University Of Piraeus
Department of Digital Systems

Run Time Evaluation of Autonomic Network Functions Trustworthiness

By
Thomas Hall

Submitted in Partial Fulfillment of the Requirements
for the Masters Degree
in Techno-economic Management
in the Department of Digital Systems at
University of Piraeus
Piraeus, March 2014

Thesis Advisor: Demestichas Panagiotis	Thesis Reader: Demestichas Panagiotis
Thesis Supervisor: Ciavaglia Laurent, Bell Labs, Alcatel-Lucent, Paris, France	Thesis Reader: Tsagkaris Konstantinos
Thesis Supervisor: Peloso Pierre, Bell Labs, Alcatel-Lucent, Paris, France	Thesis Reader: Themistocleous Marinos

Abstract

Autonomic networking and self-management are promising concepts for operating large-scale complex networks. Stability, robustness, and security issues arising from future self-organizing networks (SONs) must be understood today, in order to be incorporated into their design, standardization, and certification [1]. The success is gaining ground in addressing the perceived concerns of complexity and total cost of ownership of Information Technology (IT) systems. But we are now faced with a challenge springing from this very success. This challenge is trustworthiness and there are limited research results published in this direction. This, if not addressed will definitely undermine the success of Autonomic Computing (AC). How do we validate a system to show that it is capable of achieving a desired result under expected range of contexts and environmental conditions and beyond? We address the issue of operator trust in Long Term Evolution (LTE) SON through the design and the implementation of verification and trust evaluation mechanism which is mandatory in order to avoid reluctance from them to delegate important management tasks to some autonomic entities.

This work introduces an online trust assessment framework for autonomic

functions and proposes the relevant evaluation mechanisms. This approach addresses trust in terms of reliability and performance of the autonomic entities while it aims to be fully compliant with the Unified Management Framework (UMF [12]). It attributes and continuously updates a trust index that characterizes the behaviour of an entity according to its experiences, thus providing useful information to the operator. We present a study example, related to a SON load balancing autonomic function, to illustrate the methodology of our approach and provide implementation details. Besides, we demonstrate the feasibility and value of the proposed trust assessment framework through a set of numerical evaluations.

Acknowledgements

I would first like to thank my advisor, Professor Demestichas Panagiotis, for trusting me and giving me the chance to have a six month internship in Bell Labs of Alcatel-Lucent in Paris.

In addition, many thanks to all the co-workers from Bell Labs who helped me adapt in the company and especially Laurent Ciavaglia and Pierre Peloso for their guidance and encouragement through these six months.

Special thanks also go to my friends Theodora Kouskouli, Ioli Koukoutsaki, Panagiotis Skaltsivrehis and Bill O' Troupiotis for their relentless communication via Skype until completing this study.

Contents

List of Figures

1 Introduction

1.1 Overview

The rapid and continuous growth in the user needs and the traffic amount has led to the increase of the networks sizes and complexity. Thus, managing such large scale and interconnected networks requires highly skilled professionals to supervise and provide technical support, which greatly increases the operating expenditure of operators. As a solution to this management issue, in order to reduce both complexity of networks and operating costs, autonomic networking stands out as a promising concept able to switch the management and supervising task from human to technology.

Although many studies in the literature handle the autonomic concept and many infrastructures and results are provided, the deployment of such solutions is non-existent essentially because of the lack of trust of operators in autonomic systems. This fear is justified given the importance of the management task and the disastrous consequences that could have an incorrect decision of the autonomic system.

In this context, we propose a mechanism to enable the operators trust in

autonomic networking by verifying the reliability and performance of autonomic control loop decisions. We present an approach that intends to assign a trust index to the autonomic component by comparing the performance of its decisions to others in the same environment context. Our approach is meant to be as generic as it possible. It also aims to be pragmatic, that's why we choose to specify and implement the solution in a way compliant to the UniverSelf project "Unified Management Framework" (UMF)[12]) in order to be able to provide a holistic approach of networking self- management. Finally, we will try to present step by step every thought that was followed or rejected until the implementation in order to show all the research aspects which were explored in this field.

1.2 Organization of the thesis

The rest of the thesis is organized as follows. *Chapter 2* gives a basic description of the context of our work and the relative literature. In *Chapter 3*, we provide the road map in order to go from the trust evaluation to the certification of the autonomic systems, explaining thoroughly each one of the steps. In *Chapter 5* we discuss and analyze our proposal for an online trust evaluation mechanism which will be able to be located in the UMF. *Chapter 6* includes our simulation results and a discussion on them, and in *Chapter 7* we present our conclusions and ideas for future work.

Context and related work

2.1 Autonomic Systems

Autonomic Computing was proposed by IBM in 2001 as a solution to the increasing complexity and the evolving nature of computing systems. The principal goal of the initiative is to make computing systems able to manage themselves according to high-level objectives. The four principal aspects of self-management are: self-configuration, self-optimization, self-healing and self-protection. The architecture of the IBM Autonomic Computing solution is the MAPE (Monitor, Analyze, Plan, Execute) architecture which forms a loop of four main components as shown in the Figure 2.1 [1].

Following the concept of autonomic computing, autonomic networking aims to create self-managing networks to enable the autonomous configuration and parametrization of nodes and networks and dynamic adaptation of the networking environment at runtime. It introduces the concept of autonomic control loops which are responsible of supervising the network metrics, computing optimal solutions and administer the environment changes. In this field, many algorithmic and architectural solutions were presented such as

in [5].

With this huge effort devoted to the design and development of ASs (Autonomic Systems), emphasis is lacking on the certification of these systems. We suggest that ASs must reach trustworthy status and be certifiable to achieve the full vision of AN (Autonomic Networking). Appropriate metrics for validating AS decision, being included in a trust framework, should be defined. We identify this as the core challenge facing the success of AN. This is our main research focus.

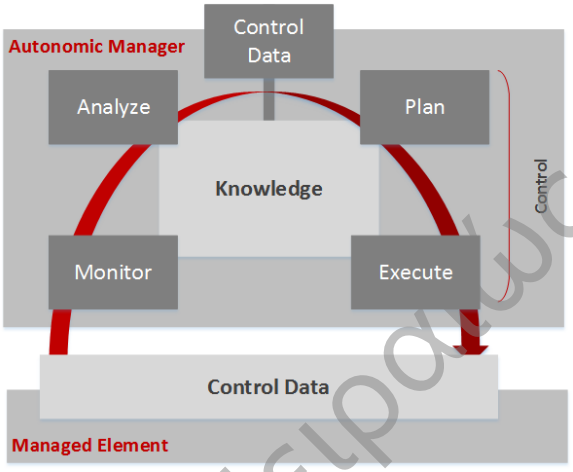

Figure 2.1: MAPE, the architecture of the IBM Autonomic Computing solution

2.2 Trust

Trust is an omnipresent notion. Trust is a term with many meanings as Oliver Williamson (2009 Nobel Memorial Prize in Economic Sciences) said. Lewis and Weigert [6] called trust a highly complex and multi-dimensional phenomenon. Like an elephant, trust is so large that it needs to be digested a bite at a time in order to make orderly progress. Researchers should agree on what trust types exist because common definitions will enable researchers to sort out findings across studies [4]. In order for the researchers to define trust and confront the relative challenges, they should focus (as we did) on

what is mentioned below [5]:

Trust constructs should be comprehensive enough to cover most of the conceptual meaning the word trust conveys in ordinary use, so that scientific work on trust will be grounded in practice otherwise, research results will not be useful to practice.

