Towards integrating multi-criteria analysis techniques in Dependability Benchmarking

Miquel Martínez Raga

Supervisors:
Dr. Juan Carlos Ruiz García (DISCA-UPV, Spain)
Dr. David de Andrés Martínez (DISCA-UPV, Spain)
Dr. Jesús Friginal López (LAAS-CNRS, France)

Master's Thesis
Master Universitario en Ingeniería de Computadores
Departamento de Informática de Sistemas y Computadores
Universitat Politècnica de València
September, 2013

UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Acknowledgements

I would like to thank my advisors, Dr. Juan Carlos Ruiz García, Dr. David de Andrés Martnez and Dr. Jesús Friginal Lópezor, for the incredible support and guidance they gave me during the progress of this master's thesis, and also for the patience they had when I got stuck and solved my doubts. After working side by side with them, I can say that they are very good professionals and awesome people, and the most important think, I consider them as friends.

Abstract

Increasing integration scales are promoting the development of myriads of new devices and technologies, such smartphones, ad hoc networks, or field-programmable devices, among others. The proliferation of such devices, with increasing autonomy and communication capabilities, is paving the way for a new paradigm known as Internet of Things, in which computing is ubiquitous and devices autonomously exchange information and cooperate among them and already existing IT infrastructures to improve peoples and societys welfare. This new paradigm leads to huge business opportunities to manufacturers, application developers, and services providers in very different application domains, like consumer electronics, transport, or health. Accordingly, and to make the most of these incipient opportunities, industry relies more than ever on the use and re-use of commercial off-the-shelf (COTS), developed either in-house or by third parties, to decrease time-to-market and costs. In this race for hitting the market first, companies are nowadays concerned with the dependability of both COTS and final products, even for non-critical applications, as unexpected failures may damage the reputation of the manufacturer and limit the acceptability of their new products. Therefore, benchmarking techniques adapted to dependability contexts (dependability benchmarking) are being deployed in order to assess, compare, and select, i) the best suited COTS, among existing alternatives, to be integrated into a new product, and ii) the configuration parameters setup that gets the best trade-off between performance and dependability. However, although dependability benchmarking procedures have been defined and applied to a wide set of application domains, no rigorous and precise decision making process has been established yet, thus hindering the main goal of these approaches: the fair and accurate comparison and selection of existing alternatives taking into account both performance and dependability attributes. Indeed, results extracted from experimentation could be interpreted in so many different ways, according to the context of use of the system and the subjectivity of the benchmark analyser, that defining a clear and accurate decision making process is a must to enable the reproducibility of conclusions. Thus, this master thesis focuses on how integrating a decision making methodology into the regular dependability benchmarking procedure. The challenges to be faced include how to deal with the requirements from industry, just getting a single score characterising a system, and academia, getting as much measures as possible to accurately characterise the system, and how to navigate from one representation to another without losing meaningful information.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Companies trying to success in the business world can be compared to a group of lions fighting to get the tastiest part of their prey. Companies always try to make the difference with their competitors developing the best products in the shortest period of time possible. But when we talk about products that require many different kind of components, deciding what components should be used and which not, is something that in most of situations can not be done simply by looking at the component's specifications. For example, when choosing between two processors that operate at different clock frequency, a higher clock frequency does not necessarily translate into a higher computational power. Therefore, tests have been developed in order to assess the relative performance of an object, so it can be compared to others. These processes are known as **benchmarking procress**.

Benchmarks can be designed for many areas of the industry, but when setting the target in computer systems, performing a benchmark represents the act of running a set of programs or operations to assess the performance characteristics of computer hardware or software. Currently there is a huge number of different computer benchmarks designed to assess many kind of systems or components, and many companies have developed well known benchmarks that are considered as standards in different fields. For example, the Embedded Microprocessor Benchmark Consortium (EEMBC) [1] develops benchmarks to assess the performance of different kind of processors, and the Transaction Processing Performance Council (TPC) [2] defines transaction processing and database benchmarks.

The execution of these benchmarks results in a single metric that allows to compare those systems running the same benchmark.

The use of benchmarks is very common and extended in the industry. Many vendors provide with their products the score they obtained performing a certain benchmark. Thus, performing benchmarking aids companies in the process of design and development of their products, by assisting them in to decide which components suit best for their purpose, so reducing a product's time to market. But when developing products that will be used by customers, companies must assure not only the performance characteristics of their products, but also the dependable ones. As products may be sensitive to perturbations that affect their behaviour, their assessment under faulty conditions let designers to know how this perturbations affect the system behaviour, and thus improve the design to prevent these situations. These kind of benchmarks are known as dependability benchmarking, and its application in real systems is quite young compared to performance benchmarking.

The single metric provided by a benchmark is a result of performing some aggregation methodology to the measures obtained by the benchmark. Even benchmarks are highly used, there are some problems related to the analysis of these measures, no matter the kind of benchmark performed (performance or dependability). Many works can be found on the literature that make use of well known or custom benchmarks to perform the assessment of systems or components in many different areas ([3], [4], [5],etc). However, there is a common problem in many of these studies when it comes to presenting their conclusions caused by a lack of standards in the process of analysis of results. Benchmarking performers apply their own criteria to the obtained results, and these criteria are not always explicit when presenting the conclusions of the work done, which makes very difficult to reproduce the same analysis done in the work. It can be said, that the study does not have the **conclusions reproducibility** feature.

The number of measures provided by a benchmark directly affects the reproducibility of the conclusions that can be extracted by performing a benchmark. When a benchmark provides a large set of measures, interpreting all the information that can be extracted from them and place it in a conclusion is a really hard task to do. In the other side, if few measures are provided, there is few information to interpret, so it does not present

a challenge, however, those few measures can present problems of representativeness of the whole system. While people from the industry prefer a single metric to help them to choose among different systems or components, people from the research community like having as much measures as possible to perform analysis the measures using different perspectives or contexts. Having this two extremes, it seems logic to think that there will be intermediate users that do not need to analyse all the raw measures obtained by the benchmark, but would prefer a set of measures that represent more general features of the system to assist them to choose between systems rather than a single metric to compare them.

The problem of having a large number of measures is more common in dependability benchmarks, where dependability features are measured in addition to the performance ones. Current methodologies used in benchmarking do not provide a multilevel analysis that would benefit many kind of users, ranging from those that require as many measures as possible, to those that prefer a single global measure to rank the system. To cover this gap, there are different methodologies that allow evaluators to perform a multilevel analysis of the results, providing conclusions at different levels for different purposes. Using this methodologies may suppose a big improvement in the benchmarking process, as benchmark performers would be able to assess systems considering aspects that would require from a certain level of expertise, but not being an expert.

With the aim of providing solutions to these problems that come with the reproducibility of the analysis of results, and avoid the loss of information at the same time, in this work we have tackled different scientific and technical challenges:

Scientific:

- Analysis and adaptation of decision-making strategies to the context of dependability benchmarking.
- Definition of a new process of multi-criteria analysis (MCA) to make more objective and reproducible the analysis step underlying the interpretation of any dependability benchmarking results.

Technical:

- Integration in an existing Resilience Evaluation Framework for Ad Hoc Networks of a web-based a TOol for Multi-Criteria Analysis of Measures named w-TOMCAM.

This document is structured to guide the reader through the steps that we followed from the detection of a problem in the dependability benchmarking process, to the definition and application of a proposal to cover the problems found. Chapter 2 analyse the different parts that compose the dependability benchmarking process, and introduce the problems that appear in relation of the analysis of results, providing an overview of the current approaches used in the literature to cope with it. Different approaches that can be used to cover the gaps found in the interpretation of measures, and that can provide dependability benchmarking of well structured and well defined standards, are described in Chapter 3. The process to integrate a version of a multi-criteria analysis technique for performing the interpretation of measures is defined in Chapter 4, and a case study is used to demonstrate its feasibility. The results of a collaboration with the work done during one of my advisor's PhD, where we applied this technique to a real framework designed to perform dependability benchmarking, are detailed in Chapter 5. Finally, the conclusions and future work are presented in Chapter 6, and the publications done with this work are presented.

Chapter 2

Background

The intrusion of a wide variety of computer components into the industry products during last decades, led to a situation where this products or their internal components where subject of evaluation processes to analyse their behaviour in presence of faults. Beyond this evaluation techniques, which are used only for evaluation purposes, the goal of dependability benchmarking is to provide generic ways to characterizing the behaviour of components and computer systems in the presence of faults, allowing for the quantification of dependability measures. The main difference between a benchmark and an evaluation and validation technique is that the benchmark represent an agreement that is widely accepted both by the computer industry and/or by the user community. The generalised idea that the main goal of benchmarks is to compare system on the basis of their results is probably caused by the success of well-established performance benchmarks. However, in this chapter we describe the different aspects of the dependability benchmarking to show that it can be used in a variety of cases. Also, we tackle the weakness present in the process of analysis of results, and some examples are used to support our statements.

2.1 Dependability Benchmarking

A dependability benchmark could be defined as "*a specification of a procedure to assess measures related to the behaviour of a computer system or computer component in the presence of faults*". The main objective of these benchmarks is to provide practical ways to characterize the dependability of computers. The characterization of the dependable attributes of a computer system and/or component allows their comparison according to this given attributes. However, dependability benchmarks can be used for other purposes as revealing weak points in a prototype during its early design phases, or monitoring improvements achieved by fault removal activities during development.

In the Dependability Benchmarking Project (DBench) [6] many researches collaborated to provide the guidelines for defining dependability benchmarks for computer systems. The parts that structure the dependability benchmarking process and the related properties that a dependability benchmark must have are described in the DBench literature.

The process of benchmarking a computer system and/or component in presence of faults to quantify their dependability features requires to follow a three step process. These steps are depicted in Figure 2.1. Many aspects must be considered in each of the steps to perform a rigorous and unambiguous dependability benchmark. These steps are described hereafter paying attention to the different aspects that they encompass.

• Experiment definition

In the first step of the process, it is important that benchmarking performers clearly identify and specify the benchmark target (BT), and the setup necessary to host and run it. This is known as System Under Benchmarking (SUB), and depending on the type of SUB where the benchmark is executed, the specification of other aspects like the workload, faultload and the measurements will vary.

The Benchmark Target. Represents the computer system and/or component which behaviour in presence of faults will be assessed.

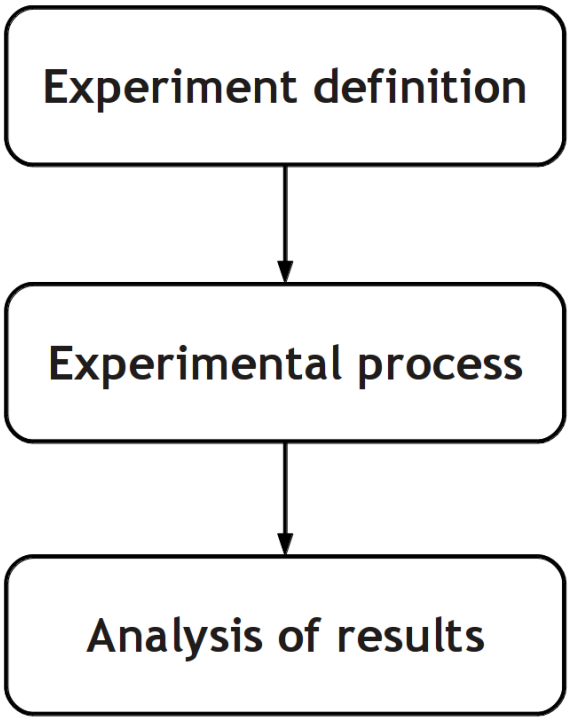

Figure 2.1: Steps of a dependability benchmark process

System Under Benchmarking. Is the system that hosts the BT. If the BT is a software component, the SUB would be represented by the hardware platform where the software is running, and the software resources used by the component. The type of SUB will influence the definition of the other aspects.

Workload. Represents a typical operational profile representative for the application domain defined for the BT. Representative workloads for different application domains are already provided by widely accepted performance benchmarks. However, synthetic workloads can be developed as long as they remain as representative as possible for the benchmark to be useful.

Faultload. Consists of a set of perturbations (faults or attacks) that are intended to emulate the real threats the system would experience. As it happens with the workload, the set of faults must be representative of those that may occur in a real application

domain.

Measurements. These are the measurements performed on the BT that allow the observation of its reactions to the applied execution profile (determined by the workload and the faultload). The measures of interest for the benchmark users are extracted from processing these measurements. To perform a proper diagnose from the execution in presence of faults, some performance-related measurements must be compared to those obtained in a normal execution (in absence of faults).

Other characteristics as the number of experiments that will be performed and the duration of these experiments must be defined in this step. However, they are not relevant for the purpose of this section.

• Experimental process

All the set up defined in the previous step is used to perform the execution of the experiment. In this step the execution profile is applied to the SUB in order to obtain the defined measurements about the outcome of the BT's behaviour. The interaction of the performer in this step is none, as any interference during the execution of the experiment would compromise the experimentation and thus, provide useless results. The measurements from the experimentation are used in the next step.

• Analysis of results

This step is an essential part of the process since the conclusions that evaluators may obtain from the benchmark rely in this part. When the execution of the experiment is over, all the measurements from the monitoring of the SUB are processed to extract relevant dependability related measures. Different kind of statistics can be applied to the measures to provide a single metric to classify the system, which method to use depends on the criteria of the benchmark performer. But if the purpose of the experiment is not only the comparison of BT, then the measures are interpreted to extract meaningful conclusions.

