An experimental methodology to evaluate the resilience of Ad Hoc Routing protocols

JESÚS FRIGINAL LÓPEZ

EDITORIAL
UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Departamento de Informática de Sistemas y Computadores

An Experimental Methodology to Evaluate the Resilience of Ad Hoc Routing Protocols

Jesús Friginal López

Ph.D. Advisors:

Dr. Juan Carlos Ruiz García

Dr. David de Andrés Martínez

Valencia, January 2013

This editorial is member of the UNE, which guarantees the diffusion and commercialization of its publications at national and international level.

First Edition, 2013

© of the present edition:
Editorial Universitat Politècnica de València
www.editorial.upv.es

ISBN: -978-84-9048-012-0 (printed version)
Ref. editorial: 5601

"Do, or do not. There is no try."

Teachings of Master Yoda

Acknowledgements

This thesis would not have been possible without a long list of people ...

Foremost, I would like to express my gratitude to my supervisors Dr. Juan-Carlos Ruiz and Dr. David de Andrés for giving me the chance to experience this exciting three-year-long journey called Ph.D. I will never forget your infinite patience, and your unplayable advices. I could not have had a better guidance than yours. Thanks for enlightening my darkness.

Besides my supervisors, I would like to thank Professor Pedro Gil, who provided me the economical support to carry out this thesis, and the remaining members of the Fault Tolerant Systems Group (GSTF) at the Technical University of Valencia (UPV). You made me feel part of the group from the moment I met you. I also must thank my fellow mates (now friends) at the GSTF lab: Jaume, Hector, Miquel, and other people who leaved the lab during this time (Jorge, Dani, Miguel-Ángel, José-Miguel, Javi and Toni). Thank you guys for your company and for all the fun[1] we have had in the last years.

At this point, I cannot forget the teachings of Dr. Guillermo Gil, my teacher of physics at the high school, who aroused in me the love for science. Furthermore, I want to thank my friends (in particular Adrian and David) who, in one way or another, have suffered with me what completing a Ph.D. means. Special thanks go to my girlfriend for her constant encouragement. Without her support this thesis could have not been possible.

Last but not the least, I would like to thank my mentors (my parents and grandparents). You showed me the way of effort and sacrifice. All I am is because of you.

In sum, thanks to all those who supported me in any respect during the completion of the thesis. This thesis belongs to all of you.

[1] Fun refers to the hard days suffering together before deadlines.

Abstract

Ad hoc networks are multi-hop wireless networks where all nodes cooperate to maintain the network connectivity without centralised administration. The use of this emerging technology extends from military to civilian applications. Routing protocols, which are key elements for these networks, are in charge of establishing routes between network nodes efficiently.

Despite the interest shown by the scientific community and industry in converting the first specifications of ad hoc routing protocols in functional prototypes, aspects such as the resilience of these protocols remain generally unaddressed in practice. Tackling this issue becomes critical given the increasingly variety of accidental and malicious faults (attacks) that may impact on the behaviour exhibited by ad hoc routing protocols. There exist many and varied challenges in the deployment of ad hoc routing protocols, but the need for methods to evaluate and justify their resilience is, without doubt, one of the most important. This lack can be addressed through the deliberate and controlled introduction of faults in the system. This technique, can be useful to measure the network behaviour in adverse conditions.

The main objective of this thesis is to design and implement a framework based on the injection of accidental and malicious faults to quantitatively evaluate their impact in routing protocols. This framework, called REFRAHN (Resilience Evaluation FRamework for Ad Hoc Networks), can be used to (i) identify sources of problems in the deployment of ad hoc routing protocols, (ii) design fault-tolerant mechanisms that address and minimise these problems, (iii) compare and select which is the routing protocol that fits the best the system requirements, and finally (iv) determine how to optimise the performance and robustness of the network, tuning the settings of routing protocols, and their dependability and security complements.

By tackling this topic, this thesis aims at making a step forward to improve the resilience aspects of ad hoc networks.

Resumen

Las redes ad hoc son redes inalámbricas multisalto, donde los nodos cooperan para mantener la conectividad de red. El uso de esta incipiente tecnología se extiende desde aplicaciones militares a civiles. Los protocolos de encaminamiento, que son elementos clave para estas redes, se encargan de establecer las rutas entre los nodos de la red eficientemente.

A pesar del interés mostrado por la comunidad científica e industrial en implementar los primeros prototipos funcionales de protocolos de encaminamiento ad hoc, aún quedan incógnitas por resolver para mejorar su resiliencia. Hacer frente a estas cuestiones es crítico dada la creciente variedad de fallos accidentales y maliciosos (ataques) que pueden afectar al comportamiento exhibido por los protocolos de encaminamiento. La necesidad de desarrollar metodologías y herramientas para poder evaluar y justificar su nivel de resiliencia, es sin duda, uno de los desafios más importantes. Este problema puede abordarse por medio de la inyección deliberada y controlada de fallos en el sistema. Esta técnica permite evaluar el comportamiento de la red en condiciones desfavorables.

El objetivo de esta tesis es diseñar e implementar un framework basado en la inyección de fallos accidentales y maliciosos para evaluar cuantitativamente el impacto de los mismos en los protocolos de encaminamiento de la red. Este framework, llamado REFRAHN (Resilience Evaluation FRamework for Ad Hoc Networks), puede servir para (i) identificar problemas en los protocolos de encaminamiento ad hoc, (ii) diseñar mecanismos que sirvan para hacerles frente, (iii) comparar y seleccionar el protocolo de encaminamiento que mejor se ajuste a los requisitos del sistema, y (iv) optimizar el comportamiento de la red, configurando los parámetros de los protocolos de encaminamiento, y de sus complementos de seguridad y confiabilidad.

Abordando la presente temática, esta tesis pretende dar un paso adelante en la mejora de los aspectos de resiliencia de las redes ad hoc.

Resum

Les xarxes ad hoc són xarxes sense fils multisalt, on els nodes cooperen per a mantenir la connectivitat de xarxa. L'ús d'aquesta incipient tecnologia s'estén des d'aplicacions militars a civils. Els protocols d'encaminament, que són elements clau per a aquestes xarxes, són els encarregats d'establir les rutes entre els nodes de la xarxa eficientment.

Malgrat l'interès de la comunitat científica i industrial per la implementació dels primers prototips funcionals de protocols d'encaminament ad hoc, encara queden incògnites per resoldre per a millorar la resiliència dels mateixos. Fer front a aquestes qüestions és crític donada la creixent varietat de fallades accidentals i maliciosos (atacs) que poden afectar el comportament exhibit pels protocols d'encaminament. La necessitat de desenvolupar metodologies i eines per a poder avaluar i justificar la seua resiliència, és sens dubte, un dels desafiaments més importants. Aquest problema pot abordar-se per mitjà de la injecció deliberada i controlada de fallades en el sistema. Aquesta tècnica permet avaluar el comportament de la xarxa en condicions de funcionament desfavorables.

L'objectiu d'aquesta tesi és dissenyar i implementar un framework basat en l'injecció de fallades accidentals i malicioses per a avaluar quantitativament el seu impacte als protocols de encaminamient de la xarxa. Aquest framework, anomenat REFRAHN (Resilience Evaluation FRamework for Ad Hoc Networks), es pot utilitzar per a (i) identificar fonts de problemes en els protocols d'encaminament ad hoc, (ii) dissenyar mecanismes que servisquen per a fer front a aquests problemes, (iii) comparar i seleccionar quin és el protocol d'encaminament que millor s'ajuste als requisits del sistema, i (iv) optimitzar el comportament de la xarxa, configurant els paràmetres dels protocols d'encaminament, i dels seus mecanismes de seguretat i confiabilitat.

Abordant la present temàtica, aquesta tesi pretén millorar els aspectes de resiliència de les xarxes ad hoc.

Contents

List of Tables

List of Figures

Acronyms

AODV Ad hoc On-Demand Distance Vector Routing

AOP Aspect-Oriented Programming

B.A.T.M.A.N Better Approach To Mobile Ad-hoc Networking

COTS Commercial Off The Shelf

ETX Expected Transmission Count

FPGA Field Programmable Gate Array

GUI Graphical User Interface

IFIP International Federation for Information Processing

IoT Internet of Things

IP Internet Protocol

LSP Logic Score of Preferences

MANET Mobile Ad hoc NETworks

MD5 Message-Digest Algorithm 5

MQT Minimum Quality Threshold

HIP Host Identity Protocol

MAC Media Access Control

MTBF Mean Time Between Failures

MTTF Mean Time To Failure

MTTR Mean Time To Repair

NIC Network Interface Card

NRPAR Neighbour Routing Packets Arrival Rate

NS Network Simulator

NTP Network Time Protocol

OLSR Optimized Link-State Routing

ORBIT Open Access Research Testbed for Next-Generation Wireless Networks

PHY Physical

REFRAHN Resilience Evaluation FRamework for Ad Hoc Networks

RF Radio Frequency

RPAR Routing Packets Arrival Rate

SIL Safety Integrity Level

SQuaRE Software product Quality Requirements and Evaluation

SSH Secure SHell

TC Topology Control

TTL Time To Live

VANET Vehicular Ad hoc NETwork

VoIP Voice over IP

WMN Wireless Mesh Network

WSN Wireless Sensor Network

Chapter 1

Introduction

Roads? Where we're going, we don't need roads
Dr. Emmett Lathrop Brown

Entities like the European platform in embedded systems ARTEMIS [1], state that the future is in networked embedded systems. This affirmation is based on the principle that networked embedded systems will make possible the development of more intelligent living, work, entertainment, energy production systems and transport environments [2]. It is estimated that the number of networked smart objects on a planetary scale will increase beyond 20 billion units by 2020 [3], thus promoting the emergence of a new type of network with ubiquitous nature called Internet of Things (IoT). As can be foreseen, no system in the future will run in isolation. IoT systems will be composed of myriads of computers, small devices, smart sensors, and conventional computers ranging from laptops to servers, being interconnected by fixed and/or wireless networks. Such systems, will constitute an essential fabric of our society, as already witnessed by the dependence placed on current computing systems for almost all of our life activities.

Following the principles of ubiquity [4], such interactions are expected to be spontaneous, opportunistic and infrastructureless.

Ad hoc networks are a new networking communication paradigm that can assist IoT systems in the deployment of this type of communications. Ad hoc networks are infrastructureless self-configuring networks that offer a quick, easy, and low-cost solution to enable wireless multi-hop communications. This new technology emerged from military [5] and rescue mission [6] scenarios, where the rapid deployment of survivable, efficient, and dynamic communications is critical.

In the last years, the unique properties of ad hoc networks have attracted the interest of companies, like TerraNet[1], Motorola[2] (wi4 MESH), and Meraki[3], a company backed in part by Google, that provides low-cost Internet access to more than 5 million clients all over the world. These companies only exploit part of the potentials offered by ad hoc networks, since their wireless solutions rely on a null, or limited, mobility of nodes. Currently, more sophisticated uses of ad hoc networks are under study in virtual classrooms [7], Vehicular Ad hoc NETwork (VANET)s [8], and ambient intelligent environments such as ambient assisted living [9]. PECES [10] and LoCon [11] projects are two European initiatives exemplifying this trend.

Communication features exhibited by ad hoc networks greatly depend on their internal routing protocols. These protocols are in charge of collecting information about network nodes in order to establish routes for those willing to communicate. However, although the deployment of ad hoc routing protocols is very simple, there exist some counterparts. On one hand, the evolutions of the interacting nodes cannot all be anticipated and controlled a priori given the dynamic conditions of the environment, e.g. the pres-

[1] http://terranet.se/index.php/?option=com_content&task=view&id=47&Itemid=87
[2] http://motorolaradiosolutions.com/en/product_lines/motowi4/wi4mesh
[3] http://meraki.com/technology/high_performance_mesh

ence of mobility and/or interferences in the wireless communication channel may favour the occurrence of accidental faults. On the other, the implicit trustworthiness relationship established between neighbour nodes may be exploited by malicious users. The occurrence of these accidental faults and attacks in the system may dramatically affect its expected behaviour. This thesis lies on the assumption that both accidental faults and malicious ones (attacks) are events that can potentially have similar effects on the system. From a high level viewpoint, they prevent systems from coping with their specified purpose [12]. Such events may partition the network or induce a certain traffic overhead, thus causing retransmission, inefficient routing and impacting the quality of service provided to the upper system layers. So, designing ad hoc routing protocols with resilience in mind, that is, with the skills of keeping the routing service working despite this type of events [13] is essential for their successful exploitation.

Public attitude towards resilience is changing, and even a small perceived risk for applications without strong resilience requirements might damage the reputation of the service providers. Consequently, the confident use of ad hoc networks requires not only the provision of mechanisms which assist ad hoc routing protocols in improving their resilience against faults, but also the definition of methodologies and the development of tools to evaluate the resilience of such protocols and their mechanisms in the presence of these faults. There exist many and varied challenges in the deployment of ad hoc routing protocols, but the need for frameworks to evaluate and justify their resilience is, without doubt, one of the most important. By the time being, most initiatives in the domain of ad hoc networks, like CeNSE [14], WiSeNts [15] or GENI [16] focus their goal in the deployment of ad hoc networks formed by massively interconnected devices measuring and processing data in real-time. However, such efforts will remain questionable in practice while suitable techniques to guarantee acceptable levels of resilience in their implementations remain unavailable. Resilience evaluation

can be seen as the lowest minimum denominator to improve traditional processes of design [17], tuning [18], benchmarking (comparison and selection) [19] and vulnerability discovery [20] of ad hoc routing protocols

To address this research gap between what is produced and what can be evaluated it is necessary to consider the sensitivity of ad hoc routing protocols to mobility and those faults they may suffer during their lifetime. To date, some works have tackled the evaluation of ad hoc routing protocols subjected to mobility [21]. However, despite their importance, few works have addressed the evaluation of ad hoc networks, in general, and that of routing protocols, in particular, in the presence of faults. More precisely, to date, no research has systematically exploited fault injection in ad hoc routing protocols to carry out their resilience evaluation. The difficulty to recreate the occurrence of faults in terms of controllability, repeatability and observability is a challenging task that has limited, so far, the achievement of this goal. Nevertheless, evaluating the impact of faults on real implementations of routing protocols running on top of real devices is also important to analyse the real-life aspects influencing the resilience of ad hoc networks in practice.

According to such premises, the main goal of this thesis is *creating a framework to evaluate the resilience of ad hoc routing protocols*. Such framework, named REFRAHN (Resilience Evaluation FRamework for Ad Hoc Networks), aims at covering an important lack in the domain of ad hoc networks, where more practical approaches are required to evaluate the resilience exhibited by ad hoc networks in adverse operating conditions.

On one hand, the innovative point of REFRAHN consists in defining a methodology for the configuration, execution, and analysis of results issued from experiments, where real (non-simulated) routing protocols deployed in real devices will be subjected to the presence of accidental faults and attacks. This methodology will be defined to facilitate not only the re-

silience assessment of ad hoc routing protocols, but also the effectiveness verification of the fault tolerance mechanisms that can potentially exist in protocols to mitigate the effect of faults.

On the other hand, REFRAHN will implement a portable tool to automate the proposed methodology and show its feasibility in practice. The goal of this tool will be providing adequate levels of experiment repeatability, controllability and observability while minimising the level of intrusiveness introduced in the system.

From what precedes, REFRAHN is expected to open a wide set of new experimental possibilities that can help users to increase their knowledge and confidence on the use of ad hoc routing protocols.

The work presented in this thesis follows the ensuing structure:

Chapter 2 introduces a brief overview of the general architecture and common design of routing protocols. Then, the different lacks existing on their current evaluation approaches are discussed.

Chapter 3 defines the REFRAHN methodology for the evaluation of ad hoc routing protocols in presence of faults. The stages concerning the definition, execution and analysis of experiments supporting the occurrence of faults at the routing level of ad hoc networks will be tacked in this chapter.

Then, Chapter 4 presents the implementation of REFRAHN, which instantiates the methodology previously proposed.

The exploitation of REFRAHN is experimentally shown in Chapter 5, where the resilience evaluation of different real (non-simulated) ad hoc routing protocols is considered the basis to support a wide variety of processes addressed to improve the confidence of ad hoc networks.

Finally, Chapter 6 summarises the main conclusions of this thesis and identifies some open challenges for further research.

Chapter 2

Evaluation of ad hoc routing protocols

Ad hoc networks rely on routing protocols to establish the network topology. Currently, the degree of maturity of ad hoc routing protocol implementations is enough for their use in real solutions. However, although some previous works approach the evaluation of such protocols in the absence of faults, there is a lack of suitable methods and tools to systematise their evaluation in the presence of the different accidental and malicious faults that can deviate their behaviour from the expected one. This fact poses the need to evaluate the resilience of ad hoc routing protocols against such faults.

This chapter studies the lacks that hinder the resilience evaluation of ad hoc routing protocols in practice.

2.1 Introduction

Ad hoc networks are an emerging paradigm in the world of telecommunications. They exploit the processing and wireless communication capabilities of new generations of mobile devices to create spontaneous and low-cost self-configuring networks without relying on any preexistent fixed infrastructure. It means that there is no infrastructure at all except the participating mobile nodes. All or some nodes within an ad hoc network are expected to be able to route data packets for other nodes in the network beyond their own transmission range. This characteristic, called multihopping, is the base for ad hoc networks.

Despite being originally developed for military purposes, the unique properties of ad hoc networks make them very suitable for civilian and commercial purposes. The restrictions on the use of ad hoc networking technology seem to be limited only by our imagination. These networks will open up new avenues across the full breadth of future information technologies. A diverse set of applications exploiting the ad hoc network communication paradigm in a number of different situations, are today a reality. Opportunistic networks [22] are an example of ad hoc network where nodes connectivity is intermittent or where link performance is highly variable. In such networks, there does not exist a complete path from source to destination for most of the time because the network is sparse or because of nodes mobility. In addition, the path can be highly unstable and may change or break quickly. Therefore, in order to make communication possible in an opportunistic network, the intermediate nodes may take custody of data during the communication blackout and forward it when the connectivity resumes. Other examples show the interest of ad hoc networks in Wireless Sensor Networks (WSN), which are typically used for monitoring and observing physical phenomena [23]. These networks have also been used for deploy-

ing communication and data networks in emergence situations, like in the hurricane Katrina [6] crisis, in 2006. Wireless Mesh Networks (WMN) are another interesting example of ad hoc network. They are capable of providing affordable Internet access to little or isolated populations far away from big city centres [24]. In the future, it is expected that many other application domains will benefit from ad hoc networks, e.g., Vehicular Ad hoc NETworks (VANET) [25] to enhance passengers comfort and safety, or aeronautical ad hoc networks [26] to increase the data rate of in-flight broadband Internet access.

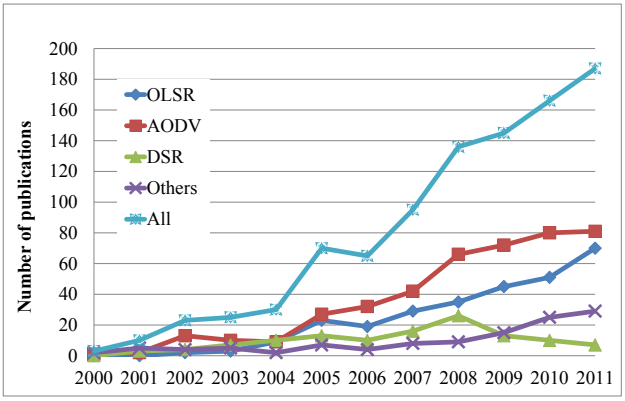


Figure 2.1: Routing protocols in literature.

Regardless the heterogeneity of these applications, all of them require the use of elements that assist network nodes in the discovery of their neighbours. Ad hoc routing protocols are the backbone of ad hoc networks. They are in charge of finding the best route from source to destination nodes in ad hoc networks. Due to their key role, the performance and robustness of an ad hoc network will greatly depend on that exhibited by the selected routing protocol. This fact justifies why routing protocols are more and more present in the current research context. Since 2000, research around routing protocols has grown in interest. Most reputed scientific databases

such as *IEEE*, *ACM*, *Springer* and *ScienceDirect* (see Figure 2.1) reflect such trend. However, although most of this success underlies on studies carried out using simulation approaches [27], research in more practical aspects is increasingly becoming more important to study the impact that any change in the execution conditions (due to the effect of mobility or the presence of accidental and malicious faults) may have on the expected behaviour of ad hoc routing protocols. These impairments to provide a confident service ask not only for solutions encompassing the design of ad hoc routing protocols with the production of adequate fault/intrusion tolerance mechanisms, but also for methodologies and tools to evaluate and guarantee the system confidence.

Traditionally, dependability has been the engineering branch devoted to address confidence problems. It studies the ability of systems to provide justifiably trusted services [70], and has been applied for decades, typically as a synonym of fault tolerance. However, fault tolerance has generally ignored the need for keeping services working in spite of continuous changes. Changes can be planned, predictable, or totally unforeseen, but in any case they constitute an unexpected aspect of the phenomena the systems may have to face. These phenomena become of primary relevance when moving to the future large, networked, evolving, ubiquitous and mobile complex heterogeneous systems, like ad hoc networks. So, designing systems, and specially ad hoc networks with resilience in mind [13], that is, with the capability to remain dependable in the presence of changes, is no more a choice but a requirement.

The importance of keeping the routing service working despite changes leads us to tackle the notion of resilience within the domain of ad hoc routing protocols. Yet, the main strengths of ad hoc networks may become their main weaknesses when analysed from the viewpoint of resilience given the nodes mobility and the dynamic nature of faults. An in-depth

analysis of such practical aspects shows that hardware gets more sensitive to manufacturing faults as the scale of components decreases. Likewise, increasing the level of sophistication of embedded software like operating systems or middleware leads to higher rates of design, programming, and configuration faults. Finally, inter-nodes wireless communications are potentially exposed to a wide variety of accidental and malicious faults given the unsteadiness and openness of the wireless communication medium.

Given such circumstances, it is necessary to adapt ad hoc routing protocols to the dynamic conditions of real environment, like mobility and the occurrence of faults, in order to deliver the best possible service. Unfortunately, to date, very few works have addressed the resilience evaluation in the domain of ad hoc networks in general, and routing protocols in particular. In essence, the difficulty to recreate the wide amount of faults threatening ad hoc networks in a controllable and repeatable way limits their consideration in the evaluation.

This chapter is devoted to analyse current challenges in the evaluation of ad hoc networks. Likewise, Section 2.2 firstly introduces basic notions about routing protocols, whereas Section 2.3 describes the key aspects behind the evaluation of ad hoc routing protocols. Then, the need for resilience evaluation frameworks is discussed in Section 2.4. Section 2.5 concludes the chapter.

2.2 Routing protocols

In ad hoc networks, nodes do not initially know the topology of their network, and must acquire that knowledge by exchanging information with their neighbour nodes. This information is used by routing protocols to control how and when information is routed among network nodes. Ac-

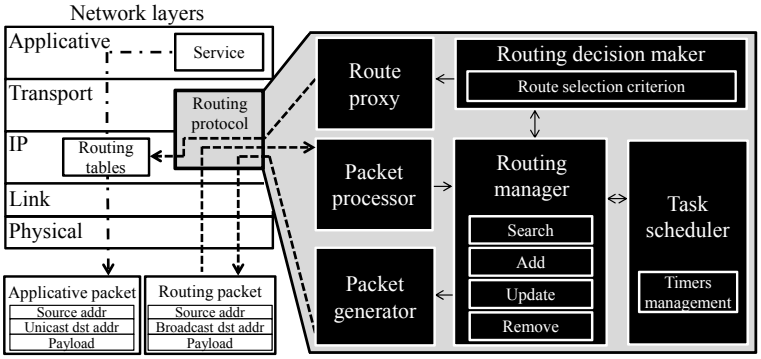


Figure 2.2: General architecture of a routing protocol.

cording to their proactiveness to exchange this information, routing protocols can be classified into reactive, if they just establish routes under demand; proactive, if they periodically exchange routing information; or hybrid, if they combine both previous characteristics. Depending on the strategy considered to compute such routes, routing protocols can be either link-state or distance vector. Link-state routing requires each node to maintain a map of the whole network. Conversely, in distance vector protocols, a given node only knows their 1-hop neighbours, but not the rest of the route.

The general architecture of a routing protocol is depicted in Figure 2.2. As can be seen, the first important thing is to establish a clear difference between the packets generated at the applicative layer (*applicative packets*) from those that are generated by the routing protocol (*routing packets*) in order to keep the network nodes interconnected. Routing protocols rely on a *packet generator* which is in charge of creating and sending routing packets.

A basic element of any routing protocol is its *task scheduler*. It manages all the internal protocol timers, which are used to periodically trigger

the broadcasting of control routing messages to the network (just in case of proactive routing protocols) and the expiration of a link (in any case) whenever the packet validity time elapsed. Valid incoming routing packets are processed by the *packet processor* and stored in the internal routing information repository of the *routing manager*. The content of incoming packets and the information stored in the routing repository is different for distance vector and link-state protocols. However, in both cases, when a change in the state of a link is discovered (i.e, a new link has been created, updated or removed), the *route proxy* reflects such change in the *routing tables* of the node, which is located in the network stack, and is not part of the routing protocol itself. The *routing manager* is responsible for handling (searching, adding, updating and removing) routes attending to the different situations experienced by the network. When computing a route from a source to a given destination node, it is possible to distinguish three different situations: (i) not finding any route, (ii) finding just one route, or (iii) finding several available routes. Dynamic factors of the environment like mobility or the presence of faults force the alternation of these three basic situations in the network. Obviously the worst case consists in not finding any route, because in such case, the communication will not be possible. Other situation may be finding one single route, but in case one node or one link in the route fails, the communication may be compromised until the routing protocol is able to discover an alternative route. Finally, the best case from the communication viewpoint consists in discovering different alternative routes. This is the most desirable case from the viewpoint of resilience as it enables the system to use backup routes in case of faults. In this situation, the *routing decision maker* will rank the different alternative routes aiming at selecting which is the best route and which remain as backup routes. The traditional Ethernet philosophy of selecting a given communication link towards a destination, among those available, with the criterion of minimising the number of remaining hops (*hop-count*) is a poor

choice in the context of ad hoc networking. Due to the actual set of features of these networks, the quality of all wireless communication links between nodes is not the same, which advices against the use of such a simple metric. Since mid-00s, the use of link-quality-based solutions has been considered fairer. Link quality is a notion that can be applied to any routing protocol regardless its nature. It is basically computed by each node according to the amount of routing information received from its neighbourhood: the higher this reception rate the better. The Expected Transmission Count (ETX) is without any doubt, the most well-known metric for characterising the quality of a link [28]. It represents the number of expected transmissions for a packet to be successfully received at its destination. This number varies from one to infinity. An ETX of one indicates a perfect transmission medium, whereas an ETX of infinity represents a non-functional link. In practice, ETX can be defined as shown in Formula 2.1.

$$ETX(i) = \frac{1}{RPAR(i) \cdot NRPAR(i)} \tag{2.1}$$

Given a sampling window in link i, $RPAR(i)$ is the Routing Packets Arrival Rate seen by a node, and $NRPAR(i)$ is the $RPAR(i)$ seen by the neighbour node. As a result, whenever two communication links (i, j) towards a destination are available, the protocol selects the one providing the better link quality.

This type of routing strategies are generally instantiated by current functional implementations of ad hoc routing protocols. This is the case of proactive routing protocols such as Optimized Link-State Routing (OLSR) [29], Better Approach To Mobile Ad-hoc Networking (B.A.T.M.A.N) [30] and Babel [31], or reactive ones such as Ad hoc On-Demand Distance Vector Routing (AODV) [32]. However, how to carry out their evaluation keeps on being a challenging task given the dynamic nature of the environments

they are deployed in.

2.3 Key aspects behind the evaluation of ad hoc routing protocols

The quantitative evaluation of ad hoc routing protocols is essential to determine whether the risks to which ad hoc networks are exposed are acceptable or not. To properly evaluate ad hoc routing protocols, it is necessary to pay attention to three basic aspects that necessary influence the quality of the results: (i) the type of platform to carry out the evaluation, which will be addressed in Subsection 2.3.1; (ii) the recreation of the conditions affecting ad hoc routing protocols, which will be tackled in Subsection 2.3.2; and (iii) the properness of the set of measures selected to represent the system behaviour, which will be analysed in Subsection 2.3.3.

2.3.1 Evaluation platforms

There exist three major strategies to address the quantitative evaluation of ad hoc routing protocols: using models, prototypes or emulating the network through emulation. The main goal of this subsection is studying current evaluation strategies and determining their lacks.

2.3.1.1 Model-based evaluation

Model-based approaches are typically used because they can be deployed in any stage of the system development, thus saving time and detecting flaws in the system design before they are too much costly to correct. They are based on formal techniques like classic process algebras [33], timed automata [34], model checking [35] or the use of Stochastic Activity Networks

(SAN) [36], an extension of the Stochastic Petri Nets formalism. These approaches are generally animated through open-source simulation tools such as Network Simulator (NS) and Glomosim, or commercial ones such as Opnet, as seen in works such as [35, 37, 38, 39, 40].

Simulation offers a high degree of controllability, repeatability and observability over results. During the study of ad hoc networks, typically few parameters are varied while most remain fixed. This allows to study the effects of certain parameters on the network performance, specially the scalability of deployments up to thousands of nodes. Simulation studies are very flexible and the related costs are normally low.

However, a simulation study has also its disadvantages. It is worth noting that, regardless the simulator used, most authors emphasise the difficulty of correctly characterising ad hoc networks due to the complexity of precisely recreate the dynamicity and heterogeneity of the network conditions, the mobility and resources-constrained nature of nodes, and the interaction between regular nodes and attackers in realistic scenarios [27]. This fact would involve building very complex models which are very costly to solve. Consequently, most of the results obtained from these works are based on assumptions which typically simplify in excess the behaviour of the system, which might lead researchers to biased or wrong conclusions. For example, simulations have been criticised for not using realistic mobility models [41] or because of assuming unrealistic wireless medium characteristics [42].

2.3.1.2 Prototype-based evaluation

In the bibliography it is possible to find different open-source prototype-based approaches to evaluate the performance of ad hoc networks. Most of them are custom-built for specific projects, and are consequently non-reusable for other experiments and purposes.

The Ad Hoc Protocol Evaluation Testbed (APE) is used for the comparative study of different ad hoc routing protocols [43]. APE uses a large space for placing the nodes, given the effective radio range of the signal emitted (from 50 m to 100 m in IEEE 802.11b/g). Moreover, node movement must be reproduced manually. Mobility in APE is introduced by providing explicit movement directions to volunteers carrying laptops. This is a choreographed mode of mobility management. Roofnet [44] and Floornet [45], are other projects deployed for conducting IEEE 802.11 measurement experiments to understand the nature of large-scale wireless networks. Since these nodes are static, there is no flexibility in the creation of different topologies.

2.3.1.3 Emulation-based evaluation

Emulators are typically used to form a virtual dynamic topology among nodes. They can be subdivided into Physical (PHY) and Media Access Control (MAC) layer emulators. In physical layer ones, all network layers except the physical layer are implemented in a real system. PHY layer emulators mangle the radio signal emitted by the wireless interfaces of the nodes to mimic the effects that radio waves would experience in a real-world setup. One possibility to do this is to attenuate the emitted signal as shown in [46]. Here, signals are fed with cables into programmable Radio Frequency (RF) attenuators. The Open Access Research Testbed for Next-Generation Wireless Networks (ORBIT), developed at Rutgers University [47], is a shared testbed that works in that way. ORBIT is a large indoor radio grid of nodes. Each node is static, and thereby it is easy to connect them to a central node using a wired interface making management functionalities easier and reliable. The mobility of each node in ORBIT is emulated through a separate mobility server, that transfers the state of a mobile node from one node in the grid to another. The

topology generation is another problem due to the static nature of the grid. Similarly, the Carnegie Mellon University Wireless Emulator (CMUWE) is a tool defined to be shared by multiple users, that supports real devices, applications, and MAC and PHY layers on a network-wide scale while maintaining experimental control and repeatability [48]. The disadvantage of this emulator is the mandatory use of custom devices which use a Field Programmable Gate Array (FPGA) to emulate the communication channel, which limits its portability.

In a MAC layer emulator, all network layers except the MAC and PHY layers are implemented in a real system. MAC layer emulators simply determine the nodes that should receive a given packet: if a node is emulated to be within radio range of another node, a filtering approach allows the exchange of packets between them, if nodes are out of reach, such packets are dropped. The filtering approach can be centralised or distributed. In a centralised system all nodes send their packets to a central controller which, in runtime, determines the nodes that should receive the packets [49]. Nevertheless, this type of approaches may induce a high level of intrusiveness in the final measures. In decentralised approaches, each node applies its own rules to determine its visibility with the rest of nodes of the network. This is the case of open-source approaches such as Castadiva [50] and MobiEmu [51], or the shared testbed Seawind [52]. By dynamically adding and removing filter rules, the emulator can also create scenarios with node movement.

2.3.1.4 Summary

Model-based proposals provide a high degree of control, abstraction, flexibility and scalability to study different types of network in a repeatable way. However, its most important disadvantage is the limited applicability

of results to the real world. Most of the results obtained from these works, are based on assumptions which typically simplify in excess the behaviour of the system, which might lead researchers to biased or wrong conclusions. Conversely, the highest degree of applicability and therefore transferability of results, is given in the case of prototype or testbeds. However, experiments are typically difficult to repeat given the complexity of actual deployments and the unsteadiness of the wireless communication medium. Furthermore, the cost from the viewpoint of the use of real hardware and the manpower required, limits the scalability of the approach. Emulation is a hybrid approach comprised of both real and virtual parts representing a trade-off between prototype-based and model-based proposals. The advantage of emulation environments over real world experiments is the possibility of scaling to larger scenarios with a minor effort. However, the degree of realism in emulation strongly depends on which components or actions of the system are real or virtualised (devices, mobility, network interfaces, etc). Table 2.1 summarises the different evaluation alternatives referred in this section.

2.3.2 Recreating the environment characteristics

Recreating as faithfully as possible the features of the environment under which ad hoc networks typically operate, is a key aspect to justify the credibility of results derived from the evaluation of ad hoc routing protocols. More precisely, the dynamic features of such environments require considering in the evaluation definition aspects such as the mobility of nodes, and the presence of internal or external threats related to the nature of the wireless medium, the limited resources of devices and the absence of a fixed infrastructure, that may lead to the occurrence of faults in ad hoc routing protocols.

Table 2.1: Comparative of reviewed platforms.

Platform	Approach			Characteristics				
	Model-based	Emulation-based	Prototype-based	Real devices	Real applications	Number of nodes	Mobility	Use of exploitation
Opnet [53]	X					>1000	X	Under commercial license
NS [54]	X					>1000	X	Free
Glomosim [55]	X					>1000	X	Free
CMUWE [48]		X		PCs[a]	X	7	X	Free, but permission required
ORBIT [47]		X		PCs	X	400	X [b]	Free, but permission required
Castadiva [50]		X		PCs, routers	X	<50	X	Free
Seawind [52]		X		PCs		<50	X	Free, but permission required
MobiEmu [51]		X		PCs	X	<50	X	Free
Roofnet [44]			X	Routers	X	24		Free
Floornet [45]			X	PCs, routers	X	37		Free
APE [43]			X	Laptops		40	X	Free

[a] A FPGA is required

[b] Only grid-based topologies are supported

It is worth noting that most of the platforms reviewed in Table 2.1 take mobility into account. However, none of them considers the occurrence of faults despite their importance during the execution of experiments in the evaluation process. This fact is mainly motivated by the difficulty to recreate the presence of faults in systems in a controllable and repeatable way [56].

2.3.3 Measures considered during evaluation

The measures considered during evaluation are expected to numerically express the behaviour of target routing protocols.

Table 2.2 reports a systematic review of what measures are typically used

in papers addressing the evaluation of ad hoc routing protocols. Results show that the performance of ad hoc routing protocols is typically characterised through measures reporting the throughput, delay, routing overhead, packet delivery ratio or jitter exhibited by the network. Surprisingly, none of the reviewed papers proposed the use of resilience-related measure to evaluate the impact of faults. This fact is intimately related to the absence of faults during the experiments execution.

Table 2.2: Measures considered by reviewed papers.

Paper reference	Measures				
	Packet loss	Delay	Routing overhead	Packet delivery ratio	Jitter
[57]	X				
[58]		X	X	X	
[59]	X			X	
[60]					
[43]					
[37]		X		X	
[38]	X	X	X		
[61]					
[39]	X				
[62]	X			X	X
[63]	X	X		X	
[64]	X	X		X	X
[40]	X	X		X	X
[65]	X	X	X		
[66]			X		
[67]	X	X	X	X	
[68]	X	X		X	
[69]	X		X	X	
Total	12 (66%)	9 (50%)	6 (33%)	10 (55%)	3 (16%)

Following section addresses both the absence of faults during the experiment execution and the lack of resilience measures in the traditional per-

formance evaluation of ad hoc routing protocols. The goal tackling this challenge is proposing the basis towards a resilience evaluation framework for ad hoc routing protocols.

2.4 Towards a resilience evaluation framework for ad hoc routing protocols

The present thesis promotes the interest of enriching current evaluation processes in ad hoc routing protocols with resilience aspects. Given the characteristics of the evaluation approaches previously analysed in Section 2.3.1, emulation seems a suitable technique to define a resilience evaluation framework given the interesting combination of experiment repeatability with the deployment on real devices. However, beyond this choice, how to cope with the introduction of the fault in the routing protocol and how to practically retrieve the necessary resilience measures from experimental environments, are essential issues (still requiring further research) that will be tackled in this thesis.

