. A. Woodward. A distance routing effect algorithm for mobility DREAM. In *The ACM Conference on Mobile Computing MOBICOM'98*, pages 76–84, Dallas, Texas, USA, October 1998.

[BDD03] Nadjib Badache, Djamel Djenouri, and Abdelouahid Dehab. Mobility impact on mobile ad hoc networks. In *ACS/IEEE International Conference on Computer Systems and Applications (AICCSA'03) conference*, Tunis, Tunisia, July 2003.

[BGdMT98] Gunter Belch, Stefan Greiner, Hermann de Meer, and Kishor S. Trivedi. *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications*. John Wiley and Sons, Inc., 1998.

[BH01] L. Buttyan and JP. Hubaux. Nuglets: a virtual currency to stimulate cooperation in self-organized mobile ad hoc networks. Technical Report DSC/2001/001, Swiss Federal Institute of Technology, Lausanne, Switzerland, January 2001.

[BH03] L. Buttyan and J.-P. Hubaux. Stimulating cooperation in self-organizing mobile ad hoc networks. *ACM/Kluwer Mobile Networks and Applications*, 8(5), October 2003.

[BLB02] Sonja Buchegger and JeanYves Le-Boudec. Performance analysis of the confidant, protocol cooperation of nodes fairness in dynamic ad hoc networks. In *Third ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'02)*, pages 80–91, Lausanne, Switzerland, June 2002.

[BLB04] Sonja Buchegger and Jean-Yves Le-Boudec. A robust reputation system for p2p and mobile ad-hoc networks. In *Second Workshop on the Economics of Peer-to-Peer Systems*, Harvard university, Cambridge, MA, USA, June 2004.

[BM92] Steven M. Bellovin and Michael Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *IEEE Symposium on Security and Privacy*, pages 72–84, May 1992.

[BM02] Suman Banerjee and Archan Misra. Minimum energy paths for reliable communication in multi-hop wireless networks. In *The 3rd ACM international symposium on Mobile ad hoc networking and computing MobiHoc'02*, pages 146–156, 2002.

[BW98] Klaus Becker and Uta Wille. Communication complexity of group key distribution. In *The 5th ACM conference on Computer and communications security*, pages 1–6, 1998.

[CBH03] Srdjan Capkun, Levente Buttyan, and Jean-Pierre Hubaux. Self-organized public-key management for mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, 2(1):52–64, January 2003.

[CE95] M.S Corson and A. Ephremides. A distributed routing algorithm for mobile wireless networks. *Wireless Networks*, 1(1):61–81, 1995.

[CG98] Tsu-Wei Chen and Mario Gerla. Global state routing: A new routing scheme for ad-hoc wireless networks. In *IEEE International Conference on Communications (ICC'98)*, pages 171–175, Atlanta, GA, USA, June 1998.

[CGM05] Marco Conti, Enrico Gregori, and Gaia Maselli. Improving the performability of data transfer in mobile ad hoc networks. In *the Second IEEE International Conference on Sensor and Ad Hoc Communications and Networks (SECON'05)*, Santa Clara, CA, USA, September 2005.

[CH03] Tzu-Chiang Chiang and Yueh-Min Huang. Group keys and the multicast security in ad hoc networks. In *The first International Workshop on Wireless Security and Privacy (WiSPr'03)*, 2003.

[CM02] Claude Castelluccia and Gabriel Montenegro. Protecting AODV against impersonation attacks. *ACM SIGMOBILE Mobile Computing and Communications Review MC2R*, 6(3):108–109, July 2002.

[CR00] C. Chiasserini and R. Rao. Routing protocols to maximize battery efficiency. In *The Military Communications Conference MILCOM'00*, pages 496–500, 2000.

[CWLG97] C. Chiang, H-K Wu, Winston Liu, and Mario Gerla. Routing in clustered multihop, mobile wireless networks. In *The IEEE Singapore International Conference on Networks*, pages 197–211, 1997.

[Dav00] Anthony Davison. *Bayesian Models, Chapter 11 in Manuscript*. Springer, 2000.

[DB02a] Sheetalkumar Doshi and Timothy X Brown. Design considerations for an on-demand minimum energy routing protocol for a wireless ad hoc network. In *International Conference on Communications (ICC02)*, 2002.

[DB02b] Sheetalkumar Doshi and Timothy X Brown. Minimum energy routing schemes for a wireless ad hoc network, 2002.