Measurements processing (from measurements to measures). At the end of the experimentation all the data is in its more basic format and usually representing different aspects of the system. For example, if the measure to obtain is throughput, in first place we have a big number of measurements about the monitoring of the network interface. This measures needs to be correlated and processed to obtain the final measures.

Interpretation of measures. This is a necessary step for benchmarking that is not typically addressed in dependability benchmarking. The number of measures obtained depends on the aspects of the benchmark applied, as the BT, the SUB and the execution profile, as different aspects are measured in different application domains. Not considering the interpretation of results in a benchmark can be acceptable if there are few measures and their interpretation is straightforward. Nevertheless, in dependability benchmarking usually a wide variety of measures are obtained (e.g., performance and dependability), thus, performers do their best using analysis methodologies to provide end users with quantitative or qualitative (depending on the performer) conclusions.

2.2 Benchmark properties

A dependability benchmark must fulfil a set of properties in order to be validated and accepted by the dependability community. These properties, defined in [6] are shown in Table 2.1.

All properties are important for a dependability benchmark, some are more relevant than others, as the questioning of those can imply the non-validation of the benchmark. Representativeness is one of them, as when benchmarking real systems or components, the users expects the results to be representative with those that would obtain in a real situation. If it is not the case, then users will not consider that benchmark as valid.

Reproducibility is another property, which necessarily must be fulfilled to consider a dependability benchmark as valid. The achievement of this property strongly depends on the amount of details given in the *experiment definition*. This is not an easy task, as the details must be general enough so it can be applied to the type of systems addressed by the benchmark, but at the same time, concrete enough to not distort the original speci-

Table 2.1: Required properties for dependability benchmarking validation

Representativeness	Concerns all the aspects of the benchmarking process as *measures*, *workload* and *faultload*. During the benchmark, all three aspects must be defined to represent the application domain as accurately as possible
Repeatability	Guarantees statistically equivalent results when the benchmark is run more than once in the same environment (i.e., using the same SUB, the same execution profile workload and faultload and the same prototype)
Reproducibility	Guarantees that another party obtains statistically equivalent results when the benchmark is implemented from the same specifications and is used to benchmark the same SUB
Portability	Refers to the applicability of a benchmark specification to various target systems within a particular application area. A portable benchmark can be implemented and run on various target systems within the application area
Non-intrusiveness	If the implementation of the benchmark (particularly to what concerns the workload and faultload) introduces changes on the system under benchmarking (either at the structure level or at the behaviour level) it means that the benchmark is intrusive. The benchmark must require minimum changes in the system under benchmarking
Scalability	A benchmark must be able to evaluate systems of different sizes. Usually, benchmark scaling is achieved by defining a set of scaling rules in the benchmark specification. Scaling rules mostly affect the workload, but other components such as the faultload may also have to be scaled
Time and Cost	The time and cost required to perform a benchmark will depend on the system. However, a trade-off needs to be achieved to perform benchmarks with acceptable times, and which perceived value is higher than the associated costs.

fications during the benchmark implementation. The fact that a benchmark is primarily meant to be used on an open basis, it is required a full understanding and interpretation of the benchmark results. The same way the benchmark results must be statistically similar when it is performed by another party, it is logical to think that from the same measures, the analysis performed by different performers should lead to the same conclusions. In this work we have coined this property as **conclusions reproducibility**. If different

performers using the same benchmark for the same application domain do not achieve the same conclusions from the same results, then that benchmark could not be considered as a representative benchmark for that domain, and thus it would be dismissed.

However, from a deep analysis in the literature, it can be seen that benchmark performers apply different methodologies from one to the other, and also, each of them apply their own criteria to interpret the measures. This situation would not represent a problem if those criteria would be explicit on their studies, allowing other performers to reproduce the reasoning process after the conclusions. Unfortunately, this is not always the case, as it happens in works like [5], where the criterion used is not explicit in the paper, and the conclusions have to be analysed to know the process followed in order to reproduce them.

Next section analyses the problem present in many works from the literature during the process of measures interpretation. The causes of the problem, and its implications in the research domain are tackled to later provide good reasoning on how this problem can be solved.

2.3 Issues in the interpretation of measures

The dependability community has made big efforts during the last decade to define and develop standards to provide benchmarks with well defined methodologies with the aim of performing widely accepted dependability benchmarks. Nevertheless, when it comes to the interpretation of measures, no standards have been defined so far. This lack of standards has led to a situation where researches interpret the measures obtained using different techniques and custom criteria. The use of custom criteria is totally acceptable, but the techniques do not provide a mechanism to make explicit these criteria. Therefore, other researchers willing to analyse the conclusions extracted from other works, sometimes find it difficult to understand the reasoning followed by the authors, unless it is precisely detailed with natural language.

The main problems that are found in the literature are due to the use of these techniques, or given the lack of them:

- Simplistic interpretation of the results that are not representative of a real context.
- The achievement of different conclusions from the same results caused by a missing well defined criterion.

With the aim of easing the understanding of the aforementioned issues, and in order to illustrate the related problems, let us use an example based on the results obtained in [4], where the authors perform a dependability benchmark to compare two well-known web servers (Apache and Abyss), running on top of three different operating systems (Windows XP, Windows 2000 and Windows 2003) through the SPECWeb99 benchmark [7]. Thus, authors aim at selecting the best combination of the pair {web server, operation system}. Despite target systems are subjected to 12 different faults encompassing both software and hardware faults, authors finally present only two types of results: those regarding the execution of the system in absence of faults (baseline) and those from the execution in presence of faults. Table 2.2 shows the results extracted from the paper.

Table 2.2: Measures characterising the behaviour of the pair {web server, operating system} in presence of faults [4].

System	**AUT** *(%)*	**AVL** *(%)*	**SPECf** *(# con)*	**THRf** *(# op/s)*	**RTMf** *(ms)*	**ACR** *(%)*
Apache-2000	93.98	95.28	13.82	79.24	382.2	97.21
Apache-XP	95.48	97.94	18.07	71.63	359.7	97.60
Apache-2003	96.77	97.62	11.27	79.21	373.1	97.29
Abyss-2000	94.36	96.35	10.32	75.96	363.7	94.78
Abyss-XP	95.97	97.31	13.71	68.22	362.0	94.50
Abyss-2003	96.25	97.53	12.91	66.18	358.7	95.55

The results of the benchmark are analysed using 6 measures (3 from performance and 3 from dependability). The set of performance measures is composed of the number of simultaneous connections (con) correctly established (SPECf); the number of operations (op) per second (THRf); and the average time in milliseconds (ms) that the operations requested by the client take to complete (RTMf). With respect to dependability, authors consider autonomy, as a percentage of administrative interventions with respect to the

number of faults injected (AUT); accuracy, as a percentage of requests with error with respect to the total amount of requests (ACR); and the percentage of time the system is available to execute the workload from the total (AVL).

In the paper the authors state that benchmark performers would consider different weights for the measures depending on the environment where the web-server would be deployed. Meaning that a performer could consider availability more important, while other could consider performance more relevant for its purpose. Therefore, we can deduce that contextualizing the measures of the experiment is important to provide meaningful conclusions. The criteria used by the performed strongly depends on the application context of the SUB. For example, if analysing the results looking for the pair {web-server, operating system} with the best performance, we conclude that the best system is the combination Apache-XP. But if we analyse them looking for the most dependable system, the conclusions would be that the best system is Apache-2003. So, if these criteria are not well detailed during the analysis, when two performers working on different application contexts interpret the measures, they will obtain different conclusions, thus causing rejection against the benchmark. But not making explicit the criteria is more common than expected, as some researchers consider that basic concepts do not need to be detailed.

Nevertheless, the authors interpret the measures assigning equal relevance to all six measures, so basically, an average value of the measures is obtained to perform the analysis. The main issue is that this kind of evaluations where all measures have the same relevance are too simplistic and do not represent a real deployment. However, more complex interpretations where the measures are contextualized into a certain application context are rarely common, very few studies can be found where this is considered [8]. To our concern, the main cause for this are the analysis methodologies used for the aggregation of measures, which do not provide of mechanisms to perform more complex interpretations.

A review of this methodologies is done in next chapter to point out their limitations, which are reflected on the process of analysis done by performers when applying them to interpret the measures.

Chapter 3

Towards the standardization of the interpretation of measures

Current widely accepted performance benchmarks are designed by committees of companies or organizations that defined reputed standards for computer benchmarks. Some steps have been done in the definition of standards for dependability benchmarking in some areas. For example, the ISO/IEC 25045 [9] defined to consider accidental faults to address the recoverability capabilities of a software system integrated to the ISO/IEC 25000 "SQuaRE" standard series [10]. Standardize the process of benchmarking is a necessary step towards expanding its use in different domains. In this chapter an analysis of the different techniques that are usually used by performers when interpreting the results is done. Given the weakeness that are found in these techniques, a two multi-criteria analysis techniques are chosen among others to show how its use could be defined as a standard for the interpretation of measures in dependability benchmarking.

3.1 Measures aggregation approaches

The number of measures obtained benchmarking a computer system and/or component is usually related with the difficulties found by performers to present the results to end users. For that reason, many benchmarks provide a single score for the system, for example, when observing the set of benchmarks provided by the EEMBC [1] or by the TPC-C [2], many of them provide a single global measure for a system by calculating a geometric mean with all the given measures. But, when providing a set of measures, it should be taken into account that there are different evaluator profiles that may need to consume these measures. For example, while people from the academia may want as many measures as possible, people from the industry in the other hand could prefer a single global measure to perform systems comparison. Thus, following this reasoning, we can conclude that there are intermediate users requiring more than a single measure, but not tens of them.

There are different approaches to represent and analyse the multiple measures obtained from evaluation. Although each approach has its own particularities, all of them have to face a common problem: *how to characterise the decision criteria within a friendly and usable model*. The choice for a representation of measures has important consequences in terms of expressiveness. Simplistic approaches may skew in excess the representation of the model, whereas representations with a high expressiveness can add unnecessary complexity to the model or can be cumbersome in its use for decision making. Therefore it is important to find an equilibrium between the possibility of representing as much situations as possible but at the same time maintaining a good degree of usability.

Measures aggregation is a common approach trying to enable meaningful comparisons among systems that eases the analysis of benchmarked systems or components. However, although these techniques are usually applied in the community of dependability 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. Common methods applied by users for aggregation range from simple mathematical operations (e.g., addition or mean average) to more serious and systematic distribution fitting [11] and custom formulae [12] approaches.

Kiviat or radar diagrams [13] are graphical tools which represent the results of the

benchmark in an easy-to-interpret footprint (Figure 3.1). 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 [13] 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 numerical value associated to the footprint shape. However, the problem of 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 systems could result quite vague [14].

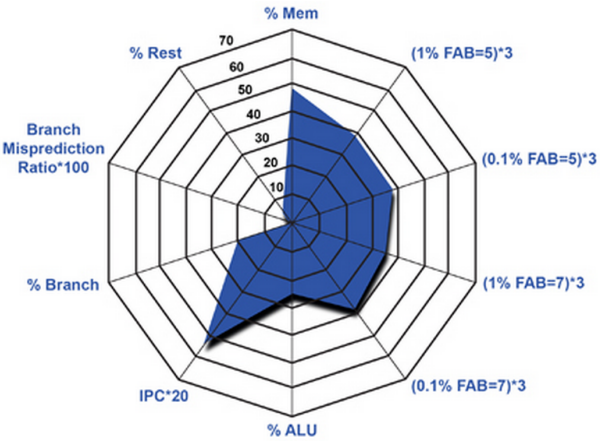

Figure 3.1: Kiviat graph representing multi variable data for the BaseFP benchmark within AutoBench 1.1

Other approaches, like the presented in [11], characterise the level of goodness of the measures according to their ability to fit with a particular statistical distribution. Nevertheless, this approach presents three main drawbacks. First, it assumes that a measure follows the same distribution for all the systems, which may be false depending on the context of use. Second, to understand this type of characterisation, it is necessary to understand the assumed statistical model, which is not straightforward. Third, the subjectivity of the probability distributions will strongly affect the sensitivity analysis. Finally, it is necessary to handle those situations when there is not enough information

to build probability distributions for evaluation data.

Alternatively, other authors, like Al-Sbou [12], propose the use of custom formulas for the aggregation of measures obtaining a single score which characterises the behaviour of the system. However, these types of formula definition is based on heuristics and lacks formal foundation and validation.

Finally, Correia et al. [15] apply the notion of thresholds to map measures into a particular scale for software systems certification. Yet, they assume all the measures have the same importance when it is not always the case.

In sum, previous methodologies lack the ability of aggregating measures into a meaningful way. Generally, these techniques focus on aggregation of results and do not provide any insights on how to cope with the interpretation of the resulting aggregated scores. Accordingly, open questions requiring further research in the domain of dependability benchmarking are (i) how to systematically aggregate such measures to capture in a single or small set of scores the information required to characterise the overall system quality, and (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 benchmark measures.

3.2 What is needed?

Previous methodologies present problems to provide performers with the necessary tools to make explicit their criteria during the experimentation, and in some situations, problems to perform complex analysis of the measures. All this issues led us to the situation where the use of new techniques to cope with this floods is required. So, based on the gaps that we need to cover, what in our opinion is expected from an aggregation methodology to analyse the measures is the following:

- Sensibility: Any change in the model should be proportionally reflected in the result of the aggregation.
- Consistency: The semantic of measures should be preserved in the model in such a way they are coherent with the result of the aggregation.

- Accuracy: The model should capture the complexity of relationships established among the characteristics of real systems.
- Usability: The model should be easy to explain and interpret.
- Navigation: The model should enable the hierarchical analysis of measures from coarse to fine-grained measures (and vice-versa).