- Trust constructs should not be so large and complex individually that they stretch trust's conceptual meaning into vagueness.
- Trust constructs should be able to convey the original meaning from prior researchers' models across disciplines in order, where possible, to build on prior research.
- Trust constructs should be measured. Defining variable-level constructs that are well defined and specific meets this challenge.
- Trust constructs should be connected in meaningful ways.
- The constructs should be parsimonious enough to be easily understood and clearly distinguishable from each other conceptually.

Trust has many definitions. Trust could be the subjective probability that an entity will perform in an expected manner beneficial for our welfare, without doing unexpected harm. Considering trust only as a subjective probability leaves out risk as an important concept related to trust. This fact has catalysed a vigorous debate between economists and social psychologists. Therefore, trust is the extent of willingness to depend on others' decisions and actions, accepting the risk of undesired outcome [6].

As mentioned previously, trust has different definitions but the common notions are confidence, belief, and expectation regarding the reliability, integrity, ability, or characteristics of an entity. Besides, trust is, in general,

subjective, because acceptably sufficient trust levels differ depending on the entity goals or requirements, although trust should always be evaluated in an objective manner. Various approaches and architectures were presented for the verification of functioning and the trust establishment in autonomic systems. Many works concern the domain of peer to peer systems and ad-hoc networks [7]. For example, a formal model for trust in dynamic networks which relies on domain theory to provide a semantic model for the interpretation of trust policies. Another solution is based on a quantitative trust establishment framework which defines two metrics (trust and confidence) and computes them using a Bayesian Network and beta distribution. Trust in P2P systems is based on the quality of services provided by the other peer and is represented as a weighted vector of all services where each weight reflects the importance of its corresponding service. These works focus more on the security aspect for distributed entities and even if we were inspired by these methods, our work is more oriented to reliability and performance analysis in a vertical hierarchy where the operator needs to trust the decisions made by its autonomic control loops at run time. There are also different approaches implemented in the relevant literature such as a pervasive supervision approach monitoring the configuration of autonomic elements, the use of multi agents, a self-testing approach that extends the MAPE architecture, an interesting architecture that integrates the concept of trust using two indices and an architecture which uses a case based reasoning approach to enhance the knowledge of the evaluation mechanism and to help the mechanism compute the same two indices [7].

Independent of the particular method that may be applied for evaluating whether a NEM is trustworthy or not, a first important step is to define observable parameters/metrics that can be measured and monitored in a

system, and that can be linked to a level of trustworthiness. In our work we combined the supervision of the executing NEM in order to obtain its decisions with the definition and the computation of an index. The index will be an expression of the evaluation of the parameter/metric that will be monitored and measured after the appliance of the NEM s action in order to observe its influence on the network. We designed a mechanism which will be a part of the UMF and compute the index above in order to evaluate the trustworthiness of running NEMs.

3 Trust evaluation until certification

3.1 Model of Trust Evaluation

The Trust Management System as a whole essentially comprises of two basic components [8]:

- Trust Value Evaluation
- Trust based Authorization

"Trust Value Evaluation" process essentially entails collecting the relevant information necessary to establish trust relationship and at the same time dynamically monitors and adjusts the existing trust relationship. This process assigns a single-valued scalar numeric value in a given range. Lower trust value signifies lack of trust, while higher value denotes more trustworthiness of an entity. A trust value of 0 represents the condition with the highest risk for an entity and the highest trust value represents the condition that is totally risk-free or fully trusted.

Trust, on the other hand, is always related to a particular context. An operator needs not trust an autonomic entity completely. The operator

only needs to calculate the trust associated with the entity within some context pertinent to a situation. The specific context will depend on the nature of application and can be defined accordingly. Based on our current model, trust is evaluated under a single context only, in order to simplify the complexity of the simulations to obtain the first results.

There are three different interested parties in the trust evaluation of an autonomic entity: operators (who want to buy and use these entities), sellers (developers of the entities who aim to sell them) and the evaluator of the entities. Trust Value for an entity is determined by the following two models either by their own or in combination:

- "Evidence-based" model, an appropriate trust value is assigned to an entity based on some evidence such as self-defence evidence etc., explicitly manifested by the entity. Trust is considered as a set of relationships established with the support of evidence.
- "Reputation-based" model, in which Direct Experience coupled with Indirect Recommendations establish the trust value of an entity. Trust is motivated from human society, where human beings get to know each other via direct interaction and through a grapevine of relationships. In a large distributed system, one of the aforementioned parties cannot obtain first-hand information about all autonomic entities. As an option, they can rely on second-hand information or recommendations. "Direct Trust" constitutes the party's own interaction experiences with the evaluated entity; if a party has first-hand experience of interacting with the evaluated entity in the past. "Indirect Recommender Trust", recommendation from peers who have interacted with the evaluated entity before.

Based on these two criterion, the trust rating value could be obtained by applying different mathematical functions/algorithms to all the relevant trust attributes applicable for an entity. All of the trust attributes ("Evidence-based" as well as "Reputation-based" attributes) would be assigned respective weights as part of the trust calculation algorithm. In a highly decentralized environment where entities are dynamic in nature, the identity of every entity is not known in advance. In such an environment, the static role assignment needs to be evolved in such a manner that it enables a dynamic trust value assignment to an entity. In very simplistic terms, "Trust based Authorization" process is a mathematical equation. On one side of the equation is the Security Demand (SD) of an entity; the threshold of the trust evaluation value, which can be admitted, is set by the operator. On the other side of the equation is the Trust Value (TV) that reveals the computation of the trust evaluation value in the way we described before. These two must satisfy a security assurance condition so that TV$\geq SD$.

In this project, the "Trust Management System" is going to have its basic components mentioned before. We are going to compute the Trust Evaluation value (trust index) using an evidence-based model. NEMs will publish their decisions to the Trust Management System after each execution loop and they are going to be evaluated, depending on the change of one or more key performance indicators of the network, respecting each time the type of the running autonomic entity. "Trust based Authorization" equation consists of two numbers expressing the trust index. On the one hand, SD reveals the threshold of the trust index, which can be admitted by the operator, while, on the other hand, TV reveals the computation of the average trust index of a NEM. In the end, after a period of time (some loops of the running NEM), we always check if the average evaluation value (trust in-

dex) of the NEM is over the defined threshold by the operator ($TV \geq SD$), because in a different case we should begin a set of actions in order to inform him about it. However, a reputation-based model could also be added if one of the parties would prefer to use prior results of the NEM or even recommendations for the NEM from another evaluator. Then, it would be more complicated to compute the trust evaluation value, because different weights would be needed for each component of it, but also very interesting, because there would be more holistic and stable perception about the trustworthiness of the NEM.