[DB04] Djamel Djenouri and Nadjib Badache. Simulation performance evaluation of an energy efficient routing protocol for mobile ad hoc networks. In *IEEE International Conference on Pervasive Services (ICPS'04)*, American University of Beirut (AUB), Lebanon, July 2004.

[DB05a] Djamel Djenouri and Nadjib Badache. New approach for selfish nodes detection in mobile ad hoc networks. In *the first IEEE/Creat-net Workshop on Integration of Security and Quality of Service (SecQoS'05), workshop of SecureComm'05*, pages 282–288, Athens, Greece, September 2005.

[DB05b] Djamel Djenouri and Nadjib Badache. A novel approach for selfish nodes detection in manets: Proposal and petri nets based modeling. In *8th IEEE International Conference on Telecommunications (ConTel'05)*, pages 569–574, Zagreb, Croatia, June 2005.

[DB06a] Djamel Djenouri and Nadjib Badache. Cross-layer approach to detect data packet droppers in mobile ad-hoc networks. In *The first International Workshop On Self-organized systems IWSOS'06*, number 4124

in LNCS, pages 163–176, Passau, Germany, September 2006. Springer-Verlag GmbH.

[DB06b] Djamel Djenouri and Nadjib Badache. New power-aware routing for mobile ad hoc networks. *The International Journal of Ad Hoc and Ubiquitous Computing (Inderscience)*, 1(3):126–136, 2006.

[DB06c] Djamel Djenouri and Nadjib Badache. Testimony-based isolation: New approach to overcome packet dropping attacks in manet. In *The 7th Postgraduate Symposium on Convergence of Telecommunications, Networking and Broadcasting*, pages 114–119, John Moors University, Liverpool, UK, June 2006.

[DBB02] Sheetalkumar Doshi, Shweta Bhandare, and Timothy X Brown. An on-demand minimum energy routing protocol for a wireless ad hoc network. *SIGMOBILE Mobile Computing Communication Revue (MC2R)*, 6(3):50–66, 2002.

[DBar] Djamel Djenouri and Nadjib Badache. A gradual solution to detect selfish nodes in mobile ad hoc networks. *The International Journal of Wireless and Mobile Computing (Inderscience)*, 2(1), 2007 (to appear).

[DD96] B.Johnson David and A.Maltz David. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, volume 353, pages 153–181. Kluwer Academic, 1996.

[DDB06] Djamel Djenouri, Abdelouahid Derhab, and Nadjib Badache. Ad hoc networks routing protocols and mobility. *International Arab jornal of Information Technology*, 3(2):126–133, 2006.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[DKB05] Djamel Djenouri, Lyes Khalladi, and Nadjib Badache. A survey of security issues in mobile ad hoc and sensor networks. *IEEE Communications Surveys*, 7(4):2–28, 2005.

[DMF05] D.Djenouri, M.Bouzenada, and F.Khiyat. Mobility and mac protocols in manets. In *The International Arab Conference on Information Technology (ACIT'05)*, pages 527–533, El-Isra Private university, Amman, Jordan, December 2005.

[DMFL05] D.Djenouri, M.Bouzenada, F.Khayat, and L.Khelladi. Scalability of wireless mac protocols in mobile ad hoc networks. In *The First IEEE International Computer Systems and Information Technology Conference 2005 (ICSIT)*, Algiers, Algeria, July 2005.

[DMM03] S. Das, B. Manoj, and C. Murthy. A dynamic core based multicast routing protocol for ad hoc wireless networks. In *Mobile Ad Hoc Networks Workshop (MADNET)*, Sophia-Antipolis, France, 2003.

[DOMB05] Djamel Djenouri, Nabil Ouali, Ahmed Mahmoudi, and Nadjib Badache. Random feedbacks for selfish nodes detection in mobile ad hoc networks. In *The 5th IEEE International Workshop on IP Operations and Management, IPOM'05*, number 3751 in LNCS, pages 68–75, Barcelona, Spain, October 2005. Springer-Verlag GmbH.

[dPML+03] R. di Pietro, L. V. Mancini, Y. W. Law, S. Etalle, and P. Havinga. A directed diffusion-based secure multicast scheme for wireless sensor networks. In *The first International Workshop on Wireless Security and Privacy (WiSPr'03)*, 2003.

[DR92] D.Duchamp and N. Reynolds. Measured performance of wireless lan. In *17th IEEE Conference on Local Computer Networks*, pages 494–499, Minneapolice, MN, USA, september 1992.