On one hand, the deliberate introduction of faults in the system in a repeatable and controllable way (fault injection) could be very useful to cover the former question. On the other, selecting a suitable set of measures which takes into account the viewpoint of resilience, addresses the second one.

2.4.1 The benefits of fault injection

The statistically low activation of faults in routing protocols in particular, makes very difficult (or at least very costly) to observe deviations in their expected behaviour. Fault injection is a well-known technique that can be useful to speed up the occurrence of faults throughout the deliberate intro-

duction of fault models in the system [71]. Thus reducing the required time to experimentally evaluate the system behaviour under adverse conditions.

This approach, introduced by Avizienis in 70s [72], enables researchers and industrials not only to characterise the system in the presence of faults, but also the effectiveness of the detection and fault-tolerance capabilities (if any), showing the potential resilience bottlenecks of the system.

Many works have applied fault injection in the last four decades to evaluate the dependability and security of systems in domains ranging from databases [73] to operating systems [74]. Even some standards [75] emphasise the need for fault injection for certification specially in the case of critical systems. The key of fault injection [56] is on clearly defining the characteristics of the faults introduced in the system, or *faultload*, according to (i) what faults to inject, which involves specifying the type of fault that will be activated in the system, (ii) how to inject them, which defines the process to introduce the fault in the system, (iii) where to inject them, which delimits the component targeted by the fault, (iv) when to inject them, thus determining the injection trigger, that can be time-driven if the fault injection is time-dependent, or event-driven, if the activation of the fault depends on the occurrence of any internal or external event, and finally (v) the duration of the fault, that can be generally permanent or transient, if it persists or not in time.

As previously stated, the first step towards the definition of a faultload, is specifying the set of faults to be injected. However, selecting a representative of faults in the domain of ad hoc routing protocols is not easy given the wide variety of both accidental and malicious faults that threaten their expected behaviour. Unlike traditional wired networks, ad hoc networks must face the openness and the unsteadiness of the wireless medium. Furthermore, most nodes in ad hoc networks have resources limitations in terms of processing, storage and energy consumption capabilities. This, in

addition to mobility, makes that ad hoc networks in general, cannot afford a centralised infrastructure to protect their communications like in a traditional network.

Some faults, like variable signal interferences, or fading, are inherited from traditional wireless networks while others, such as power consumption or topological changes, are acquired given the unique properties of ad hoc networks. In any case, these faults can be classified according to the network or node feature they take benefit from [76, 77]: resource limitations, wireless nature of communications, mobility and the absence of infrastructure.

2.4.1.1 Faults related to resources limitations

In most cases, nodes are wireless embedded devices manufactured using high scales of integration. Nodes' hardware must face hostile environmental conditions, like corrosion, strokes or fires, that can produce an irreparable *physical damage* [35] in devices. This is something inherent to outdoor wireless networks providing monitoring and controlling capabilities such as Wireless Sensor Network (WSN)s. In such context of use, nodes may stop working without previous notification, thus leading the whole system to fail if some critical sensor or control data are lost.

In addition, nodes used on ad hoc networks in general, and particularly, those used on WSN, are characterised by their limited (computational) resources. They implement tiny processors (with limited computation capabilities), short-duration batteries (which limit the nodes lifetime) and low-transmission-rated communication technologies (limiting the channel throughput). Consequently, any non-predicted *peak in the service demand* (like an environmental event leading to the processing and exchange of a big quantity of sensoring data) may exhaust the local resources of nodes or congest the network [78]. This situation can be also artificially generated

by malicious users that can saturate the wireless communication medium with a storm of (valid and/or invalid) messages just to keep nodes busy (*flooding attack*), wasting their available resources [79]. An alternative approach, typically derived from an extremely high density of nodes, may collapse the sending and reception capabilities of ad hoc routing protocols due to the massive information exchanged and processed, thus leading to a *neighbours saturation*. In all these cases, the presented faults can result in a reduction of the available bandwidth, and can even extenuate the battery of some nodes (*battery extenuation*) [80], thus leading them to switch off, consequently reducing the amount of alternative routes in the network.

2.4.1.2 Faults derived from the nature of the wireless communication medium

Ad hoc networks can be deployed in a wide variety of environments (both indoors and outdoors). However, signal quality depends on a number of different aspects, like (i) the physical characteristics (temperature, pressure, humidity, etc.) of the environment (the air in case of superficial networks, the water in case of subaquatic networks), (ii) the distance between the sender and the receiver nodes and (iii) the signal power. Accordingly, the signal can suffer from *attenuation* due to a decrease in the intensity of the electromagnetic energy at the receiver (e.g., due to long distance between nodes) [81], *multifading* [82] caused by reflection and diffraction phenomena, or *signal fluctuation* at the transmission borderline [83]. These faults may complicate the successful reception of the signal, thus not correctly reflecting whether communication between two nodes is possible or not, which may lead to an increment in the packet loss.

Since the wireless medium is simultaneously shared by multiple network devices, communications are susceptible to suffer interferences from (i) *ambi-*

ent noise [84] caused by signal overlapping due to communications located in adjacent channels (e.g., IEEE 802.11 only has three non-overlapping channels) or meteorological effects (like the solar radiation, which is specially important in networks deployed in aircrafts or space shuttles); (ii) other protocols or heterogeneous devices (like Bluetooth or microwave ovens) which inhibit the signal, denoted as *jamming attack* [85] in case of being malicious; and (iii) nodes from the same network, that can prevent other nodes from sending or receiving packets (*exposed node* problem) [86], thus increasing the rate of corrupted packets by interferences.

Finally, the wireless medium used by ad hoc networks is an open medium where anyone can listen the private communications of others without their permission. This makes the channel vulnerable to *traffic analysis attacks* [87], and *cryptanalysis* [88]. These attacks, apart from being difficult to detect given their difficult traceability, could be used to launch more sophisticated attacks.

2.4.1.3 Faults boosted by the mobility of nodes

Some types of networks, specially Mobile Ad hoc NETworks (MANET)s are characterised by the mobility of their nodes. But what may become an opportunity from the viewpoint of the user to consume a service everywhere, typically involves frequent changes of the network topology. Indeed a *wrong nodes distribution* [89] or a *wrong routing protocol configuration* [61] could boost this problem. For instance, if a routing protocol is configured to refresh the topology every 30 seconds, and a given node left the network 15 seconds ago, the network nodes have been using an expired link that does not exist anymore for 15 seconds. The same effect can be derived from nodes that accidentally *replay the sequence number* of routing packets. Mobility can be seen as a vehicle to propagate erroneous messages to other parts of

the network in case of disseminating faked or expired information about nodes and links. Attackers could benefit from this fact through *replay* [79], *sybil* [77] and *tampering* attacks [90]. This fact makes network links more unsteady, which leads to network partitions, specially at high speeds. In these cases (e.g., in vehicular networks), the *Doppler shift* [25] increases such adverse effects due to the relative speeds of the transmitter and the receiver nodes.

2.4.1.4 Challenges behind to the absence of infrastructure

Unlike traditional wireless networks, ad hoc networks do not usually rely on any fixed infrastructure. Instead, nodes must rely on each other to keep the network connected. Consequently, attackers may take benefit from this vulnerability. For example, the *sink hole attack* [91] forces the intrusion of a malicious node in an established route between two nodes by sending faked messages. Once intruded in the route, all the traffic will be forwarded through the malicious node. Thus, the attacker will be not only able to intercept but also insert messages between two victim nodes. This type of attack is used as a platform to launch a wide variety of attacks requiring a route intrusion, like *black hole* [92], *grey hole* (also referred to as *selective forwarding*) [93] or *jellyfish attacks* [91]. Apart from altering the topology, these faults can create a fictitious state of congestion and blockade of communications.

2.4.1.5 Summary

Table 2.3 lists the wide amount of faults that threaten the correct behaviour of routing protocols in ad hoc networks according to the network or node feature they take benefit from. Some of them are accidental while others are malicious. In any case, most of them have a transient nature given the

dynamic features of ad hoc network deployments. Obviously, these faults are not the only ones, but they are enough in number and importance to justify the need of evaluating ad hoc routing protocols from the perspective of their resilience.

Table 2.3: Accidental and malicious faults affecting the routing layer of ad hoc networks.

Network characteristic	**Main faults identified**
Resources limitations	Physical damage, peak in service demand, flooding attack, neighbours saturation, battery extenuation
Wireless communication medium	Signal attenuation, multifading, ambient noise, jamming attack, exposed node, traffic analysis, cryptanalysis
Mobility of nodes	Wrong nodes distribution, wrong routing protocol configuration, sequence number replay, replay attack, sybil attack, tampering attack, Doppler shift
Absence of infrastructure	Sink hole, black hole, selective forwarding attack, jellyfish attack

Figure 2.3 summarises the characteristics of such faults and the effect they may induce in the network. In this way, faults such as *physical damage* and *battery extenuation*, which affect the device lifetime, may lead the network to suffer a *node fall*. A *node fall* happens when a node stops communicating with their neighbours in a sudden way or without previous notification. Faults such as *Doppler shift*, *multifading*, *signal attenuation*, *exposed node*, *ambient noise* and *jamming attack*, that typically affect the PHY and MAC layers of the network, may create *signal problems* where the wireless medium (e.g., the air) is subjected to different physic phenomena or interferences that alter the wave transmissions. Other faults affecting the routing Internet Protocol (IP) layer, such as *wrong nodes distribution*, *wrong routing protocol configuration*, *sybil attack*, *sink hole attack*, *replay attack* and *sequence number replay* may forge or forward fictitious (non-real) routing messages, thus promoting a *topology alteration* that may lead network nodes to suffer from inefficient routing or isolation clustering ef-

fects. Alternatively, faults such as *neighbours saturation*, *tampering attack*, *peak in the service demand*, *black hole attack*, *selective forwarding*, *flooding attack* and *jellyfish attack*, affect both the routing and the service layer. These faults are characterised by creating a *communication blockade* effect that alters the expected integrity, delay or packet delivery of the data flows exchanged between a given pair of nodes. Finally, faults exploiting *traffic analysis* and *cryptanalysis* through the *eavesdropping* of frames and packets, may intercept and analyse sensitive information from the MAC, routing (IP) and service layers for a different (malicious) purpose they were conceived to.

The effects of such faults can be propagated along the network and lead to the activation of other faults. For example, a node falling or the occurrence of problems with the radio signal, may lead to a topology alteration, finally manifesting as a communication blockade between network nodes.

2.4.2 Usefulness of resilience measures

Current evaluation processes, as shown in Table 2.2, typically rely on quality-of-service (performance-related) measures like delay, packet loss or jitter to characterise the behaviour of the system. However, the special nature of ad hoc routing protocols requires additional measures to evaluate the impact of faults on the system resources or the system capability to fight against service degradation or denial.

Research in the field of resilience evaluation [13], highlights the importance of also characterising the recoverability of a system through its ability to detect, diagnose, repair, and restore the normal behaviour of the system. For instance, in case of suffering malicious attacks, the component with the lowest and more accurate detection and diagnosis times could be able to react faster and, thus, limit the effect of these faults in the system.

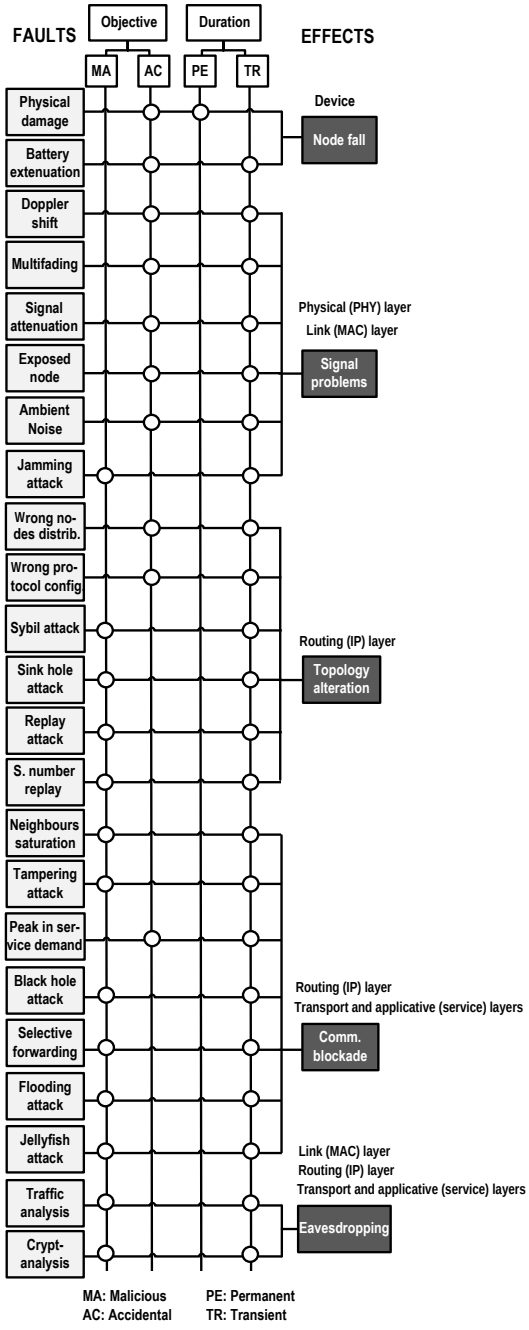


Figure 2.3: Faults characterised by their nature and effect on the network.

Likewise, low repair and restoration times are interesting to reduce the system downtime.

Generally, resilience measures can be characterised according to the following attributes [94]:

- *Availability*, that gathers those measures related to the readiness of the service, e.g., link availability, route availability, etc.
- *Reliability*, that gathers those measures addressed at measuring the continuity of the service, e.g., Mean Time To Failure (MTTF), Mean Time Between Failures (MTBF), etc.
- *Confidentiality*, that represents the degree to ensure that an unauthorized user will not be able to understand protected information in the system. Some examples that measure the confidentiality are the access controllability [75] or data encryption [75].
- *Integrity*, that gathers those measures that represent the absence of improper system alterations, e.g., link integrity, data integrity, etc.
- *Maintainability*, that gathers those measures related to the ability of the system to undergo modifications and repairs, e.g., Mean Time To Repair (MTTR), etc.

Considering resilience measures may complement traditional evaluation, for example, by assisting evaluators to select the more robust ad hoc routing protocol for a given system.

2.4.3 Discussion

To date, the evaluation of ad hoc routing protocols has been generally limited to performance aspects while non-functional aspects like resilience

surprisingly remains in the background. The introduction of faults in the system, technique known as fault injection, as well as the consideration of resilience measures could result useful to evaluate the ability of routing protocols to keep on providing the routing service despite the activation of faults. However, how to handle the introduction of faults in the environment of ad hoc networks, determining which resilience measures should be considered, and how to obtain them requires further research. These issues will be addressed in the rest of this thesis.

2.5 Conclusions

This chapter has presented the wide amount of faults that may threaten the expected behaviour of routing protocols in ad hoc networks during their deployment lifetime. The dynamic nature of faults require the provision of tools and techniques that enable academia and industry to evaluate the resilience of ad hoc routing protocols. However, currently, there is a lack of practical approaches to carry out this goal.

Next chapters address these research gaps and propose a novel resilience evaluation framework called REFRAHN (Resilience Evaluation FRamework for Ad Hoc Networks). Such framework will follow the principles of emulation, an evaluation approach encompassing the benefits from simulation and real prototypes, to experimentally evaluate the impact of faults in ad hoc routing protocols. More concretely, Chapter 3 will define the methodology of such framework. The REFRAHN methodology will be essentially divided in three basic stages: the identification and configuration of the various elements that concern the evaluation of ad hoc routing protocols, the underlying procedure to carry out the injection of accidentl fault and attacks during the experimentation, and finally, the retrieval of experimental measurements, their computation and the final provision of

performance and resilience measures. The following chapter of this thesis will address each one of these parts.

Chapter 3

A novel methodology to evaluate the resilience of ad hoc routing protocols

The presence of changes in ad hoc networks due to nodes mobility, or because of the occurrence of unexpected faults that evolve along time may compromise the basic mission of the ad hoc routing protocols: routing. So, addressing the resilience evaluation of such protocols is essential. Resilience evaluation is a well-specified procedure that enables researchers to evaluate the ability of the system to continue providing a service despite changes.

Chapter 2 showed the impairments that limit the resilience evaluation of ad hoc routing protocols in practice. These research gaps can be covered through the definition and implementation of REFRAHN, a novel unified framework that defines a methodology and implements a tool to support the experimental resilience evaluation of ad hoc routing protocols. This chapter presents the principles of the REFRAHN methodology.

3.1 Introduction

Ad hoc networks are expected to be the basis to interconnect future ecosystems of devices based on the notion of Internet of Things (IoT). However, as previously stated in Chapter 2, their main strengths may become their weaknesses when studied from the viewpoint of resilience. The occurrence of faults in ad hoc networks, and more concretely in ad hoc routing protocols, that are critical elements for such networks, may dramatically degrade the service provided by the upper layers of the network.

To date, most of the efforts done in the community of ad hoc networks concerning such issue have been focused on simulation studies proposing novel approaches to fight against the effects of such faults. However, despite their interest, the challenge is not only in proposing new fault tolerance mechanisms to face such impairments, but in providing methodologies and tools to evaluate their resilience.

The resilience evaluation of ad hoc routing protocols is essential to study how the dynamic features of ad hoc networks may affect their dependability. This aspect is very important to estimate how good (or bad) a given routing protocol or fault tolerance complement adapts to changes in the environment. Likewise, resilience evaluation could be very useful to assess the risk of subjecting (new or existing) fault tolerance mechanisms to the presence of known faults they were designed (or not) against.

Resilience evaluation is the lowest minimum denominator to support a variety of processes that are indispensable to analyse and improve the confidence of ad hoc routing protocols and their fault tolerance complements along their life-cycle. For example, results obtained from resilience evaluation could be used to discover and correct flaws in the behaviour of routing protocols and their fault tolerance complements at the design and

prototyping stages [20]. Once implemented, the comparative selection of components in presence of changes (resilience benchmarking) [19] is a process that uses the evaluation to decide which, among the existing routing protocols or fault tolerance complements, matches the best the system quality requirements. After that, resilience evaluation plays an important role to determine the impact on the system behaviour when tuning the parameterisation of routing protocols or their fault tolerance complements [18]. Finally, in case the behaviour of the system is not satisfactory enough, the resilience evaluation may guide the design of new routing and fault tolerance strategies [17] that increase the level of confidence of the network.

Unfortunately, despite the advantages of resilience evaluation to support such processes, recreating the required dynamic characteristics of ad hoc networks and their routing protocol is particularly hard. This task involves studying the evolution of mobile nodes along time as well as the faulty conditions of the environment. Specifically, it is necessary to define the experimentation in such a way it is controllable and observable, as far as possible, so that results can be comparable and repeatable.

To address this goal, this thesis presents an unified framework called REFRAHN (Resilience Evaluation FRamework for Ad Hoc Networks) that, on one hand, defines an experimental methodology to evaluate the resilience of real ad hoc routing protocols executed in real devices and, on the other, implements a tool to exploit the benefits of resilience evaluation to support processes such as the resilience benchmarking, tuning, vulnerability discovery and design of ad hoc routing protocols.

More concretely, the rest of this chapter will introduce the REFRAHN methodology, while Chapter 4 will focus on the REFRAHN implementation.

With respect to the definition of the proposed methodology, it is woth noting that ensuring an adequate level of controllability, repeatability and

observability of experiments is one of the major issues threatening the success of resilience evaluation in practice. Controllability denotes the ability to move a system around in its entire configuration space using only certain admissible manipulations. Repeatability is the variation in measurements taken by a single person or instrument on the same item and under the same conditions. Finally, observability defines how well internal states of a system can be inferred by knowledge of its external outputs.

This problem is specially critical in case of considering mobile networks, where the larger the physical distance among the nodes, the more tedious and time-consuming the task of setting up a multi-hop topology is. On one hand, several research projects, like APE at Uppsala University [83], have reported experiences of setting up a multi-hop wireless testbed spending a significant amount of time and work using volunteers to carry mobile devices in an orchestrated manner. However, this approach may affect the controllability and observability of the experimentation process. On the other hand, the radio range of wireless nodes operating in the 2.4 GHz band and using off-the-shelf IEEE 802.11b/g wireless Network Interface Card (NIC)s may vary dramatically (from 50 m to 100 m) due to ambient faults like multi-path interference, noise in the channel, etc. Thus impacting on the repeatability of experiments.

Consequently, to tackle such experimental approach, we rely on emulation. It enables researchers to use a testbed composed of real devices whereas the visibility of nodes is virtualised through software packet filtering, which, as stated in Section 2.4, is much more cheaper than using hardware attenuators. This trade-off eases the study of real ad hoc networks while avoiding the problem of deploying and controlling a large number of actual mobile nodes within a wide area.

The REFRAHN methodology copes with these impairments through the definition of three basic stages. First, it is mandatory to understand which

parameters bound the variability of results in the evaluation of ad hoc routing protocols. Then, it is necessary to address the experimental procedure to carry out the evaluation of the ad hoc routing protocols in the presence of faults. Finally, it is essential to define how to transform obtained measurements into resilience measures that can be analysed and interpreted by final users.

The rest of this section is focused on explaining such stages. More concretely, Section 3.2 defines the experiments configuration, Section 3.3 introduces the experiments execution and Section 3.4 addresses the analysis of results. Finally Section 3.5 concludes the chapter.

3.2 Experiments configuration

This section presents how to configure the resilience evaluation of ad hoc routing protocols and which elements are required from a practical experimental viewpoint. All the parameters that characterise the experimentation must be precisely determined prior to the experiments execution. As previously stated, recreating the environment characteristics and selecting a proper set of measures that reflects such characteristics is essential for the quality of evaluation results. Likely, our methodology divides such sensitive parameters into four categories: (i) the *network profile*, which configures all the parameters related to the network deployment; (ii) the *execution profile*, that animates the network deployment with realistic work- and faultloads; (iii) the *measures definition* to quantify the behaviour of the system in presence of such elements; and (iv) the *campaign configuration*, to delimits the statistical representativeness of the experimental execution.

The flow of the experiments configuration is synthesised in Figure 3.1.

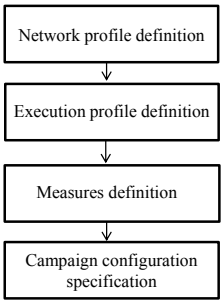


Figure 3.1: Flow of the experiments configuration.

3.2.1 Network profile definition

The *network profile* is in charge of accurately defining the network under study, its characteristics and deployment. This groups the following basic parameters.

The *topology* determines the relative position and initial distribution of the nodes within a delimited area. The *network size* is determined by the number of nodes it comprises. The *area* bounds the physical space where the nodes are confined for the duration of the experimentation. Finally, thanks to the *mobility pattern* (like Manhattan [21], random way point [95] or walking [96] models) and the *node speed*, it is possible to compute the trajectory followed by network nodes. Another parameter of prime importance is the *routing protocol* to be considered, its particular implementation and version. The selected protocol will greatly influence the resulting measures.

3.2.2 Execution profile

The execution profile, defined in terms of the *workload* and the *faultload*, encompasses the experimental operating conditions under which the evalu-

ation will be performed.

3.2.2.1 Workload

The representativeness of results obtained throughout the proposed evaluation process will depend, among other things, on the selection of a workload that matches, as much as possible, the real operation conditions of the final system. The *workload* describes the applicative traffic exchanged among nodes. Data flows are generated by the *applications* used by *source* and *destination* nodes. Real applications are preferred to increase the representativeness of results. Some example of useful applications to study particular behaviours are Voice over IP (VoIP) conferences, instant messaging or peer-to peer file exchanging. However, the use of synthetic workloads could be also interesting to study data flows from a generic viewpoint. The bitrate of synthetic workloads can be constant or variable. Conversely to real workloads, synthetic ones offer a higher level or controllability because their execution does not depend on external events (e.g., interactions with end user). In any case, the proposed methodology supports both types of workloads to increase the representativeness of results, on one hand, or recreating specific scenarios under controlled conditions on the other.

3.2.2.2 Faultload

The *faultload* defines the adverse conditions the system will face during the evaluation. One of the major issues along these years of research concerns the representativeness of injected faults [97]. In other words, the faultload should include those faults that are actually experienced during the system lifetime in order to obtain realistic evaluation results. If the injected faults are not representative, then the usefulness of the evaluation process can be compromised. This means that carefully selecting the proper fault types

(what to inject) is very important.

The proposed faultload takes into account a wide set of well-known faults that affect the behaviour of routing protocols in ad hoc networks. Such types of faults, that were introduced in Section 2.4.1, are detailed below:

F1. ***Signal attenuation*** [98]: The signal quality depends on factors such as the distance between the nodes and the signal power. This irremediably creates signal problems that lead to an increment of the packet loss rate.

F2. ***Ambient noise*** [98]: Since the wireless medium is simultaneously shared by multiple devices, communications are susceptible to suffer signal problems caused by interferences that increase the rate of corrupted packets.

F3. ***Battery extenuation*** [98]: Ad hoc network nodes, and especially those used on Wireless Sensor Networks (WSN), are characterised by their limited resources. They integrate short-duration batteries which limit to some extent the nodes lifetime. The activity of nodes can exhaust their batteries, thus leading the node to switch off and consequently impacting all the routes involving the participation of such node.

F4. ***Traffic peak*** [77]: Non-predicted peaks in the service demand may lead network nodes to suffer a communication blockade that exhausts the local resources of nodes and congests the network for a while.

F5. ***Sink hole attack*** [77]: The sink hole is an attack launched by a malicious node to intrude a communication route. Their effects cause a malicious topology alteration that benefit the attacker to execute other attacks.

F6. ***Tampering attack*** [99]: This attack violates the principle of data integrity by modifying the content of packets exchanged between a given pair of nodes. This attack leads to a communication blockade where legitimate packets addressed to destination nodes are substituted by maliciously altered ones.

F7. ***Replay attack*** [77]: The attacker overhears the network traffic to (i) capture and (ii) reproduce routing packets when they have already expired. The effects of this attack may lead to the creation of fictitious links, thus inducing an undesired topology alteration in the network.

F8. ***Selective forwarding attack*** [99]: This attack drops all those packets belonging to a target data flow. Its consequences are a communication blockade for those data flows affected by the attack.

F9. ***Jellyfish attack*** [91]: This attack delays all data packets belonging to a target data flow. Consequently, all the data flows subjected to this attack can be affected by a communication blockade.

F10. ***Flooding attack*** [77]: Nodes must overhear all messages in their radio range to determine whether they are the addressee. As a result, an attacker can saturate the medium with broadcasting storms just to keep nodes busy, totally consuming their resources and consequently causing a communication blockade in affected neighbour nodes.

F11. ***Neighbours saturation*** [77]: Routing protocols store the topology information in internal routing tables, which grow proportionally to the amount of links stored. In this way, large routing tables may result difficult to manage. Consequently, the sending and receiving capabilities of nodes may be affected, creating a communication blockade.

F12. ***Sequence number replay*** [77]: Nodes may start sending routing protocols packets without updating its identification sequence num-

ber. In case routing protocols are not protected to cope with this situation, this misbehaviour may lead to create fictitious topology alterations.

Some of these faults such as *signal attenuation*, *ambient noise*, *battery extenuation*, *traffic peak* are generic (protocol independent) while some other must be instantiated according to the particular nature of each routing protocol (protocol dependent), like *sink hole attack*, *replay attack*, *tampering attack*, *selective forwarding attack*, *jellyfish attack*, *flooding attack*, *neighbour saturation* and *sequence number replay*.

3.2.3 Measures definition

The set of measures proposed in this methodology characterises different features of routing protocols such as the performance, the resources consumption and the resilience. The goal is providing the evaluator different criteria to assess the behaviour of ad hoc routing protocols.

The subset of performance measures considered is derived from Table 2.2, which summarises the measures typically used in the evaluation of ad hoc networks. Thus, the three most widely-used measures considered by this study were taken into account: *packet delivery ratio*, defined as the percentage of applicative packets correctly delivered; *packet loss*, which is the complement of the packet delivery ratio; and *delay*, which is defined as the time elapsed since a given packet is sent by a source node until its reception by the destination node.

Considering the use of measures which assist evaluators to estimate the impact of mobility or the occurrence of faults on the resources consumption, is very important. At this point, it is proposed the use of *energy consumption*, which is specially critical in isolated environments, such as

WSN. Such measure computes the energy wasted by nodes when using their wireless NIC.

Finally, the analysis carried out in Section 2.3.3 revealed the lack of specific resilience measures in the domain of ad hoc networks. However, although a wide set of generic resilience measures can be found in the bibliography [94], it is necessary to adapt them to the necessities of ad hoc routing protocols. The subset of resilience measures considered in this methodology is a first step towards this goal. Such subset is formed by measures such as the *route availability*, defined as the percentage of time a communication route established between a source and a destination node is ready to be used; *packet integrity*, which computes the percentage of packets whose content was legitimately received; *threat exposure*, which defines the percentage of time a communication route coexists with the presence of a fault; and *fault effectiveness*, that computes the percentage of time such fault is successfully activated.

Table 3.1 summarises the purpose, formula and interpretation of the different measures considered in our experimental methodology.

3.2.4 Configuration of the experimental campaign

The proposed resilience evaluation approach consists in the practical execution of experiments. Such experiments can be grouped in experimental campaigns. Several parameters define the configuration of experimental campaigns: the *warm-up time* required for the establishment of initial routes between network nodes, the *duration* of the experiments, and the *number of repetitions* determining the proper amount of experiments that should be performed. Given the particular nature of each network deployment, determining the value assigned to these parameters is a choice that, to date, is not supported by any standardisation body. Generally, this de-

Table 3.1: Selected measures.

Performance	**Description**	**Formula**	**Interpretation**
Packet delivery ratio (%)	% of packets that arrived from source to destination in a communication route.	$\frac{\#\text{received packets}}{\#\text{generated packets}} \cdot 100$	The higher the better
Packet loss (%)	% of packets that were lost in a communication route.	$\frac{\#\text{lost packets}}{\#\text{generated packets}} \cdot 100$	The lower the better
Delay (ms)	Avg. time required by a packet to get from source to destination.	$\frac{\text{delay of received packet}}{\#\text{received packets}}$	The lower the better
Resources consumption	**Description**	**Formula**	**Interpretation**
Energy consumption (J)	Energy consumed by a node's NIC that takes part in a communication route.	$E = E_{Sending} + E_{Overhearing} + E_{Receiving}$	The lower the better
Resilience	**Description**	**Formula**	**Interpretation**
Route availability (%)	% of time the target route was ready to be used.	$\frac{\#\text{sec. the route worked}}{\#\text{experiment duration}} \cdot 100$	The higher the better
Packet integrity (%)	% of packets received at destination whose content (or payload) was not illegitimately altered.	$\frac{\#\text{non-altered received packets}}{\#\text{received packets}} \cdot 100$	The higher the better
Threat exposure (%)	% of time the communication route was exposed to any fault.	$\frac{\#\text{sec. the fault is activated}}{\#\text{experiment duration}} \cdot 100$	The lower the better
Fault effectiveness (%)	% of the threat exposure time the fault succeeded on impacting the network behaviour.	$\frac{\#\text{sec. the fault is activated}}{\#\text{sec. the fault is launched}} \cdot 100$	The lower the better

cision is taken empirically by evaluators, and despite its importance, the value of such parameters is not always found in evaluation reports. The duration of the experiments typically ranges from 60 seconds [62] to 24 hours [101], while previous works rarely take into account the warm-up time when considering the evaluation of routing protocols. However, not considering this time may result unfair when comparing routing protocols from different families, specially in experiments with short duration, as results will be biased due to the different behaviour experienced since steady operation is reduced. The number of repetitions ranges from 1 [61] to 50 [39], depending, in most cases on the standard deviation of the population sample of results. At this point, it is worth noting how results, even obtained from the same evaluation platform, are subjected to a wide variability due to,

among other factors, the range of hours when the experimentation was performed, or the amount of users accessing to the evaluation platform (in case of shared testbeds). This is a factor hindering the comparison of results.

In case it is not feasible to perform the required number of experiments, due to its huge number or its long duration, we recommend that the number of experiments performed should be bounded by the manpower involved and the time available for experimentation.

3.3 Experiments execution

One key element when tackling the experiments execution is the experimentation platform, as it strongly conditions practical aspects, like the experimental portability, scalability, controllability, observability, repeatability and intrusiveness, to successfully deploy the evaluation process.

3.3.1 Proposed architecture

As can be seen in Figure 3.2, the proposed architecture of REFRAHN consists of two different networks. The first one, the wireless (data) network, is the ad hoc network used by nodes to exchange information and constitutes the experimental network where real routing protocol targets will be deployed. Network nodes can play the role of either common or fault injector nodes (injector nodes from now on). The former send and forward traffic to other network nodes, whereas the latter are responsible for recreating the occurrence of faults in the network. The second network, the wired (control) network, connects all nodes with a special one, the experiment controller. This node is in charge of configuring all the network nodes and controlling the experimentation. It is to note the flexibility offered by the proposed architecture. Since nodes mobility is emulated, the experimental

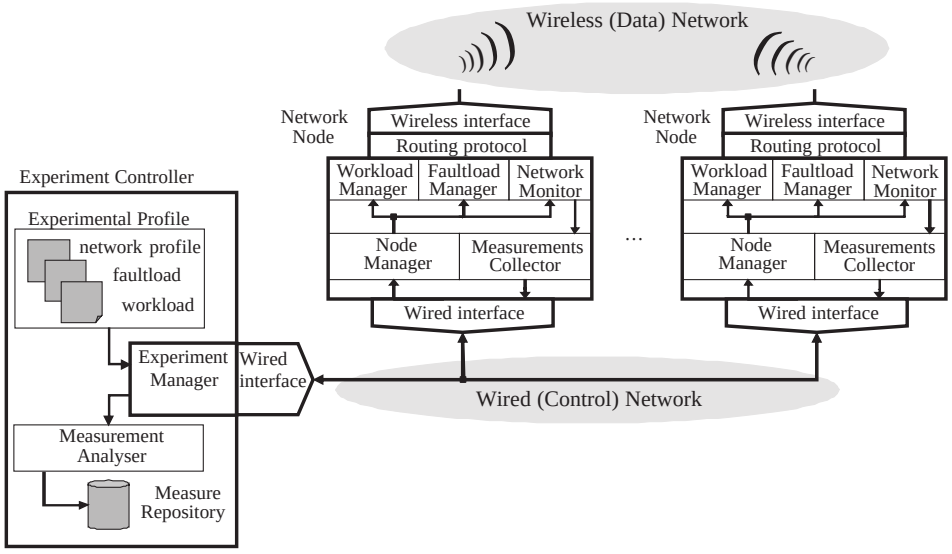


Figure 3.2: General resilience evaluation architecture.

platform may accommodate network nodes physically close. In this way, real ad hoc networks can be easily used for experimentation, since nodes behave as if they were moving around the designated area despite being static and receiving all the traffic generated by the rest of nodes. A laboratory desktop, may for instance, host more than 15 real devices. This experimental alternative enables the evaluator to considering a bigger number of nodes. This option can be really useful when addressing the typical logistic spatial limitations of research laboratories. Furthermore the fact of considering real nodes supporting available communication technologies and real routing protocol implementations eases the portability of the evaluation platform.

To carry out the fault injection process we need specific instruments and tools to inject the faults and monitor their effects. Figure 3.3 shows the interactions between REFRAHN's fault injector and its required tools. The

fault injector will used injector nodes to recreate the occurrence of faults during the experimentation. Once the faultload manager receives the faultload specification to be injected, it instruments the fault injector module to deploy the adequate actions in the right order. Essentially, the fault injector module is prepared to carry out a set of generic basic actions that can be applied at the routing level. Such a list, introduced in Figure 3.3, represents the union between the sets of actions proposed in previous works [66, 102, 103]. These actions involve (i) sniffing and storing packets from the wireless medium, (ii) processing stored information, (iii) creating packets, (iv) sending packets, (v) relaying packets, (vi) introducing a given lapse of time between one packet and the following, (vii) altering the content of legitimate packets or (viii) deleting packets. The combination of these basic actions can result into more complex faults. Additionally, the fault injection monitor can trace the faulty activity for the subsequent analysis.

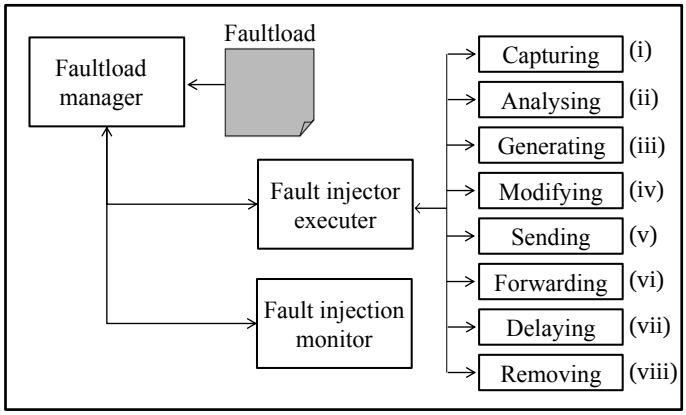


Figure 3.3: Conceptual diagram of the fault injector.

3.3.2 Experimental procedure

Once our architecture defined, the experimental evaluation can be performed as Figure 3.4 states.