3.2 Online Trust Evaluation

In the traditional network management schemes, the trust of human operators on the behaviour of the network devices was based on two aspects: exhaustive testing and certification activities before deployment at production level, and the limited capability of the network element for taking decisions and enforcing actions, which make them rather inoffensive for the overall network activity.

The introduction of autonomic elements (such as NEMs) in an operator network implies at the same time the introduction of uncertainty about the correct functioning of the intelligent devices that are able to make decisions on their own. Since the decisions taken by NEMs depend on the context, exhaustive certification activities before deployment (offline evaluation) cannot cover all the possible situations. That's why we tried to introduce in the UMF trust mechanisms that assess the functioning of the NEM (online evaluation). We distinguish between offline and online building-trust procedures. Online means that the operation of a given NEM or a set of NEMs

deployed at production level are being continuously scrutinized by the trust mechanisms during the operational phase of the NEMs. On the other hand, offline trust refers to an evaluation made prior to the deployment at production level and therefore in conditions that may try to simulate or emulate the context in production.

Testing and validation in a live system is arguably the most accurate way of verifying a system to ensure its compliance with the set system's goal. We understand that not all systems can be exposed to real life validation nonetheless researchers have identified it as one of the main approaches. Pervasive supervision is a type of online validation that will be used. It is a monitoring approach proposed in [8] to ensure process correctness of ASs. The supervision system is designed to continuously monitor the decisions of each NEM, interpret the monitored data according to certain operations requirements, like functional correctness, performance, consistency etc., and enforce corrective measures in case of requirements violation. The final step of enforcing corrective measures is still under consideration, because it is not decided yet which should be the next step after the observed violation. We are going to refer to it below as the "Call for Governance" part which follows the evaluation.

3.3 Validation

In this section we define the problem of trustworthiness, identifying its significance and the extent of its challenge. We believe that the ultimate goal of AN should be the certification of AC systems which is also stated in [11]. Yet to achieve certification requires a process and the meeting of some conditions. For unknown reasons and in a bid to get things working faster, the

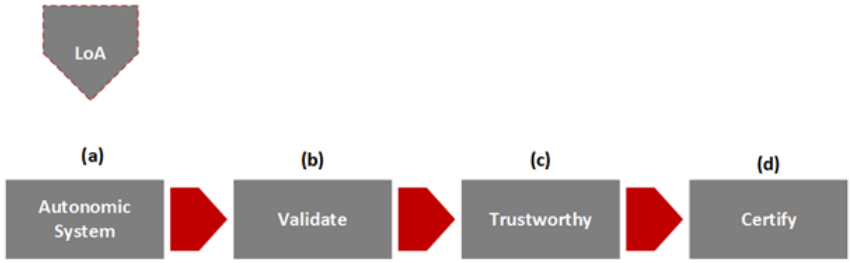

Figure 3.1: Roadmap towards the certification.

AC research community has concentrated efforts on designs and architecture with little or no emphasis on system validation. Only very few researchers have identified trustworthiness as a major AC challenge and yet fewer [9] and [10] have actually suggested or proposed techniques. The problem in clear terms is the ignoring of AS trustworthiness and the general lack of validation efforts that specifically target the dynamic aspects of these systems.

Figure 3.1 represents a section in the journey towards full AC. At point (a) we assume that a system is developed and is considered autonomous at some level. This level is determined by a LoA (Level of Automaticity) measurement methodology which needs some form of standardization. The definition of LoA at this point is prerequisite to the next step. At point (b) is the system's self-validation distributed across design-time and run-time. When it is ascertained that a system is validated then it is trustworthy and trustworthiness is a vital and foundational step on the road towards certification. It then follows that for a system to be certified, it must be trusted and only validated systems can be trusted.

The consequence of a lack of validation comes in two dimensions. On the one hand, is the risk of losing control and loss of confidence that the autonomic system will not fail. This is obvious, owing to the nature of ASs

which includes dynamic changes caused by their features in unpredictable environments and conditions. On the other hand, the issue of standardization of AS design processes is a big matter. It is very unlikely to secure standards for invalidated systems as the general standardization of a system will largely depend on the level of confirmation of its process correctness [11].

3.4 Certification

Certification of ASs is a specific work area that needs attention and we believe this can be achieved through defining proper AS validation mechanisms. AN systems are designed and deployed across many application domains to address the challenge of human management complexities. We may come to a point where these systems take over full control of operations in those domains (e.g., businesses, military, health etc.) and any failure can be extremely costly in terms of down time, danger to life, loss of control etc. This underpins the criticality of AS validation. Robust self-management in AC systems, resulting in dynamic changes, and reconfigurations require that ASs should be able to be continuously validated. Such systems are considered trustworthy and then certifiable. It is then necessary to have a testing approach that combines design/run-time elements and is also an integral part of the self-management architecture. We have a longer term vision to develop certifiable systems. By trustworthiness, we mean a state where we can be confident that an AS will remain correct in the face of any possible contexts and environmental inputs and sequences of these; this is achieved through robust validation.

The authors in [11] proposed the roadmap of Figure 3.1 towards AS trust-

worthiness and therefore the way to achieve certifiable AC systems. Firstly, characteristics or features should be identified, that a proper validation approach should possess. Then a number of inter-related steps should be taken towards certifiable ASs. They identify that the validation step is the most important towards certification and they argue that a proper validation and assessment methodology is the only reliable process towards certifiable AC systems. A proper validation/assessment approach should have the following characteristics:

- Generic: Re-usability reduces complexity and cost (in terms of time and effort) in developing validation processes for AS. A good validation approach should be flexible to be adapted to different adaptation processes and the procedure or process for this adaptation clearly detailed.
- Design/Run-time: The dynamic changes and reconfigurations in AS could result in drawbacks such as the possibilities of policy conflicts and incorrect goal specifications. It is then necessary to impose testing both at design-time and run-time.
- Integrated: Testing should be an integral part of the whole self-management architecture. Testing being integrated to the management structure achieves real time validation which is necessary to mitigate adaptation conflicts and promote consistency.
- Automatic: Validation activity should be human independent (i.e. should be triggered by a change in application context, environmental volatility or a locally-detected failure requiring reconfiguration) following a defined validation process. But proving that a validation mechanism actually meets its set requirements is another issue of con-

cern.