As will be seen in next section, all of this points can be achieved by using a MCA technique. These techniques make use of quality models to extrapolate the performers interpretation criteria into decision trees. This quality models allow the simultaneous interpretation of different kind the measures in a hierarchical way, thus easing the navigation from the low level measures to the global score.

3.3 Multi-criteria Analysis of measures

In the literature can be found many different multi-criteria analysis techniques, with the aim of not expanding on the definition of methodologies that will not be used in this work, in [16] most of this techniques are explained in detail. For now on, we focus on two of these techniques that fulfil all the desirable points presented in previous section, that an aggregation methodology must have. Their feasibility will be shown through a simple example.

For both techniques, the results used are extracted from the study done in [3], where the authors evaluate the behaviour of an ad hoc network in presence of perturbations. For this example, we selected a subset of the measures presented by the authors, first because of space constraints, and second, because we consider this reduced number of measures is enough to show the feasibility of the techniques.

The results in Table 3.1 represent the measures obtained from the ad hoc network while performing one of the attacks (*Replay attack*, *Flooding attack* and *Tampering attack*). The goal of the analysis is to determine under which attack the network behaves better. The selected measures for the example are described next:

Table 3.1: Measures obtained from the study done in [3]

Measure	Replay attack	Flooding attack	Tampering attack
Availability (%)	75.20	65.00	90.33
Integrity (%)	99.44	98.23	62.90
Throughput (Kbps)	70.90	80.18	96.45

Availability

Average percentage of time the communication route established between sender and receiver is ready to be used.

Integrity

Average percentage of the number of packets whose content has not been unexpectedly modified.

Throughput

Average throughput of the network shown in kilobits per second.

3.3.1 Analytic Hierarchy Process

One technique is the Analytic Hierarchy Process (AHP) [17]. This technique is widely used in other contexts of use for decision making processes. This technique has been highly used in many domains [18] [19] to perform the analysis of results and establish a comparison between systems. In this technique, the measures are aggregated through a decision tree, where the leafs represent the different systems under benchmark and the root is a global score for the system, Figure 3.2.

The fundamental input to the AHP is the decision maker's answers to a series of questions of the general form, "*How important is criterion A relative to criterion B?*". These are termed pairwise comparisons. For each pair of criteria, the decision-maker is then required to respond to a pairwise comparison question asking the relative importance of the two. Responses are gathered in verbal form and subsequently codified on a nine-point intensity scale, as follows:

Table 3.2: Importance scale index table

How important is A relative to B?	Preference index
Equally important	1
Moderately more important	3
Strongly more important	5
Very strongly more important	7
Overwhelmingly more important	9

2, 4, 6 and 8 are intermediate values that can be used to represent shades of judgement between the five basic assessments.

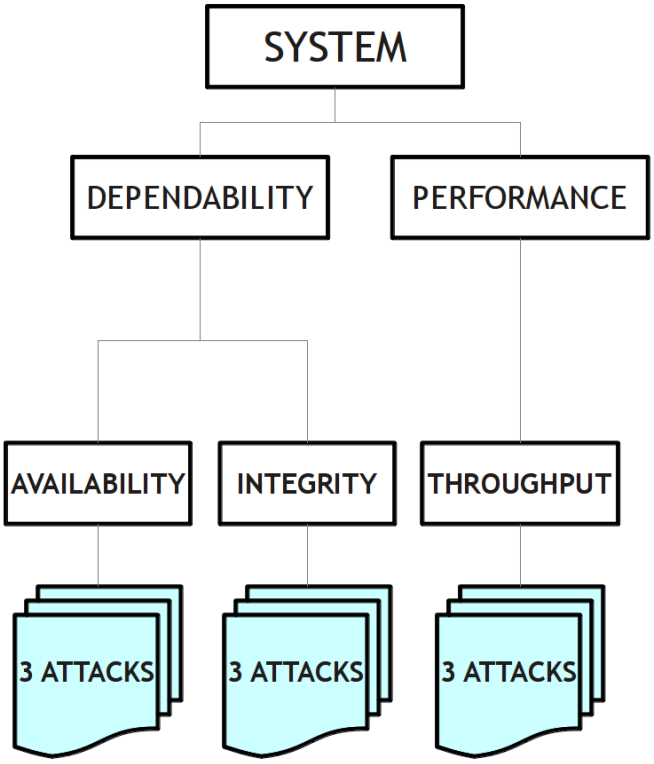

Figure 3.2: Representation of the decision tree for the analysis of the measures in Table 3.1using AHP

The comparisons are summarized in a matrix form, and the weights (or priorities) are obtained performing the geometric means of the value for each criterion, and then

normalizing the results. The weights from all the criteria that have an upper level criterion, have to sum the weight of the parent criterion. For example, if Dependability has a weight of 0.6, the weights of Availability and Integrity will sum 0.6. For this example, we consider Integrity strongly more important than Availability for the Dependability criterion, and Dependability slightly more important than performance. Table 3.3 and 3.4 shows the matrix of the pair comparison for Dependability (D) and Performance (P), and Availability (A) and Integrity (I) respectively

	D	P
D	1	2
P	1/2	1

Table 3.3: Dependability vs Performance

	A	I
A	1	1/5
I	5	1

Table 3.4: Availability vs Integrity

After calculating the weights for the higher level criteria, the weights for the lower level ones are calculated. Table 3.5 defines the weights applied to each criterion in the decision tree.

Table 3.5: Resultant weights after calculations

Criterion	**local priority**	**global priority**
System	1	
Dependability	0.667	
Performance	0.333	
Availability	0.167	$0.667x0.167 = 0.111$
Integrity	0.833	$0.667x0.833 = 0.556$
Throughput	1	$1x0.333 = 0.333$

When the priorities are established for all the criteria, a pair comparison of the results obtained for each attack shown in Table 3.1 is defined for the three measures. Being R replay attack, F flooding attack and T tampering attack, the pair comparison for each measure can be seen in Tables 3.6, 3.7 and 3.8.

After computing the priority of each attack for each measure given, all priorities are accumulated to determine under which of the three attacks the system has a better performance. The final priorities let us establish a ranking for the different attacks based

	R	F	T
R	1	2	1/2
F	1/2	1	1/3
T	2	3	1

Table 3.6: Availability

	R	F	T
R	1	1.5	5
F	1/1.5	1	5
T	1/5	1/5	1

Table 3.7: Integrity

	R	F	T
R	1	1/2	1/3
F	2	1	1/2
T	3	2	1

Table 3.8: Throughput

on their impact on the system. From lower impact to higher, the ranking (and their priorities) for the criterion used stands like this:

1. Replay attack (0.37) 2. Flooding attack (0.34) 3. Tampering attack (0.29)

3.3.2 Logic Score of Preferences

The other technique is the Logic Score of Preferences (LSP) [20]. This technique has been used in many different works to perform the analysis of multiple criteria measures in different fields of research, [21] [22]. LSP provides the necessary mechanisms for combining a large number of criteria into one score. In order to achieve this, as it happens in the AHP, an aggregation tree has to be built to define the quality model that represents the criteria applied by the user. In this tree, the leafs represent the raw measures, and the intermediate levels represent aggregation criteria for their children. As could be seen in the previous example, when using AHP, all systems are compared based on each low level criterion when the results are available. LSP process is different. The decision tree is defined in advance, before the experiment is executed, and it is applied individually to each of the SUB. With the aim of representing a similar reasoning as done with the AHP (Integrity more relevant than Availability, and Dependability slightly more relevant than Performance), Figure 3.3 depicts the quality model defined to perform LSP for the results shown in Table 3.1.

When the quality model is defined, the weights for each criterion is set. As the values of the measures obtained for the experiment are aggregated, and thus mathematical operations are performed with them, the values must be normalized in advance, as for example, it is incorrect to sum percentage with kbps. For this reason, a range of acceptable values for each measure is defined in advance. In Figure 3.3 it can be appreciated that

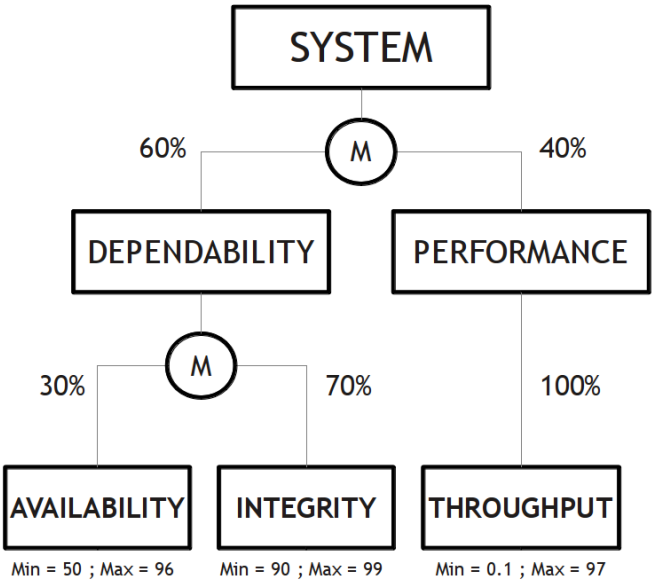

Figure 3.3: Representation of the decision tree for the analysis of the measures in Table 3.1 using LSP

for the Availability measure (given in percentage format), the minimum and maximum thresholds defined are 50 and 96, respectively. The normalization function applied will vary between two functions, depending if the measure represents a "higher the better" value, or the opposite. The measures are aggregated into higher-level features using operators and weights that determine each low-level measure's contribution to the higher-level one. In the current example, a mean operator is used in all the aggregations. The final result is a global score that can be used to compare the evaluated system.

Table 3.9: Normalized values using the thresholds defined for each measure

Measure	**Replay attack**	**Flooding attack**	**Tampering attack**
Availability (%)	54.78	32.61	87.67
Integrity (%)	100	91.44	0
Throughput (Kbps)	73.90	82.64	99.43

Table 3.9 show the value of the measures for each attack after normalizing their values. With the values already normalized, we perform the aggregation operations for

the values obtained for each attack. The result is that we obtain three different scores for the system, one for each attack, representing the quality of its behaviour under a certain attack. The operations are not shown given their simplicity, as the operation that need to be performed for the aggregation is a weighted mean. From the final score, we can determine a ranking based on the impact that a certain attack has over the system (and the SUB's score for that attack):

1. Replay attack (81.09) 2. Flooding attack (77.33) 3. Tampering attack (55.55)

3.4 Discussion

After performing the same analysis with both techniques, AHP and LSP, we can state that from the visualization of the process, the criterion used by the user to analyse the measures can be extracted. Also, the use of the same criterion in both situations has result in the same classification of the attacks according to their impact level on the system.

The use of this techniques to perform the interpretation of measures in a dependability benchmark would solve the current issues that were introduced in the previous chapter. Therefore, its integration into a dependability benchmark would absolutely improve its approval in the researchers community.

In this work, we have set our goals in the integration of one of these techniques into a dependability benchmark. With this objective in mind, we have decided to use LSP instead of AHP because to our concern, its use for unexperienced users can be easier. However, LSP can use a wide number of different operators, and aggregation combinations to define more precise and accurate representations of the evaluator's criteria, and the correct use of these tools requires from a higher level of expertise.

With the aim of simplifying its usability, in this work a deep analysis on the different operators and their combination for the LSP (defined in [23]) have been done. From this analysis we have been able to provide an *adapted* version of the LSP (aLSP) that is more easy to use.

Next chapter describes how this technique must integrated into a dependability bench-

mark and how it affects its benchmarking process. Also, a detailed explanation of the different parts of the aLSP is provided, and its feasibility is proved using the same study that has been used for AHP and LSP [3], but this time, with the whole set of measures and SUB provided by the authors.

Chapter 4

Adapted Logic Score of Preferences in dependability benchmarking

Up to this point of the document, the lacks present in the interpretation of measures in dependability benchmarking has been pointed out. An approach that has not been considered in many works in dependability benchmarked has been introduced and its applicability to cover this lacks has been demonstrated. But all have been done only considering the part of the interpretation of measures from the whole benchmarking process. In this chapter, details about how to integrate the Logic Score of Preferences technique in the process of benchmarking are given with the aim of providing with guidelines those researchers interested on using this technique in their benchmarks. With the same purpose, the adapted version of the LSP (aLSP) that simplifies the use of this technique is described in detail, and a study done in dependability benchmarking is used as a case study to make clear its usability in situations with a high number of measures and systems under benchmarking.

4.1 Integration in the benchmarking process

Using the aLSP technique in dependability benchmarking requires to modify the benchmarking process in order to add new functionalities in some of its steps. First of all, and just to make it clear, the way aLSP is integrated in the benchmarking process is the same that for the original LSP, as the modifications introduced to design the aLSP do not affect its overall behaviour, but only affect the aggregation process.

There are mainly two new functionalities that must be added to the benchmarking process. The first one is the "definition of the quality model" and the other the performance of the "aLSP analysis". The parts of the process where these functionalities must be added, are depicted in Figure 4.1.