[DRWT97] R. Dube, C. D. Rais, K. Y. Wang, and S. K. Tripathi. Signal stability based adaptive routing for ad hoc mobile networks. *IEEE Personal Communications*, 4(1):36–45, 1997.

[GCNB03] Javier Gomez, Andrew T. Campbell, Mahmoud Naghshineh, and Chatschik Bisdikian. Paro: supporting dynamic power controlled routing in wireless ad hoc networks. *Wireless Networks*, 9(5):443–460, 2003.

[GLAM99] J. Garcia-Luna-Aceves and E. Madruga. The core-assisted mesh protocol. *IEEE Journal on Selected Areas in Communications*, 17(8):13801994, 1999.

[Gol97] S. Gold. A PSPICE macromodel for lithium-ion batteries. In *The 12th Annual Battery Conference on Applications and Advances*, pages 9–15, 1997.

[G.W83] G.W.BRAMS. *Reseaux de petri: Theorie et pratique*. Edition masson, 1983.

[HBC01] Jean-Pierre Hubaux, Levente Buttyan, and Srdan Capkun. The quest for security in mobile ad hoc networks. In *MobiHoc '01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking and computing*, pages 146–155. ACM Press, 2001.

[HFLY03] Y. Huang, W. Fan, W. Lee, and P. Yu. Cross-feature analysis for detecting ad-hoc routing anomalies. In *the 23rd International Conference on Distributed Computing Systems*, Providence, RI, May 2003.

[HJP02] Yih-Chun Hu, David B. Johnson, and Adrian Perrig. Secure efficient distance vector routing in mobilewireless ad hoc networks. In *Fourth IEEE Workshop on Mobile Computing Systems and Applications WMCSA 02*, June 2002.

[HJP03] Yih-Chun Hu, David Johnson, and Adrian Perrig. SEAD: secure efficient distance vector routing for mobile wireless ad hoc networks. *Ad hoc networks (Elsvier)*, 1(1):175–192, 2003.

[HP04] Yih-Chun Hu and Adrian Perrig. A survey of secure wireless ad hoc routing. *IEEE Security and Privacy*, 2(3):28–39, 2004.

[HPJ02] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. In *The Eighth Annual International Conference on Mobile Computing and Networking MobiCom 2002*, pages 12–23, september 2002.

[HPJ03a] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Packet leashes: A defense against wormhole attacks in wireless ad hoc networks. In *Twenty-Second Annual IEEE Joint Conference of the on Computer Communications and Networking (INFOCOM 2003)*, April 2003.

[HPJ03b] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Rushing attacks and defense in wireless ad hoc network routing protocols. In *The ACM workshop on Wireless Security WiSe 2003*, San diego, CA, USA, september 2003.

[ICP+99] Atsushi Iwata, Ching-Chuan Chiang, Guangyu Pei, Mario Gerla, and Tsu wei Chen. Scalable routing strategies for ad hoc wireless networks. *IEEE Journal on Selected Areas in Communications, Special Issue on Ad-Hoc Networks*, 17(8):1369–1379, 1999.

[IET06] IETF manet working group. http://www.monarch.cs.rice.edu/ietf.html, 2006.

[IKP95] K. Ilgun, R. A. Kemmerer, and P. A. Porras. State transition analysis: A rule-based intrusion detection approach. *IEEE Transactions on Software Engineering*, 21(3):181–199, March 1995.

[JJ01] J. Jetcheva and D. Johnson. Adaptive demand-driven multicast routing in multi-hop wireless ad hoc networks. In *MOBIHOC*, page 3344, 2001.

[JJL99] Mingliang Jiang and Y.C. Tay Jinyang Li. Cluster based routing protocol CBRP, IETF draft, 1999.

[JLB04] Raja Jurdak, Cristina Videira Lopes, and Pierre Baldi. A survey, classification and comparative analysis of medium access control protocols for ad hoc networks. *IEEE Communications Surveys*, 6(1), 2004.

[JNL99] M. Joa-Ng and I.-T. Lu. A peer-to-peer zone-based two-level link state routing for mobile ad hoc networks. *IEEE Journal on Selected Areas in Communications, Special Issue on Ad-Hoc Networks*, 17(8):1415–1425, August 1999.

[JV05] Eun-Sun Jung and Nitin H. Vaidya. Power aware routing using power control in ad hoc networks. *SIGMOBILE Mobile Computing Communication Revue (MC2R)*, 9(3):7–18, 2005.