That's why it is very crucial to integrate the validation mechanism in the UMF, because if we want to reach from evaluation to certification we need to have trustworthy NEMs but also trustworthy processes to evaluate NEMs in order for the evaluation to be trustworthy too. UMF has the ideal role for our purpose, uses the best practices and standards which could lead to certification.

4 Trust Evaluation Mechanism

4.1 Background of the mechanism

Our solution consists of an On Line Trust Evaluation mechanism (OLTE), which consists a part of the UMF and is responsible to evaluate the NEMs that the operator will define. Therefore, OLTE mechanism, in order to work properly needs the components below:

- UMF
- a running NEM
- a network
- a user interface
- OLTE

4.1.1 UMF

UMF is a key achievement of the project named UniverSelf [12]. It originally targeted autonomic management of networks. The objectives of the project are to federate the network management over network segments and

to bring autonomic interworking into its maturity age. The work has followed two consecutive approaches: a top-down approach tackling autonomic management under a network wide approach from a somewhat theoretical point of view, and a bottom-up approach tackling many different autonomic solutions to network operators problem all across the different segments and layers of the network. This second approach is somewhat more pragmatic.

Considering together the two approaches, UniverSelf reached the conclusion that a management system for autonomic networks must offer the possibility of integrating autonomic functions coming from different vendors and integrating all them in a single management process. Regarding that, the UMF is the toggle point from a vision targeting autonomic management of network to its instantiation into the management of autonomic functions, which themselves manage the network (elements or resources).

Hence, the UMF is the framework to manage these autonomic functions, which requires two things: first providing these autonomic functions with a given set of capabilities, and second to impose to these autonomic functions a given set of duties.

UMF is a management framework based on three main blocks namely, Governance, Coordination and Knowledge (figure 4.1). The three together managing the autonomic functions NEMs once these autonomic functions comply with UMF specifications, i.e. once the software implementing the autonomic function is performing the duties imposed by the UMF specifications. Governance block responds to the need of the human network operators to have the possibility of supervising the functioning and controlling the behaviour not only of the underlying autonomic functionalities (NEMs), but of the management system as well (UMF core blocks). The UMF Knowl-

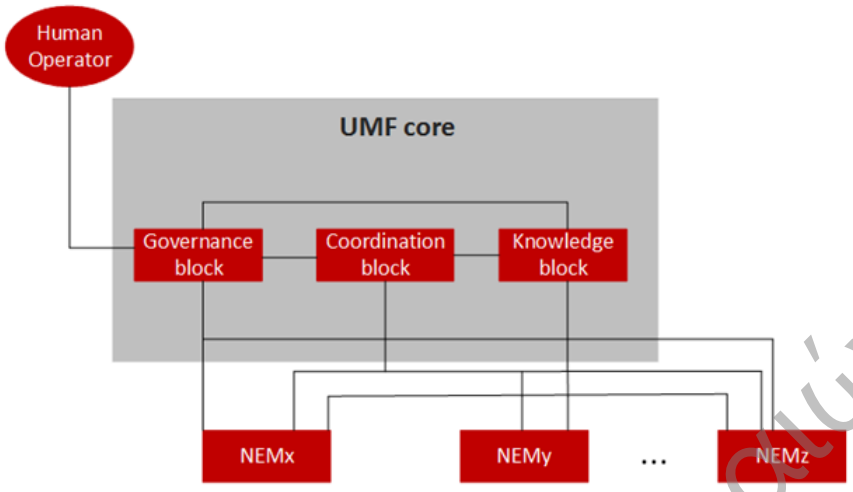

Figure 4.1: Components of UMF.

edge block (KNOW) plays the role of information / knowledge collection, aggregation, storage/registry, knowledge production and distribution across all UMF functional components (i.e., NEMs and Core blocks). The role of the coordination block is to protect the network from instabilities and side effects due to the presence of many NEMs running in parallel. It ensures the proper triggering sequence of NEMs and their stable operation. To this end, the coordination block must define conditions/constraints under which NEMs will be invoked (i.e. produce their output), taking into account operator service and network requirements e.g. the needs to optimize the use of the available network resources and avoid conflicts between NEMs that can lead to sub-par performance and even unstable and oscillatory behaviours.

4.1.2 NEM

NEM, which consists of a piece of software and/or hardware, can be applied to a network to make it more autonomic. One of the key characteristics of UMF is to allow seamless deployment and trustworthy interworking of

multiple/independent autonomic functions that will (each) ease the life of network operators. Hence, any actor of the telecommunication/networking market can develop NEMs: equipment vendor, network management system vendor, network operator, software developers etc. A NEM class is a piece of software that contains the logic achieving a specific autonomic function. Such class is deployed in a network running a UMF system and requires being instantiated on a set of concrete network elements to effectively perform its autonomic function. An instance of a given NEM class performs a given autonomic function onto a given sub-set of network elements. This is achieved by binding the code of a NEM class to a set of identified network resources/equipment. Hence, there may be multiple instances of a given NEM class inside the same network (e.g. one per area). A NEM instance is created by the UMF system in which it is being deployed. Moreover, a NEM instance is managed by the UMF system as an atomic entity, while its internal functioning can rely on separated pieces of software running on different equipment.

For the needs of our project, we will use a SON Load Balancing (LB) NEM [13]. Load balancing means that the load of a highly loaded or overloaded cell is, in one way or the other, offloaded to a neighbouring cell. To this end, there are several strategies, e.g. manipulating the interference situation or manipulating handover parameters. This work focuses on the manipulation of the emitted powers of the antennas. As emitted powers change, the total power that a user receives and the antenna that prevails change too and as a result the user can be served by another antenna. For example, if an antenna is overloaded by users, we can raise the emitted power of the neighbouring antenna in order to attract some users and disburden the other antenna. More formally, SON LB adapts the coverage zone of the relay stations by

adjusting their pilot powers while keeping traffic channels unchanged. The dynamics of the LB-SON is described by the Ordinary Differential Equation (ODE):

$$P_s = P_s * [\rho_0(P)\rho_s(P)]$$

where $P_s$is the pilot power of station s, and ρ_s its load.

4.1.3 Network

The evaluation environment could be either a real network or a simulated one. We will use a network simulated by Matlab in order for us to change any possible settings and, mainly, users' configurations in a much more flexible way than using a real network testbed. Therefore, we will be capable to try all possible scenarios with different traffic configurations. This network will be applied in a cell that we will choose from a map on a real location. Each cell is separated into 6 zones and it is on our hand to define the density of the uniformly spread users in each zone. In each cell there are 4 femtos and a central antenna, emitting power in order to serve the users. For this reason we are going to exploit the simulator of the project UniverSelf which facilitates both UMF and the network.