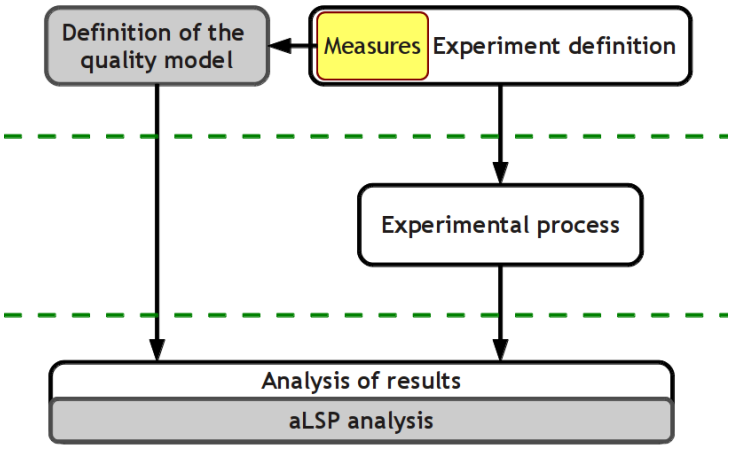

Figure 4.1: Integration of the aLSP functionalities into a benchmarking process

This technique requires the benchmark performer to define a decision tree representing the analysis criterion to perform the analysis of the measures. Although this decision tree could be defined after the benchmarking process is over, establishing the analysis criterion that will be used before the measures are available, provide the benchmark with a more reliable interpretation of measures. If performers determine their criterion after observing the measures, they could add subjectivity to the decision tree they define, trying to adapt it to make the results look as they expected before the execution.

The other functionality is, of course, the implementation of the aLSP technique. The

process of analysis of the aLSP will be executed when the measures defined in the quality model are available, this is in the part of the analysis of results of the benchmarking process.

The details of the internal functioning of the aLSP and all the features that must be taken into consideration as described in next section.

4.2 aLSP: A multi-criteria analysis technique to interpret evaluation results

The proposed multi-criteria analysis methodology does not intend to automate the task of benchmarking performers when selecting a proper system; it rather tries to support and guide the comparison of the systems or components fulfilling the system requirements for a particular application, and the selection of the most suitable one.

What makes it interesting for dependability benchmarking is its capability to systematise the way to compute the global score of a component, not only considering the measures themselves, but also formalising their interpretation attending to aspects such as the relationship among the measures, and their relative importance within a particular context of use. Accordingly, it is easy to obtain a hierarchical quality model which assists the navigation from the fine-grained measures to the coarse-grained scores, without losing the numerical perspective of results. In such a way, one can keep the consistency in the interpretation and analysis of results independently from the viewpoint (fine or coarse) acquired by the benchmark user.

4.2.1 Benchmark user and target system

The first step is to identify the benchmark targets (in case of more than one alternative), the application context where they operate in and their goal, that obviously depend on the evaluation performer. These aspects are crucial to (i) determine the requirements of the system; and (ii) fix their level of accomplishment.

System requirements can be expressed through the notion of quality model, previously introduced in standards such as [10]. A quality model is a framework to ensure that all the

information required by the stakeholder to perform the proper decision-making is taken into account to carry out the analysis of benchmark measures. With respect to this point, the rest of this methodology will introduce the instruments (thresholds, relationships, weights) required to enrich the meaning of measures within the benchmarking process.

4.2.2 Measures selection

The selection of measures is the first step to characterise the quality of the system. This stage is common to any traditional benchmarking process, and consists in choosing a set of measurable attributes (noted m_1 to m_n) that are representative of the system quality or simply of interest for the evaluation performer. In the absence of criteria for their selection, it would be convenient that such measures are non-redundant, independent and thoroughly selected attending to their capability to represent quantitative elemental aspects of the system such as delay, throughput, data availability, etc. This involves that no measure should be derived from other. According to this remark, if we are already taking into account the system throughput in presence of faults as a measure, considering any other throughput-based measure, such as a ratio between the throughputs in presence and absence of faults, would be unfairly providing more importance to throughput than the rest of measures.

4.2.3 Scales of measures

Given the heterogeneity of the measures considered in resilience benchmarking, it is easy to find different measures using distinct scales and dimensions, e.g., seconds or milliseconds if measuring time, joules if measuring energy, and so on. Obviously, this is a fact hindering analysis and comparison of measures for non-skilled users. To cope with this problem we propose the definition of quality criterion functions $c_i(m_i)$ which specify how to quantitatively evaluate each measure, i.e., they establish an equivalence between the measured value and the system quality requirements within a 0-to-100 quality scale. The result of each criterion function, known as elementary score (or elementary preference), corresponds to s_i. Formally, such elementary preferences s_i can be interpreted as the degree of satisfaction of a measure m_i with respect to the quality requirements specified by

the benchmark performer for such measure. Since all the measures are scored according to the same normalised scale, resulting elementary preferences are directly comparable. Such equivalence can be mapped to discrete or continuous functions. Equations (4.1) and (4.2) show an example of lineal increasing and decreasing functions when measures are the higher the better and the lower the better, respectively.

$$s_i = c_i(m_i) = \begin{cases} 0, & m_i \leq T_{min_i} \\ 100\frac{m_i - T_{min_i}}{T_{max_i} - T_{min_i}}, & T_{min_i} < m_i < T_{max_i} \\ 100, & m_i \geq T_{max_i} \end{cases} \tag{4.1}$$

$$s_i = c_i(m_i) = \begin{cases} 100, & m_i \leq T_{min_i} \\ 100\frac{T_{max_i} - m_i}{T_{max_i} - T_{min_i}}, & T_{min_i} < m_i < T_{max_i} \\ 0, & m_i \geq T_{max_i} \end{cases} \tag{4.2}$$

The use of minimum and maximum thresholds (T_{min_i} and T_{max_i} respectively) within criterion functions is necessary to position and compare the value of measures with respect to reference values of the applicative domain, thus easing their interpretation. For example, the interpretation of the measured throughput in a communication system (let us assume 8 Kbps) will be better if the measure is obtained from a Wireless Sensor Network in charge of monitoring temperature (where the optimum value may round 10 Kbps) rather than if it is obtained from a Wireless Mesh Network to provide Internet access (where even the minimum value allowed for a quality communication, let us assume 500 Kbps, is greater than the value obtained). The definition of thresholds gives meaning to the values obtained for each measure. Consequently providing the minimum and maximum values that can receive each measure will be very important to determine their preference. Thresholds can be obtained through previous experimentation, the opinion of experts in the domain, or certification and widely-used references.

Once measures have been scored, evaluation performers have a founded intuition about the system behaviour. In fact, they are able to determine if the individual goal for each particular measure has been accomplished or not. For example, obtaining a

score of 75% in one measure could be interpreted as a positive feedback. However, their global preferences about the system requirements are not mapped yet in the result of the evaluation. The idea of the following stage is to aggregate the characteristics of the system according to the evaluation performer's requirements and preferences.

4.2.4 Preferences aggregation

To address the aggregation of scores, this stage of our methodology structures a quality model through a hierarchy of high-level objectives, sub-objectives, etc., where previously computed scores are located at the leaves of the hierarchy. The construction of such hierarchy is relative. First, it is necessary to classify each single score regarding the system characteristic it better fits in. For example, let us assume a transactional system where four measures such as *throughput*, *delay*, *availability* and *reliability* have been considered. In this case, the first level of aggregation could group *throughput* and *delay* within the characteristic of *performance*, and *availability* and *reliability* within the characteristic of *dependability*. This classification of measures can continue grouping similar sub characteristics into characteristics. Thus, a second level of aggregation would group both *performance* and *dependability* to determine the global quality of the system.

Despite modelling the hierarchical structure of the system, not all the system requirements may have the same importance depending on factors such as the benchmark performer's preferences and the application domain. To cope with this problem, our methodology enables the refinement of the quality model using weights. Thus, it is possible to assign a weight w_i to each one of the k particular measures (or resulting sub-characteristics) within the same hierarchical level according to their relative importance or influence, in such a way that $\sum_{i=1}^{k} w_i = 1$. Weights enable to tune the way in which system characteristics contribute to the global quality of the system. For example, if taking into consideration a distributed system within a non-critical solution such as comfort electronic control in cars, probably a rapid response in terms of performance aspects will have more weight than dependability ones (e.g., weighting them 75% and 25% respectively). Conversely, if for example we refer to the Antiblock Brake System (ABS) of the vehicle, benchmarking performers may weight dependability above performance assigning weights of 75% and 25% respectively. Fig. 4.2 illustrates this last example. The number

above the tree branches indicates the weight assigned in each case.

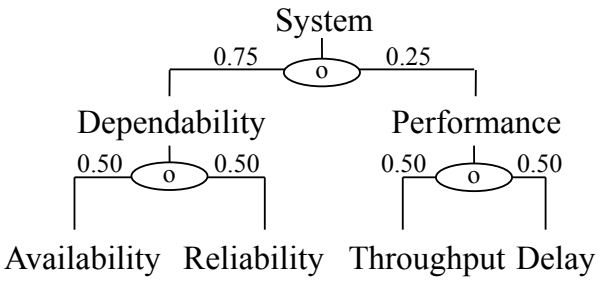

Figure 4.2: Example of weights assignment.

Once weights assigned, it is essential to determine the relation between the elements of the model. For this, different types of operators $\boldsymbol{o}$ may be used to define the conditions under which characteristics are aggregated in Fig. 4.2. The power or generalised mean [24], defined in (4.3), is a generic expression to compute an infinity of aggregation types, considering the notions of scores and weights previously stated. Such expression is equivalent to traditional arithmetic mean, widely used for aggregation, when exponent $r = 1$. However, strikingly, the use of different aggregation operators has been rarely considered despite their power to represent, for instance, a punishment in the aggregation result when requirements are not being accomplished or a reward for those requirements that satisfy evaluation criteria. Thanks to (4.3), it is possible to define as many aggregation types as values may take exponent $\boldsymbol{r}$. Indeed, authors such as Dujmovic propose up to 20 different ones [20]. However, the selection of the proper aggregation operator is a task whose complexity increases as far as more alternatives are considered. Thus, our goal is to define a reduced set of equivalence classes that intuitively represent the different possible levels of aggregation through distinct values of r.

$$S = (\sum_{i=1}^{k} w_i s_i^r)^{\frac{1}{r}} \tag{4.3}$$

To address this challenge, first, it is necessary to introduce the notion of *andness* [25], and how it relates to exponent r. The *andness* of an aggregation operator o, defined in (4.4), is a 1-to-0 coefficient where $andness = 1$ represents that all the system requirements must be satisfied at the same time, and $andness = 0$ involves that just accomplishing

any system requirement (regardless which one) is enough.

$$andness(o) = \frac{max(x) - o(x)}{max(x) - min(x)} \tag{4.4}$$

According to [20], $andness = 1$ is associated to $r = -\infty$ whereas $andness = 0$ equates to $r = \infty$. Mathematically, it is quite easy to prove how *min* is the operator $o(x)$ that makes $andness = 1$, and *max* is that making $andness = 0$. For the sake of homogeneity, let us denote *min* with $S+$ to intuitively illustrate the idea that all the system requirements keep a relationship of *strong simultaneity*. Following the analogous reasoning, let *max* be represented with $R+$ to show the notion that any accomplished system requirement *strongly replaces* the rest (despite they are not satisfied). In the middle, $andness = 0.5$ matches to *arithmetic mean*, which, as previously introduced, is represented with $r = 1$. Let us denote this operator with N to associate its use with the meaning of *neutrality*. Between $andness = 1$ and $andness = 0.5$ there is a gradation of aggregation operators that can be explained as filters that progressively boost the influence of simultaneity against replaceability in system requirements, as far as *andness* tends to 1. Mathematically, this implies minimising the influence of higher scores while maximising that of lower ones in the aggregation result. For the sake of simplicity, we have selected $andness = 0.75$ as a representative value of this range. Let us denote this operator of *weak simultaneity* as S. Conversely, the range of operators among $andness = 0.5$ and $andness = 0$ boosts the influence of replaceability with respect to simultaneity as far as *andness* tends to 0. Similarly, this implies minimising the influence of lower scores while maximising that of higher ones. We have selected the aggregation operator with $andness = 0.25$ to represent this equivalence class. Let us denote the *weak replaceability* of this aggregation operator with R. The different values exponent r takes depending on the number of inputs of the aggregation can be found in Table 4.1. For instance, considering the aggregation of 5 different scores with normalised values of 90, 70, 70, 50 and 20, with evenly distributed weights, the final score obtained for operators $R+$, R, N, S, and $S+$ are 90 (max), 72, 60 (arithmetic mean), 48, and 20 (min), respectively.

Previous simple aggregations between scores can be nested to denote those requirements having a special meaning or priority, i.e., a certain degree of mandatoriness or sufficiency for a particular system requirement within the same hierarchical level.

Table 4.1: Value of exponent r for the operators considered.

Aggregation operators	**2 inputs**	**3 inputs**	**4 inputs**	**5 inputs**
S+ (strong simultaneity)	$+\infty$	$+\infty$	$+\infty$	$+\infty$
S (weak simultaneity)	3.93	4.45	4.83	5.11
N (neutrality)	1	1	1	1
R (weak replaceability)	-0.72	-0.73	-0.72	-0.71
R+ (strong replaceability)	$-\infty$	$-\infty$	$-\infty$	$-\infty$

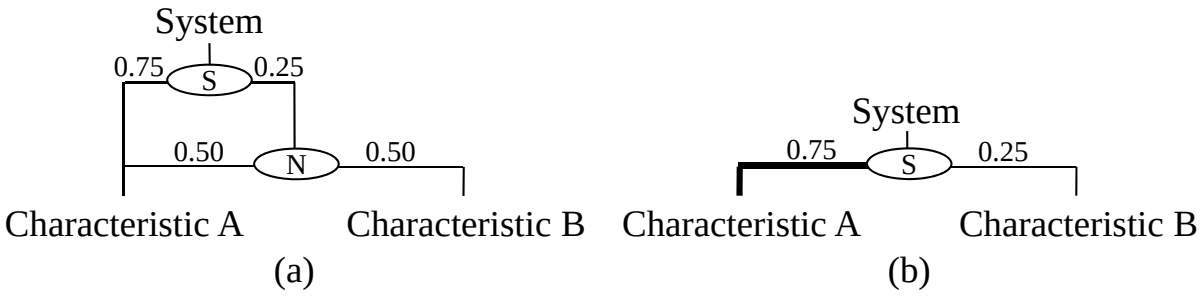

Figure 4.3: Model representing the priority of Characteristic A versus Characteristic B: (4.3a) full model showing how Characteristic A feedbacks its own simultaneity operator (Characteristic A is mandatory), and (4.3b) compact version of that model representing exactly the same hierarchy.