[KG03] Oleg Kachirski and Ratan Guha. Effective intrusion detection using multiple sensors in wireless ad hoc networks. In *36th Annual Hawaii International Conference on System Sciences (HICSS'03)*, Big Island, Hawaii, January 2003.

[KKWS04] Frank Kargl, Andreas Klenk, Michael Weber, and Stefan Schlott. Advanced detection of selfish or malicious nodes in ad hoc networks. In *1st European Workshop on Security in Ad-Hoc and Sensor Networks, ESAS'04*, Heidelberg, Germany, Augest 5-6 2004.

[KLNY03] T. Kaya, G. Lin, G. Noubir, and A. Yilmaz. Secure multicast groups on ad hoc networks. In *The first ACM Workshop on Security of Ad hoc and Sensor Networks SASN'03*, Fairfax, VA, USA, October 2003.

[KML04] Marwan Krunz, Alaa Muqattash, and S.J. Lee. Transmission power control in wireless ad hoc networks: Challenges, solutions, and open issues. *IEEE Network Magazine*, 18(5):8–14, 2004.

[KT75] L. Kleinrock and F. A. Tobagi. Carrier sense multiple-access models and their throughput-delay characteristics. *IEEE Transaction on Communications*, 23(12):1400–1416, 1975.

[KV00] Young-Bae Ko and Nitin H. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. *Wireless Networks*, 6(4):307–321, 2000.

[KV03] Pradeep Kyasanur and Nitin H. Vaidya. Detection and handling of mac layer misbehavior in wireless networks. In *IEEE International Conference on Dependable Systems and Networks (DSN'03)*, San Francisco, California, Jun 2003.

[Lee03] Yian Huang Wenke Lee. A cooperative intrusion detection system for ad hoc networks. In *The 1st ACM workshop on Security of ad hoc and sensor networks SASN'03*, pages 135–147, Fairfax, Virginia, USA, 2003.

[LGC99] S. Lee, M. Gerla, and C. Chiang. On demand multicast routing protocol. In *IEEE WCNC*, page 12981302, August 1999.

[LK00] S. Lee and C. Kim. Neighbor supporting ad hoc multicast routing protocol. In *ACM MOBIHOC*, page 3750, August 2000.

[LN02] G. Lin and G. Noubir. Secure multicast over multihop wireless ad hoc networks. In *ACM MOBIHOC*, June 2002.

[LNS03] Donggang Liu, Peng Ning, and Kun Sun. Efficient self-healing group key distribution with revocation capability. In *The 10 th ACM Conference on Computer and Communications Security CCS*, pages 231–240, Washingtion DC, USA, October 27-30 2003.

[LP03] Loukas Lazos and Radha Poovendran. Energy-aware secure multicast communication in ad-hoc networks using geographic location information. In *IEEE International Conference on Acoustics Speech and Signal Processing*, Hong Kong, China, April 6-10 2003.

[LPH03] Loukas Lazos, Radha Poovendran, and Gregory H.Cirincione. Location-aware secure wireless multicast in ad-hoc networks under heterogeneous path-loss. Technical Report UWEETR-2003-0012, University of Washington, Electrical Engineering departement, 2003.

[LTG+92] T. Lunt, A. Tamaru, F. Gilham, R. Jagannathan, P. Neumann, H. Javitz, A. Valdes, and T. Garvey. A real-time intrusion detection expert system (ides). Technical report, Computer Science Laboratory, SRI International, Menlo Park, California, February 1992.

[LWF02] Xiang-Yang Li, Yu Wang, and Ophir Frieder. Efficient hybrid key agreement protocol for wireless ad hoc networks. In *IEEE International Conference on Computer Communications and Networks ICCCN'02*, Miami, FL, USA, 2002.

[MAH00] Silja Maki, Tuomas Aura, and Maarit Hietalahti. Robust membership management for ad-hoc groups. In *The 5th Nordic Workshop on Secure IT Systems (NORDSEC'20000)*, 2000.

[MGLB00] S. Marti, T. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *ACM Conference on Mobile Computing and Networking, MOBICOM 2000*, pages 255–65, Boston, MA, USA, 2000.

[MHJ98] Broch D.A. Maltz, Yih-Chun Hu, and J. Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *The*

*fourth Annual ACM/IEEE International Conference On Mobile Computing And Networking MOBICOM'98*, pages 85–97, 1998.

[MM02a] Pietro Michiardi and Rafik Molva. Simulation-based analysis of security exposures in mobile ad hoc networks. In *The European Wireless Conference 2002*, Florence, Italy, February 2002.