4.1.4 User Interface

A user interface is needed in order for the human operator to use the mechanism to evaluate the NEMs and have the results of this evaluation. We have implemented a user interface in Java GUI as we can see in figure 4.2. Operators or, generally, high level management will choose through the interface the NEMs that should be evaluated from a list that contains all the

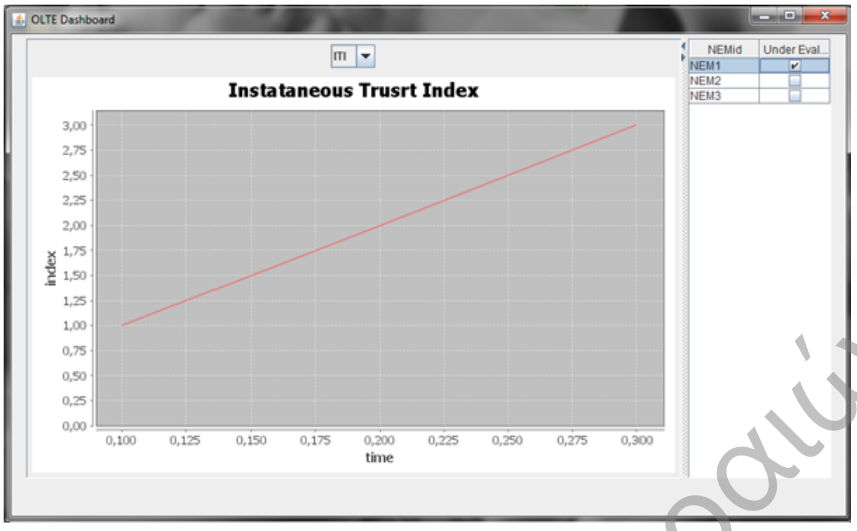

Figure 4.2: The implemented user interface.

available NEMs. After the evaluation, the management will be able to see and investigate, through the interface, the results of the evaluation. More specifically, they will be able to see the graphs of an index which will show how trustworthy or not the NEM is.

4.1.5 On Line Trust Evaluation Mechanism

OLTE mechanism should present the characteristics that can be seen below:

1. Generic/reusable: We tried to design a mechanism which presents reusability and high adaptability. By providing some guidelines it can be reused without much complexity in terms of time and effort.

2. Design/Run-time: The NEMs should be validated each time that the system has converged. What we mean by that is that after each decision/action of the NEM we are waiting a little time (converge time) before evaluating, until the system stabilizes after the applied action. The design validation process will be undertaken by the offline evalu-

ation.

3. Automatic: We tried to make OLTE as automatic as possible. However, every time a NEM should be evaluated, a reference algorithm should be chosen or developed. There is, still, much room for progress in this direction.

4. UMF compliant: It should be integrated to the UMF and cooperate with Government and Knowledge block. Evaluation should be an integral part of the whole self-management architecture. Evaluation being integrated to the management structure achieves real time validation which is necessary to mitigate adaptation conflicts and promote consistency.

Before the operation of the OLTE, high level policies should be defined by the human operator respecting the conditions that should be checked during the evaluation. On the other hand, if these conditions are violated, OLTE should inform the human operator in case he wants to stop the running NEM (Call for Governance) or enforce other precautionary measures. In order to implement the OLTE mechanism, we have to make the NEMs publish, through some policies, their actions to the Knowledge block of UMF where OLTE mechanism is established (will be explained below). A reference algorithm, a case base and a function, which will be computing the trust index, constitute the basic components of the OLTE mechanism. The reference algorithm is going to realize the same work as the NEM in order to have some equivalent decisions to these of the NEMs, which will be comparable. The performance of the reference algorithm compared to the NEM under evaluation is not important; we just want it to be trustworthy and qualitative in order to have a valid and reasonable comparison with the NEM. The

correct choice, for each type of NEM, of the perfect reference algorithm is fundamental, because it will be the tool that will help us to investigate if a NEM is trustworthy. By saying perfect, we mean the algorithm that fits best in each matter that the NEM is trying to treat, e.g. for optimization problems genetic algorithms have the best fit. The case base will be a static base of cases. The idea is that when the under evaluation NEM is known, the reference algorithm can be chosen or created. Then, we run simulations (depending on the different network configurations as far as SON load balancing NEM is concerned) using the reference algorithm in place of the NEM in order to fill the case base. The case consists of some parameters of the network, the decision of the algorithm and the evaluation of the decision. After that, we run simulations using the NEM and when the NEM takes a decision for a specific network s state, we evaluate it, depending on some key performance indicators, and compare the evaluation to this of the relevant case (same or similar network state) of the case base. In the end, we plot the distance between these two evaluations, which constitutes the trust index. We are going to be even more thorough in the next chapters.

The mechanism that we designed and implemented, through the evaluation of the NEMs, will produce knowledge to the management of the network. Given that we want the OLTE mechanism to be compliant to the UMF, and more specifically a part of the UMF, it seems obvious that it should be integrated in the Knowledge block of the UMF. To be more detailed, the OLTE mechanism will be located in the knowledge production part of the Knowledge block which facilitates other knowledge production mechanisms too. This kind of information can be delivered, after processing, to the Governance block of the UMF and from there to the human operator for further actions.

4.2 Challenges

Firstly, we worked on the Case Based Reasoning approach which was used in the relevant literature [1]. CBR, as a learning algorithm, would add knowledge to the OLTE mechanism from past experience. It is a process of arriving at the solution of a new problem on the basis of the solutions of previously-solved similar problems, such as when a doctor, lawyer, or mechanic relies on one experience to remedy a current situation. However, we noticed that the actions of the reference algorithm cannot be applied on the network, they are theoretical just to be compared to these of the NEMs, thus, there was no meaning using the CBR. Maybe, it could be useful to enhance the knowledge and the quality of the OLTE mechanism if we had a dynamic reference algorithm, running on the parallel to the NEM, which would be supported by a theoretical model or law, in order to compute the influence of the decision of the reference algorithm to the network even theoretically. Or even, by enhancing the knowledge and the quality of the validation mechanism (better decisions than these of NEMs based on their influence to the network), it could also be used to improve the NEM itself, but this is not our primordial goal. However, in most cases, finding this kind of reference algorithm is nearly impossible, and, for sure, at this stage it is infeasible.