For example, Fig. 4.3a illustrates a case where *characteristic A* feedbacks its own simultaneity aggregation (e.g., S), which basically means that satisfying that characteristic is a mandatory condition for the system. Logically, this can be seen as $A \wedge (A \vee B)$, with different degrees of *andness* depending on the selected operators. Thus, not satisfying the requirements of that characteristic would severely penalise the system. Conversely, applying a replaceability operator (e.g., R), would involve defining that characteristic as a sufficient requirement. Likewise, this could be logically expressed as $A \vee (A \wedge B)$, with the selected degrees of *andness*. Fig. 4.3b depicts exactly the same model as Fig. 4.3a but using a simplified notation to ease the use of mandatory and sufficient requirements. Thick branch represents priority requirements in such a way they become mandatory if using S or $S+$ operators, and sufficient if using R and $R+$. To complete this simplification, neutrality operator N and equitable weighs are assumed for the branches omitted.

All the concepts described during this section are applied in the next one through a case study. In the case study, the simplified notation will be used.

4.3 Evaluating perturbations on ad hoc networks with aLSP

This case study aims to show the feasibility of this methodology to determine the impact that each single perturbation has over a system when considering its injection separately from the rest of perturbations compounding the faultload. In [3], the authors perform the evaluation of two different and representative types of ad hoc networks, a static Wireless Sensor Network (WSN) where 6 real nodes execute AODV routing protocol (Network A) and a Mobile Ad Hoc Network (MANET) where 6 real mobile nodes run OLSR routing protocol (Network B), when subjected to perturbations. Such set of perturbations is formed by accidental faults like *signal attenuation* and *ambient noise*; and attacks such as *flooding attack*, *replay attack* and *tampering attack*.

The networks studied on this paper are mapped into a specific context of use, representing each one different situations of the real world. The specifications of each network are represented in Table 4.2.

Table 4.2: Experimental configuration of Network A and Network B presented in [3].

Network	**RP**	**Speed**	**Area**	**Range**	**Workload**
A	AODV	6 nodes: 0 m/s	30 x 50 m	20 m	Text data (500 bps)
B	OLSR	6 nodes: [0-3] m/s	300 x 150 m	125 m	VoIP traffic (100 Kbps)

RP: Routing Protocol

4.3.1 Measures selection

In the paper, the authors evaluate the impact of each perturbation in the network considering two performance measures: the applicative throughput (or Goodput), and the

increment of delay (or Jitter); and two measures of dependability: the percentage of packets correctly delivered (or Integrity), and the percentage of time the network is ready to be used (or Availability). Table 4.3 illustrates the values measured by the authors for each considered perturbation in Network A and Network B.

Table 4.3: Measures obtained from the case study of ad hoc networks.

			Perturbations				
	Measure	**Golden run**	**Signal attenuation**	**Ambient noise**	**Replay attack**	**Flooding attack**	**Tampering attack**
Netw. A	*Availability (%)*	92.94	73.98	88.74	93.89	51.22	90.12
	Integrity (%)	99.03	97.53	92.12	98.54	97.56	8.01
	Goodput (Kbps)	0.19	0.17	0.18	0.19	0.10	0.19
	Jitter (ms)	319.89	353.45	332.66	300.78	721.66	312.44
Netw. B	*Availability (%)*	95.14	73.9	87.00	75.20	65.00	90.33
	Integrity (%)	98.34	98.73	92.26	99.44	98.23	62.90
	Goodput (Kbps)	96.45	85.19	90.56	70.90	80.18	96.45
	Jitter (ms)	199.98	210.23	211.11	220.88	230.55	195.00

4.3.2 Scales of measure

This case study has an interesting detail that worth to be mentioned. The authors establish a discrete three level criterion (Low, Medium or High) to evaluate the impact of perturbations on the measures: *"In this way, the impact is considered low, medium or high if the measure is degraded underneath 5%, over 5% or over 10% respectively, according to the golden run results"*. Accordingly, (4.5) and (4.6) define a discrete three-level criterion function for the-higher-the-better measures (availability, integrity and goodput), and the-lower-the-better measure (jitter), respectively. In these equations, $B(m_i)$ refers to the baseline computed value for measure m_i.

$$s_i = c_i(m_i) = \begin{cases} 0, & m_i \leq 0.90 \cdot B(m_i) \\ 50, & 0.90 \cdot B(m_i) < m_i < 0.95 \cdot B(m_i) \\ 100, & m_i \geq 0.95 \cdot B(m_i) \end{cases} \tag{4.5}$$

$$s_i = c_i(m_i) = \begin{cases} 100, & m_i \leq 1.05 \cdot B(m_i) \\ 50, & 1.05 \cdot B(m_i) < m_i < 1.10 \cdot B(m_i) \\ 0, & m_i \geq 1.10 \cdot B(m_i) \end{cases} \tag{4.6}$$

4.3.3 Preferences aggregation

After identifying the three different levels quantifying the impact of perturbation on the obtained measures, authors do not detail how to determine the impact of the perturbation on the whole system. Instead, they perform a qualitative analysis (also based on three discrete levels) with no clear rules about how it was perform. Accordingly, as no special requirements for the scores aggregation are defined, equitable weights and neutral aggregations have been considered for all the branches of the proposed quality model shown in Fig. 4.4.

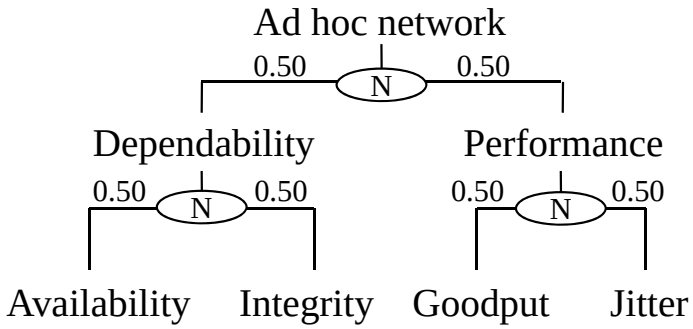

Figure 4.4: Quality model to determine the impact of each perturbation on the considered ad hoc network.

4.3.4 Hierarchical analysis

To show the power of this technique, we will perform a hierarchical comparison between the behaviour of both networks (A and B). This means that we will compare the networks based on the middle-criteria results obtained with aLSP to illustrate the great advantage that a hierarchical navigation suppose for an interpretation of measures. Table 4.4 shows the results obtained from the analysis of the tampering attack at all levels following the

criterion defined in Figure 4.4, the raw measures' values are the normalized values using equations (4.5) and (4.6)

Table 4.4: Fine to coarse-grained results of the Tampering attack in both networks

Netw. A	*Availability (%)*	100.0	*Dependability*	50.0	*Ad hoc network*	75.0
	Integrity (%)	0.0				
	Goodput (Kbps)	100.0	*Performance*	100.0		
	Jitter (ms)	100.0				
Netw. B	*Availability (%)*	50.0	*Dependability*	25.0	*Ad hoc network*	62.5
	Integrity (%)	0.0				
	Goodput (Kbps)	100.0	*Performance*	100.0		
	Jitter (ms)	100.0				

With this type of analysis, performers can analyse features of the systems that would not be possible to analyse only with the system's global score. For example, from the global score it can be seen that network A behaves better than network B when a tampering attack is performed. However, it is not until an analysis of results at a more fine-grained level (*Dependability vs Performance*) is done, that it can be stated that that a tampering attack has a lower impact in network A because its dependability features have better than the network B ones. Indeed, if we pay attention to the results is the *availability* aspect that has a better performance in network A than in network B.

The aim of this very small and simple example is enough to illustrate how the full behaviour of a system is represented using this kind of analysis. This way, in benchmarks with a large amount of measures, where the analysis is more complicated, with aLSP performers can navigate through the different levels, and choose between compare systems using the global score, or deep into more fine-grained levels to identify differences between the systems.

4.3.5 Analysis of results

The global scores obtained for each of the networks are listed in Table 4.5. As previously stated, authors make a qualitative analysis of the impact of each perturbation on each measure to determine the actual impact of the perturbation on the whole system (Low,

Medium, High). Since there is no explicit information about how this analysis is performed, we propose to determine the impact level according to the global score obtained for each perturbation. As measures are normalised according to their deviation with respect to the baseline, final scores between 100 and 70 indicate that the perturbation is barely affecting the system (low impact level), scores between 69 and 40 show a medium impact level, and scores between 39 and 0 reflect a high impact.

Table 4.5: Characterisation of the impact level according to the scores for Network A and Network B.

	Perturbation	**Score**	**Quality Model Impact level**	**Original Impact level**
Network A	Signal Attenuation	25.0	High	High
	Flooding attack	25.0	High	High
	Ambient noise	75.0	Low	Low
	Replay attack	100.0	Low	Low
	Tampering attack	75.0	Low	Low
Network B	Signal Attenuation	37.5	High	High
	Flooding attack	25.0	High	High
	Ambient noise	50.0	Medium	Medium
	Replay attack	25.0	High	High
	Tampering attack	62.5	Medium	Low

The resulting classification for perturbations affecting both networks matches that obtained in the original paper, but for the *tampering attack* on Network B, which is now classified as having a Medium instead of Low impact. This divergence obviously derives from the vague description of the characterisation performed on the original paper. This shows the necessity of precisely defining the criterion and procedure followed during the results analysis. Otherwise, the same results could be interpreted in a completely different way, preventing this process from being repeatable.

In addition to the analysis performed in the original work, and to show the potential of the proposed approach, it could be possible to define a new quality model to help evaluators when deploying a new routing protocol in the network, tuning routing

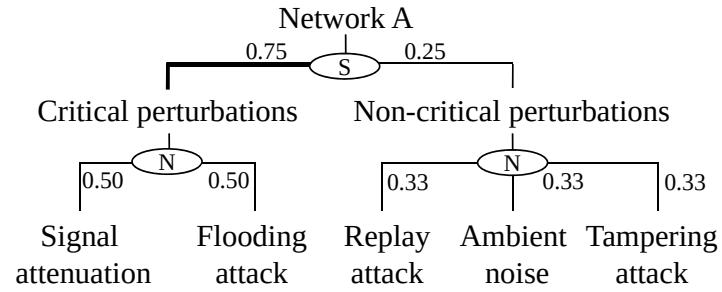

Figure 4.5: Aggregation of perturbations for Network A (WSN).

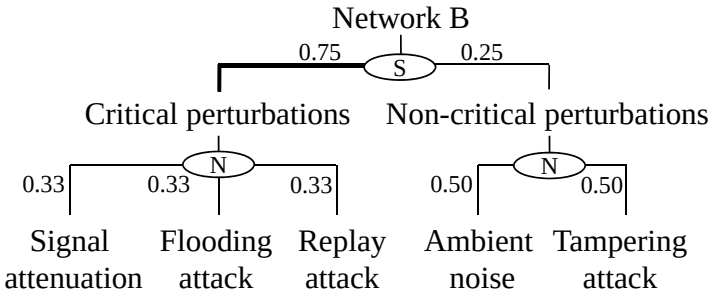

Figure 4.6: Aggregation of perturbations for Network B (MANET).

protocol parameters, or introducing new fault tolerance mechanisms, for instance. This model could take into account the information extracted from this case study, so those perturbations presenting a high impact on the system could be aggregated with equal weight under *critical* perturbations category, and those with a lower impact could be grouped under the *non-critical* perturbations category. The severity of critical perturbations could be remarked by punishing those critical scores with a low value. So, a mandatoriness relationship with the simultaneity operator S, could be used to illustrate this purpose. Medium and low impact perturbations could present different weights, like 0.75 and 0.25 respectively, to reflect their different importance. Fig. 4.5 and 4.6 show the resulting quality models for Network A and Network B respectively.

4.4 Discussion

The results obtained using aLSP in this case study show the great potential of this approach. The analysis performed was mapped to the criterion used by the authors in

the paper as faithfully as possible, and even a discrete analysis of the results was possible by tuning the normalizing functions.

Even though we have provided evidences of the great adaptability of aLSP to interpret resulting measures from a dependability benchmark, we have applied it to results obtained from previous studies done in this field so far.

With the ambition of performing all the necessary steps to prove this techniques feasibility, in this work we have gone a little bit further and we have integrated aLSP in a real framework designed to evaluate resilience (dependability in spite of changes) in ad hoc networks. The technical challenges and the results of this integration are described in next chapter.