[MM02b] Pietro Michiardi and Refik Molva. Core: A collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks. In *The 6th IFIP Communication and Multimedia Security Conference CMS'02*, Portoroz, Slovenia, September 2002.

[MR03] Hugo Miranda and Luís Rodrigues. Friends and foes: Preventing selshness in open mobile ad hoc networks. In *The 23rd IEEE International Conference on Distributed Computing Systems ICDCS'03*, pages 440–445, Providence, RI, USA, May 2003.

[MRR80] J. McQuillan, I. Richier, and E. Rosen. The new routing algorithm for the arpanet. *IEEE Transactions on Communications*, 28(5):711–719, 1980.

[MW77] J.M. McQuillan and D.C. Walden. The arpa network design decisions. *Computer Networks*, 1(5):243–289, 1977.

[MY05] Chi Ma and Yuanyuan Yang. A prioritized battery-aware routing protocol for wireless ad hoc networks. In *MSWiM '05: Proceedings of the 8th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems*, pages 45–52. ACM Press, 2005.

[Mye91] R.B. Myerson. *Game Theory: Analysis of Conflict*. Harvard University Press, Cambridge, Mass, 1991.

[NNL02] Dalit Naor, Moni Naor, and Jeffery Lotspiech. Revocation and tracing schemes for stateless receivers. *Electronic Colloquium on Computational Complexity (ECCC)*, 9(43), 2002.

[O.B85] James O.Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer, second edition, 1985.

[OH04] Masakazu Ono and Hiroaki Higaki. Power control routing for high throughput in mobile ad hoc networks. In *International Conference on Wireless Networks*, pages 584–589, Las Vegas, Nevada, USA, June 2004.

[OKS99] T. Ozaki, J. Kim, and T. Suda. Bandwidth efficient multicast routing protocol for ad hoc networks. In *IEEE ICCCN*, page 1017, October 1999.

[OY91] R. Ostrovsky and M. Yung. How to withstand mobile virus attacks. In *the 10th ACM Annual Symposium on Principles of Distributed Computing (PODC91)*, pages 51–59, Montreal, Quebec, Canada, Augest 1991.

[PB94] Charles.E. Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computer. In *ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244, London, UK, August 1994.

[PC97] V. D. Park and M. S. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *The IEEE conference on Computer Communications and Networking INFOCOM '97*, pages 1405–1413, Aprile 1997.

[PGC00] TGuangyu Pei, Mario Gerla, and Tsu-Wei Chen. Fisheye state routing in mobile ad hoc networks. In *IEEE International Conference on Communications (ICC'00)*, pages 70–74, New Orleans, LA, USA, June 2000.

[PH02] Panagiotis Papadimitratos and Zygmunt Haas. Secure routing for mobile ad hoc networks. In *In SCS Communication Networks and Distributed Systems Modeling and Simulation Conference CNDS*, San Antonio, Texas, January 2002.

[PH03] Panagiotis Papadimitratos and Zygmunt J. Haas. Secure data transmission in mobile ad hoc networks. In *ACM MOBICOM Wireless Security Workshop (WiSe'03)*, San Diego, California, USA, September 2003.

[PR99] Charles.E. Perkins and Elizabeth.M. Royer. Ad hoc on demand distance vector (AODV) algorithm. In *the 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'99)*, pages 90–100, 1999.

[PST01] Adrian Perrig, Dawn Xiaodong Song, and J. D. Tygar. ELK, a new protocol for efficient large-group key distribution. In *IEEE Symposium on Security and Privacy*, pages 247–262, 2001.

[PSW+01] Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, and J. D. Tygar. Spins: Security protocols for sensor networks. In *Seventh Annual ACM International Conference on Mobile Computing and Networks (MobiCom 2001)*, Rome, Italy, July 2001.

[Rab89] M.O. Rabin. Efficient dispersal of information for security, load balancing, and fault tolerance. *Journal of ACM*, 36(2):335–348, 1989.

[RGAM90] R.Heady, G.Luger, A.Maccabe, and M.Servilla. The architecture of a network level intrusion detection system. Technical report, Computer Science Department, University of New Mexico, August 1990.

[RP99] E. Royer and C. Perkins. Multicas using ad hoc on demand distance vector routing. In *ACM MOBICOM*, page 207218, 1999.