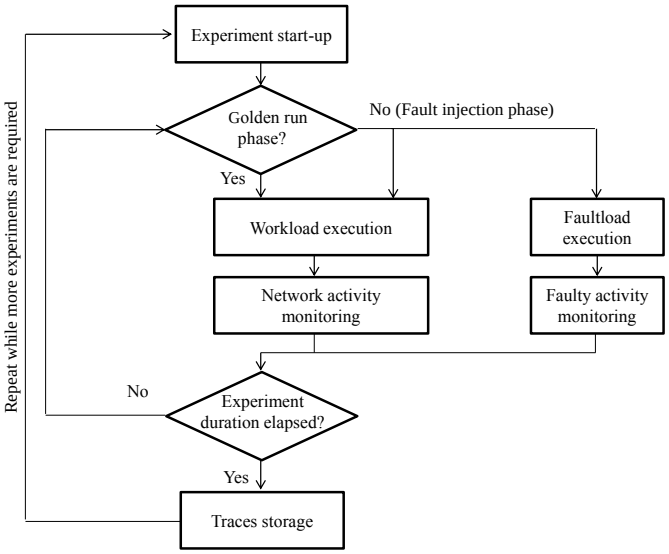


Figure 3.4: Flow of the experiments execution.

During the experiment start-up, each experiment presents an initial set-up time, required to reset the original state of the system, followed by a warm-up time, devoted to lead the targeted routing protocol to a stable state. Via the control network, the experiment controller configures each network node to play its assigned role. Common nodes are programmed to generate network traffic according to the selected workload, whereas injector nodes are in charge of introducing faults in the system according to the defined faultload. Alternatively, nodes are deployed according to the previously computed initial topology of the network. The experiment start-up requires synchronising the clock in all the network nodes to improve the accuracy of the evaluation process.

3.3.3 Golden run execution

After this, the *baseline phase*, known as *golden run* in the dependability domain [104], starts. It consists of a number of successive experiments in which the system executes the selected workload, in absence of faults, while the nodes activity is monitored. The experiment controller sets-up a number of probes in each node to monitor its activity during experimentation. Basically, these probes collect information about the amount and type of packets exchanged and the timestamp they were received and forwarded, and the energy consumed by each packet.

3.3.4 Fault injection execution

The *fault injection phase* corresponds to the execution of the workload in the presence of the faultload to evaluate the impact of faults on the system behaviour. Its execution is identical to the golden run phase, but introducing the fault into the system at a particular location at a given time. The procedure, the injection point, the trigger and the duration of each considered fault are aspects that must be specified.

3.3.4.1 F1: Signal attenuation

- **Injection procedure:** As our methodology emulates the location of the nodes, the effect of increasing or decreasing the distance among nodes is recreated through packet loss. Previous works using physical attenuators such as [105], similarly obtain a degradation in the packet loss when emulating an increment in the distance between a given pair of nodes.

- **Injection point**: The packet loss is applied over the routing and applicative packets received by the NIC of every node participating in the experiment campaign, to recreate the worst possible case.
- **Injection trigger:** The fault is activated from the beginning until the end of the experiment.
- **Fault duration:** Although generally considered transient, this fault is bounded by the experiment duration given its activation in every network node.

3.3.4.2 F2: Ambient noise

- **Injection procedure:** The interferences created in the wireless medium are emulated through packet corruption [98].
- **Injection point:** A packet corruption rate is applied over the routing and applicative packets received by the NIC of every node participating in the experiment campaign to recreate the worst possible case.
- **Injection trigger:** The fault is activated from the beginning until the end of the experiment.
- **Fault duration:** Although generally considered transient, this fault is bounded by the experiment duration given its activation in every network node.

3.3.4.3 F3: Battery extenuation

- **Injection procedure:** This fault can be emulated by disabling the packet receiving and sending capabilities of injector node [98]. Affected nodes will be seen as fallen nodes by their neighbours.

- **Injection point:** The fault disables the functions at the victim node's wireless NIC.
- **Injection trigger:** The fault is activated from the beginning until the end of the experiment.
- **Fault duration:** The fault, typically permanent, is bounded by the experiment duration.

3.3.4.4 F4: Traffic peak

- **Injection procedure:** This fault is injected by increasing the packet sending rate of a injector node up to the saturation point of the network (e.g., around 18 Mbps in IEEE 802.11g). This model emulates the case where tens of users share the same route to exchange data [77].
- **Injection point:** The fault, launched by an injector node, degrades the behaviour of all the nodes that are in the same radio range during the packet emission.
- **Injection trigger:** The fault activation is scheduled by the evaluator. By default, it is launched at the experiment beginning.
- **Fault duration:** The nature of a traffic peak is transitory. So this fault has been defined to affect a limited percentage of the experiment time, generally delimited by the evaluator.

3.3.4.5 F5: Sink hole attack

- **Injection procedure:** The sink hole attack is recreated through the exchange of routing packets between common and injector node [77].

The injector node responsible for playing the role of the attacker must create routing packets their neighbours are able to understand.

- **Injection point:** The fault is launched by an injector node. First, the injector node (M) must dynamically locate a proper data flow to identify victim nodes (A, B and C). Then, the malicious node tries to replace the position of one legitimate node (B) in the route. Likely, it announces itself as a suitable node so that neighbour nodes (A and C) take it into account to establish a new route, as Figure 3.5 illustrates.
- **Injection trigger:** This attack is triggered when the injector node and the victim nodes that are forwarding a target data traffic flow are in the same radio range.
- **Fault duration:** This fault is generally transient given the mobility, and the connection and disconnection of nodes.

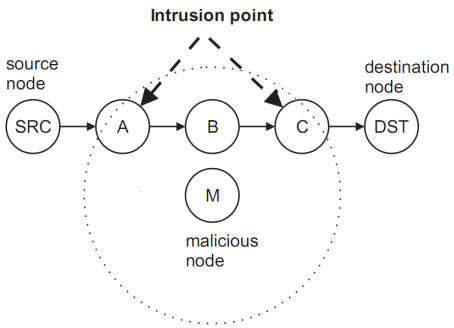


Figure 3.5: Intrusion point for sink hole attacks.

3.3.4.6 F6: Tampering attack

- **Injection procedure:** The recreation of tampering attacks requires first the execution of a sink hole attack [99]. Then, the original data

flow between a target pair of nodes is altered.

- **Injection point:** The payload of incoming target applicative packets is modified before relaying again them towards their destination.

- **Injection trigger:** This attack is triggered as soon as the sink hole succeeds.

- **Fault duration:** The duration of this attack depends on the injector node skills to keep the sink hole alive as far as possible, but it is generally transient.

3.3.4.7 F7: Replay attack

- **Injection procedure:** The injector node playing the role of replay attacker is in charge of capturing routing packets from one zone of the network and reproducing them in a different one [77].

- **Injection point:** Captured routing protocol packets will be broadcasted by the injector node, thus affecting the topology map of all the nodes that were in the same radio range during the packet replay.

- **Injection trigger:** The injector node will capture routing protocol packets during the whole experimentation time. In parallel, such packets will be replayed after a fixed period of time determined by the evaluator.

- **Fault duration:** Although launched for all the experiment time, the duration of this attack depends on the percentage of time the attacker affects a given node. Given the dynamic nature of ad hoc networks, it is considered generally transient.

3.3.4.8 F8: Selective forwarding attack

- **Injection procedure:** This attack requires the successful execution of a sink hole attack [99]. Then, the original data flow between a target pair of nodes is removed.

- **Injection point:** The injector node drops all those packets belonging to a target data flow intercepted.

- **Injection trigger:** This attack is triggered as soon as the sink hole succeeds.

- **Fault duration:** The duration of this attack depends on the injector node skills to keep the sink hole alive as far as possible.

3.3.4.9 F9: Jellyfish attack

- **Injection procedure:** This attack requires the successful execution of a sink hole attack [99]. After that, the original data flow established between a source and a destination node is delayed.

- **Injection point:** The injector node delays all those packets belonging to a target data flow intercepted a given period of time established by the evaluator.

- **Injection trigger:** This attack is triggered as soon as the sink hole succeeds.

- **Fault duration:** The duration of this attack depends on the injector node skills to keep the sink hole alive as far as possible.

3.3.4.10 F10: Flooding attack

- **Injection procedure:** The injector node responsible for introducing the flooding attack in the network will send a constant burst of packets reaching the saturation point of the network (for example, around 18 Mbps in practice for IEEE 802.11.g) [77].

- **Injection point:** The fault is launched by an injector node, thus degrading the behaviour of all the nodes that were in the same radio range during the packet emission.

- **Injection trigger:** The injector node will deploy the attack from the beginning of the experimentation time.

- **Fault duration:** Although launched for all the experiment time, the duration of this attack depends on the percentage of time the attacker affects a given node. Given the dynamoc nature of ad hoc networks, it is considered generally transient.

3.3.4.11 F11: Neighbours saturation

- **Injection procedure:** This fault is emulated by a single injector node. The idea is instrumenting such node to send a burst of fake routing protocol packets announcing a huge amount of different links [77]. The amount of new links announced in such burst could be statically assigned by the evaluator, or dynamically determined by the injection node itself through different tries.

- **Injection point:** As far as the packets sent by the injector node are received by all the neighbour nodes in the same radio range, they will believe that each packet corresponds to a different announcing node.

- **Injection trigger:** The injector node will deploy the attack from the beginning of the experimentation time.
- **Fault duration:** The fault duration is limited by the number of burst launched by the injector node. Such number is a parameter the evaluation performer should configure.

3.3.4.12 F12: Sequence number replay

- **Injection procedure:** This fault is emulated by manipulating the generation of packet sequence numbers in a single node. The idea is sending always the same packet identifier [77].
- **Injection point:** The injector node will mainly affect those nodes within the same radio range.
- **Injection trigger:** The injector node will deploy the fault from the beginning of the experimentation time.
- **Fault duration:** The fault will be deployed for the whole duration of the experiments. However, from the network viewpoint, it can be considered transient as its practical duration is bounded by the time the injector and affected nodes share the same radio range.

The degree of intrusiveness (either temporal or spatial) induced in network nodes by the fault injection process is negligible. Conversely to other fault injection methodologies [56], in any case the system execution is paused or forced to launch complex routines or tasks to induce a faulty behaviour in common nodes.

To ease the analysis of results, our approach considers that a fault injection experiment is related to one particular type of fault, thus requiring additional fault injection experiments in case of considering additional faults.

At the end of each experiment, collected measurements are transferred to the experiment controller, which is in charge of processing them to obtain the measures that characterise the network behaviour in presence of faults.

The faulty activity is monitored by the fault injection monitor. This module records every relevant event. That is, when the fault injection starts, when the fault is activated and when the fault injection finishes.

3.4 Analysis of results

At the end of the experimentation, the analysis of results is in charge of processing all the log files generated with the information collected while monitoring the system activity, as depicted in Figure 3.6. In order to reduce

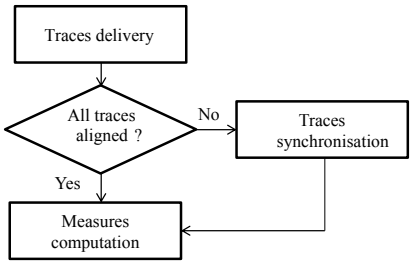


Figure 3.6: Flow of the experiments analysis.

the degree of intrusiveness on measurements that can affect the accuracy of results, all this information will be processed offline, filtered and correlated, to extract (i) the measures required to quantify the system behaviour, (ii) their average values and (iii) variability indicators such as the standard deviation or confidence intervals. Given the distributed nature of the system monitoring process, it is possible that probes installed into network nodes experience a minimal delay until the order to start or stop monitoring is

applied. To correct the possible inconsistencies caused by this delay, and its possible intrusiveness in final measures, all the traces obtained by different probes are aligned in time. This operation involves bounding the observation time of the experiment between the latest beginning timestamp and the earliest ending timestamp experienced by the traces collected from the probes of each network node.

After the analysis of traces, measures can be finally obtained by applying the expression introduced in Table 3.1.

3.5 Conclusions

The REFRAHN methodology proposed in this chapter has been designed to be a practical approach for the systematic resilience evaluation of ad hoc routing protocols. Such methodology is based on emulation to enable evaluators to use real devices and routing protocols without the physical limitations imposed by real mobility.

Fault injection is an important stage in the REFRAHN methodology to evaluate the effectiveness of routing protocols and their fault tolerance mechanisms. Accordingly, a representative set of faults in the domain of ad hoc routing protocols has been modelled to enable the recreation of faulty conditions in a repeatable way.

The impact of faults in ad hoc routing protocols is finally characterised throughout the use of different types of measures encompassing performance, resilience and resources consumption aspects. The idea of deriving measures from several viewpoints is providing evaluators a more accurate image of the target routing protocols they are evaluating.

Figure 3.7 shows the complete flow of the REFRAHN methodology, while next chapter details how such methodology can be deployed in practice.

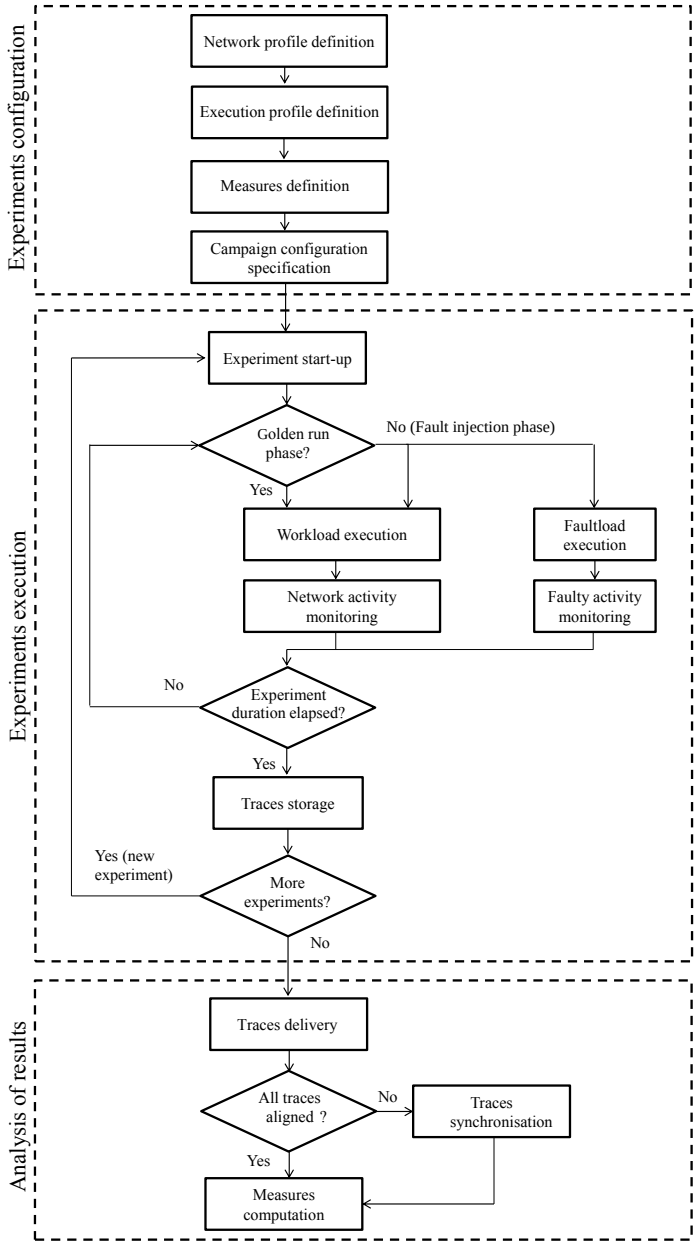


Figure 3.7: Experimental resilience evaluation flow.

Chapter 4

A tool to support our methodology

This chapter presents the first prototype of REFRAHN supporting the proposed methodology to evaluate the resilience of routing protocols. Such implementation facilitates the recreation of realistic network environments in presence of faults. Our approach allows researchers to generate network topologies, exporting them to real devices and obtaining the resulting traces. It can also generate different types of workloads and faultloads between nodes, and offers support for some well-known ad hoc routing protocols. The architecture of this tool is described along with the different processes required to conduct a fault injection campaign.

4.1 Introduction

The key principles that guide the REFRAHN implementation are (i) the miniaturisation of the area of installation of a multi-node multi-hop wireless

testbed, (ii) the control over the nodes to allow easy reconfigurability of topology in the testbed, and (iii) last, but not the least, the emphasis on building a low-cost testbed from Commercial Off The Shelf (COTS) components.

In essence, REFRAHN goes in the direction of alleviating the huge amount of time elapsed and the manpower required to carry out real experimentation with ad hoc networks. REFRAHN proposes the emulation as a way to reduce the space of experimentation and maintain the basic properties of multi-hopping while considering fault injection to introduce a wide variety set of faults within the ad hoc network.

REFRAHN implements three different stages related to the proposed methodology: (i) the definition of all the required parameters to specify the resilience evaluation to be performed; (ii) the execution of the requested fault injection experiments while monitoring the system's execution; and (iii) the analysis to determine the impact of these faults into the system's behaviour.

The rest of this chapter is structured as follows. The architecture of the REFRAHN implementation is described in Section 4.2. The experiments definition process is presented in Section 4.3. After that, Section 4.4 details the control flow for the execution of experiments and the particularities of the implementation of the fault injection. How results can be analysed to assess the resilience of targeted ad hoc routing protocols is a topic presented in Section 4.5. Section 4.6 discusses the main characteristics of the REFRAHN implementation. Finally, Section 4.7 concludes the chapter.

4.2 Architectural overview

REFRAHN has been defined to support IP as the base technology of their communications, which makes our proposal independent from the technology used in the PHY layer (Bluetooth, Zigbee, WiFi, etc). It is structured in two types of elements, the experiment controller, and the common and injector nodes. Figure 4.1 presents these basic elements. As can be seen, the emulation of mobility enables experimenters to create scenarios with dynamic topology without physically moving the nodes.

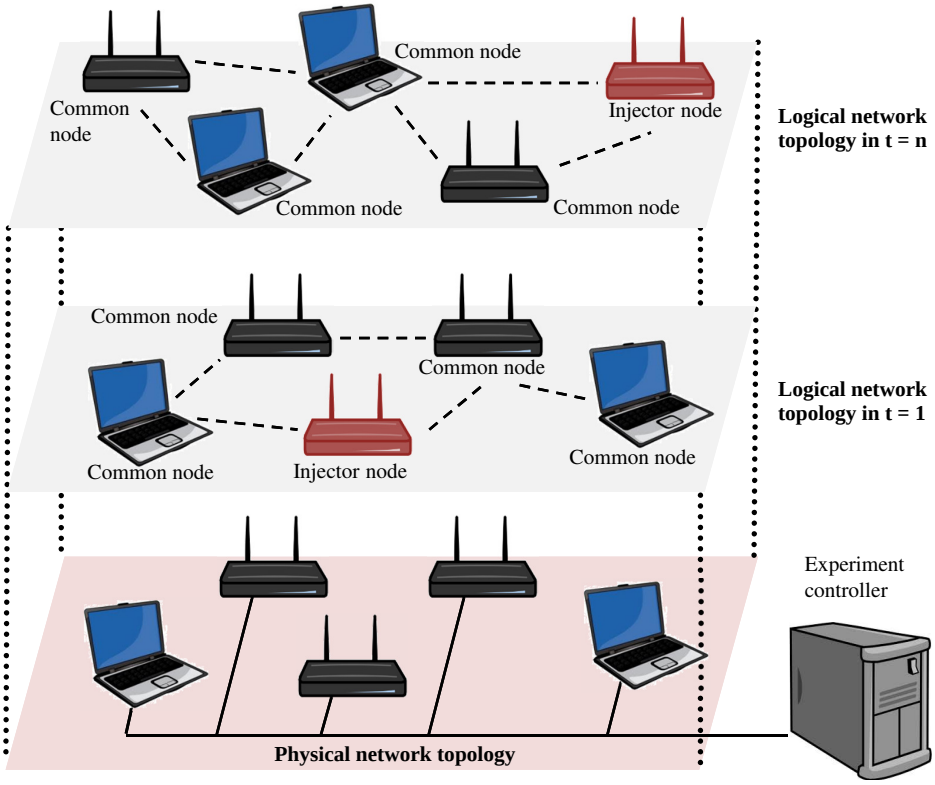


Figure 4.1: Basic elements of REFRAHN.

The experiment controller configures the network devices. The network can comprise any sort of computing device, like a laptop, a smartphone or a wireless router. Current prototype of REFRAHN, is implemented using open-source tools based on Linux/Unix, but its design, according to Chapter 3, is conceived to integrate other operating systems or tools, according to the possible needs of the evaluator.

The controller application, developed in *C* and *shell script*, controls all devices and manages network nodes dynamically according to the desired deployment scenario. Since the controlling application requires communicating with nodes to send control packets, our approach combines two different networks: the control network (wired), that connects the controller with the wireless nodes, and the wireless network, where actual tests run. The control network is a wired network that allows the controller to send configuration messages to all the nodes without creating any interference within the wireless network itself. This aspect is very important to reduce the intrusiveness of the evaluation platform itself in the evaluated system. Basically, the control network requires a switch connecting the controller to all the nodes. This communication is based on Ethernet technology, to avoid large latencies. Maintaining the internal clocks of network nodes synchronised is essential so that all the nodes participating in a experiment start at the same time, avoiding significant latency effects and maximising result accuracy. Accordingly, every node executes the Network Time Protocol (NTP) service before a new experiment starts. This service can be installed in the experiment controller in case the evaluation platform does not have Internet access to access remote NTP services.

REFRAHN comprises three different stages to manage the whole experimentation process.

- *Experiments definition*: A Graphical User Interface (GUI) imple-

mented in *java* supports the definition of the evaluation experiments by selecting the network profile and the execution profile, including the workload and the faultload.

- *Experiments execution*: Once all the parameters have been defined, REFRAHN takes control of the system to perform all the requested experiments. The coordination between the controller and each node is established using Secure SHell (SSH) through the wired network. This is a simple way to spread commands to all the nodes. First, the controller transmits the execution scripts to each node. These scripts, apart from activating the ad hoc mode in the nodes, can be customised to induce a particular behaviour in the network nodes during the execution lifetime (i.e., a given distribution of nodes affected by a given fault). The state of each node is recorded during the experimentation execution.

- *Analysis of results*: Trace logs issued from experiments subjected to fault injection are compared to fault free execution (Golden Run) traces to determine the behaviour of the system in the presence of the injected faults.

The interactions (noted from 0 to 10) between the REFRAHN's components required to complete the aforementioned three steps are detailed in Figure 4.2. Following sections are focused on explaining the details of such interactions.

4.3 Experiments definition

This process tries to gather all the necessary information to perform the desired experiments to assess the resilience of the system under study.

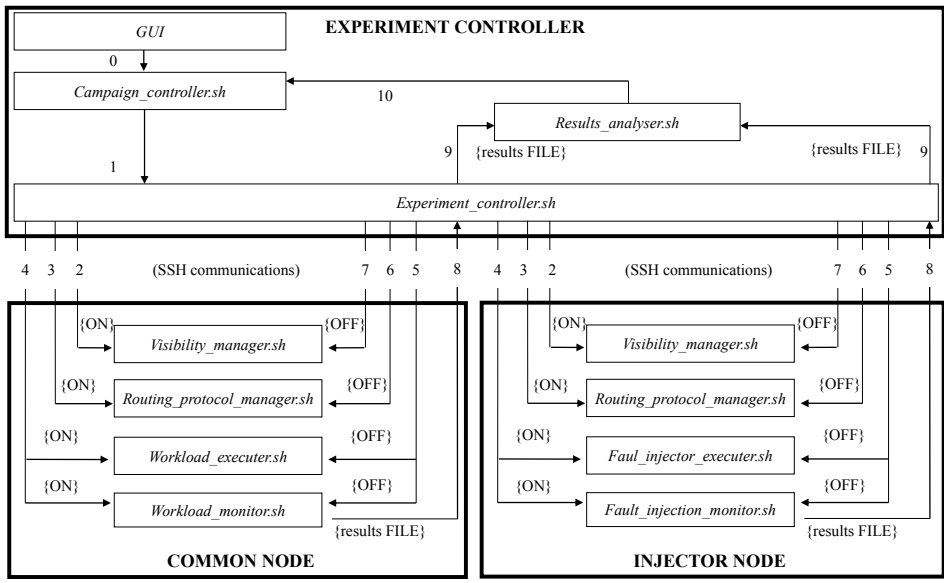


Figure 4.2: Workflow between REFRAHN components.

4.3.1 Experiments campaign

REFRAHN needs to collect some data regarding the structure of the system under study and general aspects of the experiments to be able to help the user to comfortably define the evaluation experiments. This process, referred to as interaction 0 in Figure 4.2, requires the user to provide this information by means of the graphical user interface (GUI) shown in Figure 4.3.

An experiment campaign is defined by a *name* and the number of *experiment repetitions* that bounds how many times the same experiment configuration is executed. The experiment configuration specifies the *warm up* time devoted for routing protocols to achieve a steady operating state and the *experiment time* that limits the experiment duration, as well as the *network profile* (that will be presented in Section 4.3.2) and the *execution*

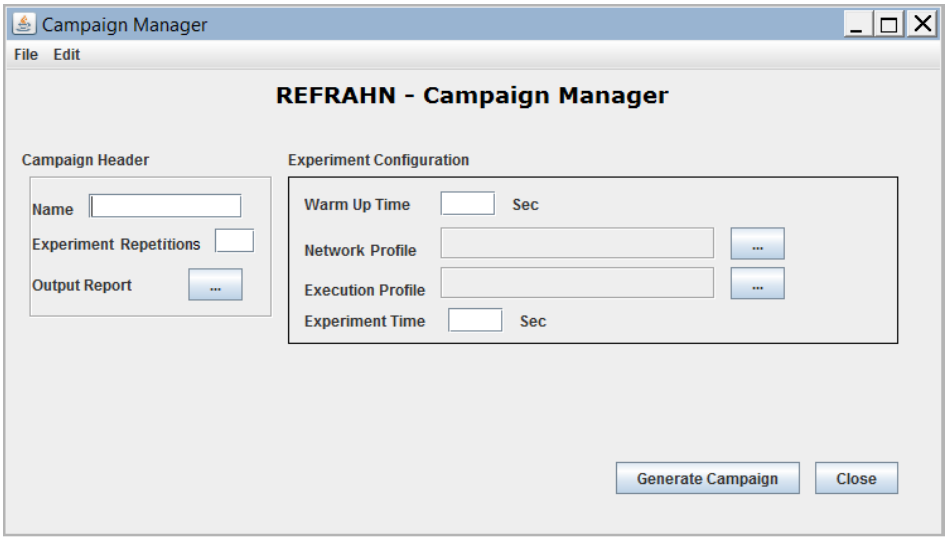


Figure 4.3: REFRAHN GUI: Form to configure the experiment campaign.

profile (that will be presented in Section 4.3.3). Finally, the *output report* specifies the path where the final report containing the evaluation measures will be stored.

4.3.2 Network profile

The network profile, as seen in Figure 4.4, enables the user to configure the nodes comprising the network. This operation involves defining their *network alias*, IP addresses, NICs and the type of applications that will be available for common nodes during the workload execution. Additionally, the REFRAHN GUI details the *network area* (length, width and radio range of each network node), the *initial distribution of nodes* in the space, the *node speed*, the *mobility pattern* and the target *routing protocol* under evaluation. To include a new type of applicative data flow in REFRAHN, it is necessary to define the path to the *script* that launches the new application.

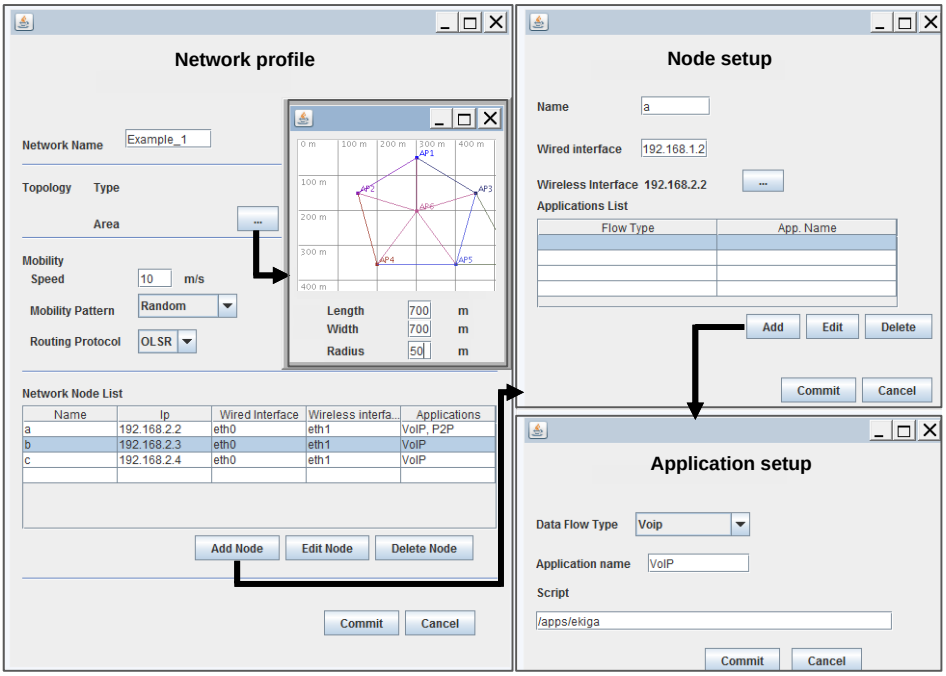


Figure 4.4: REFRAHN GUI: Forms to configure the network profile.

4.3.3 Execution profile

The configuration of the execution profile is illustrated in Figure 4.5. From the workload viewpoint, the network activity can be animated through different data flows launched by common nodes. Each data flow is characterised by an alias *flow name* to assist users to identify it, a *source* node that generates the applicative traffic using a given *application* (previously registered in the system), and a *destination* node in charge of receiving such data flow. Finally, the *deployment time* specifies the duration (in seconds) of the data flow whereas the *scheduler* determines the instant of time when the application is launched. Obviously, this time plus the *deployment time* must be shorter than the experiment duration, which limits the experiment

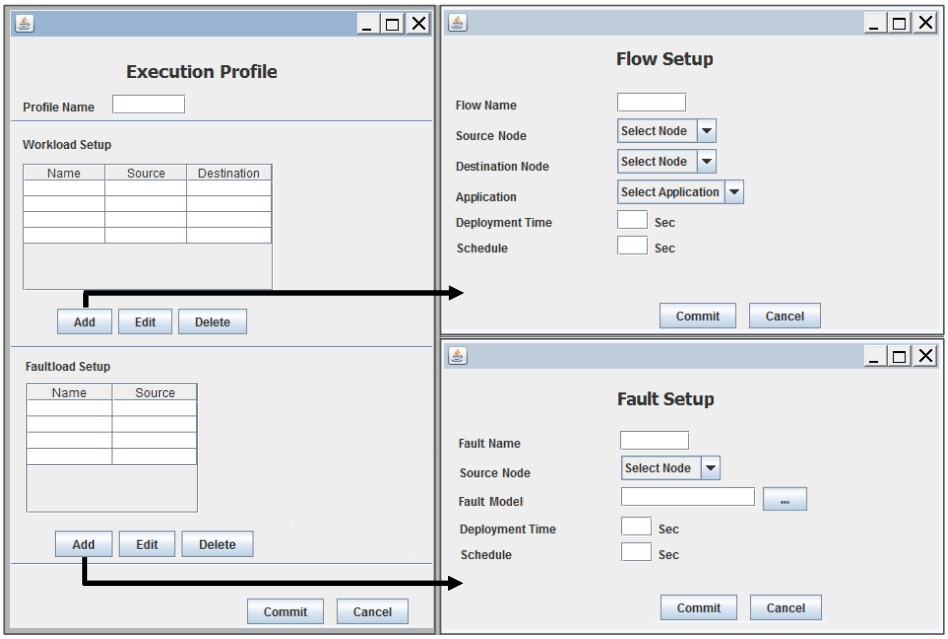


Figure 4.5: REFRAHN GUI: Forms to configure the execution profile.

execution.

Analogously, injector nodes can select the faultload through a *name* to ease its identification when there are several faultloads in the system. The fault injection *source* makes reference to the injector nodes in charge of deploying the faulty activity in the system. The type of fault introduced in the system can be specified through the *fault model*. By the time being, this information is introduced in the REFRAHN GUI indicating the path to launch a script implementing the desired fault. Finally, the fields referring to the *deployment time* and the *scheduler* mean the same that in the case of the workload configuration.

4.4 Execution of experiments

This process is in charge of controlling the experiment execution flow, including the initialisation of the prototyping system, the execution of the workload, the observation of the system's behaviour, the injection of faults, and the reset of the system to its initial state.

4.4.1 Initialisation

Once all the experimentation parameters are defined through the GUI, REFRAHN generates two scripts that characterise the experimentation, the *Campaign_controller.sh* that gathers all the parameters for the experiment campaign and launches the its execution (as interaction number 1 details in Figure 4.2), and the *Experiment_controller.sh* that manages the configuration of the components installed in network nodes and the calls to other scripts. The *Visibility_manager.sh* is the first script it calls (as interaction number 2 in Figure 4.2 shows).

4.4.2 Visibility of nodes

As already stated, nodes can remain physically stationary (in a laboratory for example) while their mobility is entirely emulated by means of the *Visibility_manager.sh* script. This script instruments the *iptables* tool [106] for this purpose. *iptables* is a generic firewall for the definition of rule sets. Each rule within an IP table consists of a number of classifiers that will trigger a connected action in case all of them are satisfied. Taking this into account, each of the nodes is provided with a *visibility script file*, which states the set of rules that must be used to emulate the visibility, and thus the mobility of the nodes during experimentation. An example of such file

is listed in Table 4.1. This script is generated by means of the Castadiva application [50], an intuitive tool, we already introduced in Chapter 2, to deploy emulated ad hoc networks topologies.

Table 4.1: Sample visibility script file.

```
#Initial visibility
/sbin/iptables -I INPUT -m mac -mac-source 00:21:00:02:45:80 -j DROP
/sbin/iptables -I FORWARD -m mac -mac-source 00:21:00:02:45:80 -j DROP
#Changes into the visibility during experimentation
sleep 136
/sbin/iptables -D INPUT -m mac -mac-source 00:21:00:02:45:80 -j DROP
/sbin/iptables -D FORWARD -m mac -mac-source 00:21:00:02:45:80 -j DROP
sleep 54
/sbin/iptables -I INPUT -m mac -mac-source 00:1A:73:A1:67:CA -j DROP
/sbin/iptables -I FORWARD -m mac -mac-source 00:1A:73:A1:67:CA -j DROP
#Delete remaining visibility instructions at the end of the experiment
sleep 62
/sbin/iptables -D INPUT -m mac -mac-source 00:1A:73:A1:67:CA -j DROP
/sbin/iptables -D FORWARD -m mac -mac-source 00:1A:73:A1:67:CA -j DROP
```

The rules inserted (-I) by the first two lines of the script file will cause every single packet received from a node with MAC address 00:21:00:02:45:80 to be dropped (-j DROP). So, in this initial topology, the current node (the one executing the script) is in the neighbourhood of the rest of nodes but the one identified by these rules. As can be seen, after 136 seconds (the node slept), some other command rules are executed to delete (-D) the initial packet discarding rules. In this case, nodes have been moving around the defined area at a given speed, and the selected node is now in radio range of every other node. 54 seconds later, due to mobility, the node falls out of range of the node with MAC address 00:1A:73:A1:67:CA. New rules are inserted to drop those packets received from that node. Finally, 62 seconds later, the experiment ends and these rules are deleted to allow for a clean start up of the next experiment.

4.4.3 Execution of the routing protocol

The target routing protocol under evaluation is launched through the *Routing_protocol_manager.sh* script after configuring the visibility of nodes (interaction number 3 in Figure 4.2). Respecting this order is essential to build communication routes which are compliant with the desired topology. Otherwise, all the nodes, that are physically in the same radio range, would be 1-hop neighbours.

By the time being, REFRAHN is ready to deploy ad hoc networks using the 4 most representative open-source routing protocols in the community: OLSR [29], AODV [32], BATMAN [30] and BABEL [31]. A description of each routing protocol will be provided in Chapter 5, when introducing our case study.

It is worth noting that the *Routing_protocol_manager.sh* script is generally executed by common nodes. Its execution in injector nodes depends on their necessity to run or not the routing protocol. Injector nodes introducing intrusion-based attacks in the system typically rely in particular packet generators to exactly forge the routing packets they require (for example to announce a fake neighbour), instead of instantiating the regular version of routing protocols. Conversely, in case of non-malicious faults, injector nodes generally execute the default implementation of routing protocols to look like common nodes while the faulty activity is injected through an additional software. Section 4.4.5 will provide more details about fault injection in REFRAHN.

4.4.4 Execution of the workload

Once routing protocols achieve a steady state after spending the warm up time, the execution of the workload starts.

REFRAHN allows defining different types of either real or synthetic traffic flows between pairs of nodes. It can result very useful when analysing the impact of using different types of workloads. The behaviour of the system is generally workload-dependent. This means that it is not the same an application generating a data flow that sends a heartbeat notification every minute than a heavy application sending real-time video and audio. The *Workload_executer.sh* script is in charge of assigning the proper workload to concerning nodes (see iteration number 4) in Figure 4.2.

On one hand, the *Workload_executer.sh* has been instrumented to send real traffic by means of actual network traffic applications. By default, well-known open-source applications based on VoIP (like Ekiga [107]) and FTP (FileZilla [108]) are supported. However, as previously stated, additional applications can be registered in REFRAHN (for example, this is the case of remote control applications for AR-drones [109] or helicopters). Table 4.2 shows an excerpt of the *Experiment_controller.sh* script devoted to the notify to the controller node when the *Workload_executer.sh* script has initiated the execution of real network traffic in a common node. As seen, the experiment controller keeps locked until variable *ready* (checked every second) receives a value distinct from 0, thus confirming when the workload application has effectively started the process. This script can be customised by modifying several variables (preceded by $ on the script file) through incoming parameters.