Chapter 5

w-TOMCAM: a web-based TOol for Multi-Criteria Analysis of Measures

Given the need of performing resilient evaluation of ad hoc networks in presence of perturbations, during the work done in [26] a Resilience Evaluation FRamework for Ad Hoc Network was developed to cover this need. This tool was developed considering that new functionalities would be added to test different scenarios and situations. Its structured design allows to easily create or adapt new modules to the framework to evaluate either different perturbations or different kind of devices. REFRAHN is an evaluation platform designed following the emulation strategy, thus, it uses real nodes that remain physically static during the evaluations but virtually in constant movement if needed, as the mobility of the nodes is simulated. As the structure of the framework is highly detailed in [26], only a brief introduction of its structure and its functioning will be provided. As a collaboration with the author of [26], Jesús Friginal López, that is also one of my mentors, the work presented in this chapter was possible. This chapter describes the process followed to develop and integrate a web TOol for Multi-Criteria Analysis of Measures (w-TOMCAM) in REFRAHN. w-TOMCAM is in charge of the interpretation of measures of the experiments performed in REFRAHN using aLSP, at the same time that has improved its usability and accessibility.

5.1 About REFRAHN

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 systems execution; and (iii) the analysis of the impact of these faults into the system.

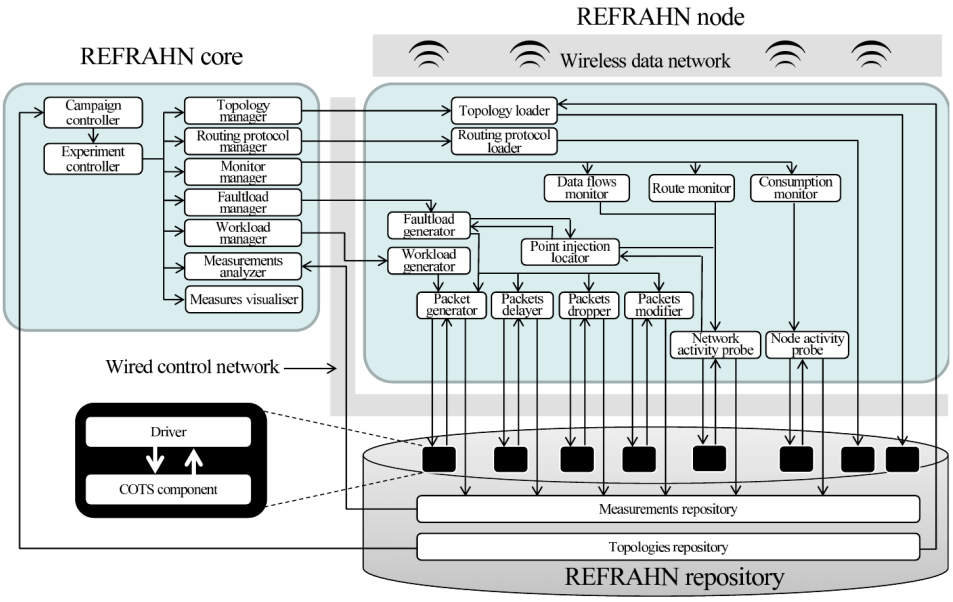

Figure 5.1: General architecture of REFRAHN

As can be seen in Figure 5.1, 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. REFRAHN 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 REFRAHN core, using Ethernet

technology to avoid large latencies. This node is in charge of configuring all the network nodes and controlling the experimentation. The fact of using two different interfaces is very important to reduce the intrusiveness of the evaluation platform itself in the evaluated system. Furthermore, the architecture is completed with a shared space, named REFRAHN repository, that can be accessed by the core and the nodes for the storage of components and data. The REFRAHN repository has been implemented using the Network File System (NFS) technology, thus enabling REFRAHN core and nodes to access files over the wired network in a manner similar to how local storage is accessed. 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, nodes are synchronised through the Network Time Protocol (NTP).

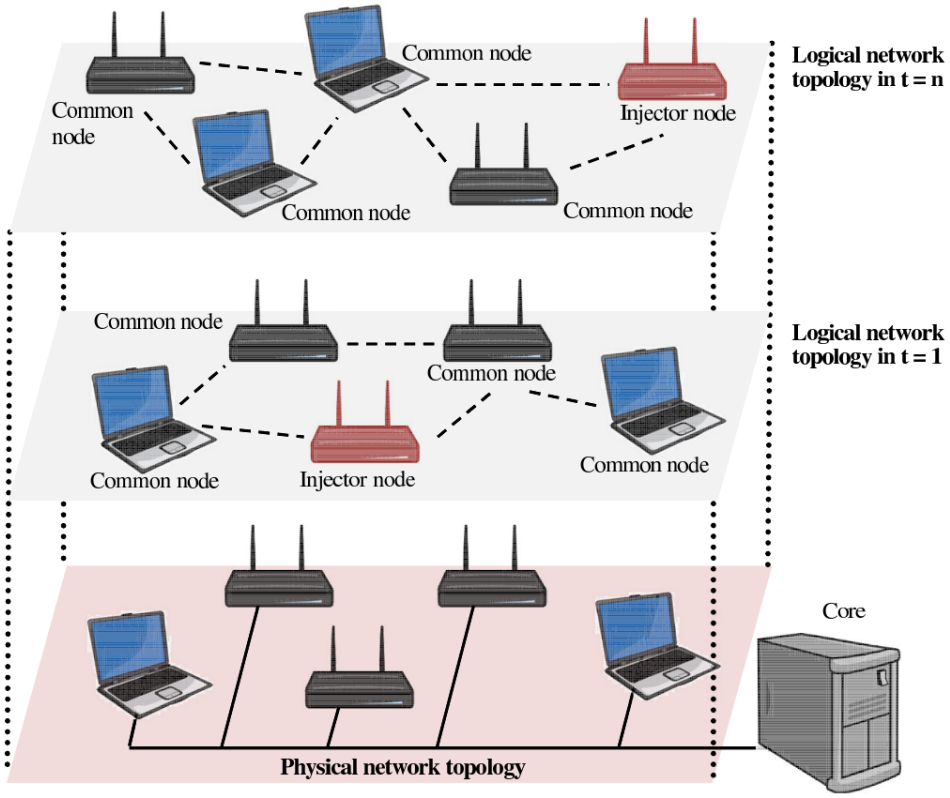

Figure 5.2: Basic elements of REFRAHN

It is to note the flexibility offered by the proposed architecture. Since nodes mobil-

ity is emulated, the experimental 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. All These ideas are represented in Figure 5.2. The considered ad hoc routing protocol can be changed to evaluate the features of different targets. Likewise, the wireless interface can also be replaced to take into account different physical wireless technologies.

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 physical layer (Bluetooth, Zigbee, WiFi, etc). It is structured in two types of elements, the core, and the common and injector nodes. The emulation of mobility enables experimenters to create scenarios with dynamic topology without physically moving the nodes. The implementation of REFRAHN is developed in C and shell script.

5.2 Overview of the aLSP process in TOMCAM

The implementation of the aLSP relies in two main parts: The definition of the quality model, and the execution of the analysis. All the information required to represent the quality model is stored in a configuration file like the one shown in Figure 5.3.

This configuration file corresponds to the quality model shown in Figure 5.4. In the file, $MEASURE\_X$ are the measures obtained from the experimentation process, and to be able to normalize the values for the analysis, their thresholds and the type of function that must be considered are provided for each one. The aggregated measures are defined as its name, followed by the operator that is used (S+, S, N, R or R+), and the next following lines determine which measures participate in the aggregated value, the weight they have in the aggregation and a flag to know if a value has priority above the others. Defining the features of the analysis with configuration files, allows evaluators to define as much configuration files as they want to analyse the results of the same experiment.

This way, the results can be analysed considering different application contexts or user requirements.

```
################
# QUALITY MODEL
################

# RAW MEASURES

MEASURE_1   MEASURE_2   MEASURE_3   MEASURE_4

# THRESHOLDS
#
# measure_name  min_threshold  max_threshold  value_type

MEASURE_1   50      100     INCREASING
MEASURE_2   30      40      DECREASING
MEASURE_3   3       16      DECREASING
MEASURE_4   120     370     INCREASING

# AGGREGATED MEASURES
#
# aggregated_name  operator_used
# first_measure    weight(%)  has_priority(1-0)
# second_measure   weight(%)  has_priority(1-0)
# ...

AGGREGATED_1 R+
MEASURE_1  25  0
MEASURE_2  75  1

AGGREGATED_2 N
MEASURE_3  40  0
MEASURE_4  60  0

GLOBAL_SCORE N
AGGREGATED_1  50  0
AGGREGATED_2  50  0
```

Figure 5.3: Configuration file for a quality model

In REFRAHN all the results from the experiments are stored in single files identified by the name of the experiment. The fact that all data is stored as plain text in files, but following a determined structure (as it happens with the configuration files to define the

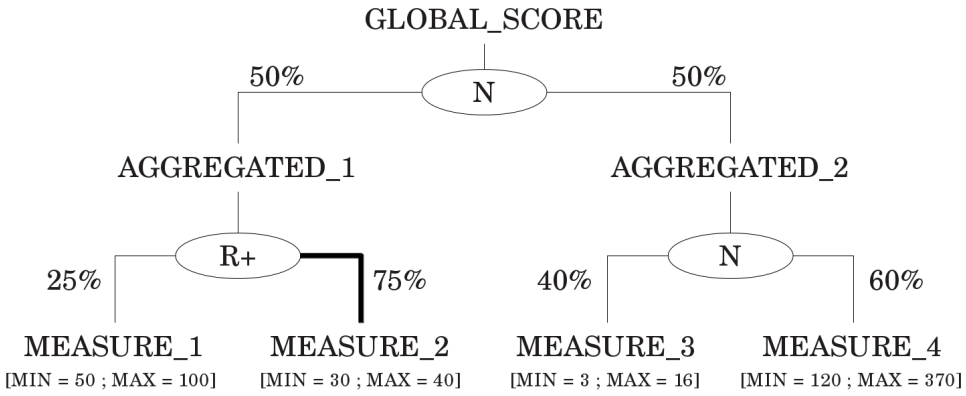

Figure 5.4: Illustration of the quality model defined in Figure 5.3

quality models), makes the interpreted programming language AWK [27] very useful for dealing with the data, thus it is used to programm the aLSP module.

When the experimentation has finished, and the measurements are stored, all the process to analyse the data is performed using AWK. First, from the file with the measurements we extract the values and store them in array, where the index of the array is the name of the measure itself. After that, all the information about the process of analysis in the configuration file, is extracted and stored in different arrays to be used to perform the analysis. Finally, when all the data is accessible, it is used in the aLSP module to provide the scores. The full code is too large to be explained in detail, however, with the aim of providing some notions about how the modules work, Figure 5.5 shows the implementation of the first function executed in the aLSP module, the one used to normalize each measure. This function implements the mathematical operations described in equations 4.1 and 4.2 showed in Chapter 4. The function type of each measure are stored in the array *functype*, this array is filled with the data extracted from the configuration file. The same happens with the arrays *min* and *max*, that store for each measure its minimum and maximum threshold respectively. As it can be appreciated by its identifier, the array *results* contains the values obtained after the experimentation of the measures selected for the analysis. Then, given a measure name, the function applies the normalizing function corresponding to its function type, and return the normalized value of the measure.

When the aLSP module has finished, evaluators have the outcomes of the execution

```
function normalize(measure_name){

  if(functype[measure_name] == "INCREASING"){
    if(results[measure_name] < min[measure_name]){
      return 0;
    }
    else if(results[measure_name] > max[measure_name]){
      return 100;
    }
    else{
      val = (100 * (results[measure_name] - min[measure_name]) /
            (max[measure_name] - min[measure_name]));
      return val;
    }
  }
  if(functype[measure_name] == "DECREASING"){
    if(results[measure_name] < min[measure_name]){
      return 100;
    }
    else if(results[measure_name] > max[measure_name]){
      return 0;
    }
    else{
      val = (100 * (max[measure_name] - results[measure_name]) /
            (max[measure_name] - min[measure_name]));
      return val;
    }

  }
}
```

Figure 5.5: Normalize function in the aLSP module

available, this outcomes represent the concluding remarks they will extract based on the analysis defined in the quality model given before the experimentation.

To ease the task defining the quality models and have the results in a more visual and accessible format, w-TOMCAM has been developed with a web interface to interact with REFRAHN and simplify the evaluation process for new evaluators. Next section describe the features of the web side of w-TOMCAM and illustrates the different steps that have to be followed to perform an evaluation in REFRAHN.

5.3 Incorporating w-TOMCAM to REFRAHN

The use of REFRAHN was limited to the group members that participate in its development. But the will to share it with other evaluators made us change our perspective on how REFRAHN should be used and accessed. The access to the core of REFRAHN was restricted to those with permissions on that machine, and as letting others use REFRAHN could not imply giving them direct access to the machine itself for security reasons, we decided to implement a web interface to make it accessible, without compromising the integrity of the platform. The features of that compose the web part of w-TOMCAM are detailed in next sections.

5.3.1 Structure

Before REFRAHN was developed as a functional tool, to perform evaluations of ad hoc networks we used initially a group of scripts and programs in C language to do the experiments. When the number of scripts and programs was increasing, REFRAHN was designed to merge all together in an more structured and controlled way. So, REFRAHN is composed by a large number of scripts and programs to control the different aspects of the experimentations, which complicate the process of inserting an upper layer to it, w-TOMCAM. To add it, we had to design an architecture that encompass the architecture defined in Figure 5.1. As can be seen in Figure 5.6, the access of the external users is limited to w-TOMCAM (orange arrow), and through it, all the set up files for the experiment are defined, the experiments are launched, and the results (and the analysis) of each experiment are seen using w-TOMCAM.