[RVW02] Daler Rakhmatov, Sarma Vrudhula, and Debrah A. Wallach. Battery lifetime prediction for energy-aware computing. In *International Symposium on low power electrics and design*, pages 154–159, Monterey, California, USA, 2002.

[SA99] Frank Stajano and Ross Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In *7th International Security Protocols Workshop*, Cambridge, UK, April 1999.

[SDL+02] Kimaya Sanzgiri, Bridget Dahill, Brian Neil Levine, Clay Shields, and Elizabeth Belding-Royer. A secure routing protocol for ad hoc networks. In *The 10th IEEE International Conference on Network Protocols (ICNP 02)*, November 2002.

[S.G02] Matthew S.Gast. *802.11 Wireless Networks*. O'Reilly and Association, Inc, 1 edition, 2002.

[SGR97] P. F. Syverson, D. M. Goldschlag, and M. G. Reed. Anonymous connections and onion routing. In *in IEEE Symposium on Security and Privacy*, 1997.

[SH95] S.Kumar and E. H.Spafford. A software architecture to support misuse intrusion detection. In *the 18th National Information Security Conference*, pages 194–204, 1995.

[Sha79] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, November 1979.

[SJ96] S.Murthy and J.Garcia. An efficient routing protocol for wireless networks. *ACM Mobile Networks and Application Journal, Special Issue on routing in mobile Communication Networks*, 1(2):183–197, October 1996.

[SMF+02] J. Staddon, S. Miner, M. Franklin, D. Balfanz, M. Malkin, and D. Dean. Self-healing key distribution with revocation. In *the IEEE Symposium on Security and Privacy*, The Claremont Resort Oakland, CA, USA, May 2002.

[SNFR03] Vikram Srinivasan, Pavan Nuggehalli, Carla F.Chiasserini, and Ramesh R.Rao. Cooperation in wireless ad hoc networks. In *The 22th IEEE Annual Joint Conference on Computer Communications and Networking INFOCOM'03*, San Francisco, California, USA, April 2003.

[SSB99] P. Sinha, S. Sivakumar, and V. Bharghavan. Mcedar: Multicast core extraction distributed ad hoc routing. In *IEEE WCNC*, page 13131317, August 1999.

[Sta03] William Stallings. *Cryptography and Network Security principles and practices*. Pearson Education Inc, 3 edition, 2003.

[STW96] Michael Steiner, Gene Tsudik, and Michael Waidner. Diffie hellman key distribution extended to group communication. In *ACM Conference on Computer and Communications Security*, pages 31–37, 1996.

[STW98] Michael Steiner, Gene Tsudik, and Michael Waidner. A new approach to group key agreement. In *International Conference on Dis-tributed Computing Systems*, pages 380–387, 1998.

[SWR98] Suresh Singh, Mike Woo, and C. S. Raghavendra. Power-aware routing in mobile ad hoc networks. In *The fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking MOBICOM'02*, page 187190, 1998.

[SYQH04] Samba Sesay, Zongkai Yang, Biao Qi, and Jianhua He. Simulation comparison of four wireless ad hoc routing protocols. *Information Technology Journal*, 3(3):219–226, 2004.

[Toh96] Chai-Keong Toh. A novel distributed routing protocol to support ad hoc mobile computing. In *The IEEE 15th Annual International Phoenix Conference on Computers and Communincation sIPCCC'96*, pages 480–486, Phoenix, AZ, USA, March 1996.

[Toh02] C.-K. Toh. *Ad hoc Mobile Wireless Networks : Protocols and Systems*. Prentice Hall PRT, 2002.

[WGL98] Chung Kei Wong, Mohamed G. Gouda, and Simon S. Lam. Secure group communications using key graphs. In *The ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication*, pages 68–79, 1998.

[WL06] Weizhao Wang and Xiang-Yang Li. Low-cost routing in selfish and rational wireless ad hoc networks. *IEEE Transaction on Mobile Computing*, 5(5):596–607, 2006.

[YD02] Alec Yasinsac and Jim Davis. Modeling protocols for secure group communication in ad hoc networks. In *The tenth International Workshop on Security Protocols*, Cambridge, UK, April 2002.

[YK03] Seung Yi and Robin Kravets. Moca : Mobile certificate authority for wireless ad hoc networks. In *The second anunual PKI research workshop (PKI 03), Gaithersburg*, 2003.

[YML02] H. Yang, X. Meng, and S. Lu. Self-organized network layer security in mobile ad hoc networks. In *ACM MOBICOM Wireless Security Workshop (WiSe'02)*, Georgia, Atlanta, USA, September 2002.