On the other hand, synthetic traffic is created by means of the well-known application *iperf* [110]. This is an open-source packet generator for Linux to establish CBR (Constant Bit Rate) or VBR (Variable Bit Rate) UDP/TCP data flows among a source and a destination node.

Table 4.2: Sample of the *Experiment_controller.sh* file.

```
...
#launching real application in common network node...
eval "ssh $app_user@${control_network}$node $app_command &"
ready=0
#waiting for the application to start...
while [ $ready -eq 0 ]
  do
  ready=$(ssh ${control_network}$node "ps -u $app_user | grep $app_name | wc -l" )
  sleep 1
  done
...
```

4.4.5 Execution of the faultload

The fault injection procedure is executed in parallel with the workload (as interaction number 4 in Figure 4.2 states). Section 4.4.5.1 will introduce the different open-source Commercial-Off-The-Shelf (COTS) components considered by REFRAHN to instrument to faultload execution. Figure 4.6 relates the basic actions previously introduced in Figure 3.3 to the tools selected by REFRAHN to deploy them. Once such tools presented, Section 4.4.5.2 will support the faultload implementation according to the specification of Section 3.3.4.

4.4.5.1 Injection nodes instrumentation

Instead of building new tools from the scratch to deploy required actions, we rely on well-known COTS components to ease the portability and maintainability of REFRAHN.

Traffic monitors are suitable tools to *capture* and *analyse* network traffic. It is to note that *tcpdump* [111] is a good solution among the available traffic monitors considered nowadays, given the wide variety of filtering parameters that can be tuned, like the timestamp, IP and MAC addresses,

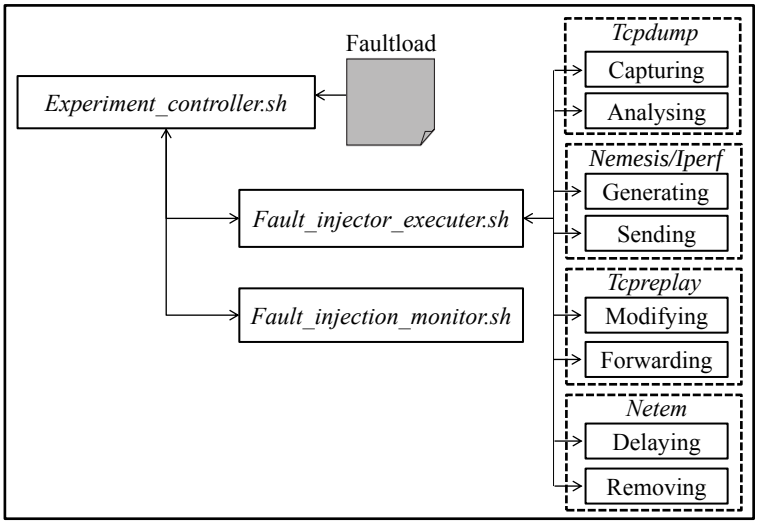


Figure 4.6: COTS used by REFRAHN's fault injector.

ports and so on. Furthermore, it implements several verbose modes to include more or less detail in reported logs. The fact that it can be launched by command line in a console, eases its integration within REFRAHN.

To instantiate packet *generation* actions, the *nemesis* utility [112] has been used. *nemesis* is a packet injector tool enabling REFRAHN to forge packets addressed to some of the most well-known communication protocols nowadays (such as ARP, ICMP, UDP, TCP, etc). In case of considering the injection of a particular type of packet not implemented natively by *nemesis*, the *Fault_injector_executer.sh* must (i) build the adequate payload according to the structure of the protocol message and (ii) including it within a conventional packet to send it using *nemesis*. This is very common for routing protocol messages typically encapsulated within UDP or TCP packets. Unfortunately, *nemesis* presents some deficiencies when needing to forge a continuous data flow. Indeed, current version of *nemesis* cannot neither tune the amount of packets to send nor the duration of the sending

period. An alternative tool called *iperf*, also considered for the workload generation, has been taken into account for this purpose.

Packet *modifying* and *forwarding* actions can be implemented through tools such as *tcprewrite* and *tcpreplay* respectively, both included within the *tcpreplay* [113] suite. Such tools edit packets already captured and replay them at arbitrary sampling speeds onto the network.

Finally, network emulation libraries such as *netem* [114] are very useful to recreate packet *removing* and *delaying*. The current version of *netem* emulates packets variable delay, loss, duplication and re-ordering in the node's network interface card. *netem* is generally enabled by default in Linux-based kernels. Furthermore, it is possible to model their activation according to a given uniform or non-uniform distribution. For example, Table 4.3 shows an excerpt of the script to recreate packet removing. Such script creates a filter for all the outgoing packets addressed to a given port. Filtered packets are then subjected to a percentage of packet loss (from 0% to 100%) corresponding to the level desired by the evaluator.

Table 4.3: Script to introduce a given packet loss $LOSS for those packets received in port $PORT.

```
...
tc qdisc add dev $DEV root handle 1:  prio
tc qdisc add dev $DEV parent 1:3 handle 30:  \
    netem loss ${LOSS}%
tc filter add dev $DEV protocol ip parent 1:0 \
    prio 3 u32 match ip dport $PORT 0xffff flowid 1:3
...
```

The resources consumption required by these components is less than 2% in terms of CPU and memory for regular laptops and 5% in wireless routers according to our previous experience, so the level of intrusiveness introduced in the nodes is practically negligible.

4.4.5.2 Fault injection

The *Fault_injector_executer.sh* configures previous tools to orchestrate the faulty activity deployed by injector nodes. For simplicity, just a summary of most important details is provided below for each one of the faults considered by the first implementation of REFRAHN.

F1. ***Signal attenuation***: The percentage of packet loss induced by this fault is emulated by the *netem* tool following a normal distribution. By default, this percentage is set to 5% during the experimentation time.

F2. ***Ambient noise***: The percentage of packet corruption induced by *ambient noise* is introduced by the *netem* tool following a normal distribution. By default, such percentage is set to 5% during all the experiment time.

F3. ***Battery extenuation***: The effects of *battery extenuation* are emulated by means of the *netem* tool introducing a packet loss in both receiving and sending packets. By default, packet loss is set to 100% for the whole experimentation duration.

F4. ***Traffic peak***: The massive generation of packets is emulated using the *iperf* tool. The effect of peaks has been recreated by default by tuning 2-second bursts of 18 Mbps every 10 seconds of the experimentation.

F5. ***Sink hole attack***: The selection of the injection point once detected the suitable data flow to launch the intrusion is determined executing the actions specified in Figure 4.7. This step consists in selecting the candidate victim nodes (A and C) to deploy the route intrusion. This phase could be represented by a conceptual diagram based on

attack trees. The leaves of the diagram represent basic actions that can (OR) or must (AND) be deployed to intrude the target route according to the ad hoc routing protocol specification. The diagram must be read from left to right and from bottom (basic actions) to top (goal). The information required to execute these actions is provided by *tcpdump*. Once victim nodes identified, the route intrusion requires knowing the syntax and semantics of the packets exchanged by routing protocols. The *fault_injector_executer.sh* script is in charge of providing *nemesis* correct payloads in the right order to successfully join the targeted route. Some examples of successful intrusion processes will be provided in Section 5.2.4.2.

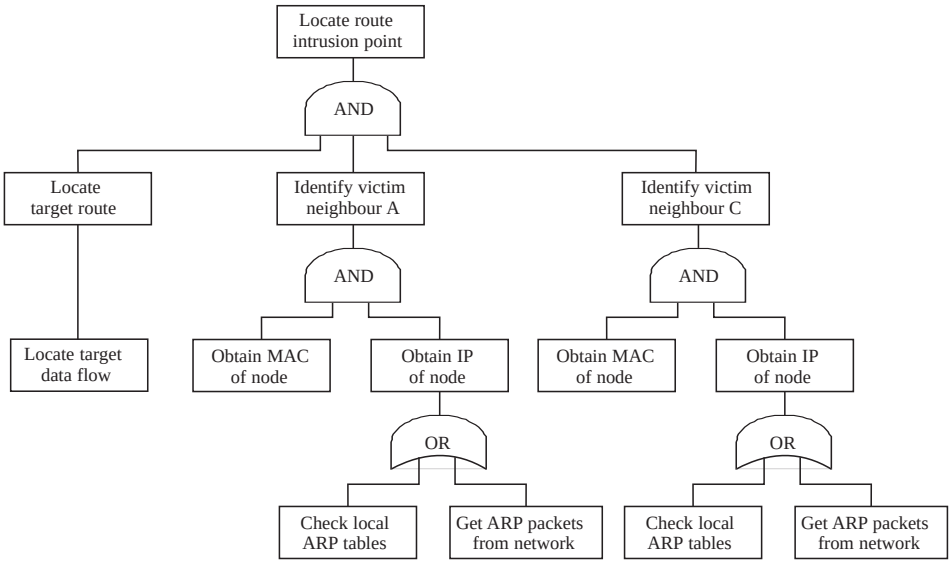


Figure 4.7: Location of the intrusion point.

F6. ***Tampering attack***: This attack requires the successful execution of a *sink hole attack*. Once the injector node located within the communication route, it uses the *tcprewrite* tool to edit the payload of

the applicative packets it forwards. By default, the original payload is changed by a string set to 0s.

F7. ***Replay attack***: The injector node first captures the routing packets exchanged in the network using *tcpdump*. Filtering captured packets by port will be useful to identify targeted routing protocol packets. Then, by default, captured routing packets will be replayed 30 seconds later using the *tcpreplay* tool for all the experimentation time. The fact of replaying routing packets pursues inducing topology incoherences in the network.

F8. ***Selective forwarding attack***: This attack can only take place after a successful *sink hole attack*. After that, the injector node configures the *netem* tool to drop all the applicative packets belonging to a target data flow.

F9. ***Jellyfish attack***: This attack requires a successful *sink hole attack*. After that, the injector node delays all data packets using the *netem* tool. By default, the delay has been tuned to 2 seconds since it is time enough to realise their effects.

F10. ***Flooding attack***: In this attack, the injector node must generate a heavy network traffic. Conversely to the effects of *traffic peak*, this attack generates a continuous broadcast data flow using *iperf*. This data flow is tuned by default to send a rate of 18 Mbps for all the experimentation time.

F11. ***Neighbours saturation***: Injector nodes emulate this fault using *nemesis* to randomly forge fake routing packets. By default, such routing packets are sent in a burst of packets announcing 400 new fake links between nodes.

F12. ***Sequence number replay***: Injector nodes use the *nemesis* tool to forge routing packets that fix the sequence number of routing protocol packets. By default, the duration of the fault was set to the experiment duration.

4.5 Analysis of results

As shown in Figure 4.2, monitoring scripts are executed in parallel with the workload and faultload executer scripts (see iteration number 4 in Figure 4.2 for more information). At the end of the experimentation time, the *Experiment_controller.sh* stops the activity of the scripts previously started up. In order to avoid the intrusiveness of this operation in final results, scripts are stopped sequentially. First, the *Workload_executer.sh*, the *Fault_injector_executer.sh* and their respective monitors, the *Workload_monitor.sh* and the *Fault_injection_monitor.sh* (as iteration number 5 in Figure 4.2 states). Then, REFRAHN stops the *Routing_protocol_manager.sh* (as iteration number 6 in Figure 4.2 shows), and finally the *Visibility_manager.sh* is switched off (as iteration number 7 in Figure 4.2 confirms). After that, the *Experiment_controller.sh* remains waiting for the trace logs from the monitoring probes installed in both the common and injector nodes (see iteration number 8 in Figure 4.2). After that, the *Experiment_controller.sh* analyses them to determine the impact of injected faults on the behaviour of the ad hoc network.

Three different logs are generated by the nodes while monitoring the behaviour of the ad hoc network during the experimentation. The *Workload_monitor.sh* manages information related to the activity deployed by common nodes (*common node log*) and ping requests (*ping log*), injector nodes use the *Fault_injection_monitor.sh* script to obtain the *injector node log*. The whole set of data stored in these logs must be processed, cor-

related, and analysed to extract those values needed to estimate desired measures (see iteration number 9 in Figure 4.2).

REFRAHN estimates the impact of faults in ad hoc networks using performance, resources consumption and resilience measures as specified in Section 3.2.3. Generic performance measures, typically taken into account in current literature such as the *packet delivery ratio*, *packet loss* and *delay*, have been selected. The resources consumption is computed throughout the *energy consumption*. Finally, this subset of measures is completed with some resilience measures: *route availability*, *packet integrity*, *threat exposure time* and *fault effectiveness*.

The *common node log* is useful to compute performance- and resources-consumption-related measures. The other two logs, the *ping log* and *injector node logs*, are required to estimate resilience-related measures. The rest of this section details how expected measures from performance (addressed in Section 4.5.1), resources consumption (addressed in Section 4.5.2) and resilience (addressed in Section 4.5.3) can be deduced from all these experimental measurements.

4.5.1 Performance measures computation

Each node must collect all the information related to the workload activity (involving the applicative traffic sent or received) using *common node logs*. Such logs will be used to compute the *packet delivery ratio*, *packet loss* and *delay*. Tools, like *tcpdump*, are good candidates to help generating these logs given their flexibility (e.g., -vv option in *tcpdump* prints a wide variety of useful information, like the Time To Live (TTL), packet ID or the total length of packets). Regarding the intrusiveness, *tcpdump* is a light process running through the line command, which monitors all the network activity with a low CPU and memory usage.

Table 4.4 lists an excerpt of the information collected by these means, which includes (i) a timestamp stating when the packet was sent/received, (ii) the MAC addresses of the source and destination nodes, and (iii) the packet header.

As previously indicated in Table 3.1, the *packet delivery ratio* is computed as the relationship among the amount of packets delivered to the destination node and the total amount of packets sent by the source node. The amount of packets delivered can be computed from the source node's log whereas the set of packets received is estimated from the destination node's log, with respect to the same data flow.

Packet loss is computed as the relationship among the amount of packets not delivered to the destination node and the total amount of packets sent by the source node (see Table 3.1). The amount of packets not delivered can be computed as the difference between the set of packets sent (from the source node's log) and the set of packets received (from the destination node's log) with respect to the same data flow. It can be alternatively estimated as 100-*packet delivery ratio*.

A similar process can be followed to compute the average *delay* of data flows (see Table 3.1). In this case the traffic log of the source and destination nodes is required to compute the delay as the difference between the

Table 4.4: Sample entry of a traffic log.

timestamp	source MAC	destination MAC	length	Packet ID	Dst. port	...
...	...	...	...	...	...	...
1233226811.175453	00:21:00:02:46:66	00:1a:73:a1:62:e9	50	1	3333	...
1233226812.365123	00:21:00:02:46:66	00:1a:73:a1:62:e9	50	2	3333	...
1233226813.578987	00:21:00:02:46:66	00:1a:73:a1:62:e9	50	3	3333	...
...	...	...	...	...	...	...
1233226931.795546	00:21:00:02:46:66	00:1a:73:a1:62:e9	50	6000	3333	...
...	...	...	...	...	...	...

timestamps of the packets sent and received with the same packet identifier.

4.5.2 Resources consumption measures computation

The *energy consumption* is the measure considered to estimate the resources consumption. It is computed as the energy required by common nodes' NIC to send, receive and overhear packets (see Table 3.1). To estimate this measure, it is necessary to compute the amount of packets sent, received and overheard by a node from all their neighbourhood (including both applicative and routing packets). It is worth noting that the notion of overheard traffic makes reference to those packets listened by a node even when it is not their addressee. Accordingly, it is necessary to filter from the traffic log of each node (i) those packets sent by the node (ii) those packets addressed to the node itself, and (iii) those which are not. Then, it is necessary to multiply each amount of filtered packets by the energy required to send, receive and overhear a packet respectively. To make our measurements more realistic, we offline obtain this value directly from the wireless NIC using hardware probes. Although this process is hard and requires the use of oscilloscope, taking the measurement once is enough. After that, the value is stored in REFRAHN to be used in the following experiment campaigns.

A generic expression to estimate the energy consumed (in Joules) to send a packet p from a wireless network interface card is $E_s(p) = P_s(p) * t_s(p)$, where $P_s(p)$ is the power consumed to send a packet and $t_s(p)$ the time required to transmit it. $P_s(p)$ can be computed as $P_s(p) = V_{in}\frac{v_s(p)}{R}$ [115], where V_{in} is the input voltage (e.g., about 3.3 V for current laptops), $v_s(p)$ is the voltage required to send a packet, and R is a test resistance (1 Ω is generally enough). Accordingly, we get $E_s(p) = 3.3\frac{v_s(p)}{1}t_s(p)$ Joules for our example, where $t_s(p)$ depends on the packet size (the larger the packet, the

longer the time to send it). Figure 4.8 depicts an example of measurement of $t_s(p)$ and $v_s(p)$. It shows an oscilloscope screenshot while monitoring the sending of a routing packet of 150 bytes, where $v_s(p) = 216mV$ and $t_p(s) = 1ms$ for a conventional wireless card. Finally, the total energy consumed by a wireless card to send packets during experimentation can be approximated as $E_s = E_s(p) * N_s$, where N_s is the total amount of packets sent. Similarly, we can also compute the energy consumed by receiving (E_r) and overhearing (E_o) packets, to estimate the total energy consumed by a wireless card as $E = E_s + E_r + E_o$.

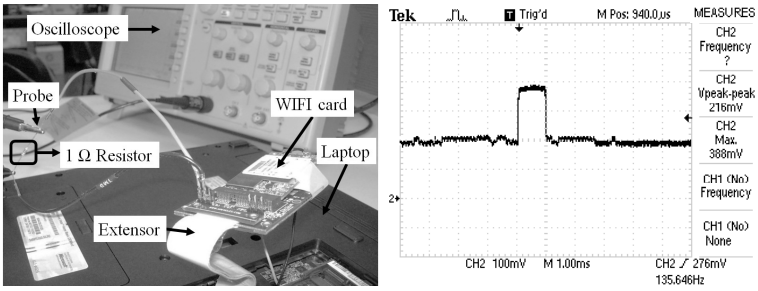


Figure 4.8: Oscilloscope sample representing the voltage consumed by an IEEE 802.11 b/g Broadcom WMIB184G card sending a packet of 150 bytes.

4.5.3 Resilience measures computation

The *route availability* measure represents the average probability of a packet to be delivered from source to destination nodes (see Table 3.1). In order to estimate this probability it is necessary to deploy some mechanism in the ad hoc network to determine whether the communication between source and destination is possible at any time. Since the network workload may not ensure that applicative packets are continuously exchanged between nodes, ICMP ECHO REQUEST (ping) messages are continuously sent from source to destination. This activity is reported by *ping logs* (see Table 4.5).

Each log's line represents a small slot of time the experimentation has been divided into. The first item in each line is a timestamp identifying that time slot. Several ping packets are sent from source to destination in each time slot to account for ambient noise or other phenomena that could lead to missing pings. Thus, the second element of each line indicates whether at least one packet was received by the destination node and, hence, the communication between nodes was available.

Table 4.5: Ping log sample.

timestamp	ICMP Reply packets received
...	...
1233226773.899245	Yes
1233226774.523744	Yes
1233226775.495388	No
1233226776.936998	No
1233226777.011103	Yes
...	...

The rest of resilience measures (*packet integrity*, *threat exposure* and *fault effectiveness*), can be derived from the information provided by the injector node. As shown in Table 4.6, the *injector node log* lists all the events it induces on the network. The first element of each log's line is a timestamp stating when an event occurred. This is followed by an event identifier and description. In the particular case of the injector log reported in Figure 4.6, the reader can see the log of a *selective forwarding attack*. Event E1 notifies the detection of a target data flow, whereas Events E2 and E3 notify the execution of the *sink hole attack* and its successful intrusion respectively. Finally, Event E4 confirms the packet dropping of the targeted data flow. In this case, the injector node only disrupts packets (i.e. data flows) sent by node 192.168.2.56 to 192.168.2.55 in a port range between 5000 and 5099.

Using this log, the *packet integrity* of a data flow (see Table 3.1) is computed as the ratio between (i) the time when the data flow is not affected by

Table 4.6: Sample of an injector node log reflecting a selective packet dropping attack.

timestamp	event notification
...	...
1233226824.839366	E1 DETECTION data flow between 192.168.2.56 and 192.168.2.55
1233226836.093937	E3 OFF INTRUSION data flow between 192.168.2.56 and 192.168.2.55
1233226836.116471	E2 GENERATING malicious routing packets between 192.168.2.56 and 192.168.2.55
...	...
1233226858.660319	E3 ON INTRUSION data flow between 192.168.2.56 and 192.168.2.55
1233226858.684511	E2 GENERATING malicious routing packets between 192.168.2.56 and 192.168.2.55
1233226859.797914	E4 ON DATA FLOW DISRUPTION between ports 5000 and 5099
...	...

attacks altering the packet content and (ii) the total experimental time.

The estimation of the *threat exposure* can be computed analysing the intervals of time when common nodes are in the radio range of injector nodes, i.e., when they are susceptible to suffer the fault effects (see Table 3.1). This information can be easily extracted from the injector logs. It is calculated as the ratio between the time when the fault is activated and the total experimentation time. In the example of Table 4.6, the time when the fault is activated refers to the difference between Event E1 (the injector node detects the data flow) and Event E3 ON (the intrusion has been successfully completed).

Finally, *fault effectiveness* is computed as the ratio between the time when the fault is activated and the experimentation time spent by the injector node to successfully activate the fault (see Table 3.1). In the example of Table 4.6, *fault effectiveness* is computed as the difference between Event E2 (the injector node starts generating fake packets addressed to the routing protocol) and Event E3 ON (the intrusion has been successfully completed).

4.5.4 Measures report delivery

After the measures computation, the *Results_analyser.sh* script transfers a measures report to the *Campaign_controller.sh*, which stores it in the path previously indicated by the evaluation user, as iteration number 10 in Figure 4.2 shows. Table 4.7 presents an extract of the measures report stored by REFRAHN for a VoIP data flow established between nodes A and C. The average and standard deviation values are provided to estimate the statistical representativeness of the experiments.

Table 4.7: Sample of measures report stored in the system.

```
...
PERFORMANCE MEASURES
Delay (ms): Node A → Node C (VoIP (Ekiga))
   mean: 2802.57
   std dev: 569.64
Packet loss (%): Node A → Node C (VoIP (Ekiga))
   mean: 23.96
   std dev: 9.08
...

RESOURCES CONSUMPTION MEASURES
Energy consumption (J): Node A
   mean: 11.20
   std dev: 1.10
Energy consumption (J): Node B
   mean: 10.34
   std dev: 0.90
Energy consumption (J): Node C
   mean: 10.31
   std dev: 0.88
...

RESILIENCE MEASURES
Route availability (%): Node A → Node C (VoIP (Ekiga))
   mean: 83.24
   std dev: 9.42
Packet integrity (%): Node A → Node C (VoIP (Ekiga))
   mean: 53.97
   std dev: 4.87
...
```

4.6 Tool features

REFRAHN satisfactorily copes with the requirements of controllability, repeatability, observability, portability and low intrusiveness presented in the introduction of this chapter.

With respect to controllability, REFRAHN is able to define and manage experimental campaigns manipulating all type of parameters concerning nodes configuration, such as their spatial location, mobility pattern, the routing protocol target considered and the work- and fault- load, among other parameters. Furthermore, the execution of the experiments is totally automated through a serial of scripts.

The degree of control achieved through to the mobility emulation and the precise fault injection, enables REFRAHN to execute repeatable experiments. This feature makes of REFRAHN, an interesting testbed in the domain of ad hoc networks to conduct reproducible fault-injection experiments for ad hoc routing protocols.

REFRAHN also provides a good level of observability. Our tool uses methods based on the analysis of experiments to collect packet traces from different types of nodes (both common and injector nodes), at different levels (software and hardware). This information is filtered and correlated to compute measures that require data from different logs, and aggregated to deliver average values.

Taking the scalability issues into account, REFRAHN is able to deploy network topologies formed by tens of nodes in a reduced space. This is a good balance between the need to scale network deployments and the space limitations of most research laboratories.

Apart from these basic characteristics, it is worth noting the notable degree of portability provided by REFRAHN to support the evaluation of a wide

variety of current real ad hoc routing protocol prototypes running on real devices implementing IP-based communications.

Finally, the low intrusiveness of REFRAHN is guaranteed not only because the managing operations between the experiment controller and network nodes use a different control network, but also because the tools installed in network nodes deploy light processes, which, in no case saturate the nodes capacity during the experiment duration.

4.7 Conclusions

The implementation of REFRAHN shows the feasibility of the resilience evaluation methodology presented in Chapter 3. This tool implements a fault injection approach for the controllable, repeatable, observable portable and low-intrusive injection of faults in real routing protocols running on real devices.

REFRAHN relies on the use of a controller node that assigns a particular role to the nodes of the ad hoc network. Accordingly, nodes can act as common (regular) nodes or injector (faulty) nodes. Likewise, REFRAHN (i) allows for the recreation of different network topologies applying the benefits of mobility emulation, (ii) implements a considerable set of fault models considered representative for current ad hoc routing protocols, and (iii) presents a very simple GUI that any non-skilled user may employ to deploy a complete experimental campaign.

Apart from coping with the resilience evaluation of ad hoc routing protocols, REFRAHN could be exploited to support the processes of design, benchmarking, tuning and discovery of vulnerabilities of ad hoc routing protocols, since resilience evaluation is the pillar of all of them. In this way, external evaluator, auditors or certifiers could benefit from REFRAHN

to instrument the benchmarking process of various protocol candidates to select the most suitable one according to particular requirements (e.g., a critical application deployed in an aircraft, or a domestic entertainment system), thus easing the analysis of results issued from experimentation. Network administrators, typically in charge of managing and configuring the components within the system, may rely on this tool for the fine tuning of the system behaviour, thus obtaining optimal configurations in presence of faults. Finally, component developers (e.g., routing protocol designers) might use REFRAHN for guiding the robust design of routing protocols and detecting flaws that can lead components to exhibit vulnerabilities at runtime.

Next chapter illustrates the exploitation of REFRAHN for different processes supported by the resilience evaluation of ad hoc routing protocols.

Chapter 5

Exploitation of REFRAHN

Once the methodology of REFRAHN has been shown, and the main features of the current implementation have been presented, it is desirable to show some examples of the possible exploitation of REFRAHN.

This chapter considers various case studies to show the interest and applicability of REFRAHN for different user profiles, different purposes, and different types of networks.

5.1 Introduction

The flexibility of REFRAHN enables users to recreate the dynamic characteristics of ad hoc networks, including a wide variety of fault models. This fact poses an interesting discussion about the use of REFRAHN to support processes that require the resilience evaluation of ad hoc routing protocols to improve the confidence of ad hoc network solutions along their life-cycle. The final aim of REFRAHN is to improve the practical aspects that limit, by the time being, the exploitation of ad hoc networks in our daily life.

This section is structured as follows. First, Section 5.2 presents a case study to show the feasibility of the experimental resilience evaluation for ad hoc routing protocols. This section shows the power of REFRAHN to evaluate different routing protocols subjected to the presence of all the accidental faults and attacks previously considered in Chapter 3 and Chapter 4.

Then, the rest of this chapter is devoted to show the exploitation of results issued from experimentation to carry out the design, benchmarking, fine tuning and discovery of vulnerabilities of ad hoc routing protocols and their fault/intrusion tolerance complements. More concretely, Section 5.3 shows the usefulness of REFRAHN to guide the discovery of new vulnerabilities on ad hoc routing protocols. Section 5.4 poses the problem of determining which is the best alternative when different candidate routing protocol are eligible. Section 5.5 illustrates a typical need of system administrators: the process of tuning a system component. Our fault injection methodology can assist the user to evaluate the impact of considering one parameterisation setup or another, or what is more, guiding the selection of the optimum configuration for a given component. Section 5.6 shows the applicability of REFRAHN for the discovery and correction of flaws and vulnerabilities in those routing protocols and fault-tolerance strategies under development. Concretely, as a result of applying the fault injection capability of REFRAHN in this case study, a novel adaptive fault-tolerance technique as been proposed to mitigate the impact of *ambient noise* in proactive routing protocols. Finally, Section 5.7 concludes the chapter.

5.2 Experimental resilience evaluation

This section presents a case study where a simple ad hoc network has been recreated. All the fault models implemented by REFRAHN have been introduced in different experiment campaigns to determine whether

our approach can be effectively used for the resilience evaluation of ad hoc routing protocols.

The rest of this section is structured as follows. Subsection 5.2.1 presents the routing protocol targets. Then, subsection 5.2.2 introduces the experimental testbed. After that, subsections 5.2.3 and 5.2.4 define the network profile and the execution profile configurations respectively. The number and duration of experiments is determined in subsection 5.2.5. Subsection 5.2.6 presents some considerations about REFRAHN measures that are taken into account in subsection 5.2.7 in the analysis of results. Finally, subsection 5.2.8 concludes the section.

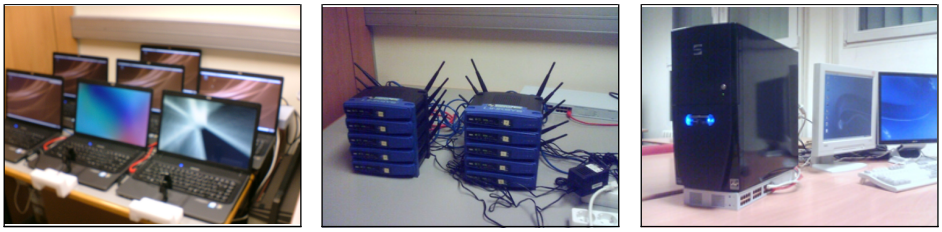
5.2.1 Routing protocol targets

For the sake of representativeness, the experimental routing protocol target considered in this case study is well-known by the ad hoc networks community and extensively used for experimentation: The Optimized Link-State Routing (OLSR) protocol. OLSR [29] is a proactive protocol which maintains routing information within network nodes to ease the quick establishment of routes among them. Network nodes continuously disseminate HELLO messages to announce their presence in their neighbourhood, and Topology Control (TC) packets to disseminate such information to the rest of the network. So as to ensure that every single node shares the same vision of the network topology, all this information is distributed by following an optimised flooding procedure. This study will consider different versions of an implementation of OLSR called *olsrd* (from *www.olsr.org*), which instruments a Link Quality extension to compute the minimum path among two nodes. Versions *v.0.4.10* (released in 2006), *v.0.5.6* (released in 2008) and *v.0.6.0* (released in 2010) have been considered for experimentation.

Moreover, an additional version implementing a Message-Digest Algorithm

5 (MD5) encryption-based mechanism was included in the case study to evaluate its effectiveness versus default versions. The proposed mechanism consists in a plugin that can be added to *olsrd v.0.6.0* (from now on *v.0.6.0+md5*). This plugin is in charge of establishing a 3-way handshaking between every pair of neighbour nodes. To verify the exchange of routing packets, a signature (or hash) is included at the end of any routing packet. Such signature is computed by cyphering, using the *md5* algorithm, the content of the outgoing routing packet with a timestamp and a 128-bits symmetric key. Such symmetric key should be known by all legitimate network nodes. As far as this signature is checked for any incoming routing packet, the protocol protects the routing integrity against malicious outsiders.

5.2.2 Experimental testbed



(a) (b) (c)

Figure 5.1: Experimental testbed consisting on 7 laptops (a), 10 routers (b) and a server controller (c).

The experimental testbed configured to deploy the experimentation required in this chapter relies on both regular and tiny devices. As Figure 5.1 shows, regular nodes were implemented by 7 HP 530 laptops with a processor of 1.6 GHz and 512 MB of RAM running Ubuntu 7.10 OS (See 5.1a). Tiny devices consisted of 10 Linksys WRT54GL routers with a processor of 200 MHz and 16MB of RAM running OpenWRT White Russian

OS (see Figure 5.1b). Considered nodes were equipped with both a wired Ethernet and a wireless IEEE 802.11b/g interface. The controller used to orchestrate the interactions between network nodes consisted, as illustrated in Figure 5.1c, in a desktop PC equipped with a 64-bits AMD Athlon processor running Debian Lenny.

5.2.3 Network profile configuration

Figure 5.2 depicts the nodes deployment for two different scenarios. The goal is showing the flexibility of REFRAHN to recreate both static (*Network A*) and mobile (*Network B*) scenarios.

As ad hoc networks performance degrades with an increasing number of hops, routing protocols tend to minimise the length of routes. So, effective routes rarely expand beyond 4-5 hops. Results from [116] show that the probability of finding a route formed by more than 4 hops in that study was less than 6%. Accordingly, the topology and mobility of the considered scenarios have been tuned to follow this trend and represent, as close as possible, the behaviour of real ad hoc networks. For instance, the topology specified for *Network A* has been defined so that nodes A and F are 3 hops distant. As *Network B* represents a scenario addressed to people, speeds ranging from 0 to 3 m/s have been considered adequate for nodes mobility. The topology evolves dynamically as depicted in Figure 5.2. For the sake of simplicity, a snapshot of this evolution is shown every 120 seconds. The number of hops along the route formed by nodes A and F ranges from 1 to 4 hops. Furthermore, it is to note that, precomputed topology changes, causes nodes involved in traffic forwarding to move away from the route, whereas new nodes appear as suitable candidates for routing. As far as these nodes are not identified yet by their new neighbours, the routing protocol requires some time to form a route again.

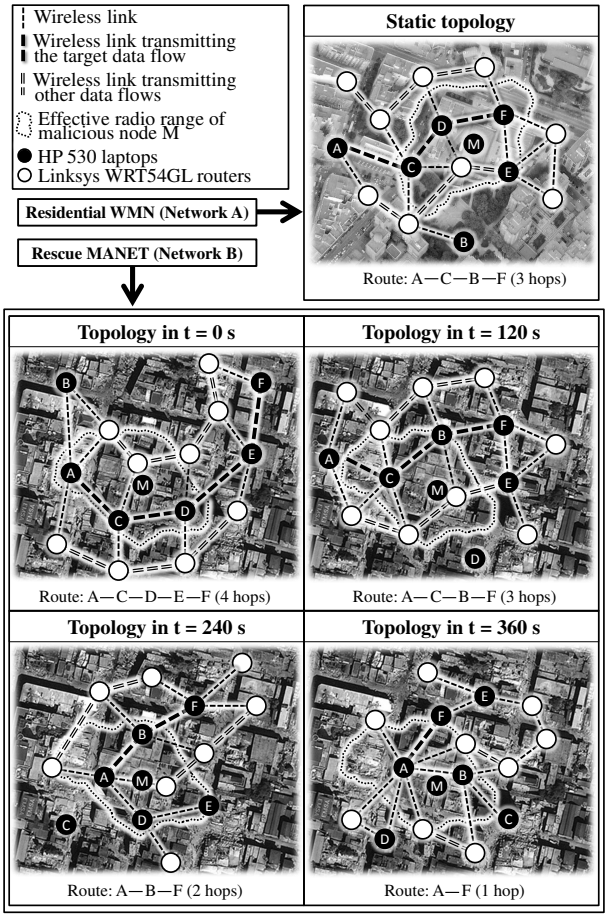


Figure 5.2: Topology evolution in considered scenarios.

5.2.4 Execution profile configuration

The execution profile defines the configuration of the workload and faultload considered in the case study.

5.2.4.1 Workload

Network A recreates the case of a WMN to provide low-cost Internet to a residential district. The applicative traffic was defined in terms of a synthetic UDP constant bit rate data flows of 200 Kbps, which is similar to the rates observed in real daily scenarios[1]. *Network B* recreates a rapid MANET deployment to assist the victims of a natural disaster. The workload defined for the *Network B* consisted in a synthetic UDP constant bit rate data flows of 2 Mbps, which was specially conceived to exchange a huge amount of information (e.g., real-time video streaming).

Three different data flows are established for each experiment. However, measurements are collected only from the data flow established along laptop nodes labelled A to F (see Figure 5.2), where the impact of monitoring probes is lower than in routers in terms of intrusiveness. The rest of the data flows are exchanged among router nodes which are 1-hop neighbours of nodes A to F. This intends to emulate the real conditions of a wireless network, where the transmission, reception and overhearing of packets are influenced by the traffic conditions imposed by nodes in the same radio range.

[1] *http://dashboard.open-mesh.com/overview2.php?id=Hillsdale*

5.2.4.2 Faultload

All the faults implemented by REFRAHN have been used in the faultload on this case study. In total, a set of 11 representative faults in the domain of ad hoc routing protocols have been introduced in the system: *signal attenuation*, *ambient noise*, *battery extenuation*, *traffic peak*, *olsrd tampering attack*, *olsrd replay attack*, *olsrd selective forwarding attack*, *olsrd jellyfish attack*, *flooding attack*, *olsrd neighbour saturation* and *olsrd sequence number replay*. It is worth noting that the *sink hole attack* was not injected alone but as a prerequisite to launch intrusion-based attacks. In our case, node M from Figure 5.2 will play the role of the injector node for all the faults but *signal attenuation* and *ambient noise*, executed by every single node because these faults typically affect a wider zone of the network. Please refer to Section 3.3.4 for more information about these fault models, and Section 4.4.5.2 for their implementation and default configuration.

It is worth mentioning that routing-protocol-dependant faults have been instantiated to *olsrd*, our target routing protocol in this case study.