The web service has been developed for w-TOMCAM using HTML5, CSS3, PHP and Javascript technology. An Apache HTTP server 2.0 has been setted up on the Core of REFRHAN, a MySQL database is used to store the information of the experiments. The information obtained from the experiments is also stored in different files, as a set up file for the experiment, the quality model for the analysis, the results obtained, etc. The use of PHP allows to access the content of the database, and also the execution of the necessary scripts and programs for the execution of the experiments. Basically, all the information of the configuration files is introduced through the web interface, and they

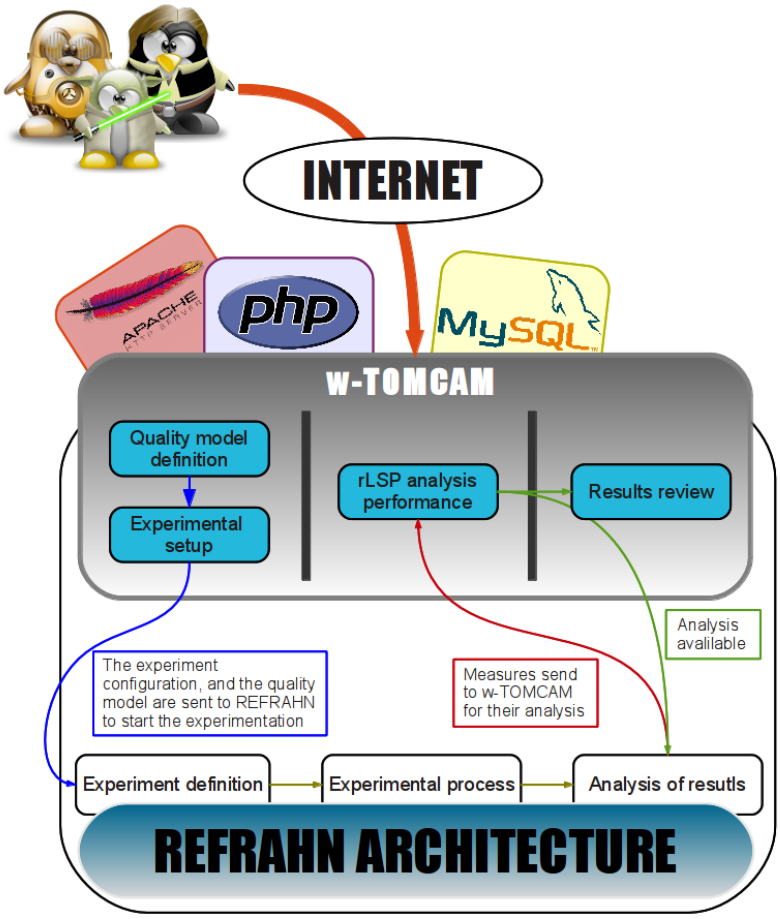

Figure 5.6: Global schema of the interaction between users, w-TOMCAM and REFRAHN

are created using PHP functions.

In order to launch an experiment (or a set of them), as it happens with the evaluation process that is divided in different states, an step by step process trying to match the evaluation states has to be followed through the web interface. The different web pages from the interface and their role in w-TOMCAM are described next.

5.3.2 Depicting the interface

Like every website, w-TOMCAM has an initial (or home) page used to describe the purpose of the project, and instruct possible evaluators on how to use the website to evaluate their experiments. This page, shown in Figure 5.7, is quite simple and doesn't

have many things that need to be explained as its only function is to describe the site's purpose.

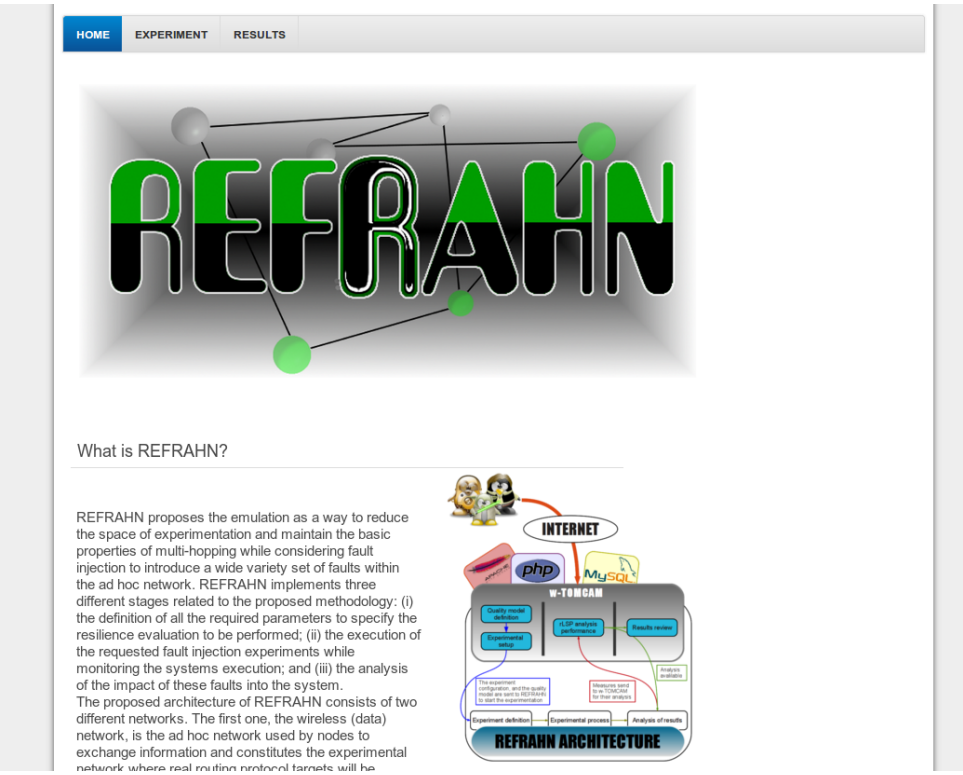

Figure 5.7: Home page

The different flow of information that the user must follow during the process of performing an evaluation is inspired in the steps of the benchmarking process. As some of the functionalities that will be added to REFRAHN in the future are still under research, like the definition of topologies, w-TOMCAM does not provide an interface to interact with them yet, so the user only needs to insert information in two screens. The first screen is the one where the quality model for the aLSP is defined, and the other is the one in charge of the set up information (campaign name, experiment name, number of participant nodes, topology used, etc) for the experiment.

Figure 5.8 shows the web page that is used to define the quality model for the analysis of results. As can be seen, to match the information given in the configuration file shown in Figure 5.3, we make use of dynamic tables. In the first table we add the

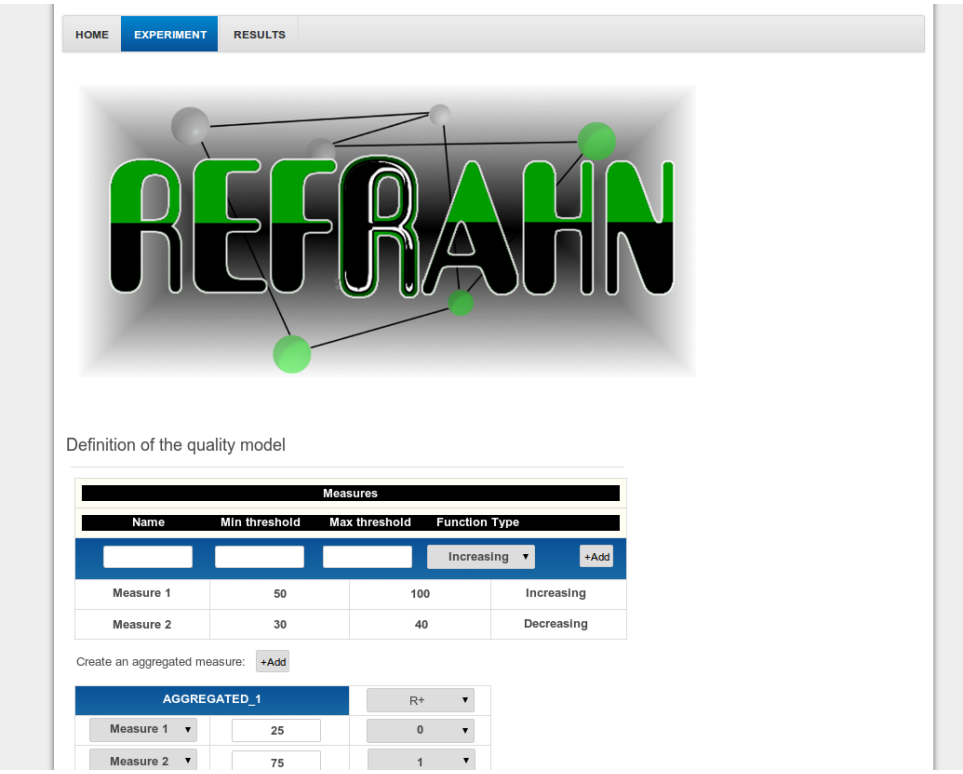

Figure 5.8: Definition of the quality model for the aLSP

rows dynamically, where each row represents a raw measure from the experiment. This measures conform the lower level of the quality model, and for each one, its minimum and maximum threshold, and what type of function should be used to normalize its value are indicated. To define an aggregated measure, a new table is created for each one, indicating the measures name and the operator used in the aggregation. Each row of this table represent either a raw measure or a previously defined aggregated one, and each column indicates the name, the weight of the measure in the aggregation and a flag to indicate weather this measure has priority over the rest, respectively. With the submit button in the bottom of the page, the user indicates that the quality model is finished, so the user is taken to the next page to define those aspects from the set up of the experiment.

In the experiments configuration page, the necessary information about the features of the experiment is introduced through the form that appears in Figure 5.9. The user is

requested to introduce the identifier for the campaign and for the experiment, this way, different experiments will be associated to a campaign, thus grouping them all under the campaign identifier.

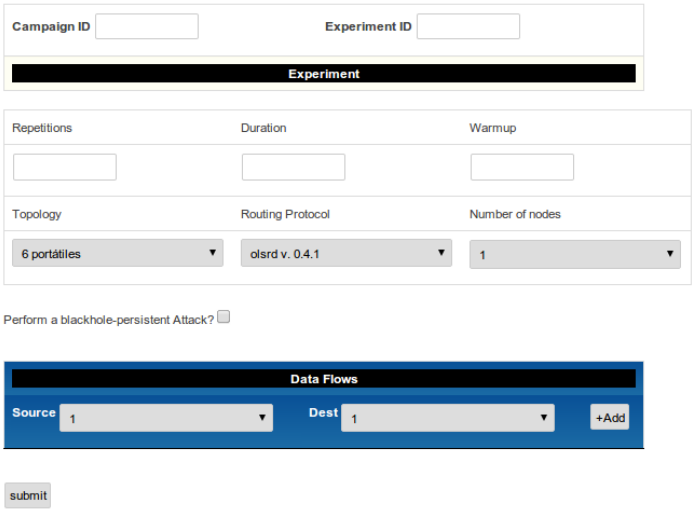

Figure 5.9: Configuration of an experiment

An evaluator can determine the number of times an experiment must be repeated, this is because some evaluators repeat the same experiment several times to obtain (if possible) results statistically more significant. This is done, for example, by obtaining all the values from the same measure in the different executions of the experiment, and then use the mean value as the good one. The duration of the experiment, and the warm up time can be set, if none of these values are introduced, default values are applied (no repetitions, 300 seconds and 30 seconds, respectively). Right now it is not possible to create a new topology, so there are several topologies already available that can be used for the experiments. Even though it is not done yet, when a version of REFRAHN is able to be released, the descriptions of each available topology will be provided. The number of nodes participating in the experiment, and the routing protocol that will be running in those nodes must be determined. REFRAHN can introduce different kind of perturbations in the system, such as jamming attack, tampering attack, packet replay attack, etc. However, as the w-TOMCAM is quite young, the only perturbation that can be used during the experimentation by now, is the black hole attack. If the evaluator

wants to perform a black hole attack during the experimentation, a list of the participating nodes appear to let the user choose which will behave as the attacker. The bottom table indicates between which nodes a data flow will be executed during the experimentation. When the submit button is pressed, the experiment is launched and the user is returned to the home page.

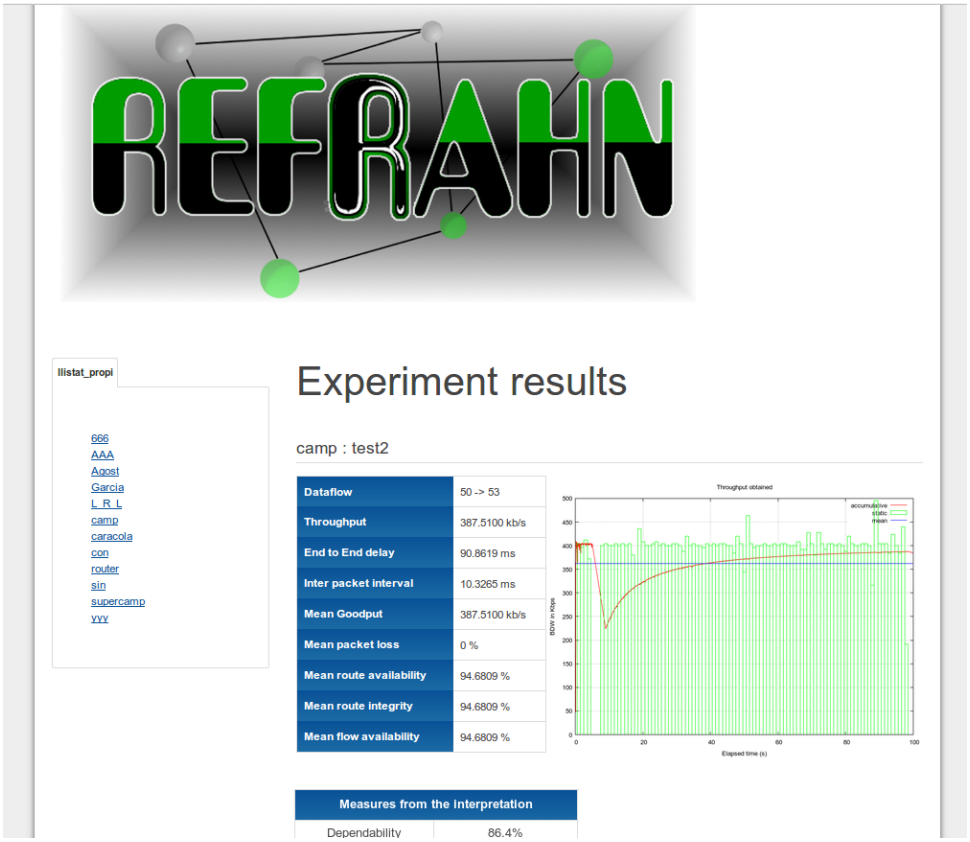

Figure 5.10: Results page

The results of an experiment are not available until the experiment has concluded. Figure 5.10 illustrates how the results are provided to the users. On the left side of the screen there is a drop down list that contains the different campaigns, and when a campaign is clicked, the experiments of that campaign are shown. To view the results of an experiment, the user only needs to click on the name of the experiment, and then the values of the raw measures, and the graphical representation of the throughput during the execution of the experiment is shown. The table under the raw measures show the

results obtained after executing the aLSP.