[YNK01] Seung Yi, Prasad Naldurg, and Robin Kravets. Security-aware ad-hoc routing for wireless networks. In *ACM Workshop on Mobile ad hoc networks, Mobihoc*, pages 299–302, 2001.

[ZA02] Manel Guerrero Zapata and N. Asokan. Securing ad hoc routing protocols. In *The ACM Workshop on Wireless Security (WiSe 2002)*, September 2002.

[ZBG98] X. Zeng, R. Bagrodia, and M. Gerla. Glomosim: A library for the parallel simulation of large-scale wireless networks. In *The 12th Workshop on Parallel and distributed Simulation. PADS'98*, pages 154–161, Banff, Alberta, Canada, May 1998.

[ZCY03] Sheng Zhong, Jiang Chen, and Yang Richard Yang. SPRITE: A simple, cheat-proof, credit-based system for mobile ad-hoc networks. In *The 22th IEEE Annual Joint Conference on Computer Communications and Networking INFOCOM'03*, San Franciso, CA, USA, April 2003.

[ZH99] Lidong Zhou and Zygmunt J. Haas. Securing ad hoc networks. *IEEE network magazine*, 13(6):24–30, 1999.

[Zim95] P. Zimmermann. *The Official PGP User's Guide*. MIT Press, 1995.

[ZLH03] Y. Zhang, W. Lee, and Y. Huang. Intrusion detection techniques for mobile wireless networks. *Wireless Networks*, 9(5):545–556, September 2003.

[ZLLY05] Sheng Zhong, Li (Erran) Li, Yanbin Grace Liu, and Yang (Richard) Yang. On designing incentive-compatible routing and forwarding protocols in wireless ad-hoc networks: an integrated approach using game theoretical and cryptographic techniques. In *The 11th annual international conference on Mobile computing and networking (MobiCom'05)*, pages 117–131. ACM Press, 2005.

[ZXJ04] Sencun Zhu, Sanjeev Setia Shouhuai Xu, and Sushil Jajodia. Gkmpan: An ef.cient group rekeying scheme for secure multicast in ad-hoc networks. In *First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous'04)*, pages 42–51, Augest 22-26 2004.

# Appendix: Petri nets and linear algebra concepts

**Definition 1** ([G.W83]). *A marked Petri Net is a 5-tuple* $< P, T, I, O, M_0 >$ *where* $P$ *and* $T$ *are non empty finite sets of PLACES and TRANSITIONS respectively.* $(P \cap T = \emptyset)$ *I is the so called Input function:* $I : P \times T \rightarrow \mathbb{N}$*, where* $\mathbb{N}$ *is the set of non negative integer numbers. The value I(p,t) is the weight of the directed arc from the place p to the transition t. O is the so called Output function:* $O : T \times P \rightarrow \mathbb{N}$*, where the value O(t,p) is the weight of the arc from the transition t to the place p. So the 4-tuple (P,T,I,O) is a bipartite (directed) multigraph whose arcs connect nodes from two distinct sets (P and T).* $M_0$ *is the Initial marking of places:* $M_0$*:* $P \rightarrow \mathbb{N}$*, where the value* $M_0(p)$ *is the number of the so called* tokens *that are located in the place p. Note that each of the applications I and O may be represented by a matrix whose lines represent places and columns represent transitions.*

**Definition 2** ([G.W83]). *We say transition t is* ***enabled*** *from the marking M and its* ***firing*** *leads to the marking* $M'$*, and we not* $M(t > M'$*, iff (if and only if)* $\forall p \in P$*,* $M(p) \geq I(p,t)$*. We say that* $M'$ *is* ***reachable*** *from M.* $M'$ *is given by:*
$M'(.,t)$=$M(.,t)$+$O(.,t)$-$I(.,t)$*, such that (.,t) denotes the column regarding the transition t within the matrix.*

*We can also write* $M'(.,t)$=$M(.,t)$+$C(.,t)$*, where C=O-I is called the incidence matrix.*
*The set of marking reachable from the marking M in the network Net is denoted by* $R(< Net, M >)$*, this reachablility may be direct (using one transition) or indirect (using a sequence of transitions) as well.*

**Definition 3.** ***(Sink state [G.W83]):*** *a marking M is called a sink state iff it enables no transition, i.e.:*
$\forall T_i \in T$, $\exists p \in P$ *such that* $M(p) < I(p, T_i)$.