5.2.5 Number and duration of experiments

Results were obtained from 1440 experiments which required 9 days for their execution. That number of experiments was defined by considering 2 network types $\times$ (1 Golden run + 11 fault models) $\times$ 4 target routing protocols (*v.0.4.10*, *v.0.5.6*, *v.0.6.0*, and *v.0.6.0+md5*), resulting in a total of 96 different experimental configurations which were executed 15 times to increase the statistical representativeness of results. Experiments lasted 9 minutes each, with 1 minute (empirically established) devoted to the warm up of the protocols, and 8 minutes for running the workload and collect the measurements.

5.2.6 Considerations about the measures selection

The set of measures selected was already presented in Section 3.2.3. However, it is important to explain some considerations referring to packet loss and delay.

To enrich the accuracy of the analysis of results, the percentage of packet loss has been categorised. We distinguish whether packets were lost due to the mobility of nodes or not, being then classified as short service interruptions ($< 15s$), and long service interruptions ($> 15s$).

The same is apply to delay. Although the delay is measured in milliseconds, it can be characterised according to its duration in different categories. Thus, we distinguish among the percentage of normal delays ($< 400ms$), percentage of long delays ($400-1200ms$) and percentage of very long delays ($> 1200ms$).

5.2.7 Analysis of results

Figure 5.3 depicts the results of experiments. Considered routing protocols have been analysed under three different perspectives: performance, resilience, and resources consumption.

5.2.7.1 Performance analysis

When considering the effect of mobility in both scenarios, an immediate conclusion from the viewpoint of performance is that, regardless the *olsrd* version used, results obtained in (static) *Network A* are generally better than those obtained in (mobile) *Network B*. Nodes in *Network B* must rebuild network routes several times (as defined in Section 5.2.3), which greatly affects those *olsrd* versions requiring more time to establish routes.

5. EXPLOITATION OF REFRAHN

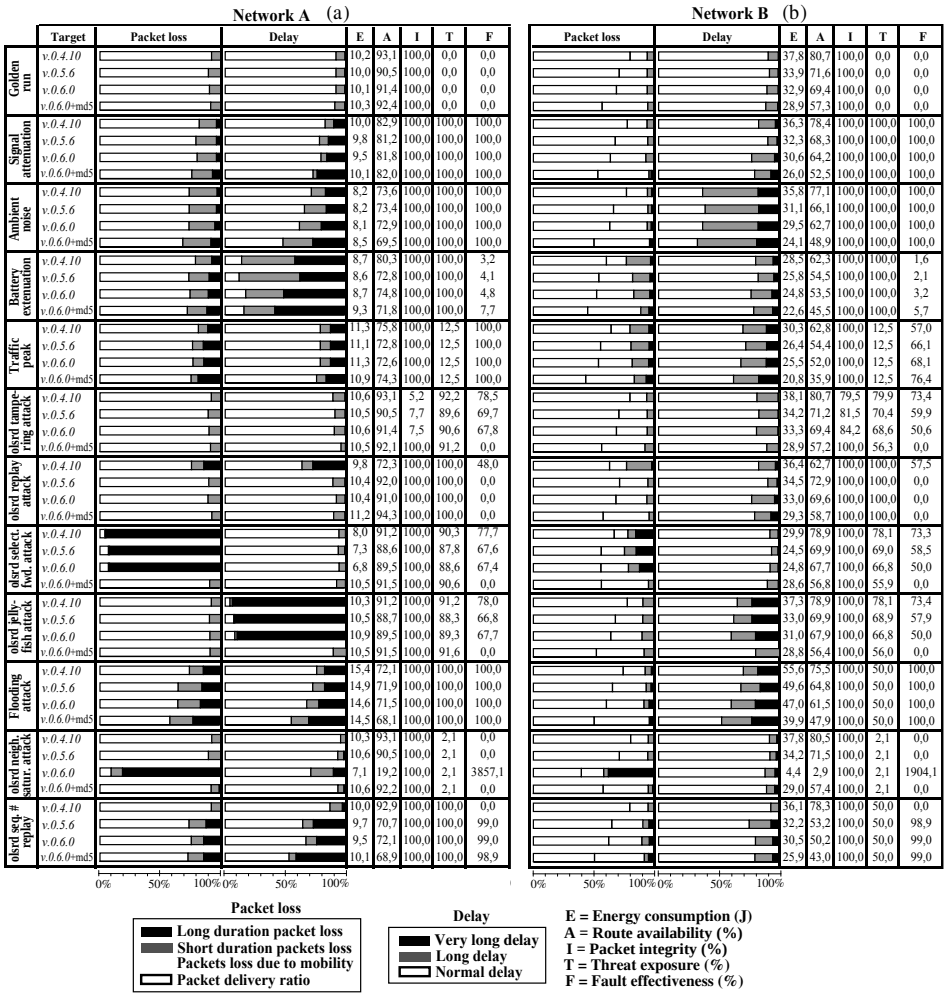


Figure 5.3: Measures obtained during experimentation.

According to the golden run phase (fault-free execution of the experiment), *v.0.4.10*, *v.0.5.6*, and *v.0.6.0* required an average time of 22.71s, 37.40s, and 40.71s, respectively, to establish a 3-hops route. However, these two last versions may be considered as equivalent, as the standard deviation overlaps. Finally, *v.0.6.0+md5* needed 59.40s to establish a route, mainly due to the underlying handshaking mechanism. The time devoted to establish a route has a significant impact on the service delivery. The longer it is, the lower the performance exhibited by the ad hoc network.

5.2.7.2 Resilience analysis

From the viewpoint of resilience, this study reflects that, in *Network B*, mobility may result either beneficial or harmful depending on the considered routing protocol version. In general, results are very similar for those threats that affect the whole network, e.g. ambient faults like *signal attenuation* and *ambient noise*. In most cases, mobility assists routing protocols to leave the area of influence of the nodes originating the fault. For example, the resilience of non-secured versions in presence of intrusion-based attacks improves as the attacker (node M) leaves the radio range of victim nodes. Consequently, despite the waste of time devoted to establishing new routes, the threat exposure rate decreased from 20 to 50 percentage points (which is proportional to the time node M was in the vicinity of the route). Then, the threat exposure gap between non-secure and secure versions is reduced in presence of mobility for intrusion-based attacks (i.e., *selective forwarding attack*, *tampering attack* and *tampering attack*). Regarding the fault effectiveness, it can be seen that threat exposure rates do not directly relate to the impact caused on the network behaviour. For example, in the case of *neighbours saturation*, the threat exposure time was only of 10s, but the effects of the fault persisted in the route even when the threat was not longer active (3857.1% of the threat exposure rate for *Network A* and

1904.1% for *Network B*). Furthermore, *v.0.6.0+md5* presents the lowest fault effectiveness for all the considered threats, but for *battery extenuation* and *traffic peak*, due to its lower adaptiveness to establish new routes.

If we focus on the effectiveness of the secure mechanism, it can be seen that intrusion-based attacks like *jellyfish*, *selective forwarding* and *tampering attacks* resulted useless against *v.0.6.0+md5*. This is specially perceptible in the delay, packet loss and integrity of *Network A*, which are a 90% worse than the golden run phase for non-secured versions. However, we can highlight that, although faults provoking long service interruptions in terms of packet loss and delay (*flooding attack*, *traffic peak*, *battery extenuation*, *ambient noise* and *signal attenuation*) impact all the considered *olsrd* versions, *v.0.6.0+md5* is, curiously, the most affected one. This result shows that despite protecting routing with cryptography, it is not exempt of performance problems, which are in some cases, paradoxically more severe than for non-secure versions.

5.2.7.3 Resources consumption analysis

The energy consumption is clearly related to the number of packets sent, received, and overheard by nodes, and the amount of information they contain. As including a 128-bits signature, the size of routing protocol packets for *v.0.6.0+md5* is larger that for the rest of versions. Furthermore, the fact of executing the 3-way handshaking increases the amount of packets exchanged between nodes. These reasons result in an increase of a 20% of energy consumption for the routing protocol traffic. Although routing protocol traffic represents less than 5% of the total energy consumed by a node, in environments like Wireless Sensor Networks, it could pose a serious problem for nodes lifetime, thus compromising the availability of the network. Regarding the applicative traffic, it must be noted that the

energy consumed by nodes along the target route increased a 42%, just by overhearing the traffic from 2 additional data flows (see Section 5.2.3). It is also worth noting the higher energy consumed by *Network B*, obviously motivated by its heavier workload. As the best possible scenario is that where nodes consume the least, readers may misunderstand the results when analysing them just from a strict energy viewpoint. For instance, some faults (like *ambient noise* or *battery extenuation*), decrease the energy consumed by nodes with respect to that consumed by the golden run phase. However, this energy saving is related to packet loss as can be deduced from correlating this information with availability (if packets are not flowing along the route, nodes do not consume any energy). Hence, although faults may benefit nodes by reducing its energy consumption, this cannot be really considered a benefit for the network, as faults affect the final service provided to the user. On the other hand, *flooding attack* and *traffic peak* are the only faults increasing the energy consumed by nodes. This increase is specially important for *flooding attack* (around 50%), thus manifesting again the importance of the overheard traffic.

5.2.8 Summary

Far from the pre-established idea that prevention mechanisms increase the resilience of a system in general, this case study shows that such statement depends on the context of use of the network. A particular security mechanism enhancing the routing protocol robustness in a particular context (e.g. a static network) does not necessarily ensure the same degree of robustness in a different context. In fact, as the *olsrd* version secured with a *md5* encryption shows, such mechanism may not only decrease the network performance (which could be acceptable with the goal in increasing resilience), but it can also decrease the resilience of the routing protocol in presence of certain faults or network conditions. According to our results,

for instance, *olsrd v.0.6.0+md5* provides a good protection against faults (mainly attacks) requiring route intrusion (*sink hole attacks*). However, for other types of faults (like, for example, *ambient noise*, *flooding attack* or *traffic peak*) the time required by the encryption-based handshaking mechanism is so long that degrades the protocol performance and its resilience. Consequently, it seems important that, in order to be useful in a wide variety of contexts, routing protocols make a step beyond routing adaptiveness by integrating adaptiveness in their prevention mechanisms.

5.3 Vulnerability discovery

The presence of vulnerabilities in ad hoc networks, and more precisely in routing protocols is something indisputable. They are an open door enabling the activation of faults that can compromise the behaviour of the whole network. Because of this, the detection (in time) of vulnerabilities in routing protocols is essential to guarantee the confident use of system.

The analysis of experimental results issued from resilience evaluation may be very useful to carry out the discovery and assessment of vulnerabilities in practice. Thus, the results previously obtained by REFRAHN in Section 5.2.7 could be interesting to analyse curious or unexpected phenomena in the behaviour of the routing protocols targeted, possibly hiding the presence of important vulnerabilities.

Following this line, this section shows three interesting results that could disclose the presence of vulnerabilities in some of the routing protocols evaluated in Section 5.2. More precisely, Subsection 5.3.1 reveals a possible vulnerability in *olsrd v.0.4.10* related to the acceptance of routing packets already received. Subsection 5.3.2 states a possible vulnerability in *olsrd v.0.4.10* that enables the creation of links with a reduced exchange of rout-

ing packets. Subsection 5.3.3 shows a possible vulnerability in *olsrd v.0.6.0* based on a wrong management of the neighbour tables. Finally, Subsection 5.3.4 presents some conclusions.

5.3.1 Accepting routing protocol packets with a replayed sequence number

Results from Section 5.2.7 showed a significant packet loss in the network when subjecting *olsrd v.0.4.10* to the presence of a *replay attack* (around 20% higher with respect to the golden run phase). However, the surprising result is that the rest of the versions were not affected at all.

After studying the source code of the different versions in more detail, we found that *v.0.4.10* was the only one that does not implement a mechanism to reject packets whose sequence number has been already received and, thus, takes them into account to establish a (probably fake) route. *v.0.5.6*, and *v.0.6.0* versions (as expected) reject packets with duplicated identifiers emitted by the injector node M, thus relying on other nodes to establish the target route and increasing the delay. However, due to that vulnerability, routing packets with a duplicated identifier are not rejected by *v.0.4.10* but taken into account to establish a (probably fake) route.

Paradoxically, *olsrd v.0.4.10* benefits from this vulnerability when the network is subjected to the presence of a *sequence number replay* fault. The fact that the rest of the content of replayed packets is legitimate leads *olsrd v.0.4.10* to process a wider amount of information with respect to the rest of versions (that directly reject the packets), which involves an important improvement in the overall network behaviour (see this effect on Figure 5.3).

5.3.2 Establishing links with a reduced number of actions

The evaluation results obtained from Section 5.2.7 showed an interesting result. Surprisingly, the oldest version of *olsrd* (*v.0.4.10*) required the shortest time to establish a 3-hops route (22.71s), which represents from 15 to 18 seconds less than recent versions (*v.0.5.6* and *v.0.6.0* respectively), despite all of them were executed with an identical configuration. Although on one hand this result improves the performance of *olsrd v.0.4.10*, on the other, the protocol is less robust against the occurrence of intrusion attacks.

To study more in detail this phenomenon, Figure 5.4 presents the instantiation of the *sink hole attack* for *olsrd*. The injector node (node M) induces a propitious network topology for the attack success. To cope with that goal, the injector node induces a possible routing link between targeted victim nodes (call them nodes A and C) by faking HELLO and TC olsrd messages. Such messages enable nodes to determine optimal routes for their communications. Thus, the injector node forces its intrusion in the route by (maliciously) misusing the aforementioned messages. It starts injecting in the network HELLO messages declaring victim nodes as neighbours, and TC messages to announce the links with them. To obtain a symmetric link, the injector node needs victim nodes to generate TC messages announcing reciprocal links. To fake the generation of these messages at the victims side, the injector node forges fake TC messages announcing such links. It sets the victim's address in the originator field of the TC message.

Experiments showed that, in all considered cases, omitting one of these basic actions prevented the malicious node from intruding the route but in one particular case: when acknowledgement from intrusion target was not broadcasted.

Table 5.1 shows which was the content of routing tables of the nodes selected as the intrusion point (A and C) for *olsrd v.0.5.6* and *olsrd v.0.6.0* when

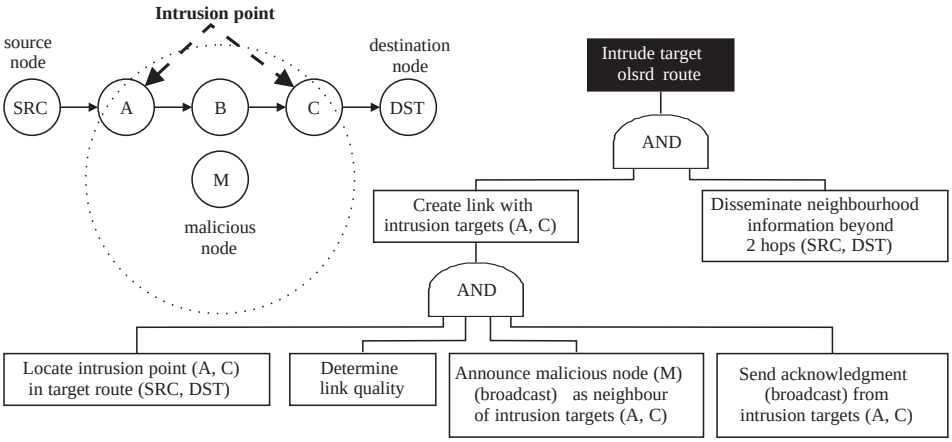


Figure 5.4: OLSR route intrusion procedure.

omitting the acknowledgement from intrusion targets. It must be noted that the best possible value for the field *link quality* is 1.00, whereas INF (infinity) states that there is not a link between these nodes (please refer to Section 2.2 for more details about the computation of link quality). In this case, although each common node creates a link with the malicious one, they are not aware of the link created by the other node. This is the behaviour defined in the protocol specification.

Table 5.1: Routing tables for *olsrd v.0.5.6* and *olsrd v.0.6.0*.

Node A			***Node C***		
DST	**Next**	**LQ**	**DST**	**Next**	**LQ**
A	M	1.00	C	M	1.00
C	M	INF	A	M	INF

DST = destination node, LQ = link quality

Nevertheless, routing tables of the target common nodes for *olsrd v.0.4.10* (see Table 5.2) show very different results: the route A $\leftrightarrow$ B $\leftrightarrow$ C has been dismissed in favour of a new route A $\leftrightarrow$ M $\leftrightarrow$ C, since these new links present the best possible link quality. This procedure reveals a possible

vulnerability, according to the protocol specification, in the implementation of this version.

Table 5.2: Routing tables for *olsrd v.0.4.10*.

Nodes A and C					
Before intrusion			***After intrusion***		
DST	**Next**	**LQ**	**DST**	**Next**	**LQ**
A	B	2.34	A	M	1.00
B	A	1.68	M	A	1.00
B	C	1.57	M	C	1.00
C	B	1.52	C	M	1.00

DST = destination node, *LQ* = link quality

In other words, *olsrd* version *v.0.4.10* is not fully OLSR compliant, while *olsrd v.0.5.6* and *olsrd v.0.6.0* are. This can be considered as a vulnerability in protocol version *v.0.4.10* that can be exploited by attackers to intrude a route more easily (with a reduced set of actions). However, the side effect seems to be a reduction in the time required to create new communication links, which explains the better packet delivery ratio of this version with respect to the rest of considered versions (see Figure 5.3).

5.3.3 Incorrect management of the neighbour lists of routing protocols

A deeper analysis of the results obtained from Table 5.3 revealed an interesting result concerning the occurrence of *neighbours saturation*. More precisely, in presence of this fault, versions *v.0.4.10*, *v.0.5.6*, and *v.0.6.0+md5* (who rejects non-encrypted routing packets) behave as the golden run phase. However, when disabling the encrypted mechanism (*v.0.6.0*), links saturation causes a dramatic packet loss of 81%.

At first sight, it seems reasonable that the problem could be exclusively related to the massive announcement of new links but, some additional ex-

perimentation, showed that there were more things to analyse. In order to determine the precise behaviour of the protocol in presence of neighbour saturation, a simple experiment with just one common node and one injector node was performed. Surprisingly, the common node was not affected at all by the packet sent and continued its normal operation without any problem. After analysing the traffic generated from several experimentations considering a progressive growing number of common nodes, we could determine that it was not the announcement of new nodes what directly caused the instability of the routing protocol, but the ulterior massive exchange of these links among legitimate neighbour nodes.

The effect induced by a high density of neighbours in the vicinity of a given node can be characterised by its failure mode, thus it is possible to distinguish three different failure modes: normal behaviour, hang or crash. Typically, hang mode can be identified because despite the protocol remains operating, its communication capabilities to send and receive routing packets has been disabled persistently. Conversely, crash mode directly involves stopping the protocol execution.

In order to identify the occurrence of previous failure modes, different probes have been introduced in the system. Concretely, to monitor whether the process associated to the routing protocol is alive, the *top* tool was used. Furthermore, an *olsrd*'s heartbeat plugin was instantiated to determine when the protocol hangs. In essence, this plugin periodically saves the system timestamp while the protocol works properly. So, by comparing the last watchdog timestamp with the current system timestamp, it is possible to determine whether *olsrd v.0.6.0* hanged during the experimentation, and when. Table 5.3 summarises considered failure modes.

Once the failure modes defined and how to identify them in practice, REFRAHN was configured to launch different experimental campaigns to study the impact of neighbour saturation in more detail. More than 200

Table 5.3: Characterisation of failure modes for *neighbour saturation* faults.

Failure mode	Routing process under execution (*olsrd* process running)	Routing capabilities enabled (heartbeat timer running)
Normal behaviour	Yes	Yes
Hang	Yes	No
Crash	No	No

experiments were executed considering 3 rates of massive exchange of routing links (100, 300, 500), representing the advertisement of new nodes in the neighbourhood.

The results obtained are shown in Figure 5.5. Increasing the number of announced new links from 100 to 300 caused *olsrd v.0.6.0* to crash in 60% of the experiments with just 4 nodes in the vicinity, whereas announcing 500 new links just collapsed the network and increased the percentage of hangs. So, it can be observed that networks with an increasing number of nodes are very likely to cause stability problems in the protocol, which may result in either hang or crash, depending on the density of nodes in the same neighbourhood. So, not only the size of the network is important, but the density of nodes in the same neighbourhood is critical. Experiments performed in a worst-case scenario consisting in a neighbourhood of 10 legitimate nodes with an increasing number of announced links, showed that protocol hangs begin to appear just by announcing 50 new links.

Although the origin and the effect of the problem is determined, the error propagation mechanism that leads to those particular failure modes remained unexplored. To increase our understanding of this error mechanism, experiments were repeated using the *ddd* debugging tool. After this process, the critical section where the fault (the massive announcement of neighbours) becomes (either a crash or hang) failure could be determined within the source code. Given the link-state nature of OLSR, *olsrd v.0.6.0*,

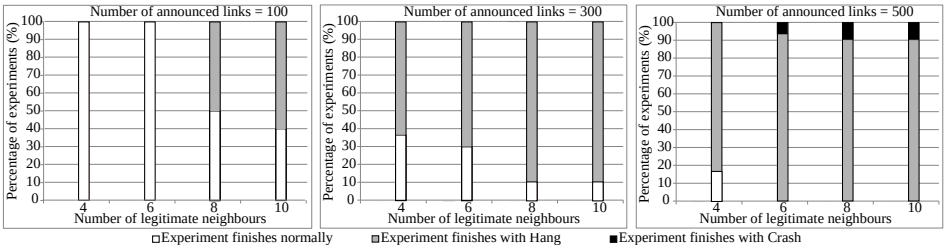


Figure 5.5: Impact of *neighbour saturation* according to the number of nodes and the new links announced.

constructs a full map of the connectivity to the network, showing how nodes are connected. Indeed, after analysing the obtained traces, the problem was tracked down to the management of the neighbours table when it is big enough (*olsr_{lookup, delete, insert}_neighbor_table* functions).

5.3.4 Conclusions

This section has shown the exploitation of REFRAHN for the vulnerability discovery.

It is worth noting that not only newest release implementations are subjected to the presence of flaws. Vulnerabilities can be present even in stable versions that, theoretically, have had more time to be tested. This fact has been corroborated from the vulnerabilities disclosed in this section. Two different vulnerabilities concerning the older implementation of *olsrd v.0.4.10* were discovered, while just one was reported for *olsrd v.0.6.0*.

Despite falling out of this thesis, the exploitation of vulnerabilities related to other routing protocols has been also taken into account. More precisely, a vulnerability exploiting the intrusion in AODV, a reactive routing protocol, was explored in [117].

5.4 Resilience benchmarking

Resilience benchmarks are well-specified procedures which enrich the notion of traditional performance benchmarking to enable the objective evaluation, comparison and selection of components and systems according to their dependability in the presence of changes. This fact increases the complexity of conventional benchmarks. Indeed, in addition to the different considered benchmark targets, the workload required to exercise the system, and the performance measures defined to characterise the system's behaviour, it is also necessary to establish the faultload and the set of measures to characterise the system reaction against considered changes, like mobility and the occurrence of faults [104].

From a practical viewpoint, these questions lead to some serious challenges in their ulterior analysis. The problem appears when the combination of multiple factors (like different workloads, faultloads, or benchmarking targets) within the same experimentation, requires taking into account a wide set of heterogeneous measures. This may lead to a potential explosion in the number of results that hinders the correlation of such results and, more importantly, the inference of any conclusion and/or ranking from the performed experimentation. Even if the effort is performed, the analysis and interpretation of results remains a complex and error-prone process requiring a very deep resilience expertise, which limit the interest of resilience benchmarking for many engineers. The main problem underlying this situation relates to the fact that most benchmarks limits its purpose to the obtention of measures, providing only some guidelines on how such measures can be later used to infer a target ranking or quality score.

Measures aggregation is a common approach trying to (i) enable meaningful comparisons among systems and (ii) ease the analysis of evaluated systems or components. However, although these techniques are usually applied in

the community of benchmarking, it is surprising that so far there is still a lack of unified criteria when addressing the aggregation of measures and their subsequent analysis. This leads to situations where different conclusions can be issued from the analysis of the same results, due to the fact that, without any systematic approach for the measures analysis, different benchmarking users can potentially apply different aggregation strategies and even perform a subjective interpretation of provided scores. Accordingly, open questions requiring further research in the domain of resilience benchmarking are (i) how to systematically aggregate such measures to capture in a single or small set of scores the information required to characterize the overall system quality, (ii) how to ensure the consistency of interpretations issued from the use of such scores with respect to the conclusions obtained from the direct analysis of measures, (iii) how to capture enough information to enable the hierarchical analysis of measures from coarse to fine-grained measures (and vice-versa), and (iv) how to do it in a way which is easy to explain and interpret.

Aforementioned questions can be addressed through the definition of a quality model (as already discussed in Section 3.2.3). A quality model requires the specification of benchmark measures and the complex relations between them and system requirements. The underlying mathematical tool that will support our approach is called Logic Score of Preferences (LSP) [118], which mathematically models the quality requirements of the system for the system evaluation, comparison, and selection. Beyond the mere use of LSP as a tool to support the measures aggregation, this section shows how conventional resilience benchmarks (in our case in the domain of ad hoc routing protocols) must evolve to exploit all the power offered by LSP.

The rest of this section is structured as follows. Subsection 5.4.1 reports the different alternatives for measures aggregation and analysis. Subsection 5.4.2 introduces the LSP technique. Subsection 5.4.3 applies the LSP

technique to analyse and interpret the results obtained from three out of the four implementations of *olsrd* considered in Section 5.2. Subsection 5.3.4 ranks such results from a resilience benchmarking viewpoint. Finally Subsection 5.4.5 presents conclusions.

5.4.1 Measures aggregation approaches

The literature offers different graphic and analytic alternatives to synthesise the measures obtained during the evaluation of a target system.

Kiviat or radar diagrams [119] are a graphical tool which represent the results of the benchmark in an easy-to-interpret footprint. Kiviat diagrams can show different measures using only one diagram and, although some training is required, the comparison of different diagrams is fairly simple. The scalability of Kiviat diagrams enables the representation of up to tens of measures. However, managing such a huge amount of information may make difficult the interpretation and analysis of results.

The problem previously stated is solved in [119] throughout the use of an analytical technique named the *figure of merit*, which imposing certain restrictions to the graph axes, synthesises all the measures into a unique value related to the footprint shape. However, the problem associated to this solution, as it happens with most techniques using the mean or the median, is that valuable information could be hidden behind a unique number, and consequently, the comparison between protocols could result quite vague [120].

Other approaches, like the presented in [121], characterise measures according to their ability to fit within a particular statistical distribution. Nevertheless, this approach presents two main drawbacks. First, it assumes that a measure follows the same distribution for all the systems, which may not be true depending on the context of use. And second, to

understand this type of characterisation, it is necessary to understand the assumed statistical model, which is not straightforward.

Finally, other authors, like Al-Sbou [122], propose the use of custom formulas for the aggregation of measures obtaining a single score which characterises the behaviour of the system. However, these formulas are based on heuristics and lack formal foundation and validation.

In sum, these techniques lack the ability of aggregating measures into a meaningful result that: (i) is easy to explain and interpret; (ii) is representative of real systems and allows their comparison and ranking, and (iii) captures enough information to enable the hierarchical analysis of measures from coarse to fine-grained measures (and vice-versa).

Aforementioned issues can be addressed through the definition of a quality model. A quality model is a (simple or complex) hierarchical abstraction of the quality requirements of the system that synthesises the attributes from performance, resilience and resources consumption into one single score representing the global quality of the system. This approach can assist evaluators to guide the analysis of results from a coarse- to a fine-grain. The underlying mathematical principles of the quality model proposed in this section rely on the Logic Score of Preferences (LSP) technique.

5.4.2 Logic Score of Preferences

The LSP technique computes the global score of a system through the recursive decomposition of their characteristics into subcharacteristics and so on, until obtaining quantifiable attributes (or measures). However, what makes it interesting with respect to to the rest of approaches presented in Section 5.4.1 from a resilience benchmarking viewpoint, is its capability to navigate from the fine-grained measures to the coarse-grained scores, without losing the numerical viewpoint of results. Thus, keeping the con-

sistency in the interpretation and analysis of results independently from the viewpoint (fine or coarse) acquired by the benchmark user.

In general terms, LSP computes the global score (S) of a system using Formula 5.1. This formula is a generalisation of the Pythagorean means including arithmetic, geometric and harmonic means.

$$S = (\sum_{i=1}^{k} w_i s_i^r)^{\frac{1}{r}} \quad | \quad \sum_{i=1}^{k} w_i = 1 \tag{5.1}$$

In Formula 5.1, s_i represents the elementary score (also referred to as elementary preference) of the k (sub)characteristics of the targeted system. Each elementary score can be seen as an abstraction of the quality observed for each system characteristic. The difficulty is in how to quantitatively obtain each s_i. For this purpose, the LSP technique uses criterion functions to establish an equivalence between the measures obtained from the system (also referred to as system attributes a_i) and the system quality requirements. The output of such criterion functions is s_i, a normalised score within a 0-to-100 scale, where 0 is the worst and 100 is the best possible value. Thus, each s_i can be interpreted as the degree of satisfaction of an attribute a_i with respect to the quality requirements specified by the benchmark performer for such attribute. Since all the attributes are scored according to the same scale, resulting elementary preferences are directly comparable. Such equivalence can be mapped to a discrete function, thus establishing different quality levels, or to a continuous one.

Once all the elementary scores belonging to the same characteristic computed, it is possible to aggregate them into an aggregation block to obtain one representative score for such system characteristic. As far as the k elementary preferences that compound the aggregation block defined by each characteristic may not have the same importance, a weight w_i, illustrating

such influence, must be assigned to each elementary preference within the same hierarchical level. The summary of weights must be equal to one.

In this line, it is also necessary to define the degree of mandatoriness that must be fulfilled for each aggregation block. The power r, described in detail in [118], represents one logic operator in charge of defining the type of relationship (from orness to andness) required for the different elementary scores within the same aggregation block. In [118], the author defines up to 20 different logic operators which describe a mandatoriness gradation among the requirements of the system. Such gradation ranges from the full conjunction (logic AND) which illustrates the *simultaneity* among all the requirements, to the full disjunction (logic OR) which represents the notion of *replaceability*, where meeting just one requirement is enough (see Figure 5.6).

Operation	Symbol	r2	r3	r4	r5
DISJUNCTION	D	+infty	+infty	+infty	+infty
STRONG QD (+)	D++	20.630	24.300	27.110	30.090
STRONG QD	D+	9.521	11.095	12.270	13.235
STRONG QD (-)	D+-	5.802	6.675	7.316	7.819
MEDIUM QD	DA	3.929	4.450	4.825	5.111
WEAK QD (+)	D-+	2.792	3.101	3.318	3.479
WEAK QD	D-	2.018	2.187	2.302	2.384
SQUARE MEAN	SQU	2.000			
WEAK QD (-)	D--	1.449	1.519	1.565	1.596
ARITHMETIC MEAN	A	1.000	1.000	1.000	1.000
WEAK QC (-)	C--	0.619	0.573	0.546	0.526
WEAK QC	C-	0.261	0.192	0.153	0.129
GEOMETRIC MEAN	GEO	0.000			
WEAK QC (+)	C-+	-0.148	-0.208	-0.235	-0.251
MEDIUM QC	CA	-0.720	-0.732	-0.721	-0.707
HARMONIC MEAN	HAR	-1.000			
STRONG QC (-)	C+-	-1.655	-1.550	-1.455	-1.380
STRONG QC	C+	-3.510	-3.114	-2.823	-2.606
STRONG QC (+)	C++	-9.060	-7.639	-6.689	-6.013
CONJUNCTION	C	-infty	-infty	-infty	-infty

Figure 5.6: Aggregation operators proposed by Dujmović, and r value for 2, 3, 4 and 5 inputs for the aggregation block.

The main concepts of LSP are illustrated throughout Figure 5.7. The aggregation process is repeated recursively grouping more and more general characteristics until obtaining a global score of the system. Thus, a coarse-

grained analysis of each benchmarking target can be performed. Then, the analysis can be progressively refined using the available intermediate scores until considering the appropriate level of analysis. The benefit of using LSP is on systematising the way in which scores are obtained from measures, naturally establishing a hierarchical approach for their analysis. Following subsections are devoted to apply the notions of LSP to a subset of the results obtained in the case study presented in Section 5.2.

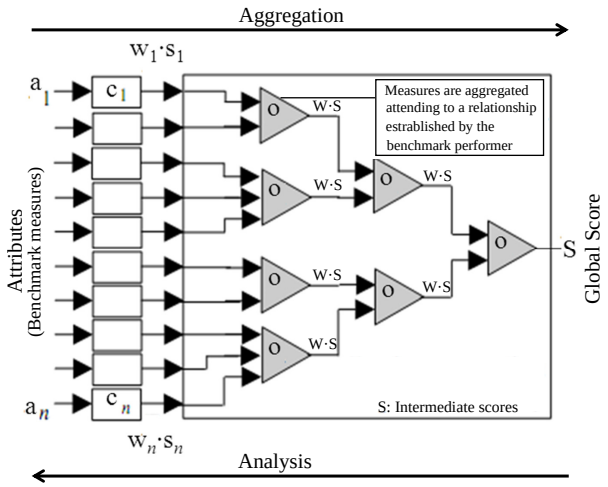


Figure 5.7: Representation of the LSP technique.

5.4.3 Experimental consideration of the case study

First, we need to specify the experimental conditions to be considered in the resilience benchmarking. Thus, three out of the four versions of *olsrd* routing protocol have been selected as benchmark targets: *v.0.4.10*, *v.0.5.6* and *v.0.6.0*. Concretely, the scenario considered to deploy this study was *Network A*, a static WMN. To complete the specification of the benchmark, it would be desirable to contextualise the scenario. So, let us imagine that

such WMN provides support for a given voting application used by the audience of a conference (about 50 people) to give real-time feedback to the speaker. The nature of such application requires a high availability and a low delay from the underlying ad hoc network to provide a quick response to the speaker, while ensuring the maximum level of integrity in the vote emitted by conference attendees.

Given the applicative context considered, the occurrence of some accidental faults or attacks is much more probable than the occurrence of others. For example, considering the presence of a potential attacker within the audience trying to disturb the communications is more probable than suffering a battery extenuation, since devices can be connected to the electric line. So, from the wide variety of faults considered by REFRAHN while running the application (refer to Section 4.4.5.2 for more details), we have selected a subset of the 5 most harmful ones. Consequently, we have considered *ambient noise* because it is quite common that the attendees' devices are simultaneously executing other tasks requiring the exchange of additional traffic, which may create interferences with the target voting application; Alternatively, intrusion-based attacks such as *selective forwarding*, *jellyfish* and *tampering* should be taken into account given the sensitiveness of the information exchanged by the application. Finally, launching a *flooding attack* would not be difficult for an attacker, and may also compromise the requirements of the application previously presented. Table 5.4 summarises the list of faults considered.

Table 5.4: Faults considered during the experimentation.

Fault	**Type**	**Origin**
Ambient noise (A)	Accidental	Natural
Selective forwarding attack (S)	Malicious	Human-made
Jellyfish attack (J)	Malicious	Human-made
Tampering attack (T)	Malicious	Human-made
Flooding attack (F)	Malicious	Human-made

Then, it is necessary to define the different measures that will be used to assess the quality of the considered benchmark targets.

5.4.3.1 Measures selection

The goal of the measures selection is to characterise the quality of the system through a complete and not redundant set of elemental attributes or measures (a_1 to a_n). This set defines a block and can contain a different amount of attributes. The blocks composition continues grouping different characteristics until the global score of the system is computed.

For the purpose of this case study we consider three characteristics: performance, resilience and resources consumption. In addition, different attributes have been identified for each characteristic (as depicted in Figure 5.8) in order to refine the proposed model. Concretely, given the application requirements previously introduced in Section 5.4.3, it seems essential to analyse the delay experienced in the communication between the attendees and the speaker, as well as the route availability to determine if attendees were able to send their votes, and the packet integrity to estimate the percentage of legitimate votes received by the speaker. Packet loss and energy consumption are additional attributes to complement the quality model. More details about these attributes can be obtained from the complete list of measures proposed by REFRAHN in Table 3.1.

Resulting attributes will be the fine-grained measures we are able to obtain throughout our benchmarking experiments.

5.4.3.2 Fine-grained experimental results

Table 5.5 shows the results obtained from experimentation. However, estimating and comparing the impact of the selected faults on each single mea-

Ad hoc routing protocol
1. Performance
   1.1. a_1: Packet loss (%)
   1.2. a_2: Delay (ms)
2. Resilience
   2.1. a_3: Route Availability (%)
   2.2. a_4: Packet Integrity (%)
3. Consumption
   3.1. a_5: Energy consumption (J)

Figure 5.8: LSP hierarchy illustrating the case study.

Table 5.5: Measures considered for the case study.