4.2.1 Case Base

But we retained the idea of the Case Base, a database full of cases that each case is a network state with the antenna powers, the action of the NEM or the reference algorithm (the new antenna powers) followed by the corresponding trust evaluation value. Therefore, we surely need a reference

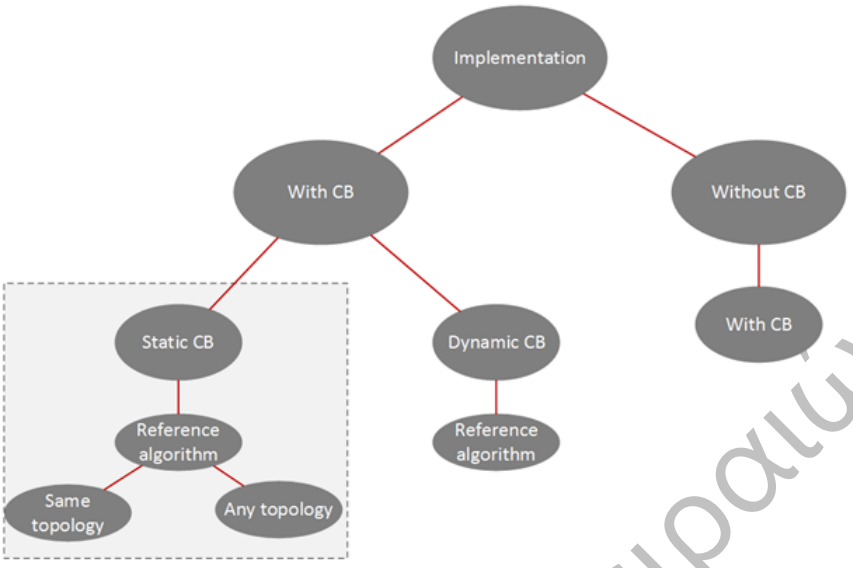

Figure 4.3: Different implementation scenarios.

algorithm to produce similar actions to these of the NEM and, possibly, a CB to store these actions and retrieve them in case the reference algorithm couldn't run on the parallel to the NEM.

As we can see in figure 4.3 There are two basic scenarios depending on the CB: to design a mechanism with a CB or without a CB. If we design the mechanism without a CB or with a dynamic CB, we need a reference algorithm that will be able to run on the parallel to the NEM and produce similar actions on the parallel in order to compare them. However, in fact, this is really difficult, because there is huge variety of already existent NEMs, they are evolving continuously and we dont know what kind of NEMs will be created in the future. Thus, we are not able to have a specific reference algorithm for each one, or even we could have but it would have needed much effort, time and money (such as those of developing a NEM), comparing to the initial goal of the mechanism which is the evaluation of the NEM. And even more, after that the procedure wouldn't be generic and au-

tomated. It could be great if there has been some effort for standardization and categorization of NEMs in order to make our life easier. For example, in this work we evaluate a SON load balancing NEM, which can be classified in the category of optimization NEMs. Our reference algorithm is a genetic algorithm which is generally pretty efficient for optimization problems. So, it could be also used for all the NEMs that belong in the same category. As a consequence, the developers of NEMs would be required to create a NEM respecting some standards and classifying it in a category and, as a result, the design of the evaluation process of the NEM would be easier, more efficient and more generic.

4.2.2 Reference Algorithm

Therefore, we are going to use a static CB which is going to be filled with the decisions of a reference algorithm, before the NEM is going to be operational but it will be used during the online trust evaluation of the running NEM. Trying to do the reference algorithm more generic we decided to use a genetic algorithm for this reason.

Genetic Algorithms are adaptive heuristic search algorithm premised on the evolutionary ideas of natural selection and genetic. The basic concept of GAs is designed to simulate processes in natural system necessary for evolution, specifically those that follow the principles first laid down by Charles Darwin of survival of the fittest. As such they represent an intelligent exploitation of a random search within a defined search space to solve a problem. First pioneered by John Holland in the 60's, GAs have been widely studied, experimented and applied in many fields in engineering worlds. Not only do GAs provide alternative methods to solve problems, they consistently out-

perform other traditional methods in the most of the problems. Many of the real world problems involved finding optimal parameters, which might prove difficult for traditional methods but ideal for GAs. Holland said that computer programs that "evolve" in ways that resemble natural selection can solve complex problems even their creators do not fully understand. However, because of its outstanding performance in optimization, GAs have been wrongly as a function optimizer.

GA maybe is not suitable to be applied to a system because it will cause instability, but it is perfect for optimization problems. Thus, it will be used to find which are the best antenna powers for each network's configuration that provide the best load balancing (giving to the system some time to converge in each simulation). In a genetic algorithm, a population of candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem is evolved toward better solutions. Each candidate solution has a set of properties (its chromosomes or genotype) which can be mutated and altered. The evolution usually starts from a population of randomly generated individuals, and is an iterative process, and the population in each iteration is called generation. In each generation, the fitness of every individual in the population is evaluated; the fitness is usually the value of the objective function in the optimization problem being solved. In our work, the individuals are the powers of the antennas and the fitness function is the most fundamental KPI of the SON load balancing NEM which is the load. The more fit individuals are stochastically selected from the current population, and each individual's genome is modified (crossed and randomly mutated) to form a new generation. The aim of the crossing mechanism is to enrich the diversity of the population by manipulating the structure of chromosomes. The crossings are planned with two parents and generate two

offspring. This is an essential step in the genetic algorithm as it allows the exploration of the search space. The mutation is defined as the random modification of the value of an allele in a chromosome. It plays the role of noise, prevents the evolution of solutions from stagnation and ensures that the global optimum can be achieved. This operator avoids premature convergence to local optima. The new generation of candidate solutions is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population. In our case, the GA terminates when we have the most equivalent load for a set of antenna powers.

In a more generic point of view, for each optimization NEM, given that we know its inputs, outputs, some key performance indicators and what else designer of the NEM believes that is important, we can use, as a reference algorithm, a genetic algorithm to find which inputs could optimize the key performance indicators for different configuration scenarios. It could also happen for another category of NEMs if we can find another type of reference algorithm which could serve this type of NEMs.

4.3 Functionality of the mechanism

4.3.1 Inputs/Outputs of the Mechanism

OLTE is introduced in the UMF as an operation of the Knowledge Production function, since it builds knowledge about the performance of the NEMs. For its proper functioning, OLTE needs as input what is described below:

- A list of NEMs, chosen by the operator, to be evaluated, and conditions

that should always be checked during the evaluation, in case they were violated. When the trust index, for example, drops below the threshold, the operation of the NEM is not considered trustworthy anymore and Governance block should be alerted. This information will be received in the form of policies sent by the Governance core block.

- Every possible static information about NEMs that is needed. Information about the exact functioning of the NEM would be precious for the development of a suitable reference algorithm e.g. if the NEM's function is based on a mathematical or physics equation, which are the inputs and the outputs of the NEM etc. Most influenced parameters (KPIs) by the NEM's actions, on which OLTE should be based in order to evaluate the decisions of the NEM and the reference algorithm, are also needed. Nearly all the above information can be found to the manifest of the NEM, or if it can't, it should be written by the developers of the NEM.

- OLTE should also be informed on the NEMs decisions and actions (which decision/action, in which context), which requires a registration of the OLTE mechanism to the appropriate information from the NEM.