5.4 Conclusions

REFRHAN has proved to be a powerful tool for performing resilience and performance evaluations of ad hoc networks. As a proof of that are the publications done by my mentors in this field. Even though the integration of w-TOMCAM with REFRAHN to make it accessible through the web is still in an early stage of development, when finished it will commit its purpose on aiding other evaluators to perform evaluation of ad hoc networks.

There are many features that need to be implemented, or that are already under research, but as they do not belong to the purpose of this work, however, they are describe in Section 6.3.

Chapter 6

Concluding Remarks

6.1 Lessons Learnt & Conclusions

Performance benchmarks appeared when there was a demand from manufacturers for techniques to assist the selection components or systems for their products among all those available in the market. Then, with the definition of standards for the development and design of benchmarks, some benchmarks became very popular and nowadays are used by manufacturers to certify the performance of their products.

With the aim of preventing the loss of performance of this products when adverse conditions were present, researchers and people from the industry started to perform evaluations of their products under faulty conditions. But the need for well defined and characterised processes of evaluation lead to the definition of standards to perform dependability benchmarking. However, dependability benchmarking is quite young compared to performance benchmarking, and even though many well defined rules for their design and development have been purposed, few of them have the acceptance of a big part of the dependability community, and thus became standards.

Some of this standards can be found in the Dependability Benchmarking Project [28], where a group of experts in the field cooperated to make it possible. Among these standards, a set of properties that a dependability benchmark must fulfil in order to be accepted by benchmark performers, nevertheless, some aspects of the benchmarking process still require from further attention. This is the case of the interpretation of

the benchmarking results. The lack of well defined rules or standards to determine the interpretation of measures process have made performers to use different techniques and criteria, leading to a situation where a performer has difficulties to reproduce other's conclusions. The impossibility sometimes of reproducing the interpretation of measures followed in a benchmark makes performer to question the whole process. Therefore, providing dependability benchmarks with tools to make the criterion followed in the analysis explicit, is necessary to increase the reliability in this kind of benchmarks.

The source of this problem is in most cases the methodologies used to perform the interpretation of measures. These methodologies lack of mechanisms to make explicit interpretation criteria, and thus they limit the complexity of the analysis that can be performed, providing simplistic interpretations. These issues can be handled with the use of more tool-rich techniques that allow the analysis of multiple measures defining different criteria. This is the case of multi-criteria analysis techniques.

Multi-criteria analysis techniques make use of quality models to define in advance the criterion that performers will follow to interpret the measures. This definition in advance provides the analysis with a complete objectivity, as the results of the benchmark are not influencing the interpretation criterion of the performer. Therefore, the use of a quality model to define the criteria applied fulfils the conclusions reproducibility property, as when using the same quality model, the analysis will be performed always the same way, thus the conclusions obtained will not change.

Nevertheless, some of these techniques require from a certain level of expertise in order to use them. This can be an issue when benchmark performers are not skilled in these kind of techniques. For that reason, potential techniques to be used in a dependability benchmark need to be adapted to ease their use for non-skilled performers. Techniques like the adapted version of the Logic Score of Preferences presented in this work, aLSP, can be easily integrated in already developed benchmarks, and suppose a great distinction from other benchmarks were approaches result simplistic and poor in representativeness.

6.2 Publications

During the preparation of this master's thesis, contributions were sent to a recognized conference and a workshop on Dependable Computing, such as the LADC (Latin-American Symposium on Dependable Computing) or the EWDC (European Workshop on Dependable Computing). Also contributions were sent to a workshop on measurements and networking, M&N(IEEE International Workshop on Measurements and Networking), and to a journal of Network and Computer Applications which is classified as a Q1 journal. Yet another contribution was done during a research day hosted by ITACA-UPV. Contributions are classified according to their current state, accepted or under review.

Accepted:

- Miquel Martínez, Jesús Friginal, David de Andrés and Juan-Carlos Ruiz. Developing a Resilience Evaluation Framework for Ad hoc Networks. *6th Latin-American Symposium on Dependable Computing (LADC), February 2013, Rio de Janeiro (Brazil), Pages 71-72.*

  In this paper we described the development process followed to deploy a resilience evaluation framework for ad hoc networks, pointing out the underlying challenges of developing such an evaluation framework.

- Miquel Martínez, Jesús Friginal, David de Andrés and Juan-Carlos Ruiz. Open Challenges in the Resilience Evaluation of Ad Hoc Networks. *European Workshop on Dependable Computing (EWDC), May 2013, Coimbra (Portugal), Pages 194-197*

  In this paper we analysed the different challenges that must be considered for the evaluation of ad hoc networks, and provided solutions and guidelines to cope with some of them.

- Miquel Martínez, David de Andrés, Juan-Carlos Ruiz and Jesús Friginal. Analysis of results in Dependability Benchmarking: Can we do better?. *IEEE International Workshop on Measurements and Networks (M&N), October 2013, Naples (Italy).*

In this paper we introduce the notion of multi-criteria analysis techniques as a new approach to cover the gaps present in dependability benchmarking when it comes to interpret the results of the benchmark.

- Miquel Martínez, David de Andrés, Juan-Carlos Ruiz and Jesús Friginal. Reducing the subjectivity in the analysis of Resilience Benchmarking results. *Jornada de investigacin del IUI ITACA, Valencia, Spain, June 28, 2013.*

  This contribution was a poster with an overview of the aLSP technique and its application in the interpretation of measures to provide objective analysis through the use of a quality model.

Under review:

- Jesús Friginal, David de Andrés, Juan-Carlos Ruiz and Miquel Martínez. REFRAHN: A Resilience Evaluation Framework for Ad Hoc Networks. *Journal of Network and Computer Applications*

  With this paper we provide a detailed analysis on the different parts of REFRAHN, we describe its internal process (and how to perform it) and give some guidelines for researches willing to develop their own dependability benchmarking platform.

6.3 Future Work

From the work done during this master's thesis, some unanswered questions from different origins have emerged. All these questions are related to the dependability benchmarking, but while some of them raise research targets in the field of the interpretation of results in dependability benchmarking, the others are focused on technical aspects of the implementation of these benchmarks.

The research goals that we establish for future work follow the lead of the current work. As a first step, it is necessary to perform a deeper analysis into multi-criteria analysis techniques, and carry out an study to compare the different MCA techniques and their feasibility to be integrated into a dependability benchmarking. The main purpose is to be able to divide into characteristics the techniques, so different elements can be

identified, and thus they can be analysed individually to determine their contribution to the whole technique. If possible, we could define evaluation processes to determine a MCA technique's suitability to be applied into a dependability benchmark. So, developing a benchmark for MCA techniques.

The technical target is focused on the improvement of w-TOMCAM. Currently w-TOMCAM is used for performing interpretation of measures using aLSP from the experiments done in REFRAHN, but the possibilities of w-TOMCAM are unlimited. A web tool can be provided to w-TOMCAM to allow evaluators to create their own topologies, and also, a deep research must be done in the area of mobility patterns. It is necessary to consider the integration of real traces to define the mobility patterns of the nodes, this way, w-TOMCAM will acquire a high level of representativeness in the results of the benchmarked ad hoc networks.

During next years, our goal is that this current future work becomes part of a PhD, thus facing all the technological problems and coping with the research challenges mentioned.

Bibliography

[1] EEMBC, "Embedded Microprocessor Benchmark Consortium." [Online]. Available: http://www.eembc.org/, 2010.

[2] TPC, "Transaction Processing Performance Council." [Online]. Available: http://www.tcp.org/, 2012.

[3] J. Friginal, D. de Andres, J.-C. Ruiz, and P. Gil, "On selecting representative faultloads to guide the evaluation of ad hoc networks," in *Dependable Computing (LADC), 2011 5th Latin-American Symposium on*, pp. 94 –99, april 2011.

[4] J. Dures, M. Vieira, and H. Madeira, "Dependability benchmarking of web-servers," in *Computer Safety, Reliability, and Security* (M. Heisel, P. Liggesmeyer, and S. Wittmann, eds.), vol. 3219 of *Lecture Notes in Computer Science*, pp. 297–310, Springer Berlin Heidelberg, 2004.

[5] M. Vieira and H. Madeira, "A dependability benchmark for oltp application environments," in *Proceedings of the 29th international conference on Very large data bases - Volume 29*, VLDB '03, pp. 742–753, VLDB Endowment, 2003.

[6] DBench, "Dependability Benchmarking Project." IST Programme, European Commission, IST 2000-25425, [Online]. Available: http://www.laas.fr/DBench, 2012.

[7] "Standard Performance Evaluation Corporation." [Online]. Available: http://www.spec.org/, 2010.

[8] J. Friginal, D. de Andres, J.-C. Ruiz, and P. Gil, "Using performance, energy consumption, and resilience experimental measures to evaluate routing protocols for ad

hoc networks," in *10th IEEE Symposium on Network Computing and Applications (NCA)*, 2011.

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

[10] 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.

[11] G. Concas, M. Marchesi, S. Pinna, and N. Serra, "Power-laws in a large object-oriented software system," *IEEE Trans. Softw. Eng.*, vol. 33, pp. 687–708, October 2007.

[12] Y. A. Al-Sbou, R. Saatchi, S. Al-Khayatt, R. Strachan, M. Ayyash, and M. 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*, pp. 479–484, 2008.

[13] M. F. Morris, "Kiviat graphs: conventions and figures of merit," *ACM/Sigmetrics Performance Evaluation Review*, vol. 3, no. 3, pp. 2–8, 1974.

[14] D. 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.

[15] J. P. Correia and J. Visser, "Certification of technical quality of software products," in *Proceedings of the International Workshop on Foundations and Techniques for Open Source Software Certification*, pp. 35–51, 2008.

[16] J. Figueira, S. Greco, and M. Ehrgott, *Multiple criteria decision analysis: state of the art surveys*, vol. 78. Springer, 2005.

[17] T. Saaty, "What is the analytic hierarchy process?," in *Mathematical Models for Decision Support* (G. Mitra, H. Greenberg, F. Lootsma, M. Rijkaert, and H. Zimmermann, eds.), vol. 48 of *NATO ASI Series*, pp. 109–121, Springer Berlin Heidelberg, 1988.

[18] N. Liu, J. Zhang, H. Zhang, and W. Liu, "Security assessment for communication networks of power control systems using attack graph and mcdm," *Power Delivery, IEEE Transactions on*, vol. 25, no. 3, pp. 1492–1500, 2010.

[19] C. W. Karvetski, J. H. Lambert, and I. Linkov, "Scenario and multiple criteria decision analysis for energy and environmental security of military and industrial installations," *Integrated Environmental Assessment and Management*, vol. 7, no. 2, pp. 228–236, 2011.

[20] 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.

[21] F. D. Backere, H. Moens, K. Steurbaut, K. Colpaert, J. Decruyenaere, and F. D. Turck, "Towards automated generation and execution of clinical guidelines: Engine design and implementation through the icu modified schofield use case," *Computers in biology and medicine*, vol. 42, pp. 793–805, 2012.

[22] A. Passuello, O. Cadiach, Y. Perez, and M. Schuhmacher, "A spatial multicriteria decision making tool to define the best agricultural areas for sewage sludge amendment," *Environment International*, vol. 38, no. 1, pp. 1 – 9, 2012.

[23] J. Dujmovic, R. Elnicki, U. of Florida, and U. S. N. B. of Standards, *A DMS Cost/benefit Decision Model: Mathematical Models for Data Management System Evaluation, Comparison and Selection.* National Bureau of Standards, 1981.

[24] P. Bullen, *Handbook of Means and Their Inequalities.* Mathematics and Its Applications, Springer, 2003.

[25] R. Yager, "A note on weighted queries in information retrieval systems," in *Journal of The American Society for Information Science*, vol. 38, pp. 23–24, 1987.

[26] J. Friginal López, *AN EXPERIMENTAL METHODOLOGY TO EVALUATE THE RESILIENCE OF AD HOC ROUTING PROTOCOLS*. PhD thesis, UPV, 2013.

[27] A. V. Aho, B. W. Kernighan, and P. J. Weinberger, *The AWK programming language*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1987.

[28] DBench project consortium. online: http://spiderman-2.laas.fr/DBench/Deliverables/ETIE1.pdf, 2012.