Target	Fault	Packet loss (%)	Delay (ms)	Route availability (%)	Packet Integrity (%)	Energy (J)
v.0.4.10	*A*	27.4	48.2	73.6	100.0	8.2
	S	93.5	42.0	91.2	100.0	8.0
	J	7.6	1966.2	88.7	100.0	10.3
	T	8.2	39.7	93.1	5.2	10.6
	F	25.5	62.9	72.1	100.0	15.4
v.0.5.6	*A*	27.3	55.6	73.4	100.0	8.2
	S	90.8	55.1	88.6	100.0	7.3
	J	9.8	1811.4	88.7	100.0	10.5
	T	9.9	39.9	90.5	7.7	10.5
	F	27.1	64.5	71.9	100.0	14.9
v.0.6.0	*A*	27.1	52.3	72.9	100.0	8.1
	S	90.1	53.4	89.5	100.0	6.8
	J	8.7	1798.1	89.5	100.0	10.9
	T	9.4	56.5	91.4	7.5	10.6
	F	26.6	66.6	71.5	100.0	14.6

sure is a complex task given the lack of criteria to determine the thresholds which separate the correct from the incorrect behaviour of the network in terms of the considered measures. Obviously, this requires an effort we already experienced when analysing the results shown in Table 5.3. Additionally, it is worth noting that measures cannot be independently analysed, since this could lead the evaluator to misleading conclusions, e.g., although some faults like *ambient noise*, and *selective forwarding* attack may benefit nodes by reducing its energy consumption, this cannot be really considered a benefit for the network, as such faults affect the final service provided to the user. This fact may favour that the more measures we consider, the

more difficult to obtain an accurate global vision of the fault impact on the protocol.

Normalising the value of these measures can be useful to ease their comparison.

5.4.3.3 Definition of criterion functions

In order to simplify the application of LSP in our case study, we have considered two generic continuous criterion functions, one increasing (see Figure 5.9), to compute the elementary score of the-higher-the-better measures such as route availability and packet integrity, and one decreasing (see Figure 5.10), to compute the-lower-the-better measures such as packet loss, delay and energy consumption.

$$s_i = c_i(a_i) = \begin{cases} 0, & a_i \leq X_{min_i} \\ 100\dfrac{a_i - X_{min_i}}{X_{max_i} - X_{min_i}}, & X_{min_i} < a_i < X_{max_i} \\ 100, & a_i \geq X_{max_i} \end{cases}$$

Figure 5.9: Increasing criterion function used in the case study.

$$s_i = c_i(a_i) = \begin{cases} 100, & a_i \leq X_{min_i} \\ 100\dfrac{X_{max_i} - a_i}{X_{max_i} - X_{min_i}}, & X_{min_i} < a_i < X_{max_i} \\ 0, & a_i \geq X_{max_i} \end{cases}$$

Figure 5.10: Decreasing criterion function used in the case study.

These functions are parameterised according to the bounds (X_{min} and X_{max}) that delimit the quality threshold for a given attribute. These thresholds are useful to contextualise selected measures with respect to meaningful values in the applicative domain which ease their interpreta-

tion at the analysis stage.

The increasing criterion function applies to those measures whose quality increases as their value does. In such function, X_{min} establishes a threshold below which the value of the measure is a 0% quality, while X_{max} defines the threshold from which that value is a 100% quality. The reverse interpretation applies to the decreasing criterion function.

In order to determine the aforementioned thresholds, the evaluator should consider all the network requirements for an acceptable quality of communications, addressing aspects like the number of users, the communication technology, the context of use and the type of service delivered. Depending on such aspects, the measures obtained may be interpreted in one way or another. Given the nature of the voting application used for this benchmark, let us consider the requirements defined in [123] for an application addressed to enable lecturers to get feedback from audience in auditoriums or convention centres. Regarding the performance constraints provided by authors, a packet loss lower than 10% ($X_{min} = 10\%$) would represent an excellent quality whereas a packet loss of 45% ($X_{max} = 45\%$) would be in the bounds for an acceptable communication. In the case of delay, the common limits of a medium quality communication [124] range from $40ms$ to $400ms$ ($X_{min} = 40ms$ and $X_{max} = 400ms$ respectively). As far as measuring route availability, packet integrity and energy consumption was not a requirement in [124], such thresholds are estimated by the author of this thesis. Let us consider an acceptable route availability between $X_{min} = 75\%$ and $X_{max} = 90\%$ ("one nine") where a downtime of 100 milliseconds per second is good enough for the type of application considered. Concerning the packet integrity, a data corruption affecting more than 1% of packets can compromise the trustworthiness of a voting application. Accordingly, the thresholds were fixed to $X_{min} = 99\%$ and $X_{max} = 100\%$. Finally, regarding the energy consumption, a previous fault-free experimentation showed

a stationary consumption of 10J. In order to compute the thresholds, we assigned this value to the minimum threshold ($X_{min} = 10J$) and considered that an increase of 50% would be enough in our case to compute the maximum threshold ($X_{max} = 15J$). Table 5.6 summarises these thresholds.

Table 5.6: Experimental quality thresholds considered in criterion functions.

Measure	**Criterion function**	X_{min}	X_{max}
Packet loss (a_1)	Decreasing	10%	45%
Delay (a_2)	Decreasing	$40ms$	$400ms$
Route availability (a_3)	Increasing	75%	90%
Packet integrity (a_4)	Increasing	99%	100%
Energy consumption (a_5)	Decreasing	$10J$	$15J$

5.4.3.4 Aggregation of scores

In order to simplify our case study, all the attributes (a_i) have been considered equally significant, thus performing a fair assignment of weights. However, as far as a low consumption is not usually a mandatory requirement in fixed networks such as WMNs, we have reduced its weight to 10% (0.1 out of 1). Remaining 90% (0.9 out of 1) is equally shared between performance (0.45 out of 1) and resilience (0.45 out of 1), which are much more important in this case. If we had considered e.g., a WSN, where the reduction of battery consumption is a must, then the weight assigned would be much more significant. The weight assignment of system attributes and characteristics is modelled in Figure 5.11.

Regarding the aggregation relationship between the attributes and characteristics considered in our case study, we have determined that all of them should be compliant to the thresholds defined. One of the functions which fits the best with this requirement is the weak quasi-conjunction (denoted as $C-$ according to the LSP notation), represented by $r = 0.261$ for aggre-

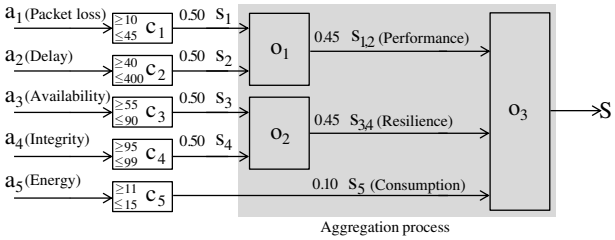


Figure 5.11: Complete quality model applied in our case study.

gation blocks with two inputs. This operator denotes 60% of mandatoriness within the operators scale proposed by Djumović.

Once obtained the global score of the system in presence of one fault type, the same reasoning can be applied to aggregate such scores into a global score of the system in presence of faults. In our case, all the faults have been considered equally important. In the same line, the system was required to achieve a minimal degradation in presence of all the faults. The same $C-$ operator used previously was used again in the aggregation of faults.

5.4.4 Hierarchical ranking of results

The multiple intermediate results obtained when applying the LSP technique enable to systematise the analysis of the benchmarking results at different levels. Table 5.7 lists the scores that represent the quantitative quality, globally and for each selected characteristic, of the considered protocol versions in presence of representative faults.

5.4.4.1 Ranking per global quality

According to these results (presented in a 0-to-100 scale), the best candidate to be integrated into the final deployment are either *olsrd v.0.5.6* and *olsrd*

Table 5.7: Results obtained from applying the LSP technique.

Target	Fault	Performance	Resilience	Consumption	Global score per fault	Global score Perform.	Global score Resil.	Global score Consum.	Global score
v.0.4.10	A	71.73	73.85	100.00	75.23	36.24	32.12	62.15	41.81
	S	5.76	100.00	100.00	32.46				
	J	9.20	100.00	100.00	38.85				
	T	99.49	7.15	100.00	37.73				
	F	72.11	71.08	0.00	47.91				
v.0.5.6	A	70.28	73.49	100.00	74.38	40.22	32.55	61.97	46.62
	S	8.81	97.98	100.00	36.89				
	J	8.90	98.13	100.00	37.44				
	T	98.42	7.15	100.00	37.50				
	F	69.44	70.70	2.50	55.71				
v.0.6.0	A	71.21	72.57	100.00	74.40	41.70	31.95	66.43	47.47
	S	8.88	99.28	100.00	37.28				
	J	8.90	99.28	100.00	37.69				
	T	97.69	7.15	100.00	37.34				
	F	70.51	69.94	10.00	60.09				

v.0.6.0, as they maximise the global score (47.47 points and 46.62 points respectively), whereas the remaining version (*olsrd v.0.4.10*) presents lower quality, 41.81 points. These results illustrate, in a certain way, the influence of the vulnerability previously detected in *olsrd v.0.4.10* in Section 5.3.2, that favours the intrusion of attackers. This fact states that, regardless the coarse- or fine-grained viewpoint of the analysies, results are solid.

5.4.4.2 Ranking per characteristic

Alternatively, evaluators may be interested in just focusing on one particular characteristic to evaluate the quality of the protocols and, in that case, *olsrd v.0.5.6* and *olsrd v.0.6.0* obtain the best scores for performance (41.70 points and 40.22 points respectively), *olsrd v.0.6.0* is the best option with respect to consumption (66.43 points) and there is a triple draw (around 32 points) if taking resilience into account.

5.4.4.3 Ranking per fault type

If we focus on the behaviour of each protocol in presence of a particular fault, *olsrd v.0.5.6* and *olsrd v.0.6.0* (in no particular order) take the lead when subjected to *selective forwarding attacks*. Regarding the impact of *flooding attacks*, version *olsrd v.0.4.10* presented the best results. *Jellyfish attacks*, *tampering attacks* and *ambient noise* impact all the considered protocols in the same way and no one could be taken as the best option to face that particular fault.

5.4.4.4 Summary

Following this analysis, the quality of *olsrd v.0.5.6* and *v.0.6.0* do not really differ in presence of the considered faults, and they could be used indistinctly but when the network is perturbed by *flooding attacks* or consumption is the main concern of the evaluator, where *olsrd v.0.6.0* reacts better.

5.4.5 Conclusions

This section has shown the applicability of REFRAHN to perform resilience benchmarking. The innovative point added has consisted in the definition of a quality model to systematise the analysis of resilience benchmarking, which is typically hard given the complexity of considering a wide set of measures. For this purpose, a well-known measures-aggregation technique in the domain of software engineering was used: the Logic Score of Preferences (LSP). Conversely to other measures-aggregation techniques, LSP plays an active role during the benchmark definition. It addresses how to adequately select and gather the types of measures to represent the system, thus assisting the benchmark user to minimise errors during the results in-

terpretation. Given their benefits, the scores obtained from LSP can be used as a complement to traditional measures.

The LSP technique results a very useful approach to overcome the problem of measures scalability and eases a more concise vision of the system. Nevertheless, regarding previous results, the application of this technique requires the adequate definition of the quality thresholds (X_{min} and X_{max}) for each criterion functions, the weight (w_i) assigned to each score within the same aggregation block, and the operator type (o_i) in charge of the measures aggregation. All these aspects highly depend on the applicative context the ad hoc networks is conceived to be deployed in. Despite the selection of these parameters may result subjective, using the LSP techniques forces the benchmark performer to make explicit such parameters, which eases the transparency and comparison between systems. This is an advantage with respect to traditional benchmarking, where the criteria considered for the system quantification keep on being generally subjective but remain hidden to the benchmark report consumer.

Considering the points previously detailed is a first step towards the characterisation of the wide amount of applicative domains ad hoc networks are present in, such as Wireless/Underground and Subaquatic Sensor Networks, Wireless Mesh Networks and Mobile and Vehicular Ad hoc Networks, among others. We argue that this type of approaches can be useful not only to quantify the impact of faults with respect to the actual application context (where components and systems are planned to be deployed), but for the comparison and selection of those targets which best fit the system requirements.

5.5 Fine tuning

System tuning answers one of the fundamental questions a system administrator may ask about her system: how to improve the system behaviour without considering the cost of buying and/or changing a given component. Well-tuned systems enable system administrators to save both time and economical resources. However, tuning always involves balancing a trade-off between the system properties in terms of performance, resilience, resources consumption and cost. So, a deep knowledge of the system is required to take optimal parameterisation decisions.

In our case, this knowledge involves determining which is the effect of tuning ad hoc routing protocol parameters and their fault/intrusion tolerance mechanisms in dynamic adverse conditions. Obviously, this issue requires providing methodologies and tools to introduce such conditions in the system, and track down the protocol behaviour.

REFRAHN could be useful for this purpose. This idea is illustrated through a case study where an actual (non-simulated) deployment of a fault detection (packet loss *watchdog*) and a fault prevention (*MD5* handshaking) mechanism have been analysed and tuned. The tuning of such mechanisms does not come for free. So, experimental results will show their impact on the performance, resilience, and resources consumption of one implementation of *olsrd*. Subsection 5.5.1 introduces the experimental consideration of the case study. Subsection 5.5.2 and 5.5.3 explain the tuning of the *MD5* handshaking mechanism and *watchdog* mechanism, respectively. Finally, Subsection 5.5.4 presents the conclusions of this study.

5.5.1 Experimental considerations of the case study

For simplicity, this case study will focus on the impact of two of the most representative attacks in ad hoc routing protocols when addressing packet loss: *selective forwarding attack* and *flooding attack* according to the scenarios deployed in Figure 5.2. The modelisation and implementation of these faults was introduced in Section 3.3.4 and Section 4.4.5.2 respectively.

Following subsections analyse the details that lead *olsrd* to exhibit a degraded behaviour when considering its fault detection and prevention complements. Then, an alternative parameterisation to improve such behaviour is proposed.

5.5.2 Parameterisation of fault detection mechanisms

According to our results (see Figure 5.3), *selective forwarding* attack and *flooding attack* greatly affects the packet loss of the network. Thus, the challenge consists in correctly detecting the degradation it introduces in the network.

5.5.2.1 Watchdog: a mechanism to detect packet loss

The most common approach under research for the detection of a *selective forwarding* attack is the deployment of a *watchdog* in each network node [125]. From an abstract viewpoint, a *watchdog* is a mechanism installed in every network node to monitor the operation of their neighbours, thus intending to detect any undesired behaviour. The operation of a real *watchdog* mechanism (available at http://safewireless.sourceforge.net) is depicted in Figure 5.12.

This *watchdog* will trigger an alert to notify the malicious behaviour of

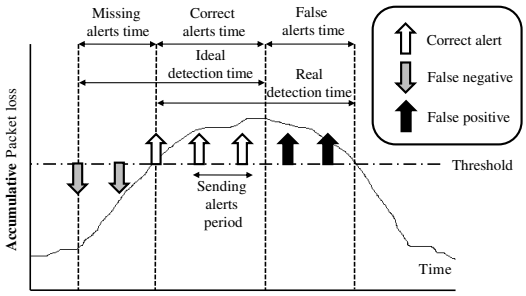


Figure 5.12: Behaviour of *watchdog* mechanisms in terms of generated alarms according to a given threshold for accumulated packet losses.

a node whenever the accumulated packet loss observed reaches a certain threshold. As the *watchdog* will receive the same packets as the nodes under observation, it will know whether incoming packets should be forwarded. In case that after a given time a packet has not been forwarded, the *watchdog* will increment the accumulated packet loss for that node. When it behaves as expected and forwards the received packet, its accumulated packet loss will decrease. Clearly, the accuracy of the actual detection of wrong behaviours is related to the selection of an adequate threshold. A very low threshold is useless, as small packet losses due to reception buffers being full or the node being computing some other high priority process, for instance, may lead to an early alert generation, thus expelling legitimate nodes from the network. Likewise, a very high threshold is also useless, as only very large packet losses will be detected, generating late alerts when the network has already collapsed.

Following this reasoning, and as can be seen in Figure 5.12, there exists a period of time during which, although a node stops forwarding packets, no alarm is triggered as the given threshold has not been reached yet. During this time, the *watchdog* provides *false negative* signals (no problem is signalled although it should be). Once the threshold is reached, the *watchdog*

correctly notifies the problem detected, until the node begins to forward packets again and its accumulated packet loss slowly decreases. However, due to the dynamics of this procedure, until the accumulated packet loss is not again below the considered threshold, the *watchdog* continues signalling the problem, although there is not any actual packet loss. During this period of time, the *watchdog* provides *false positive* signals (it signals a non-existing problem). Hence, determining the proper threshold to balance the detection time and accuracy, i.e., reducing the period of false negatives and false positives to maximise the period of correct detections, is of prime importance for the use of *watchdog* mechanisms. Likewise, as the computed accumulated packet loss depends on the number of observed packets stored in the *watchdog* buffer, its size may also influence the behaviour of the detection mechanism and, thus, it should also be precisely determined.

5.5.2.2 Problem pathology

In order to study the relationship between all these parameters, preliminary experiments were performed varying the accumulated packet loss threshold (0%–50%) and the buffer size (200–10000 packets) in scenarios with continuous presence of packet loss, with routers and laptops acting as network nodes. The packet size was set up to 1500 bytes, which is the worst case scenario established for Ethernet packets. Results are illustrated in Figure 5.13, which were obtained for a *watchdog* router with a buffer size of 1000 packets. The outcome of this experimentation is that a threshold of around 30% of accumulative packet loss, for all the considered cases, minimises both the periods of false negative and false positive signalling, thus maximising the period of correct detection.

Once the threshold defined, it is necessary to determine the optimum buffer size to maximise the resulting detection capabilities of the *watchdog*. As

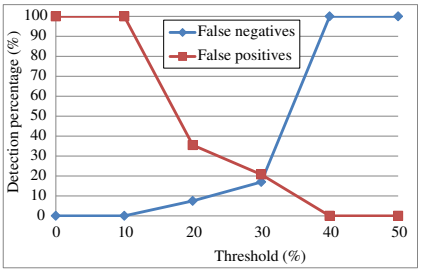


Figure 5.13: Percentage of time the *watchdog* is providing false negative and false positive signals.

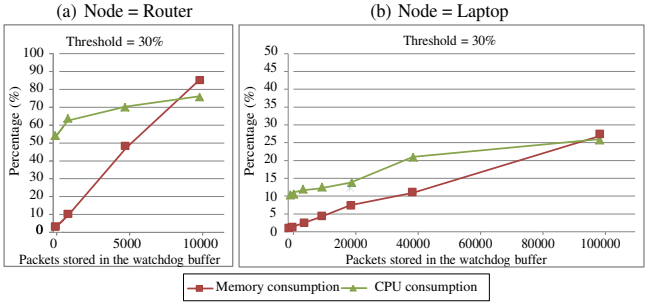


Figure 5.14: Percentage of memory and CPU consumed by the *watchdog* process with a 30% threshold.

one may hypothesise, increasing the buffer size will probably maximise the correct detection period, but also will increment the memory consumed to store the received packets and the processing power required to compute the detection algorithm. Hence, more experiments were performed to determine which were the physical limits imposed by the kind of devices acting as network nodes and, thus, find a trade off between the resources allowance and the desired detection capabilities. Figure 5.14 depicts the percentage of memory and CPU consumed by the *watchdog* process when running on a router and a laptop with a threshold of 30% and a buffer size ranging from 0 to 100000 packets.

As shown in Figure 5.14a, the CPU consumption of the *watchdog* process for a router is quite high (around 50%) even for very small buffer sizes. However, the memory consumption becomes the limiting factor and prevents the buffer size from growing beyond 10000 packets, as the operating system aborts the process for consuming the effective memory (14.2 MB). On the other hand, Figure 5.14b shows that the higher quantity of resources available in the laptop allows the buffer size to scale up to 100000 packets for just around a 25% of memory and CPU consumption. This result illustrates that more powerful devices may be configured to increase their detection capabilities at the expense of increasing their resource consumption, especially in terms of memory.

5.5.2.3 Parameterisation proposal

According to our particular study, network administrators should configure the *watchdog* to use an optimum threshold of 30% of accumulative packet loss. Additionally, they must determine the allocation of resources required. This can be made (i) statically, that is, setting a fix limit for the resources the *watchdog* process may consume, thus determining its final detection capabilities, or (ii) dynamically, i.e., enabling the on-line adjustment of the *watchdog* detection capabilities according to the resources required by the processes running on the node (consuming more resources when they are available to enhance their detection capabilities).

In order to reduce the large memory consumption of the *watchdog* process, thus allowing not so powerful devices (like routers) to trade resources for better detection capabilities, experiments were repeated to obtain the outputs provided by the *watchdog* when running in debugging mode. After analysing the resulting traces, we could determine that the whole packet (including its header and payload) was buffered, which greatly increases the

memory consumed for larger buffer sizes. Accordingly, the *watchdog* implementation has been improved to include an additional parameter that allows the administrator to determine whether the whole packet (regular configuration), or just its header (optimised configuration), is going to be buffered. Figure 5.15 depicts the results obtained for a router running the *watchdog* using the proposed parameterisation (also referred to as optimised version of the *watchdog*).

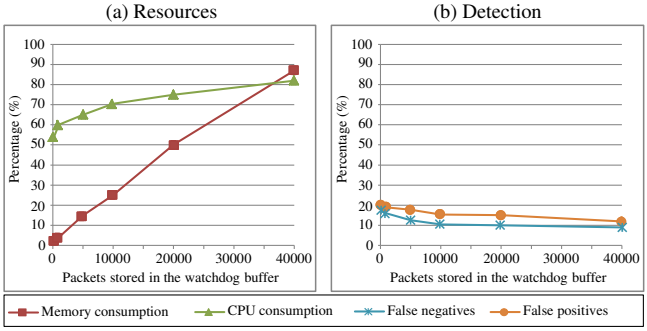


Figure 5.15: Resource consumption and detection capabilities of the optimised *watchdog* running on a router with 30% threshold.

Figure 5.15a shows that for given buffer size, the optimised version of the *watchdog* saves up to 60% of memory. This means that, as more resources are now available, the buffer size may increase to enhance the detection capabilities of the *watchdog*, as depicted in Figure 5.15b.

Finally, experimentation deploying both the default and the optimised *watchdog* mechanism with a 30% threshold and a buffer size of 10000 packets was performed. Results depicted in Figure 5.16 present a percentage of correct detections of 90% and 83% with respect to *selective forwarding* and *flooding* attacks using the optimised version of the *watchdog*. This involves an improvement from 7% to 10% in the detection capabilities with respect to the default version of *watchdog* according to our experiments. Conversely, in the presence of mobility, they worsen 28 and 33 percentage

points (pp) respectively. This results encourage thus to limit the use for the *watchdog* for static networks or look for better mechanisms for mobile scenarios.

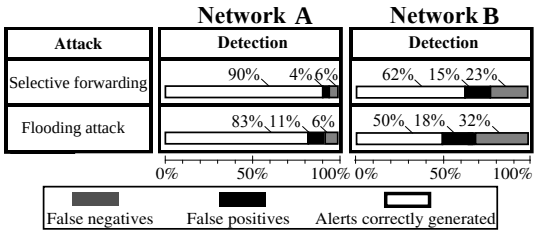


Figure 5.16: *Watchdog* detection capabilities for static (Network A) and mobile (Network B) scenarios.

5.5.3 Parameterisation of handshaking mechanisms

Attending to the results obtained by *watchdogs* in mobile scenarios, it seemed convenient to include a complementary mechanism that could avoid the success of certain attacks in this kind of scenarios. The latest version of *olsrd* includes a secure plugin that can be activated for all the packets to be signed using an encryption algorithm (MD5) with a key shared by all network nodes. This mechanism, which guarantees the integrity of exchanged packets and prevents untrusted nodes from entering the network, was already considered as a target in the previous evaluation of Section 5.2.

5.5.3.1 Limitations of a MD5 handshaking mechanism

As it was seen when addressing the analysis of results concerning the security version of *olsrd*, (referred to as *v.0.6.0+md5*) in Table 5.3 for the static (*Network A*) and mobile (*Network B*) scenarios, unsigned packets are directly discarded, greatly reducing the packet loss due to *selective for-*

warding attacks. Nevertheless, it was very surprising that the packet loss due to the nodes mobility also increases, thus leading to a lower percentage of packets correctly delivered in mobile scenarios. Furthermore, despite implementing this security mechanism, it was worth noting how the packet delivery ratio in presence of *flooding attacks* was reduced in both networks A and B. In sum, despite their benefits, this security plugin led to longer route formation times and, thus, higher packet losses.

A deeper analysis of resulting traces revealed that, if any of the challenge packets exchanged during the handshaking is missed, the handshaking protocol should be restarted from the very beginning. Additionally, this problem is only detected when the timeout for the handshaking packets expires, which is set by default to 30s in the considered protocol, thus justifying the longer delay in the formation of the route.

5.5.3.2 Parameterisation Proposal

It seems clear that network administrators should determine the right duration of this timeout for their network. Shorter timeouts will decrease the route formation time in presence of attacks. However, such timeouts should be long enough to allow for packets to be sent, processed and replied in time without unnecessarily reinitiating the handshaking protocol.

As the handshaking traffic is less than 5% of the total traffic generated by *olsrd*, we propose to follow a redundant approach to tolerate the loss of some of these packets without waiting for the timeout to expire. Handshaking packets could be replicated to ensure that some of them will reach their destination in time, even in presence of attacks. The generated traffic overhead is negligible, in the order of 1 KB after 100s of execution in the worst case (restarting the handshaking after every timeout), and duplicated packets already processed are directly discarded without requiring

either memory or CPU additional resources. Figure 5.17 depicts the formation time of a 3-hop route, in presence of a 4 Mbps *Flooding* attack, with a decreasing handshaking timeout duration and an increasing number of replicated packets.

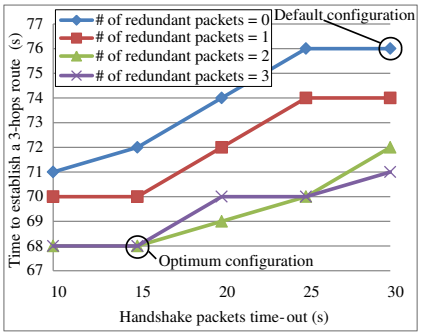


Figure 5.17: Handshake parameterisation used to establish a 3-hops route in presence of a 4-Mbps *flooding attack*.

As illustrated in Figure 5.17, the route is established after 76s using the default configuration of the secure plugin, whereas the optimum configuration achieves a route formation time of 68s just by decreasing the timeout to 15s and duplicating the handshaking packets sent. Those results were validated by repeating the previous experiments in presence of a 4 Mbps *Flooding* attack, using the default and the optimum parameterisation of the secure plugin, for a static and a mobile scenario. Results are listed in Table 5.8.

The proposed parameterisation slightly improves the behaviour of the protocol in the static scenario, reducing, in average, 8 seconds the route formation time and 4 percentage points the packet loss, but it achieves a great success in enhancing the plugin performance in presence of mobility, decreasing these same values a total of 42 seconds and 9 percentage points.

Table 5.8: 3-hops route in presence of a 4-Mbps Flooding attack using the default and the optimised secure plugin.

olsrd version	**Network A**		**Network B**	
	Route formation (s)	Packet loss (%)	Route formation (s)	Packet loss (%)
v.0.6.0+md5 (default parameterisation)	76.0 ± 10.3	32.8 ± 6.1	152.3 ± 12.7	53.8 ± 7.2
v.0.6.0+md5 (optimised parameterisation)	68.0 ± 8.9	28.5 ± 5.5	110.7 ± 10.1	44.9 ± 5.6

5.5.4 Conclusions

In addition to decisions already taken by protocol designers, programmers and system integrators, there exist a number of parameterisation issues that must be carefully considered by system administrators at deployment-time to make ad hoc networks really useful in real-life contexts of use.

The presented research demonstrates the interest of REFRAHN to drive the resilience parameterisation of ad hoc routing protocols while maintaining high levels of performance and reasonable levels of resource consumption. This idea has been successfully applied to static and mobile ad hoc scenarios integrating both high-end and resource-constrained devices.

Although results have been obtained from OLSR, a number of conclusions can be generalised and applied to any type of ad hoc routing protocol. First of all, it is to note that mobility of nodes require detection mechanisms to be designed taking mobility in mind. The values for such thresholds must be established paying particular attention to maximise the level of detection while minimising the number of generated false detections (false positives and negatives). Second, authentication protocols in ad hoc networks are very sensitive to packet loss, which may significantly slow down the performance of the protocol if values for timeouts are used for packet resending are not carefully selected. However it requires dynamic solutions in mobile contexts of use. This last lesson learned can be even formulated in a more

general way. Since too many aspects handled by ad hoc routing protocols are dynamic, it is impossible to fix every resilience problem relying only on static parameterisation strategies. Some ideas have been provided on how to design self-adaptive resilience mechanisms able to adjust their configuration and behaviour attending to available resources in nodes (processing, memory and so on) and the state of their execution environment. Such ideas will be explored in next section.

5.6 Design of new fault tolerance mechanisms

As stated along this thesis, packet loss is a major practical impairment for the resilient use of ad hoc networks. However, as concluded in the previous section, this type of problems cannot be just addressed using static fault tolerance approaches.

The interferences created by *ambient noise* are a major source of packet corruption, that irremediably end up manifesting as packet loss in the domain of ad hoc networks [126]. There exist some strategies aimed at minimising the impact of ambient noise in the resilience of ad hoc networks. For example, protocols can be statically tuned to reduce the frequency of sending routing protocol packets to increase the lifetime of their links. But the overhead derived from such tuning and the pertinence of the resulting configuration along the time must be carefully considered. In [127], for instance, authors proposed an automatic approach to manage link communication faults in WMNs by inferring suitable configurations from network model simulations. Facing dynamic faults, such as *ambient noise*, asks for more evolvable strategies (beyond static tuning) to adapt at runtime the level of link protection against *ambient noise* in practice.

REFRAHN is used in this section to guide the design of new fault toler-

ance strategies when the tuning of current routing strategies is not enough. Subsection 5.6.1 uses the fault injection capabilities of REFRAHN to study the behaviour exhibited by actual implementation of three state-of-the-art proactive routing protocols, OLSR [29], Babel [31] and B.A.T.M.A.N [30] against *ambient noise*. Then, an adaptive approach to improve their behaviour is proposed and evaluated in Subsection 5.6.2. Finally, Subsection 5.6.3 present conclusions.

5.6.1 Experimental considerations of the case study

The fastest recovery of route failures provided by proactive routing protocols with respect to reactive ones has increased the research efforts on the former ones in a wide range of domains from WMNs to VANETs.

The implementation targets considered for our experimentation are more recent steady open-source versions of the current proactive routing protocols. Accordingly, well-known implementations of OLSR (*olsrd v.0.6.0*[1]), Babel (*babeld v.1.1.1*[2]) and B.A.T.M.A.N (*batmand v.0.3.2*[3]) have been taken into account.

5.6.1.1 Proactive routing protocols under study

OLSR [29] is the most well-known link-state protocol and one of the most widely-used proactive routing protocols nowadays. OLSR uses an optimised flooding mechanism, where only special nodes called Multi-Point Relay (MPR) are responsible for broadcasting the routing information along the network. Although the initial specification of OLSR (RFC 3626) established route computation using hop-count as metric, current specification

[1] www.olsrd.org
[2] www.pps.jussieu.fr/jch/software/babel/
[3] www.open-mesh.org

OLSRv2 promotes the use of link quality extensions.

B.A.T.M.A.N [30] is a novel proactive distance-vector routing protocol. For each node, B.A.T.M.A.N periodically sends out broadcast messages to inform neighbours of its existence. This process is repeated until the routing information reaches all network nodes. For each link the routing packets arrival rate is advertised so that neighbour nodes can determine the link quality. B.A.T.M.A.N uses link quality metrics to estimate the quality of network links.

Babel (RFC 6126) [31] is the most recent protocol under consideration. It is a distance-vector routing protocol that has two main characteristics to optimise its relay mechanism. On one hand, it uses history-sensitive route selection to minimise the impact of route flaps. In such a way, the route selection favours the previously established path rather than alternating between two routes. On the other hand, it forces a request for routing information each time it detects a link failure from one of its preferred neighbours. This is a best-effort mechanism to reduce the reconfiguration time of the network. Babel uses a metric of quality for network links.

5.6.1.2 Ambient noise characterisation

To ease its interpretation, *ambient noise* was characterised in 3 different intervals: a high *ambient noise* coinciding with the workday involving a packet corruption ranging from 35% to 50%, a moderate ambient noise matching with the lunch breaks with a packet corruption ranging from 5% to 35% and a low ambient noise causing a packet corruption ranging from 0% to 5% at night. Despite not being so frequent, ambient noise leading to packet corruption in the range from 50% to 100% typically represents additional external faults like malicious attacks from signal inhibitors or accidental faults like interferences from microwave ovens.

5.6.1.3 Experiment setup

To carry out our experiments and assess the resilience of routing protocols against *ambient noise*, a new network deployment was considered this time. Concretely, a WMN implementing the topology depicted in Figure 5.18 recreates the map of our department. Routes of 4-hops distance (A-B and A-C) have been considered as representative in our study. This experimental setup takes into consideration the two basic types of routes already explained in Section 2.2, route A-B with alternative paths and route A-C without alternative paths. In the second case, routing protocols typically change the old best route by the new best one from the set of available alternative routes in case of updates in their links' quality. The study of this case has been simplified by considering just two alternative routes, which is the minimum number of routes to perform a route switch. Thus, route A-B can be established through nodes x_i (x_1, x_2 and x_3) and y_i (y_1, y_2 and y_3) respectively. Given the unavoidable presence of physical obstacles like walls or other objects, the studied routing protocols usually find the best route from A to B (and vice-versa) through x_i nodes most of the times, rather than through y_i nodes.

According to the previous setup, the goal will be to study the impact of *ambient noise* in the target routes. The faultload in charge of easing this task will be deployed as follows. On one hand, route A-C will be subjected to the presence of a gradually increasing *ambient noise*. As far as there are no alternative routes, this experimentation will be useful to study the effects of route partitioning. The presence of *ambient noise* along the whole route will be emulated by injecting an *ambient noise* equivalent to a given packet corruption, thus creating a homogeneous *ambient noise*. On the other hand, the experimentation on route A-B will be oriented to analyse the effects of an heterogeneous presence of *ambient noise* by considering two different areas of noise in the network. One zone, delimited by all

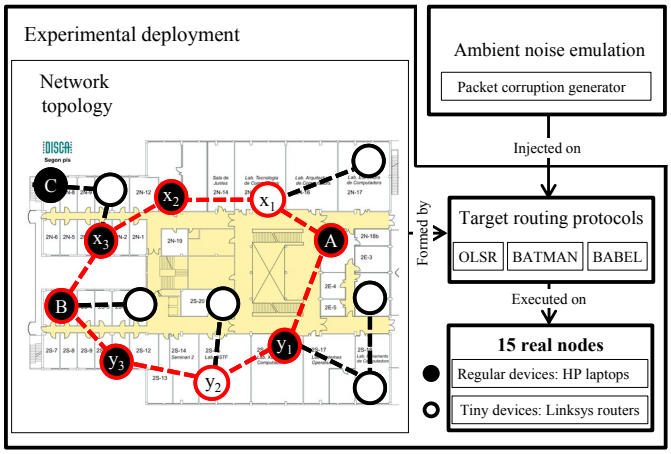


Figure 5.18: Experimental WMN deployment.

the nodes in route A-B but x_3, will be subjected to the same gradually increasing presence of packet corruption. Conversely, remaining node (x_3, typically taking part in the active route by default) will be exposed to a constant extreme presence of *ambient noise* of 95% packet corruption. Forcing this worst-case situation, we ambition to study the effects of route switching in route A-C from x_i to y_i nodes, and consequently analysing its impact on the reconfiguration time of the protocol to offer a new alternative route.

To animate routes A-B and A-C, a workload consisting on UDP constant bit-rate data flows of 200 Kbps is established to compute the packet delivery ratio of the route.

In order to limit the influence of real *ambient noise* in our results, our experimentation was carried out at night, assuming an acceptable intrusiveness of 0% to 5% of real (non-emulated) packet corruption for wireless networks. In summary, two weeks of experimentation were devoted for this purpose. In total, 600 experiments of 300s each divided in two experimental

campaigns (one per type of route) were executed.

5.6.1.4 Impact of ambient noise

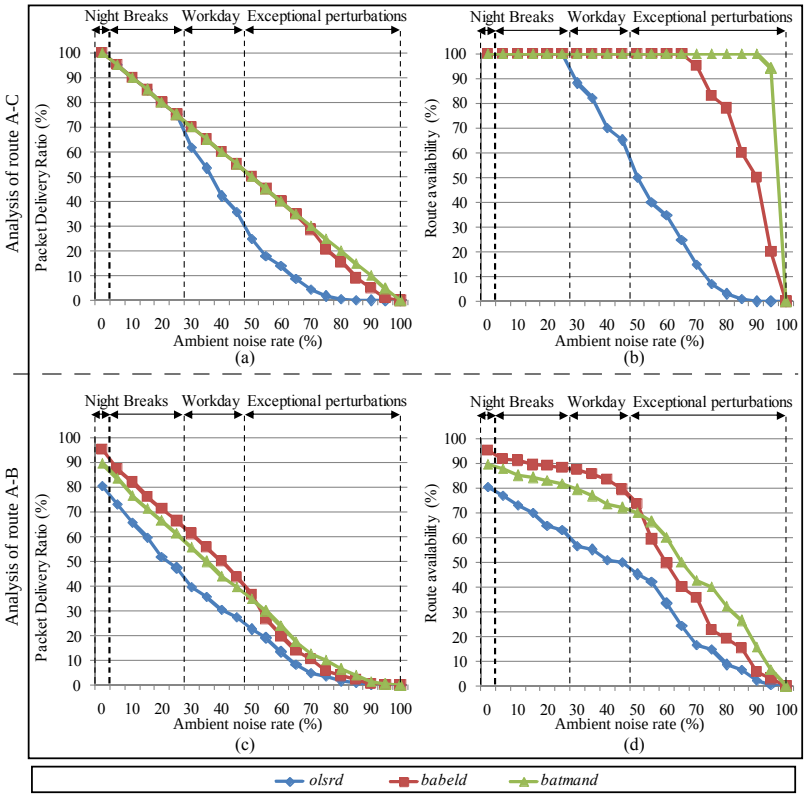


Figure 5.19: Experimental results obtained applying the default configuration.