- Information on the network state (before and after the NEM's decision) will be monitored by the simulator, because it is really important due to their existence in the case. Generally, the network's state should consist of all the necessary features that fully characterize a state of the network like the number of the users, the positions of the antennas, the traffic etc. in order to find the same or the similar case from the case

base using the network's state as a key. In our project the network's state consists of the density of users in the different areas of the map which is going to be analysed better in Chapter 5.

The information in the three last items can be retrieved through the "Information Collection Dissemination" function, which allows the access to the NEM Registry and the NEM historical information.

As output, trust evaluation provides information on the NEMs "trustworthiness", in the form of a trust index. This trust index is a measurement of the performance and reliability of the NEM. When the computed trust index drops below the threshold indicated by the policies, an alert should be sent to Governance block of the UMF through the Call for Governance interface. GOV then can set this mistrusted NEM in an "under trial" state, in order to allow the observation of the behaviour and decision making process of the NEM, but without allowing it to enforce actions onto the network.

4.3.2 Operation of the Mechanism

Here are the steps towards the computation of the trust index:

1. We run the reference algorithm (genetic algorithm) to fill the CB covering all the possible configuration scenarios (=network states) before we deploy the NEM.
2. Simulation of the NEM under evaluation starts.
3. When the NEM is taking an action, we search in the CB in order to find similar or even same cases (distance) to the New Case using the Network's state (configuration) as a key.
4. Given a KPI we evaluate both the action of the NEM and this one of

the case of the reference algorithm.

5. We plot the index which is created by the distance of the two trust evaluation values.

This method can be used in any category of NEMs given that some guidelines will be followed in order for the adaptation process to succeed:

1. Get the inputs/outputs/KPIs of the under evaluation NEM.
2. Adapt a reference algorithm depending on the type of the NEM.
3. Run the simulations to build the CB.
4. Run the simulations with the NEM comparing the evaluation of its decisions to this of an evaluation of a similar case from the CB.

It is very important, that each time the evaluator has another NEM to evaluate, needs to know some fundamental information about the NEM (inputs, outputs, KPIs etc) in order to be capable to find or build the reference algorithm. Sometimes, the reference algorithm could be a theoretical law or equation which could give perfect results but most of the times, this could be very difficult. After that, the evaluator can run as many simulations as needed to fill the case base. In case that the CB has not exactly the same cases as the potential cases of the NEM, the evaluator needs to create a similarity function that the mechanism will use to find the best case that fits each time. The similarity function will define the best case that looks alike to this of the NEM given that it respects some distance measures. All these above can be positioned in the UMF and do not need any other information but only NEM's information (most of them in the manifest) and some information about the network which can be taken by the simulator.

5 Results

In the beginning, there is map, from a real location, which is separated into cells (figure 5.1). Each cell is separated into six territories as it is shown in figure 5.2. Territory 0 is the territory which consists of all the other sub-territories. Territories 2, 3, 4 and 5 are located around the femtos and territory 1 is around the central antenna. Each one of these territories is a rectangular, but the territory around the central antenna takes over twice the surface than the others. The network is going to work over one of these cells.

The core parameters of the network are the topology of the antennas and the femtos, and the density of the users in each territory of the cell. We decided to keep a fixed topology in each simulation and change the density of the users. The fixed topology consists of a central antenna in the middle of the cell and 4 femtos in the corners of an imaginary square in the cell. The density of the users is computed by a rate which means how many users appear in a territory each second. Therefore, this rate is a 1 x 6 array which is filled with integer or decimal numbers between 0 and 3 (for example: uar = [0, 1, 2, 0, 3, 1]). The users arrival rate (uar) is used to measure the

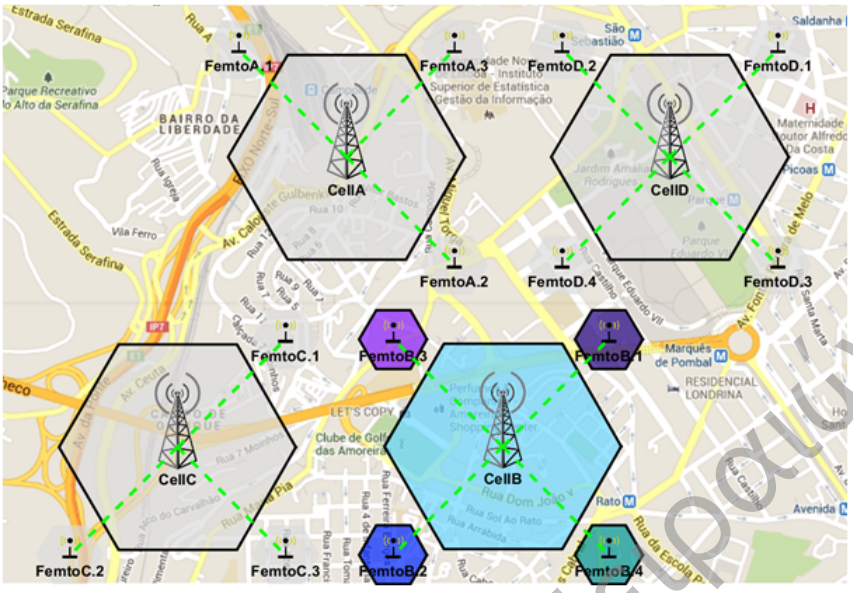

Figure 5.1: The map including the cells, the antennas and the femtos.

number of arrivals (na) each second by the following equation:

$$na = sum(rand(1, 100) < uar/100)$$

which statistically has been proved that na is approximately equal to uar. As a result, each second some users are going to be normally distributed with certain positions in the territories that are mentioned before.

Thus, we should compute all the possible combinations of users' densities for the forthcoming simulations. In this computation, we should take into account the symmetries between the femtos, because each one has the same influence in the network because of the symmetric positions that they are located. For example, if we observe the arrays A = [1, 1, 1, 0, 0, 0], B = [1, 1, 0, 1, 0, 0], C = [1, 1, 0, 0, 1, 0] and D = [1, 1, 0, 0, 0, 1], we can notice that they are equivalent. The last four digits are the arrival rates for the territories around the femtos and the first two digits are always the same. As a consequence, we can understand that we do not need all the possible

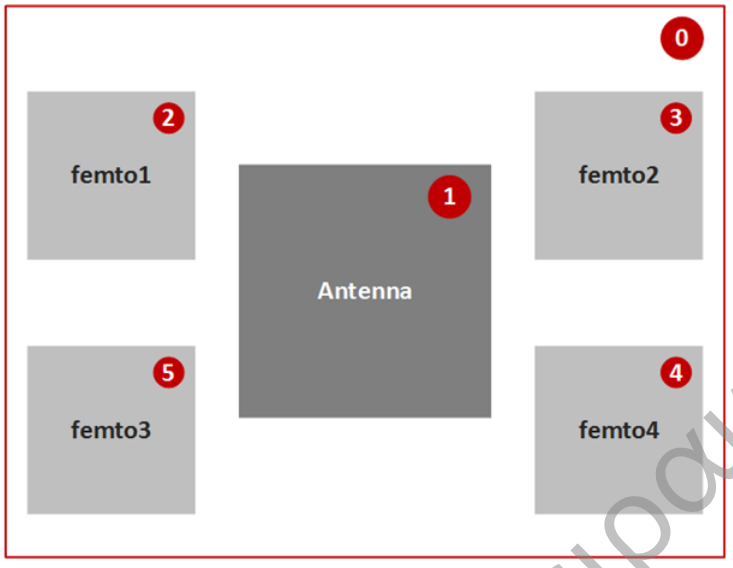

Figure 5.2: The different territories in the cell.