**Definition 4.** ***(Host state [G.W83]:)*** *A host state* $M_h$ *is a marking reachable from any marking M reachable from the initial marking* $M_0$*, formally speaking:*
$M_h$ *is a host state iff* $\forall M \in R(< Net, M_0 >)$, $M_h \in R(< Net, M >)$.

**Definition 5** ([G.W83]). *A norm for a marking $M_a$ is an application v from the markings set to $\mathbb{N}$, such that $\forall M \in R(< Net, M_0 >)$, v fulfils the following conditions:*
*i)* $v(M) = 0 \Leftrightarrow M = M_a$
*ii)* $v(M) \neq 0 \Rightarrow \exists M' \in R(< Net, M >)$ *such that* $v(M') < v(M)$.

**Definition 6** ([G.W83]). *The function $\Delta$ is defined as follows: $\Delta(t, f) = f^t C \overrightarrow{l_t}$, Such that:*
*i) $f^t$ is the transpose of the vector $f \in \mathbb{N}^{np}$, where np is the number of places in the petri net, and*
*ii) $\overrightarrow{l_t}$ is the vector representing the characteristic function, i.e.: $\overrightarrow{l_t}(q) = 1$ if t=q, otherwise $\overrightarrow{l_t}(q) = 0$, where $t \in T$ (T is the transitions set).*

**Definition 7** ([G.W83]). *The t's reaching threshold regarding a vector $f \in \mathbb{N}^{np}$ is* $\sigma(t, f) = \sum_{p \in P} f(p) I(p, t)$.

**Definition 8** ([G.W83]). *For $f \in \mathbb{N}^{np}$, we define:* $\| f \| = \{p / f(p) \neq 0\}$.

**Definition 9.** ***(substitutable place [G.W83]):****A place p of a marked network $< P, T, I, O, M_0 >$ is substitutable iff:*
$\exists m > 0, \exists H \subset T, \exists F \subset T, H \neq \emptyset, F \neq \emptyset$ *such that:*
*i)* $\forall f \in F, I(., f) = m.\overrightarrow{l_p}$ *and* $O(p, f) = 0$ *(the only f's input is p, and p is not an f's output)*
*ii)* $\exists f \in F, O(., f) > 0$ *(at least one F's transition has an output).*
*iii)* $\forall h \in H, I(p, h) = 0$ *(p is not an entry of h)*
$\exists k_h \in \mathbb{N}, k_h \neq 0, O(p, h) = m.k_h$
*iv)* $\forall t \notin H \cup F$ *I(p,t) = O(p,t) =0 (each transition which is neither related to H nor to F is then unrelated to p).*

**Definition 10.** ***(place substitution [G.W83]):*** *The reduced network obtained by p's substitution from $(P, T, I, O, M_0)$ is $(P_r, T_r, I_r, O_r, M_{0_r})$ such that:*
$P_r = P - \{p\}$
$T_r = \bigcup_{h \in H} R(h) \bigcup T - (H \bigcup F)$
*where R(h) is the transitions set obtained as follows:*
*Let's consider:* $POST(h) = \{O(., h) - K_h.m.\overrightarrow{l_p} + \sum_{f \in F} n_f O(., f)$ *such that* $\sum_{f \in F} n_f = k_h\}$,
*the different possible values of $n_f$ fulfilling the condition represent the possible combinations to make the POST(h) set.*
*R(h) is the set of transitions each of which has I(.,h) as an input vector, and an element from POST as an output vector.*

*Both $I_r$ and $O_r$ are obtained by removing the entries related to $(H \cup F)$ transitions and adding those related to R(h) transitions.*
*Let* $\overline{M_{0_r}} = \{M_0 - Q_M m \overrightarrow{l_p} + \sum_{f \in F} n_f O(., f)$ *such that* $\sum_{f \in F} n_f = Q_M\}$ *where* $Q_M$ *is the integer part of* $M(p)/m$
$M_{0_r}$ *elements are obtained by removing the* $p^{th}$ *entry from each* $\overline{M_{0_r}}$*'s element.*

Note that the initial marking in this case (in the net resulted from the substitution) is generally a set of marking and not inevitably one marking like the initial net. The concept of the initial marking given in definition 1 should merely be generalized to a set of application instead of one application. That is, $M_0 = \{m_0 : P \rightarrow \mathbb{N}\})$ [G.W83]. However, in all our study we did not use such a generale concept, as the petri nets of our model (both the initial and the reduced one) include only one initial marking. Thus, we do not need to generalize definition 1.