Figure 5.19a illustrates the results of the experiments obtained from route A-C, measuring the packet delivery ratio through x_i nodes. As observed when *batmand* is in charge of routing, the packet delivery ratio decreases proportionally with the amount of *ambient noise* introduced. *babeld* also

presents very similar results, while *olsrd* starts degrading from 30% *ambient noise*. If analysing these results, the case representing the ideal packet delivery ratio loss should involve an identical decrement with respect to the *ambient noise* introduced. Any case where the packet delivery ratio decreases faster than the *ambient noise* introduced, necessarily involves the presence of any additional effect impacting on the packet delivery ratio. To study this effect in more detail, Figure 5.19b shows the route availability of the different routing protocols considered. In this graphic, it is possible to appreciate the robustness of the route exhibited by each routing protocol against *ambient noise*. As can be deduced, any route availability below 100% impacts negatively on the global packet delivery ratio of the route. Consequently, the longer the protocol can maintain the route available, the better for the packet delivery ratio. In this sense, *batmand* deserves being considered the best routing protocol since their route A-C strongly resists and does not create network partitioning until introducing 95% of *ambient noise* in the system. *babeld*, starts degrading the route availability with 70% of *ambient noise*. Finally, *olsrd* starts suffering the impact of *ambient noise* (around 30%) within the daily levels of *ambient noise* (breaks and workday).

Now let us focus on the results obtained for route A-B. Figure 5.19c shows the packet delivery ratio when the route traverses a zone of extreme *ambient noise* (through node x_3) and the route A-B must be dynamically re-established through y_i nodes. In this case, the major difference introduced in Figure 5.19c with respect to Figure 5.19a is that packet delivery ratio never achieves 100% even in absence of *ambient noise*. Figure 5.19d illustrates the route availability for this type of experiment to explain this result. From this graphic it is possible to deduce that the application of the new route is not for free and involves a cost, even in absence of *ambient noise*. For the time required to apply the new route when the old one is no longer available (reconfiguration time), the communication between A and

B is not possible. Consequently, the longer this time, the major impact on the route availability. Concretely, *babeld* is the protocol presenting the best behaviour up to considering an *ambient noise* of 50%, but beyond this rate, *batmand* tolerates the presence of *ambient noise* slightly better. In any case, *olsrd* is the worst option. These results are consistent with the reconfiguration time measured in Table 5.9, and show how this time is one of the main reasons of route unavailability in WMNs.

Table 5.9: Reconfiguration time in route A-B.

Protocol version	**Reconfiguration time (s) per (corrupted) packet loss (%)**									
	0%	10%	20%	30%	40%	50%	60%	70%	80%	90%
olsrd v.0.6.0	58	81	106	130	147	165	200	250	274	293
babeld v.0.9.6	13	27	33	38	50	80	151	193	242	283
batmand v.0.3.2	28	45	51	62	87	90	120	172	203	253

The considerable differences in the behaviour of protocols lead us to analyse their source code to understand why protocols behave in that way. After analysing the code in detail throughout debugging tools like *gdb* [128], we observe that, the previous ranking established is not casual. Behind the name of different variables and constants (always consistent with their respective specification), there are three common parameters characterising the behaviour of proactive routing protocols against *ambient noise*. Table 5.10 identifies them and states their default values. T, is the default period to send a routing packet advertising a given link, and Twindow is the validity time determining the temporal window after which the protocol decides whether discarding a link or not. Basically, the use of these two time-related parameters is located within the task scheduler module already presented in Figure 2.2. The Minimum Quality Threshold (MQT) defines the minimum acceptable quality before removing a link. This quality-based parameter is used within the routing manager to decide about the routing capacity of a given link (see Section 2.2). After analysing the target routing protocols, the conditions that must be satisfied to remove a link have

been ordered from the most reactive to the most conservative as follows: *olsrd* and *babeld* require (i) the expiration of the Twindow or (ii) exceeding the MQT. Conversely, *batmand* only requires the expiration of Twindow before removing the link. Surprisingly, the notion of MQT is never taken into consideration in *batmand*. Unlike *babeld* and *olsrd*, the link quality is only updated in *batmand* when getting new information through incoming routing packets. If focusing on the configurations of T (shown in Table 5.10) to analyse how many opportunities has a protocol to refresh their link quality within a Twindow, it is easy to estimate that *olsrd* can only send 6 packets to update the quality of the link, whereas *babeld* admits up to 15 packets, and *batmand* has 200 new opportunities. Obviously, the more opportunities to update the link quality, the quicker the routing protocol can react against unexpected changes.

5.6.1.5 Tuning the routing protocol configuration

Essentially, as can be deduced, the success of *batmand* in our results is likely not due to its conservative policy, but to the configuration of parameters considered, which provides *batmand* a major frequency to send packets and consequently more opportunities to update the quality of their links than the rest of routing protocols considered. To make the comparison between the routing protocols considered fairer, let us apply the parameters configuration used in *batmand* to the rest of protocols. Additional experimentation was required to achieve this goal. The results finally obtained are shown

Table 5.10: Critical parameters for the route availability in WMNs.

Protocol implementation	Twindow (s)	T (s)	MQT (%)
olsrd v.0.6.0	30	5	10
batmand v.0.3.2	200	1	none
babeld v.1.1.1	60	4	0

in Figure 5.20. Figures 5.20a and 5.20b show the results of packet delivery with respect to the *ambient noise* rate. One can observe above all a considerable improvement in *olsrd* in particular for route A-C. According to route A-B, the enhancement on the packet delivery ratio concerns both *olsrd* and *babeld*. As far as the packet delivery ratio depends on the route availability, let us analyse deeply the results from that viewpoint.

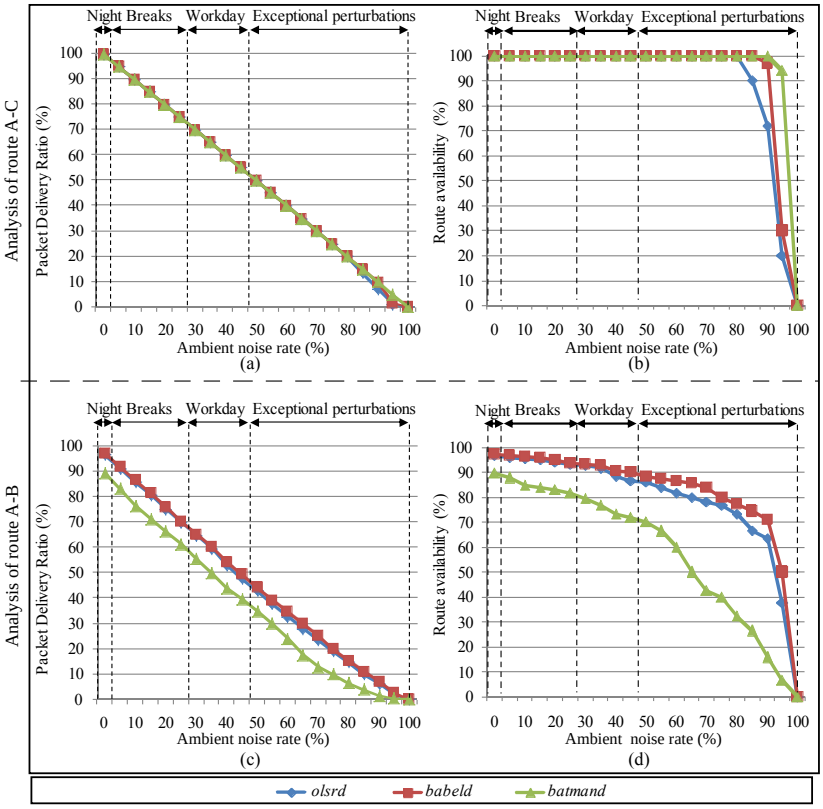


Figure 5.20: Experimental results obtained appying the *batmand*-like configuration.

If focusing on Figure 5.20b, it is worth mentioning how *olsrd* (above all) and *babeld* improve the robustness of the route A-C. With the *batmand*-

like configuration, the route degradation in *olsrd* starts around 80% of *ambient noise* rate, enhancing 50 percentage points (pp) with respect to its default configuration. In the case of *babeld*, the improvement is around 20 pp. *batmand* keeps on being the best protocol in this scenario, but the differences with the other protocols have been strongly reduced.

If considering now route A-B (through node x_3), an interesting result can be observed from Figure 5.20d. In this case, the result is striking not so because *olsrd* and *babeld* increase their route availability (as expected if reducing the period T between packets sent), but because they behave better than *batmand*, which was considered the best routing protocol for route A-C. Evidently, something else apart from the parameterisation must be influencing these results. The conservative policy of *batmand* seems to be its drawback when facing dynamic changes of routes caused by an extreme *ambient noise*. As the network topology in Figure 5.18 states, when x_3 dramatically starts losing packets, the route A-B through x_i nodes is no longer available. However, node A has no indication that the route through nodes y_i is better than the offered through x_i until the next routing packets from B to A arrive through y_i and enhance the route quality of x_i. Obviously, the route reconfiguration time is intimately related to the *ambient noise* in the network. The harder the *ambient noise* conditions, the longer the reconfiguration time. Conversely to *batmand*, *olsrd* and *babeld* implement instruments like the MQT which promote the protocol reaction to minimise the reconfiguration time, which, as shown in Figure 5.20d, have been proven useful. In this sense, the protocol in node A is able to react earlier not only because of receiving packets from y_i, but because packets from x_i announce a broken link with x_3 once the MQT exceeded.

As seen, the selection of a suitable parameterisation can improve the robustness of routing protocols against *ambient noise*. However, it is necessary to analyse the cost to pay in terms of the routing overhead introduced in

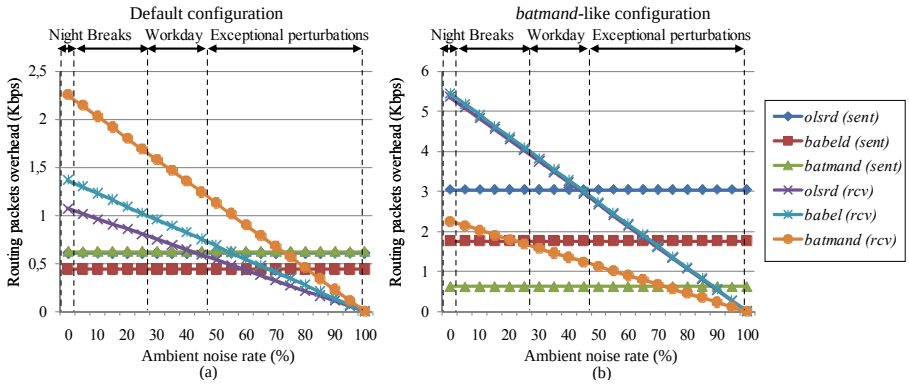


Figure 5.21: Routing overhead induced by routing packets sent and received.

the network before taking any decision.

Figure 5.21a and Figure 5.21b, study the average routing overhead introduced by each node when applying the default and the *batmand*-like parameterisation respectively. If analysing Figure 5.21a, *batmand* is the protocol with the highest routing overhead in terms of both packets sent and received when applying the default configuration (50% more than *olsrd* and *babeld* in the case of routing packets sent, and 127% and 78% in the case of *olsrd* and *babeld* respectively for the routing packets received. However, the trend changes when applying the *batmand*-like parameterisation. In this case, *olsrd* obtains the highest routing overhead in terms of packets sent (400% more than *batmand* and 58% more with respect to *babeld*) while *olsrd* and *babeld* increase the received routing overhead 161%. Since the considered routing protocols send packets with the same period T, these differences can be explained due to the average size of the routing packets sent by each routing protocol (380B in *olsrd*, 220B in *babeld* and 78B in *batmand*). In this case, the higher size of *olsrd* packets penalises its routing overhead. However, it is worth noting the lack of mechanisms to prevent the flooding of routing packets in *babeld* and *batmand*. This means that the cost of such

protocol in terms of packets sent is quadratic and depends on the number of nodes $O(n^2)$. This fact could benefit *olsrd* if performing more experiments in a network including a wider amount of nodes, given the optimisation mechanism based on multi-point relay which provides *olsrd* a cost $O(n)$.

As graphics show, regardless the configuration used, the routing information rate sent is always constant because routing protocols periodically send the same (or quasi the same) amount of information. This routing overhead is characterised for being ambient-noise independent. Conversely, the routing information received highly depends on the presence of *ambient noise* and its reception is directly proportional to the amount of packet corruption induced by the *ambient noise*. This fact makes that the ratio between the routing packets sent and received is quite disproportional as the *ambient noise* increases. Indeed, there are situations where the routing protocol could afford sending less routing packets to keep on maintaining the routes alive in presence of few *ambient noise*. However, in situations with a severe presence of *ambient noise*, the amount of routing packets received, decreases to the extent of provoking the mentioned problems of network partitioning and long convergence (or reconfiguration) times. One could think on parameterising the routing protocol according to the level of *ambient noise* in the network, however any off-line configuration of these protocols stops being valid when the conditions of the environment, and specially of the *ambient noise*, vary over time. Given these situations, the provision of adaptive strategies to balance the routing overhead could result very useful to introduce the necessary routing overhead in the network to keep the links alive.

5.6.2 Link-quality-based adaptive replication of packets

This section faces the problem of *ambient noise* in proactive routing protocols proposing a generic adaptive strategy which enables the routing protocol to increase the routing overhead only when required. Replication is a well-known technique in the domain of fault tolerance that can be used for this purpose. The use of packet replication in this solution is devoted to ensure the reception of the routing information even in the presence of a high level of *ambient noise* that disturbs the communication between nodes. So, this approach can be useful in environments affected by *ambient noise* when links run the risk of disappearing or it is necessary to speed up the reconfiguration time.

5.6.2.1 Analytical overview of the technique

This technique is based on the principles of T, Twindow and MQT previously identified in Section 5.6.1.4. As stated, nodes (re-)compute the quality of each one of their links each T. A link is lost whenever (i) its quality is lower than the MQT accepted by the protocol or (ii) no routing message is received for a period Twindow, despite its link quality.

Far from tuning their value, the algorithm proposed in this section is applied to the default configuration of the routing protocol, (but it may be applied to any other). Thus, our algorithm estimates an evolution factor m from the current link quality lq_i and the previous one lq_{i-1}. The key to compute m can be easily understood through the graphic in Figure 5.22.

If applying basic algebraic notions, given two points A (x_2, y_2) and B (x_1, y_1) in Cartesian axis, it is possible to determine the equation of the

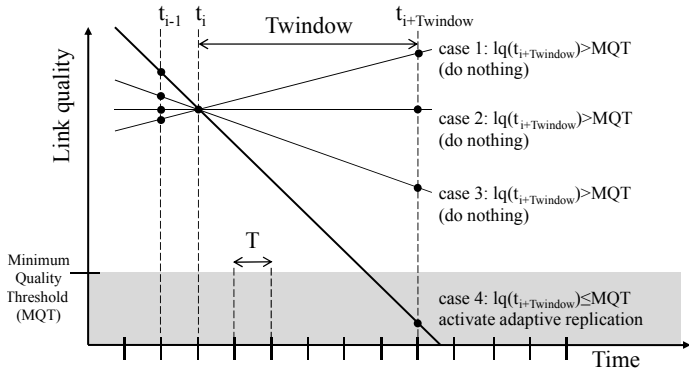


Figure 5.22: Link-quality-based adaptive packet replication technique.

linear function for any point C (x, y), as Formula 5.2 shows.

$$y - y_1 = m(x - x_1) \quad | \quad m = \frac{y_2 - y_1}{x_2 - x_1} \tag{5.2}$$

If replacing points A and B in Formula 5.2 by (t_i, lq_i) and (t_{i-1}, lq_{i-1}) respectively where t_i represents the current time (T), lq_i is its respective link quality, t_{i-1} and lq_{i-1} represent that information for last T, and point C is replaced by ($lq_{i+Twindow}$, $t_{i+Twindow}$) we can obtain Formula 5.3.

According to Formula 5.3, it is possible to forecast the link quality $lq_{i+Twindow}$ for a given time $t_{i+Twindow}$ according to the trend pointed by m.

$$lq_{i+Twindow} = m \cdot Twindow + lq_i \quad | \quad m = \frac{lq_i - lq_{i-1}}{T} \tag{5.3}$$

Then, if the estimated link quality after Twindow time $t_{i+Twindow}$ is lower than the Minimum Quality Threshold MQT, the replication level R, which counts the number of routing packets to be replicated, increases in a factor δ

to keep the link alive regardless the effect of current ambient noise. Else if R is greater than 0, it is decreased in δ packets to consider the situation when the network has overcome the risk to remove the link (either because the ambient noise has been reduced, or because of the effect of the adaptive packet replication). δ represents the number of packets to be added or subtracted to R. The idea of this strategy is reacting (in time) against a possible link removal.

5.6.2.2 Implementation of the algorithm

Our technique is included within the routing manager module of the routing protocol (see Figure 2.2). Table 5.11 shows the pseudo-code that has been implemented in C language for each routing protocol considered in this section. The real conditions of the network in practice impose limiting the amount of replicas to N_{max} and *delta* (δ). If considering a very severe *ambient noise*, the fact of sending more and more replicas will only contribute to increase, even more the effect of *ambient noise*. The value of N_{max} has been empirically computed for our deployment to 10 packets in order not to exceed the routing overhead obtained when applying the *batmand*-like configuration beyond 150%. In the same line, *delta* was limited to 1 packet. The fine tuning of these parameters falls out of the scope of this section. However, the approach followed in Section 5.5 could be used as a reference to carry out this task.

As previously stated, *batmand* presents certain limitations like the absence of a MQT. However, given the genericity of our approach, nothing impairs assigning a $MQT = 0$ to *batmand* in our algorithm, or to any other proactive routing protocol which does not consider its use. Our technique is applied before sending a routing packet every time T. Then, the algorithm proposed must obtain the value of lq_i and lq_{i-1}. The value of lq_i can

Table 5.11: Link-Quality-based Adaptive Packet Replication.

```
01: #DEFINE Nmax
02: #DEFINE delta
03: float current_lqi, previous_lqi;
04: int R = 0;
05: /*every time link quality is computed for the current link*/
06: for each T
07: /*forecast link quality*/
08: lqi_in_Twindow = ((current_lqi - previous_lqi)/T * Twindow) + current_lqi
09: /*determine the number of replicas to send*/
10:  if (lqi_in_Twindow <= MQT) then
11:    if (R < Nmax) then R += delta;
12:  else
13:    if (R > 0) then R -= delta;
14:  /*send the replicas required [0, Nmax]*/
15:  send_broadcast(R, routingPacket);
16:  /*save the variables for the next iteration*/
17:  previous_lqi = current_lqi;
```

be easily obtained from the current state of the routing manager module. However, not all the protocols consider storing the previous state. Accordingly, the algorithm must store lq_i to provide lq_{i-1} in next iteration of T. This cost is negligible in terms of memory footprint even for the tiny devices considered in our experimentation.

The next step involves computing $lq_{i+Twindow}$ through the expression in Formula 5.3. In case this value is underneath MQT, the value of R indicating the number of replicated packets that will be sent in T, is progressively increased only if its current value is lower than N_{max}. Otherwise, in case the link has overcome the risk of disappearing, the number of replicas is progressively reduced up to 0, thus restoring the default behaviour of the protocol. The value of R is also stored to increase or decrease it in the following iteration, depending on the state of the link.

In any case, all the replicated packets send the same information, so, any packet already received will be discarded. Our goal thus is not sending new packets with further information, but increasing the probability of broad-

casting the same information at least once. As all the protocols natively implement the mechanism to discard replicas, no additional strategy has been required to be introduced in our algorithm in the reception of packets.

Given the simplicity of the operations considered, and the time elapsed between the iterations, the overhead introduced in the protocol in terms of CPU is also negligible (less than 1%).

5.6.2.3 Assessing the adaptive replication of packets

Additional experimental campaigns were required to show the effectiveness of the algorithm proposed. Concretely, let us first analyse the Figure 5.23, which represents the routing overhead introduced by the routing protocols implementing the algorithm proposed. Basically, when applying our technique, all the routing protocols balance their routing overhead to adapt their behaviour in a context-aware way. In terms of routing information sent, this balance goes from the regular behaviour of the protocol (see Figure 5.21a) to a behaviour similar to the experienced when applying the *batmand*-like configuration (see Figure 5.21b). In the first case, no additional routing packet is sent, so the intrusiveness introduced in the network is null. Conversely, when the *ambient noise* increases and the routing protocol requires a major effort to maintain their routes, it is allowed to increment the amount of routing information sent. Unlike the regular behaviour of the routing protocols considered, what is constant using our technique is not the rate of routing packets sent, but the rate of received ones. The goal thus, is maintaining the routing capability as longer as possible, even with a severe amount of *ambient noise*. If comparing the new results with the previous routing overhead involving the packets sent (shown in Figure 5.21b), all the protocols reduce the amount of packets sent up to around 70% packet loss caused by the packet corruption of *ambient noise*. In this

condition of extreme necessity for the links survival, the routing protocols using our technique are forced even to increase the routing overhead introduced with respect to the *batmand*-like configuration. Indeed, *olsrd*, *babeld* and *batmand* increment, in average, 13, 15 and 150 percentage points respectively in this aspect. However, the major difference is that now, the route availability increases in these conditions. Beyond this *ambient noise* rate, the routing packets received decrease given the practical bound imposed by N_{max} to limit the packet replication indefinitely.

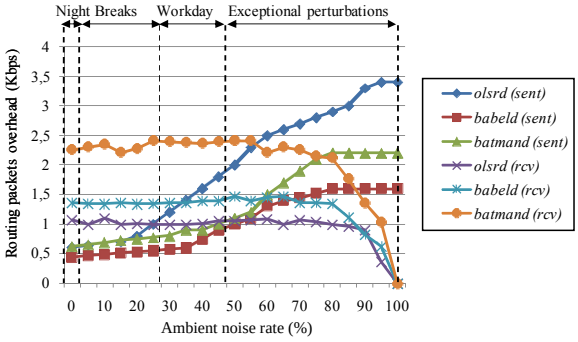


Figure 5.23: Routing overhead after applying the link-quality-based packet replication.

If taking these results in mind when comparing the route availability obtained when applying our technique, with those provided previously in Figure 5.19c and Figure 5.19d (for the default configuration) and Figure 5.20c and Figure 5.19d (for the *batmand*-like configuration), the benefits of our technique can be observed. Concretely, the regular behaviour of the targeted routes A-C and A-B is absolutely improved regarding the default configuration in all the protocols (see Figure 5.24b and Figure 5.24d respectively). In the case of the *batmand*-like configuration (see Figure 5.24c), results are very similar for *olsrd* and *babeld* (less than 3% of difference), but taking into account that the routing overhead introduced has been widely

reduced for the ranges of breaks and workday (more than 150% in all the cases), where the protocols will operate most of the time. Additionally, it is worth noting that thanks to this technique *batmand* speeds up its reconfiguration time, and consequently increments its route availability with respect to its default configuration from 5% to 10%.

All these improvements are observed in terms of the packet delivery ratio in Figure 5.24a and Figure 5.24c, thus enhancing the general behaviour of the WMN with respect to the regular behaviour of the protocol.

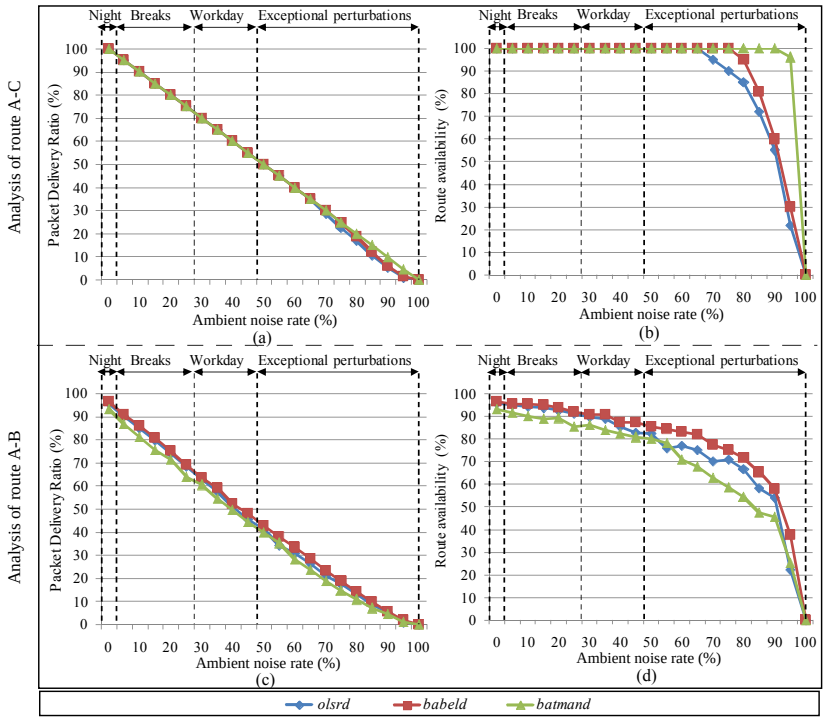


Figure 5.24: Experimental results after applying the link-quality-based packet replication.

5.6.3 Conclusions

The contribution in this section has gone in the direction of exploiting REFRAHN for (i) assessing the behaviour of three state-of-the-art proactive routing protocols in presence of *ambient noise* and (ii) improving their robustness, while reducing the cost of routing overhead. Although results have been obtained from OLSR, Babel and B.A.T.M.A.N, a number of conclusions can be generalised and applied to any type of proactive routing protocol incorporating similar link-quality-based parameters. From such ideas, a novel strategy to fight against *ambient noise* is proposed as a complement to the existing solutions. Concretely, its novelty is on promoting the dynamic adaptiveness of the routing protocol to the network environment to determine the optimum amount of routing information that must be exchanged among nodes in a given moment.

Given the genericity of this approach, it could result of interest, not only to WMNs in particular, but also to any type of ad hoc network (sensor networks, mobile ad hoc networks or vehicular ad hoc networks) in a wide range of context of use.

5.7 Conclusions

This chapter has shown the benefits of REFRAHN to carry out the resilience evaluation of different ad hoc routing protocols in the presence of a wide set of faults. After that, obtained results have been exploited to perform a serial of processes aimed at improving the robustness of ad hoc networks.

Likely, Section 5.3 has shown the interest of REFRAHN for the discovery of vulnerabilities in ad hoc routing protocols, which could be really useful

for routing protocol designers.

As stated in Section 5.4, system evaluators can take benefit of REFRAHN to perform resilience benchmarking to compare and select the routing protocol that matches the best the system requirements.

Section 5.5 has shown how the resilience of ad hoc networks can be improved by simply acting on the parameters of components that result critical for the global behaviour of the ad hoc network: the routing protocols and their fault-tolerance mechanisms. Identifying and tuning such parameters is a task that can be easily piloted by networks administrators through REFRAHN.

Finally, Section 5.6 has illustrated how routing protocol designers can also use the fault injection capabilities of REFRAHN to guide the design of new fault tolerance strategies for a system when tuning the parameters of routing protocols is not enough to ensure a reasonable level of performance and resilience. This section has shown the actual need of routing protocols for adopting dynamic strategies which enrich the already considered notion of link quality to improve the resilience of network links against packet losses caused by faults. According to this reasoning, our contribution to this research gap has gone in the direction of proposing and evaluating a novel adaptive link-quality-based packet replication technique to mitigate the impact of *ambient noise* on proactive routing protocols.

Chapter 6

Conclusions

Ad hoc networks have attracted a lot of interest from both academia and industry in the past few years due to the new possibilities they open to deploy rapid and low-cost wireless networks. They allow for the spontaneous formation of communication networks without dedicated infrastructure. However, ad hoc networks are not yet ready for large-scale deployments. Aspects like mobility or the presence of accidental and malicious faults (or attacks) in dynamic scenarios, may result in a number of changes in the execution environment that normally impact on the system behaviour degrading (or even denying) the service provided by the network. Hence, the operation contexts where ad hoc networks typically develop their activity may be subjected to continuous changes that, obviously, will force their adaptation to cope with the new functional requirements that may arise. Clearly, non-functional mechanisms, like those deploying the fault tolerance in the system, will also be affected by this adaptation.

The system's ability to resist the occurrence of faults, attacks and other changes in its environment, while offering a dependable and safe service at any time, is known as resilience. Resilience is an essential non-functional

aspect when studying the effect of such changes in ad hoc networks. However, it is to note that current market demands a reduction in the cost of production and the time to commercialisation of solutions and services. Under these conditions it is very difficult to guarantee an acceptable level of resilience. This problem becomes specially meaningful in those application contexts where the incorrect behaviour of the network may imply a severe economic or human loss. Even in less critic environments, resilience may have a decisive impact on the reputation of the service provider, thus conditioning the level of penetration of the product within the market.

This fact has increased the need for designing new and efficient techniques and tools to evaluate not only the functional aspects of ad hoc network systems, but also non-functional ones. Nevertheless, the difficulty in recreating the presence of changes, specially those referring to faults and attacks, in terms of controllability, repeatability, observability and portability is a challenging task in practice that, to date, has limited the achievement of this goal.

This thesis has explored existing gaps in the practical evaluation of ad hoc routing protocols, which are the most sensitive elements in the behaviour of ad hoc networks. In such a way, if the routing protocol fails, communications will be rarely possible beyond one hop.

The absence of works in this domain, has lead us to articulate our research around three basic questions: Which are the steps required to evaluate the resilience of ad hoc routing protocols in practice? Which faults should be considered and how to recreate them in realistic experimental conditions? Which processes may benefit from resilience evaluation?

The research developed around these question has led us to propose a novel experimental framework. Such framework, named REFRAHN (Resilience Evaluation FRamework for Ad Hoc Networks) defines a methodology and

implements a tool to carry out the resilience evaluation of ad hoc routing protocols, addressing essential issues such as (i) the definition of experiments which emphasise the need for recreating of the dynamic characteristics of real ad hoc network deployments, specially the mobility of nodes and the occurrence of faults; (ii) the execution of experiments, which considers the use of real devices executing real (non-simulated) routing protocols; and (iii) the ulterior analysis of measurements to deduce a complete set of performance, resilience and resources consumption measures.

REFRAHN pays special attention to the practical aspects of the evaluation related to the observability, controllability, repeatability and portability of results. So, on one hand, only the mobility of network nodes is emulated during the execution of experiments. A devoted control network is used to manage the visibility of nodes, thus (i) limiting the occurrence of intrusiveness-related problems that typically affect the quality of the evaluation results, and (ii) minimising the physical space occupied by the experimental platforms considered. On the other hand, REFRAHN is (i) flexible enough to enable the injection of the most representative faults in the domain of ad hoc routing protocols, such as *ambient noise*, *flooding attacks*, *intrusion-based attacks* and so on, and (ii) scalable enough to consider and include new types of faults in the future.

Through experimentation, REFRAHN has shown not only the coherence of evaluation results with what could be expected from the occurrence of faults in different network deployments, but it has shown also its usefulness to support different processes requiring the resilience evaluation of ad hoc routing protocols, such as (i) the detection of flaws and vulnerabilities in protocol implementations, (ii) the resilience benchmarking of different candidates, (iii) the fine tuning of routing protocols and their fault/intrusion tolerant complements and (iv) the design of new fault tolerance mechanisms when tuning is not enough to face existing problems.

Ad hoc networks are now in a stage where more practical aspects addressing the resilience evaluation of ad hoc routing protocols need to be investigated. This is essential so as to drive a stronger market penetration in the context of medium- and large-scale wireless networks and to enable the use of new applications and services in existing networks. Such aspects are the basis to stimulate market competition, as well as business models and realistic use cases to make this technology appealing for public institutions and private companies.

Section 6.1 proposes some working lines that, following the research presented in this thesis, aim at coping with these ambitious challenges. Section 6.2 lists the papers published of in the framework of this thesis. Finally Section 6.3 thanks the funds received to support this work.

6.1 Future work

The work initiated in this thesis does not finish in the present document. Many ideas can be suggested while yet leaving room for improvements. Hereafter, some future research and work directions that can be immediately continued are cited.

6.1.1 Future research

- *Refining and improving the fault-generator model:* More research is required to increase the type of faults considered by this work. In essence, the faults considered in this thesis are impairments to primary attributes of resilience (such as availability and integrity). However, different faults not considered here may impact on secondary attributes such as authentication or privacy, more oriented towards

the domain of security. For example, data mining attacks could infer patters about the behaviour of users or the type of information they exchange even if the traffic is encrypted. There exist techniques addressed to preserve anonymity on communications, like the Host Identity Protocol (HIP), that can be applied to ad hoc networks, as seen in [129]. Therefore, we could think of extending our fault models to cover scenarios that enable testing the robustness of the fault-tolerance implementations addressed to protect such network attributes.

- *Generic self-adaptive fault-tolerance*: On one hand, as seen along the thesis, the need for fault tolerance mechanisms is undeniable to enhance the resilience of ad hoc routing protocols. On the other hand, the wide diversity of routing protocols requires adapting the implementation of each fault tolerance strategy to each particular protocol, which may turn into a problem if routing protocol developers have limited skills on fault tolerance. Likewise, Aspect-Oriented Programming (AOP) and reflection paradigms could be a very interesting approach for the definition of adaptive fault tolerance approaches. On one hand, AOP [130] strongly eases the reuse and automation in the deployment of a given piece of code within a given component, which could be useful for the deployment of fault-tolerant mechanisms in the system. On the other hand, reflection [131] enables the real-time harvest of sensitive information from the system, which can be very useful to define self-adaptive strategies against faults. Despite being exploited in other domains, the combination of such paradigms has not been considered yet in the domain of ad hoc networks. Currently, we are already performing first steps towards the definition of fault-tolerance mechanisms for ad hoc networks based on such principles.

- *Making the mobility of nodes more realistic using real traces*: A geolocated dataset is a real collection of mobility traces for each one of the nodes comprising a network. This information may be collected either by locally logging the movements of each geolocated node, or centrally by a server with the capability of tracking the location of nodes [132]. One possible research line would be obtaining geolocated datasets to infer the movement from real traces for our testbed nodes. This approach could be useful to assess the impact of accidental faults and attacks in ad hoc networks where nodes move according to real traces, consequently improving the credibility of the evaluation results. The Crawdad project is an example of a public repository giving access to geolocated datasets, which can be used for this research purpose [133].

- *Towards standardisation and certification:* The development of procedures and guidelines for the certification of the resilience of software components has always been a need for the industry related to safety-critical systems, as unexpected failures may endanger the environment and human life. A number of standards, like [134], has adopted a Safety Integrity Level (SIL) ranging from 1 (minor property and production protection) to 4 (catastrophic community impact), as a statistical representation of the dependability of safety instrumented system. Accordingly, in order to meet the overall safety requirements, the most common safety standards such as IEC 61508 for electronic systems [134], DO-178B for airborne systems [135], and EN 50128 for railway systems [136], present very strict requirements on software development and testing. However, once the integration of untrusted third party COTS components is indispensable to meet time-to-market and development costs (as routing protocols in the domain of ad hoc networks), the resilience assessment of even non-critical systems is a must. Following this trend, the ISO/IEC

Software product Quality Requirements and Evaluation (SQuaRE) [137], which defines procedures for evaluating the quality of software components, has extended its scope to enable the evaluation of the system recoverability [138]. Although being a remarkable effort, this approach strictly focuses on recoverability, thus neglecting the rest of dependability-related attributes, and may only be used for fault-tolerant systems, thus reducing its applicability and usefulness. It would be interesting to integrate resilience evaluation into the ISO/IEC SQuaRE standard to tackle existing limitations in the quality evaluation of software products like ad hoc routing protocols not only in presence of accidental faults (as proposed by the ISO/IEC 25045 module for recoverability), but also addressing the occurrence of malicious ones (attacks). The resulting approach would enrich the ISO/IEC SQuaRE standard and enlarge its application domain for evaluating the quality of software COTS components from a performance and a resilience point of view even for non-critical applications.

6.1.2 Future development and empirical work

- *Enhancing the tool usability*: Implementing a graphical user interface to support the interactive analysis of results could be useful to ease the communication between REFRAHN, and the final user. This step is essential to enhance the usability of our approach after the experimentation deployment.

- *Addressing scalability issues*: Currently, our testbed is composed of 17 nodes. The plan is not only to extend it to a bigger testbed with larger number of nodes, but also considering additional types of wireless technologies, like Zigbee, Bluetooth, Wimax [139], and so on, to explore the influence of a given underlying physical layer in the net-

work robustness. From our experiences considering WiFI, managing stacks of more than 20 routers may revert on an increment of interferences, thus increasing the intrusiveness on the results obtained, and limiting the correctness of their analysis. A solution scaling our current testbed while avoiding such inherent problems may consist on adopting a hybrid approach. On one hand, including additional stacks of routers limiting the height of each stack to 15 nodes. On the other, reducing the transmission power of the NICs through hardware attenuators to limit the existence of interferences caused by adjacent nodes. This solution enables increasing the scalability of our testbed up to hundred of nodes.

- *Integrating simulation in REFRAHN*:

  Simulation and real-life experimentation are destined to complement each other. On one hand, results from real-life experiments are required to validate the mathematical models and the simulations tools and provide them with adjustments to refine their accuracy and extend their scope. On the other, simulation is essential to show the feasibility of prototypes when the number of nodes of testbed limits the scalability of required network deployments.

  Incorporating simulation within REFRAHN could be very useful to execute the very same experiment configuration into both emulation- and simulation-based evaluation platforms, and compare the trend of the results obtained (back to back testing [140]). In case of providing statistically similar results, experimenters may decide between simulations if they are interested in saving time (for example, if they just need to validate or refuse a given hypothesis, or they want to perform a large experiments campaign in a wider scale) or real experiments if they want to perform a more thorough experience (typical choice in advanced steps of the software life-cycle) for their further tests.

6.2 Dissemination of this thesis

The research work related to this thesis has resulted in 17 publications: 1 journal article (indexed as *Q1* by the Journal Citation Reports (JCR) database), and 16 conference papers (4 of them indexed as *A*, and 2 of them indexed as *B*, by the Computer Science Conference Ranking or the Computing Research and Education (CORE) lists). Furthermore, some of the conferences where these papers have been presented are recommended by the International Federation for Information Processing (IFIP) Working Group 10.4 on Dependable Computing and Fault Tolerance.

6.2.1 Journals

- **Jesús Friginal**, David de Andrés, Juan-Carlos Ruiz, Pedro Gil. Towards Benchmarking Routing Protocols in Wireless Mesh Networks, Ad Hoc Networks, 2011, Volume 9, Issue 8, Pages 1374-1388, ISSN 1570-8705.
  (JCR 1st quartile)

6.2.2 Indexed conferences

- **Jesús Friginal**, Juan-Carlos Ruiz, David de Andrés, Antonio Bustos. Mitigating the Impact of Ambient Noise on Wireless Mesh Networks Using Adaptive Link-Quality-based Packet Replication. 42th IEEE/IFIP International Conference on Dependable Systems and Networks (DSN) [1] , 2012, Boston (USA), Pages 8, ISBN 1-4673-

[1] Conference recommended by the IFIP Working Group 10.4 on Dependable Computing and Fault Tolerance

1625-5.
(CORE A)

- **Jesús Friginal**, David de Andrés, Juan-Carlos Ruiz, Pedro Gil. Using Performance, Energy Consumption, and Resilience Experimental Measures to Evaluate Routing Protocols for Ad Hoc Networks. 10th IEEE Symposium on Network Computing and Applications (NCA), 2011, Cambridge (USA), Pages 139-146, ISBN 978-0-7695-4489-2.
(CORE A)

- **Jesús Friginal**, David de Andrés, Juan-Carlos Ruiz, Pedro Gil. Resilience-Driven Parameterisation of Ad Hoc Routing Protocols: olsrd as a Case Study. 30th IEEE Symposium on Reliable Distributed Systems (SRDS), 2011, Madrid (Spain), Pages 85-90, ISBN 978-0-7695-4450-2.
(CORE A)

- **Jesús Friginal**, David de Andrés, Juan-Carlos Ruiz, Regina Moraes. Using resilience Benchmarking to Support ISO/IEC SQuaRE. 17th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC) [1] , 2011, Pasadena (USA), Pages 28-37, ISBN 978-0-7695-4590-5.
(CORE B)

- **Jesús Friginal**, David de Andrés, Juan-Carlos Ruiz, Pedro Gil. Attack Injection to Support the Evaluation of Ad Hoc Networks. 29th International Symposium on Reliable Distributed Systems (SRDS), 2010, New Delhi (India), Pages 21-29, ISBN 978-0-7695-4250-8.
(CORE A)

- David de Andrés, **Jesús Friginal**, Juan-Carlos Ruiz, Pedro Gil. An Attack Injection Approach to Evaluate the Robustness of Ad Hoc Networks. 15th IEEE Pacific Rim on Dependable Computing

(PRDC) [1] , 2009, Shanghai (China), Pages 228-233, ISBN 978-0-7695-3849-5.
(CORE B)

6.2.3 Other papers

- Antonio Bustos, **Jesús Friginal**, David de Andrés, Juan-Carlos Ruiz. An Aspect-Oriented Approach to Face Neighbour Saturation Issues in Proactive Ad hoc Routing Protocols: olsrd as a case study, 1st International workshop on Approaches to Mobiquitous Resilience (ARMOR), 2012, Sibiu (Romania), Pages 6, ISBN 978-1-4503-1150-2.

- **Jesús Friginal**, David de Andrés, Juan-Carlos Ruiz, Pedro Gil. On Selecting Representative Faultloads to Guide the Evaluation of Ad Hoc Networks, 5th Latin-American Symposium on Dependable Computing (LADC) [1] , 2011, Sao Jose dos Campos (Brazil), Pages 94-99, ISBN 978-1-4244-9700-3.

- **Jesús Friginal**, Juan-Carlos Ruiz, David de Andrés, Pedro Gil. Coarse-grained Resilience Benchmarking Using Logic Score of Preferences: Ad Hoc Networks As a Case Study, 13th European Workshop on Dependable Computing (EWDC), 2011, Pisa (Italy), Pages 23-28, ISBN 978-1-4503-0284-5.

- **Jesús Friginal**, Juan-Carlos Ruiz, David de Andrés, Pedro Gil. Hierarchical Analysis of Resilience Benchmarking Results Using LSP: Ad Hoc Networks As a Case Study, Jornadas de Paralelismo (JP), 2011, La Laguna (Spain), Pages 373-378, ISBN 978-84-694-1791-1.

- **Jesús Friginal**, David de Andrés, Juan-Carlos Ruiz, Pedro Gil. Characterising Networking Problems in Ambient Intelligence Networks, 4th Symposium of Ubiquitous Computing and Ambient In-

telligence (UCAMI), 2010, Valencia (Spain), Pages 13-22, ISBN 978-84-92812-61-5.

- **Jesús Friginal**, David de Andrés, Juan-Carlos Ruiz, Pedro Gil. Key Factors for Attack Injection on MANETs: Towards Enhancing Experiment Representativeness, Fast Abstract on the 8th European Dependable Computing Conference (EDCC), 2010, Valencia (Spain), Pages 51-52, ISBN 978-84-692-9571-7.

- **Jesús Friginal**, Juan-Carlos Ruiz, David de Andrés, Pedro Gil. Attack Injection for Performance and resilience Assessment of Ad-Hoc Networks, 12th European Workshop on Dependable Computing (EWDC), 2009, Toulouse (France).

- **Jesús Friginal**. Towards Benchmarking the Performance and resilience of Ad Hoc Networks in Presence of Attacks, Student Forum of the IEEE/IFIP 39th International Conference on Dependable Systems and Networks (DSN), 2009, Estoril (Portugal).

- Juan-Carlos Ruiz, **Jesús Friginal**, David de Andrés, Pedro Gil. Towards Assessing the Resilience of Ad-hoc Proactive Routing Protocols against Dataflow Disruption, 1st Sharing Field Data and Experiment Measurements on Resilience of Distributed Computing Systems workshop (RCDS), 2008, Naples (Italy).

- Juan-Carlos Ruiz, **Jesús Friginal**, David de Andrés, Pedro Gil. Black Hole Attack Injection in Ad hoc Networks, Supplementary volume of the IEEE/IFIP 38th International Conference on Dependable Systems and Networks (DSN), 2008, Anchorage (USA), Pages 34-35.

6.2.4 Awards

The work presented in this thesis has been recognised by the Intel Doctoral Student Honour Programme[1] 2012, a competitive fellowship programme for outstanding PhD students in Engineering, Computer Science, Social Science, and other technical related majors focusing on high computing technologies from EU, Russian and Swiss universities.

6.3 Projects supporting this thesis

This thesis has been funded by the following projects:

- Generación e Integración automática de Mecanismos para la mejora de la Confiabilidad y la Seguridad de circuitos VLSI (GIMCS), PAID-06-10-2388. Project sponsored by the UPV.
- Sistemas EMpotrados SEguros y Confiables bAsados en comPonentes (SEMSECAP), TIN-2009-13825. Project sponsored by the Spanish Ministry of Science and Innovation. Partners: UPV (SP). More info at http://semsecap.stf.webs.upv.es
- Reaction After Detection (RED), CP3-011. Project sponsored by the European EUREKA-CELTIC program. Partners: Alcatel-Lucent (FRA), Telefónica (SP), CRP Henri Tudor (LUX), EADS (FRA), Exaprotect (FRA), France Télécom (FRA), Telecom Bretagne (FRA), GMV (SP), Innovae (SP), Telindus (LUX), Thales (FRA), UPM (SP), UPV (SP). More info at http://projects.celtic-initiative.org/red

[1] https://intelfellowships.com/eu

References

[1] ARTEMIS Platform: Advanced Research and Technology for EMbedded Intelligence and Systems. online: http://www.artemis.eu, 2012. 1

[2] Seventh Framework Programme Calls. online: http://cordis.europa.eu/fp7/ict/programme/home_en.html, 2012. 1

[3] Global Technological and Societal Trends. online: http://www.internet-of-things-research.eu/documents.htm, 2012. 1

[4] Mark Weiser. Some computer science issues in ubiquitous computing. *Commun. ACM*, 36(7):75–84, July 1993. ISSN 0001-0782. 2

[5] M. Natu and A. S. Sethi. Adaptive fault localization in mobile ad hoc battlefield networks. In *IEEE Military Communications Conference*, pages 814–820, 2005. 2

[6] U.S. Executive Office of the President. The Federal Response to Hurricane Katrina: Lessons Learned, 2006. 2, 9

[7] S. Vijayaragavan et al. A Performance Study of Reactive Multicast Routing Protocols in Virtual Class Room Using Mobile Ad Hoc Network. *Journal of Computer Science*, 5(11):788–793, 2009. 2

[8] Valery Naumov et al. An Evaluation of Inter-Vehicle Ad Hoc Networks Based on Realistic Vehicular Traces. In *Int. Symp. Mobile Ad Hoc Net. & Computing*, pages 108–119, 2006. 2

[9] EU FP7. Ambient Assisted Living Joint Programme. *http://www.aal-europe.eu/Projectdoc.html*, 2012. 2

[10] PECES project: PErvasive Computing in Embedded Systems (FP7-INFSO-ICT-224342). online: http://www.ict-peces.eu, 2012. 2

[11] LocOn project: Platform for an inter-working of embedded localisation and communication systems (FP7-INFSO-ICT-224148). online: http://www.locon.org, 2012. 2

[12] Paulo Veríssimo, Nuno Ferreira Neves, Christian Cachin, Jonathan A. Poritz, David Powell, Yves Deswarte, Robert J. Stroud, and Ian Welch. Intrusion-tolerant middleware: the road to automatic security. *IEEE Security & Privacy*, 4(4):54–62, 2006. 3

[13] Jean-Claude Laprie. Resilience for the scalability of dependability. In *Proceedings of the Fourth IEEE International Symposium on Network Computing and Applications*, NCA '05, pages 5–6, 2005. ISBN 0-7695-2326-9. 3, 10, 29

[14] Central Nervous System for the Earth (CeNSE). online: http://www.hpl.hp.com/news/2009/oct-dec/cense.html, 2012. 3

[15] Cooperating Embedded Systems for Exploration and Control featuring Wireless Sensor Networks (WiSeNts). online: http://www.embedded-wisents.org, 2012. 3

[16] GENI project. The Global Environment for Network Innovations (GENI). [Online]. Available: http:www.geni.net/, 2012. 3

[17] Fan Bai, Ganesha Bhaskara, and Ahmed Helmy. Building the blocks of protocol design and analysis: challenges and lessons learned from case studies on mobile ad hoc routing and micro-mobility protocols. *SIGCOMM Comput. Commun. Rev.*, 34(3):57–70, July 2004. ISSN 0146-4833. 4, 37

[18] José García-Nieto and Enrique Alba. Automatic parameter tuning with metaheuristics of the aodv routing protocol for vehicular ad-hoc networks. In *Proceedings of the 2010 international conference on Applications of Evolutionary Computation - Volume Part II*, EvoCOMNET'10, pages 21–30, 2010. ISBN 3-642-12241-8, 978-3-642-12241-5. 4, 37

[19] David Johnson and Albert Lysko. Comparison of manet routing protocols using a scaled indoor wireless grid. *Mob. Netw. Appl.*, 13(1-2): 82–96, April 2008. ISSN 1383-469X. 4, 37

[20] Shah-An Yang. *Development and evaluation of methodologies for vulnerability analysis of ad-hoc routing protocols.* PhD thesis, College Park, MD, USA, 2007. AAI3297357. 4, 37

[21] Jian-Kai Chen, Chien Chen, Rong-Hong Jan, and Hsia-Hsin Li. Expected link life time analysis in manet under manhattan grid mobility model. In *Proceedings of the 11th international symposium on Modeling, analysis and simulation of wireless and mobile systems*, MSWiM '08, pages 162–168, 2008. ISBN 978-1-60558-235-1. 4, 40

[22] Chung-Ming Huang, Kun-chan Lan, and Chang-Zhou Tsai. A survey of opportunistic networks. In *Proceedings of the 22nd International Conference on Advanced Information Networking and Applications - Workshops*, AINAW '08, pages 1672–1677, 2008. ISBN 978-0-7695-3096-3. 8

[23] Haowei Bai et al. Wireless sensor network for aircraft health monitoring. In *Proceedings of the First International Conference on Broadband Networks (BROADNETS)*, pages 748–750, 2004. ISBN 0-7695-2221-1. 8

[24] I. F. Akyildiz and others. Wireless mesh networks: a survey. *IEEE Radio Communications*, 43:S23–S30, 2005. 9

[25] M. Asefi, J.W. Mark, and Xuemin Shen. A mobility-aware and quality-driven retransmission limit adaptation scheme for video streaming over vanets. In *IEEE Global Telecommunications Conference (GLOBECOM)*, pages 1 –5, dec. 2011. 9, 27

[26] E. Sakhaee, A. Jamalipour, and N. Kato. Aeronautical ad hoc networks. In *Wireless Communications and Networking Conference, 2006. WCNC 2006. IEEE*, volume 1, pages 246 –251, april 2006. 9

[27] Stuart Kurkowski, Tracy Camp, and Michael Colagrosso. Manet simulation studies: the incredibles. *SIGMOBILE Mob. Comput. Commun. Rev.*, 9(4):50–61, October 2005. ISSN 1559-1662. 10, 16

[28] Xian Ni, Kun-chan Lan, and Robert Malaney. On the performance of expected transmission count (etx) for wireless mesh networks. In *Proceedings of the 3rd International Conference on Performance Evaluation Methodologies and Tools (ValueTools)*, pages 77:1–77:10, 2008. 14

[29] T. Clausen and P. Jacquet. Optimized Link State Routing Protocol(OLSR). *RFC 3626*, 2003. 14, 74, 95, 143

[30] OpenMesh. Open Mesh, Better Approach To Mobile Ad hoc Networking (B.A.T.M.A.N.). [Online]. Available: http://www.open-mesh.net/, 2012. 14, 74, 143, 144

[31] Juliusz Chroboczek. BABEL. [Online]. Available: http://www.pps.jussieu.fr/ jch/software/babel/, 2012. 14, 74, 143, 144

[32] C. Perkings. Ad hoc On-Demand Distance Vector(AODV) Routing. *RFC 3561*, 2003. 14, 74

[33] Ansgar Fehnker, Lodewijk Van Hoesel, and Angelika Mader. Modelling and verification of the lmac protocol for wireless sensor networks. In *Proceedings of the 6th international conference on Integrated formal methods*, IFM'07, pages 253–272, 2007. ISBN 978-3-540-73209-9. 15

[34] Simon Tschirner, Liang Xuedong, and Wang Yi. Model-based validation of qos properties of biomedical sensor networks. In *Proceedings of the 8th ACM international conference on Embedded software*, EMSOFT '08, pages 69–78, 2008. ISBN 978-1-60558-468-3. 15

[35] Stéphane Maag and Fatiha Zaidi. Testing methodology for an ad hoc routing protocol. In *Proceedings of the ACM international workshop on Performance monitoring, measurement, and evaluation of heterogeneous wireless and wired networks*, PM2HW2N '06, pages 48–55, 2006. ISBN 1-59593-502-9. 15, 16, 24

[36] Marcello Cinque, Domenico Cotroneo, Catello Di Martinio, and Stefano Russo. Modeling and assessing the dependability ofwireless sensor networks. In *IEEE Symposium on Reliable Distributed Systems*, SRDS '07, pages 33–44, Los Alamitos, CA, USA, 2007. IEEE Computer Society. ISBN 0-7695-2995-X. 16

[37] Valery Naumov et al. An Evaluation of Inter-Vehicle Ad Hoc Networks Based on Realistic Vehicular Traces. In *Int. Symp. on Mobile Ad Hoc Net. and Comp.*, pages 108–119, 2006. 16, 21

[38] Azzedine Boukerche. Performance Evaluation of Routing Protocols for Ad Hoc Wireless Networks. *Mobile Networks and Applications*, 9 (4):333–342, 2004. ISSN 1383-469X. 16, 21

[39] Imad Aad et al. Impact of denial of service attacks on ad hoc networks. *IEEE Trans. on Networking*, 16(4):791–802, 2008. ISSN 1063-6692. 16, 21, 46

[40] Darlan Vivian et al. Evaluation of QoS Metrics in Ad Hoc Networks with the Use of Secure Routing Protocols. In *Network Operations and Management Symp.*, pages 1–14, 2006. 16, 21

[41] J. Yoon, M. Liu, and B. Noble. Random waypoint considered harmful. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 2, pages 1312 – 1321 vol.2, march-3 april 2003. 16

[42] Henrik Lundgren, Erik Nordstrom, and Christian Tschudin. The gray zone problem in ieee 802.11b based ad hoc networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 6(3):104–105, June 2002. ISSN 1559-1662. 16

[43] Erik Nordström et al. A testbed and methodology for experimental evaluation of wireless mobile ad hoc networks. In *Testbeds and Research Infrastructures for the Development of Networks and Communities*, pages 100–109, Italy, 2005. 17, 20, 21

[44] Roofnet testbed. [Online]. Available: http://pdos.csail.mit.edu/roofnet, 2012. 17, 20

[45] floornet. Floornet: A Wireless Multihop Testbed. [Online]. Available: http://http://floornet.org/, 2012. 17, 20

[46] Paul Johnson, Ehsan Nourbakhsh, T. Ryan Burchfield, Jeff Dix, Ravi Prakash, Subbarayan Venkatesan, and Neeraj Mittal. Assert: Advanced wireless environment research testbed. In *SenSys '09: Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, pages 297–298, 2009. 17

[47] ORBIT project. The ORBIT radio grid emulator (ORBIT). [Online]. Available: http://www.orbit-lab.org/, 2012. 17, 20

[48] The Carnegie Mellon University Wireless Emulator. [Online]. Available: http://www.cs.cmu.edu/ emulator/, 2012. 18, 20

[49] Tao Lin, S.F. Midkiff, and J.S. Park. A dynamic topology switch for the emulation of wireless mobile ad hoc networks. In *27th Annual IEEE Conference on Local Computer Networks (LCN)*, pages 791 – 798, nov. 2002. 18

[50] J. Hortelano et al. Castadiva: A Test-Bed Architecture for Mobile AD HOC Networks. In *IEEE Symposium on Personal, Indoor and Mobile Radio Communications*, pages 1–5, 2007. 18, 20, 73

[51] Mobility Emulator (MobiEmu). [Online]. Available: http://mobiemu.sourceforge.net/, 2012. 18, 20

[52] Seawind, a Network Emulator for Wireless Links . [Online]. Available: http://www.cs.helsinki.fi/research/iwtcp/seawind/, 2012. 18, 20

[53] Opnet. OPNET Simulator. [Online]. Available: http://www.opnet.com, 2012. 20

[54] Network Simulator. [Online]. Available: http://www.nsnam.org/, 2012. 20

[55] GloMoSim. [Online]. Available: http://pcl.cs.ucla.edu/projects/glomosim/, 2012. 20

[56] Mei-Chen Hsueh, Timothy K. Tsai, and Ravishankar K. Iyer. Fault injection techniques and tools. *IEEE Computer*, 30(4):75–82, 1997. 20, 23, 58

[57] R. V. Boppana et al. Evaluation of a Stastical Technique to Mitigate Malicious Control Packets in Ad Hoc Networks. In *Int. Symp. on World of Wireless, Mobile and Multimedia Networks*, pages 559–563, USA, 2006. 21

[58] Eleonora Borgia. Experimental Evaluation of Ad Hoc Routing Protocols. In *Int. Conf. on Pervasive Computing and Communications Workshops*, pages 232–236, USA, 2005. ISBN 0-7695-2300-5. 21

[59] Fei Ye Azzedine. Experimental Evaluation of Ad Hoc Testbed in Indoor Scenarios. In *Asia-Pacific Conference on Communications*, pages 1–5, Busan, 2006. 21

[60] T. Clausen et al. The Optimized Link State Routing Protocol, Evaluation through Experiments and Simulation. In *Int. Symp. on Wireless Personal Multimedia Com.*, page 6, 2001. 21

[61] C. Gomez et al. Improving Performance of a Real Ad Hoc Network by Tuning OLSR Parameters. In *Symposium on Computers and Communications*, pages 16–21, USA, 2005. ISBN 0-7695-2373-0. 21, 26, 46

[62] Koojana Kuladinithi et al. DEL - Experimental Performance Evaluation of AODV Implementations in Static Environments, 2007. 21, 46

[63] Giuseppe De Marco et al. Experimental Performance Evaluation of a Pro-Active Ad-hoc Routing Protocol in Out- and Indoor Scenarios. In *Int. Conf. on Advanced Networking and Applications*, pages 7–14, USA, 2007. ISBN 0-7695-2846-5. 21

[64] Makoto Ikeda et al. Experimental and Simulation Evaluation of OLSR Protocol for Mobile Ad-Hoc Networks. In *International Conference on Network-Based Information Systems*, pages 111–121, Turin, Italy, 2008. ISBN 978-3-540-85692-4. 21

[65] C. Mbarushimana et al. Comparative Study of Reactive and Proactive Routing Protocols Performance in Mobile Ad Hoc Networks. In *Int. Conf. on Advanced Information Networking and Applications Workshops*, pages 679–684, 2007. ISBN 0-7695-2847-3. 21

[66] Peng Ning and Kun Sun. How to Misuse AODV: A Case Study of Insider Attacks Against Mobile Ad-Hoc Routing Protocols. In *IEEE Information Assurance Workshop*, pages 60–67, 2003. 21, 49

[67] Nadia Qasim et al. Mobile Ad Hoc Networking Protocols' Evaluation through Simulation for Quality of Service. *IAENG Int. Journal of Computer Science*, 36(1):1–10, 2009. 21

[68] Peizhao Hu et al. Experimental Evaluation of AODV in a Hybrid Wireless Mesh Network. In *Work. on the Internet, Telecommunications and Signal Processing*, page 6, Australia, 2006. 21

[69] Yuxia Lin et al. Experimental Comparisons Between SAODV and AODV Routing Protocols. In *Workshop on Wireless Multimedia Networking and Performance Modeling*, pages 113–122, Montreal, Canada, 2005. ISBN 1-59593-183-X. 21

[70] IFIP Working Group 10.4 on DEPENDABLE COMPUTING AND FAULT TOLERANCE. Online: http://www.dependability.org/wg10.4/, 2012. 10

[71] Jean Arlat, Martine Aguera, Louis Amat, Yves Crouzet, Jean-Charles Fabre, Jean-Claude Laprie, Eliane Martins, and David Powell. Fault

injection for dependability validation: A methodology and some applications. *IEEE Trans. Softw. Eng.*, 16:166–182, February 1990. ISSN 0098-5589. 23

[72] Algirdas Avizienis. Fault-tolerant systems. *IEEE Trans. Computers*, 25(12):1304–1312, 1976. 23

[73] D. Costa, T. Rilho, and H. Madeira. Joint evaluation of performance and robustness of a cots dbms through fault-injection. In *Proceedings International Conference on Dependable Systems and Networks (DSN)*, pages 251 –260, 2000. 23

[74] K. Kanoun, Y. Crouzet, A. Kalakech, A.-E. Rugina, and P. Rumeau. Benchmarking the dependability of windows and linux using postmark/spl trade/ workloads. In *16th IEEE International Symposium on Software Reliability Engineering (ISSRE)*, pages 10 –20, nov. 2005. 23

[75] International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC). ISO/IEC 25000. Software Engineering - Software product Quality Requirements and Evaluation (SQuaRE) - Guide to SQuaRE. Geneve ISO, 2010. 23, 31

[76] Luciana Moreira. Ft-cowisenets: A fault tolerance framework for wireless sensor networks. *International Conference on Sensor Technologies and Applications*, pages 289–294, 2007. 24

[77] Bing Wu et al. A survey on attacks and countermeasures in mobile ad hoc networks. In *Wireless/Mobile networks security*. Springer-Verlag, 2006. 24, 27, 42, 43, 53, 55, 57, 58

[78] R. Prasad and A. Mihovska. *New Horizons in Mobile and Wireless Communications: Ad hoc networks and PANs*. Mobile Communications. Artech House, 2009. ISBN 9781607839736. 24

[79] Daniele Raffo. *Security Schemes for the OLSR Protocol for Ad Hoc Networks*. PhD thesis, Université Paris 6 and INRIA Rocquencourt, 2012. URL `http://perso.crans.org/~raffo/papers/raffo-phdthesis.ps.gz`. 25, 27

[80] Intae Kang and R. Poovendran. Maximizing static network lifetime of wireless broadcast ad hoc networks. In *IEEE International Conference on Communications (ICC)*, volume 3, pages 2256 – 2261 vol.3, may 2003. 25

[81] Jae seung Yeom, N. Wisitpongphan, S. Panichpapiboon, and O.K. Tonguz. A testbed emulator for cross-layer studies in mobile ad hoc wireless networks. In *3rd International Conference on Testbeds and Research Infrastructure for the Development of Networks and Communities (TridentCom)*, pages 1 –10, may 2007. 25

[82] Ahmad Al Hanbali, Eitan Altman, and Nain Philippe. A survey of tcp over ad hoc networks. *IEEE Communications Surveys and Tutorials*, 1(4):22–36, 2005. 25

[83] Henrik Lundgren, David Lundberg, Johan Nielsen, Erik Nordström, and Christian Tschudin. A large-scale testbed for reproducible ad hoc protocol evaluations. In *WCNC'02: Proceedings of the IEEE Wireless Communications and Networking Conference*, 2002. 25, 38

[84] Anmol Sheth, Christian Doerr, Dirk Grunwald, Richard Han, and Douglas Sicker. Mojo: a distributed physical layer anomaly detection system for 802.11 wlans. In *MobiSys '06: Proceedings of the 4th international conference on Mobile systems, applications and services*, pages 191–204, 2006. ISBN 1-59593-195-3. 26

[85] J. Ben-Othman and A. Hamieh. Defending method against jamming

attack in wireless ad hoc networks. In *IEEE 34th Conference on Local Computer Networks (LCN)*, pages 758 –762, oct. 2009. 26

[86] Y. Zhou et al. Balancing the hidden and exposed node problems with power control in csma/cabased wireless networks. In *Proceedings of IEEE Wireless Communications and Networking Conference (WCNC)*, pages 683–688, 2005. 26

[87] Xinwen Fu. *On traffic analysis attacks and countermeasures*. PhD thesis, College Station, TX, USA, 2005. AAI3246468. 26

[88] Chun-Ta Li and Yen-Ping Chu. Cryptanalysis of threshold password authentication against guessing attacks in ad hoc networks. *I. J. Network Security*, pages 166–168, 2009. 26

[89] J. Hoydis, M. Petrova, and P. Mahonen. Effects of topology on local throughput-capacity of ad hoc networks. In *IEEE 19th International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pages 1 –5, sept. 2008. 26

[90] Hoang Lan Nguyen and Uyen Trang Nguyen. A study of different types of attacks on multicast in mobile ad hoc networks. *Ad Hoc Netw.*, 6(1):32–46, 2008. ISSN 1570-8705. 27

[91] Imad Aad. Impact of denial of service attacks on ad hoc networks. *IEEE/ACM Trans. Netw.*, 16(4):791–802, 2008. ISSN 1063-6692. 27, 43

[92] J.-C. Ruiz et al. Black Hole Attack Injection in Ad hoc Networks. In *Supplemental Volume of IEEE Dependable Systems and Networks*, pages G34–G35, USA, 2008. 27

[93] Michele Nogueira Lima, Aldri L. dos Santos, and Guy Pujolle. A survey of survivability in mobile ad hoc networks. *IEEE Communications Surveys and Tutorials*, 11(1), 2009. 27

[94] A. Avizienis et al. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33, Jan–March 2004. ISSN 1545-5971. 31, 45

[95] Christian Bettstetter, Hannes Hartenstein, and Xavier Pérez-Costa. Stochastic properties of the random waypoint mobility model. *Wirel. Netw.*, 10(5):555–567, September 2004. ISSN 1022-0038. 40

[96] Marvin McNett and Geoffrey M. Voelker. Access and mobility of wireless pda users. *SIGMOBILE Mob. Comput. Commun. Rev.*, 9 (2):40–55, April 2005. ISSN 1559-1662. 40

[97] Roberto Natella, Domenico Cotroneo, Joao A. Duraes, and Henrique S. Madeira. On fault representativeness of software fault injection. *IEEE Transactions on Software Engineering*, 99(PrePrints), 2011. ISSN 0098-5589. 41

[98] D.F. Macedo, L.H.A. Correia, A.L. dos Santos, A.A.F. Loureiro, and J.M. Nogueira. Evaluating fault tolerance aspects in routing protocols for wireless sensor networks. In *Fourth Annual Mediterranean Ad Hoc Networking Workshop*, 2005. 42, 52

[99] A.D. Wood and J.A. Stankovic. *A taxonomy for denial-of-service attacks in wireless sensor networks.* Chapter 32. CRC Press LLC, 2005. 43, 54, 56

[100] European New Car Assessment Programme (EuroNCAP). online: www.euroncap.com/, 2012.

[101] P. Serrano, C.J. Bernardos, A. de la Oliva, and I. Soto. Lessons learned from the deployment of a multihop ieee 802.11g testbed using cots devices. In *Wireless Conference (EW), 2010 European*, pages 667 –674, april 2010. 46

[102] Anderson Morais, Ana Cavalli, and Eliane Martins. A model-based attack injection approach for security validation. In *Proceedings of the 4th international conference on Security of information and networks*, SIN '11, pages 103–110, 2011. ISBN 978-1-4503-1020-8. 49

[103] Yi an Huang and Wenke Lee. Attack analysis and detection for ad hoc routing protocols. In *In Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection (RAID*, pages 125–145, 2004. 49

[104] Karama Kanoun and Lisa Spainhower, editors. *Dependability Benchmarking for Computer Systems.* Wiley and IEEE Computer Society Press, 2008. 51, 114

[105] J. Yeom et al. A testbed emulator for cross-layer studies in mobile ad hoc wireless networks. In *International Conference on Testbeds and Research Infrastructure for the Development of Networks and Communities (TridentCom)*, page 10, 2007. 51

[106] Patrick McHardy et al. Iptables. [Online]. Available: http://www.netfilter.org, 2012. 72

[107] Ekiga official site. *http://ekiga.org*, 2012. 75

[108] FileZilla. [Online]. Available: http://filezilla-project.org/, 2012. 75

[109] Parrot AR.Drone. [Online]. Available: ardrone.parrot.com/, 2012. 75

[110] Iperf tool. [Online]. Available: http://sourceforge.net/projects/iperf/, 2012. 75

[111] Van Jacobson, Craig Leres, and Steven McCanne. Tcpdump/Libpcap. [Online]. Available: http://www.tcpdump.org/, 2012. 76

[112] Nemesis tool. [Online]. Available: http://sourceforge.net/projects/nemesis/, 2012. 77

[113] Tcpreplay suite. [Online]. Available: http://tcpreplay.synfin.net/, 2012. 78

[114] Network Emulator (Netem). [Online]. Available: http://www.linuxfoundation.org, 2012. 78

[115] Laura Marie Feeney. An energy consumption model for performance analysis of routing protocols for mobile ad hoc networks. *Mob. Netw. Appl.*, 6:239–249, 2001. 85

[116] Vincent Lenders, Jorg Wagner, and Martin May. Measurements from an 802.11b mobile ad hoc network. In *WOWMOM '06: Proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks*, pages 519–524, 2006. ISBN 0-7695-2593-8. 97

[117] Jesus Friginal, David de Andres, Juan-Carlos Ruiz, and Pedro Gil. Attack injection to support the evaluation of ad hoc networks. *IEEE Symposium on Reliable Distributed Systems (SRDS)*, pages 21–29, 2010. ISSN 1060-9857. doi: http://doi.ieeecomputersociety.org/10.1109/SRDS.2010.11. 113

[118] J.J. Dujmovic and R. Elnicki. *A DMS Cost/Benefit Decision Model: Mathematical Models for Data Management System Evaluation, Comparison, and Selection.* National Bureau of Standards, Washington D.C., No. GCR 82-374. NTIS No. PB 82-170150, 1982. 115, 119

[119] Michael F. Morris. Kiviat graphs: conventions and figures of merit. *ACM/Sigmetrics Performance Evaluation Review*, 3(3):2–8, 1974. 116

[120] David de Andres. Using dependability, performance, area and energy consumption experimental measures to benchmark ip cores. In *Forth Latin American Symposium on Dependable Computing (LADC)*, 2009. 116

[121] Giulio Concas, Michele Marchesi, Sandro Pinna, and Nicola Serra. Power-laws in a large object-oriented software system. *IEEE Trans. Softw. Eng.*, 33:687–708, October 2007. ISSN 0098-5589. 116

[122] Yazeed A. Al-Sbou, Reza Saatchi, Samir Al-Khayatt, Rebecca Strachan, Moussa Ayyash, and Mohammad Saraireh. A novel quality of service assessment of multimedia traffic over wireless ad hoc networks. In *Proceedings of the 2008 The Second International Conference on Next Generation Mobile Applications, Services, and Technologies*, pages 479–484, 2008. ISBN 978-0-7695-3333-9. 117

[123] Jasmeet Chhabra et al. Real-world experiences with an interactive ad hoc sensor network. In *Proceedings of the 2002 International Conference on Parallel Processing Workshops*, pages 143–151, 2002. ISBN 0-7695-1680-7. 125

[124] Weiquan Lu, Winston K. G. Seah, Edwin W. C. Peh, and Yu Ge. Communications support for disaster recovery operations using hybrid mobile ad-hoc networks. In *Proceedings of the 32nd IEEE Conference on Local Computer Networks*, pages 763–770, 2007. ISBN 0-7695-3000-1. 125

[125] Jorge Hortelano, Juan-Carlos Ruiz, and Pietro Manzoni. Evaluating the usefulness of watchdogs for intrusion detection in vanets. In *IEEE International Conference on Communications Workshops (ICC)*, pages 1–5, 2010. 132

[126] Maxim Raya, Jean-Pierre Hubaux, and Imad Aad. Domino: a system

to detect greedy behavior in ieee 802.11 hotspots. In *Proceedings of the 2nd international conference on Mobile systems, applications, and services (MobiSys)*, pages 84–97, 2004. ISBN 1-58113-793-1. 142

[127] Nikolay Tcholtchev and Ranganai Chaparadza. *Autonomic Fault-Management and resilience from the perspective of the network operation personnel*. 2010. 142

[128] GDB: The GNU Project Debugger. online: http://sources.redhat.com/gdb/, 2012. 149

[129] Marga Nacher, Carlos T. Calafate, Juan Carlos Cano Escriba, and Pietro Manzoni. Quantifying traffic anonymity in manets: A case study. In *Ubiquitous and Future Networks (ICUFN), 2010 Second International Conference on*, pages 171 –176, june 2010. doi: 10.1109/ICUFN.2010.5547234. 169

[130] G.C. Murphy, R.J. Walker, and E.L.A. Banlassad. Evaluating emerging software development technologies: lessons learned from assessing aspect-oriented programming. *IEEE Transactions on Software Engineering*, 25(4):438 –455, 1999. 169

[131] Juan Carlos Ruiz, Marc-Olivier Killijian, Jean-Charles Fabre, and Pascale Thévenod-Fosse. Reflective fault-tolerant systems: From experience to challenges. *IEEE Trans. Comput.*, 52:237–254, 2003. ISSN 0018-9340. 169

[132] Sebastien Gambs, Marc-Olivier Killijian, and Miguel Nuñez del Prado Cortez. Show me how you move and i will tell you who you are. *IEEE Transactions on Data Privacy*, (4):103 –126, 2011. 170

[133] CRAWDAD: A Community Resource for Archiving Wireless Data At Dartmouth. online: http://crawdad.cs.dartmouth.edu/, 2012. 170

[134] IEC 61508. Functional safety of electrical/electronic/programmable electronic safety-related systems. IEC, 2010. 170

[135] DO-178B. Software Considerations in Airborne Systems and Equipment Certification. Radio Technical Commission for Aeronautics, 1992. 170

[136] EN 50128. Railway applications. Communications, signalling and processing systems. Software for railway control and protection systems. British Standards Institution, 2001. 170

[137] ISO/IEC 25010. Software Engineering - Software product Quality Requirements and Evaluation (SQuaRE) - Guide to SQuaRE. Geneve ISO, 2010. 171

[138] ISO/IEC 25045. Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - Evaluation module for recoverability. Geneve ISO, 2010. 171

[139] Wimax Forum . [Online]. Available: http://www.wimaxforum.org/, 2012. 171

[140] M. A. Vouk. Back-to-back testing. *Inf. Softw. Technol.*, 32(1):34–45, January 1990. ISSN 0950-5849. 172