users' configurations but we should remove all the equivalent. Therefore, supposing that decimal numbers won' be used, we computed all the possible combinations for numbers 0, 1, 2 and 3. By applying the mathematical laws of combinations and permutations we found that we needed 559 users combinations and, as a result, 559 different simulations. Unfortunately, we had to wait for about over a month, for the simulations to finish, in order to be able to fill our CB. So, we decided to run only a small (16 configurations) but illustrative part of the possible combinations, in order to be capable to present some tangible results of this work.

As it was mentioned in the previous chapters, in our work, we will use, for our simulations the SON load balancing NEM. Load balancing is an important mechanism for resource management. It distributes the traffic load and resources between different network entities in a manner that responds to a certain objective such as the optimization of the usage of network resources. Furthermore, load balancing is an online traffic engineering technique able

LOAD BALANCING NEM	
Inputs	Load of the Antennas
Actions/Solutions	Set of antenna powers
Used Function	$P_i(t+1) = P_i(t) * (1 - a(L_i(t)L_0(t)))$
Utility Function	$(\mid L_i L_0 \mid) or \sum(\mid L_i L_0 \mid)/4$

Table 5.1: The trust evaluation value of the NEM in contrast to this of the reference algorithm.

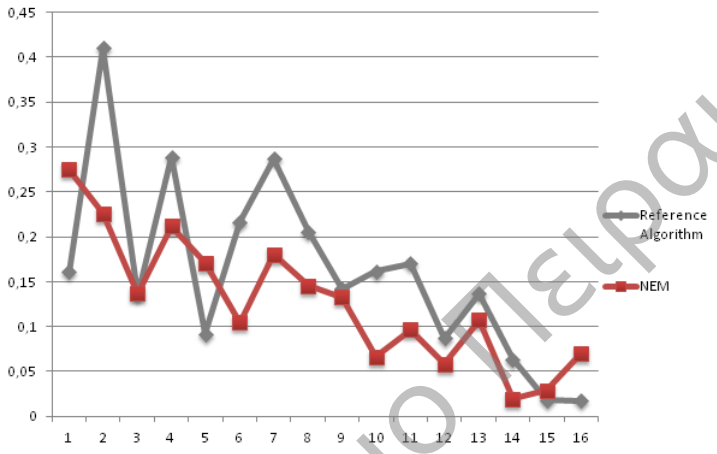

Figure 5.3: The trust evaluation value of the NEM in contrast to this of the reference algorithm.

to dynamically react to real time events. In autonomic networking concept, load balancing is realized by autonomic control loops to optimize the traffic amount sent in the network and to enhance the network ability to satisfy the future demands. The input of our NEM is the load of the antenna, the new set of the antennas' powers constitute the actions of the NEM and the used function as well as the utility function can also be seen in the table 5.1.

Firstly, we will present the instantaneous trust index (ITI) of the NEM for the 16 simulations in the figures that follow. Moreover, we will present a specific case that a deliberate error will be inserted in the code of the NEM in order to observe the differences in the ITI and, finally, we will discuss the results.

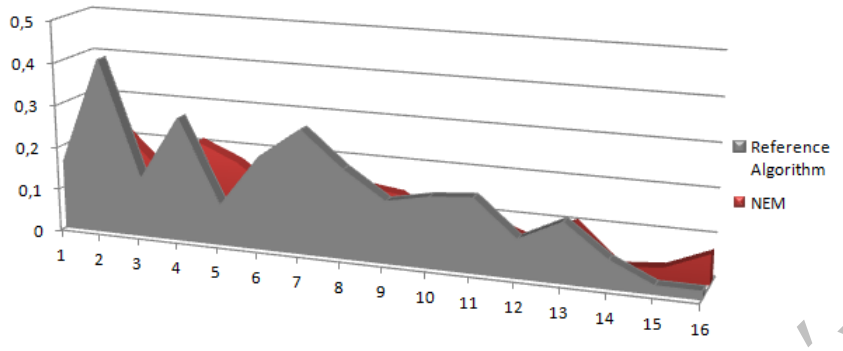

Figure 5.4: Comparison on the decisions of the NEM in contrast to these of the reference algorithm.

In figure 5.3, we can see a graph providing the evaluation of the decisions both of the NEM and the reference algorithm for the sixteen chosen users' configurations concerning the KPI of the load balancing. From 1 to 16 we add users to the users' configurations and as a result the users become more uniformly distributed to every cell (for example users' configuration 1 is represented in the code by the array [0, 0, 1, 0, 0, 0] while the configuration 16 is represented by the array [1, 1, 1, 1, 1, 1]). Therefore, our first observation is that the decisions both of the NEM and the reference algorithm, and as a result the evaluation of the decisions, are getting better while users are added in each territory of the cell. Secondly, our reference algorithm seems to be quite good (it could be even better if we had more available time for simulations to increase the number of random choices, of crossovers and mutations of the genetic algorithm) given that the two graphs follow each other. Most of the times NEM has better decisions but sometimes our algorithm seems to be more efficient as we can see from figure 5.4.

In figure 5.5, we present the ITI of the NEM comparing its decisions to these of the reference algorithm. The ITI is computed by subtracting the trust evaluation value of 1. That means that when ITI has the highest value, the

Figure 5.5: Trust index of the NEM.

NEM is most trustworthy and as the ITI decreases, the NEM becomes less trustworthy. As a first conclusion, the linearity of the graph and the fact that ITI presents very high values, near or equal to 1, reveal the stability of the decisions of the NEM.

In the next set of experiments, we intentionally inserted an anomaly within the functioning of the NEM. Figure 5.6 provides the comparison between the evaluation of the decisions of the NEM in contrast to these of the reference algorithm. It is obvious, that the faulty NEM takes a lot worse decisions than before due to the deliberate inserted anomaly, but this inefficiency and lack of trustworthiness of the NEM is identified by our mechanism.

As a result, NEM's decisions, in this experiment, are always worse, in terms of evaluation, than these of the reference algorithm, in contrast to the previous experiment, as it can be seen in figure 5.7.

Figure 5.8 provides the obtained results in terms of ITI evolution. The decrease of the NEMs performances, clearly depicted by the curve, allows the

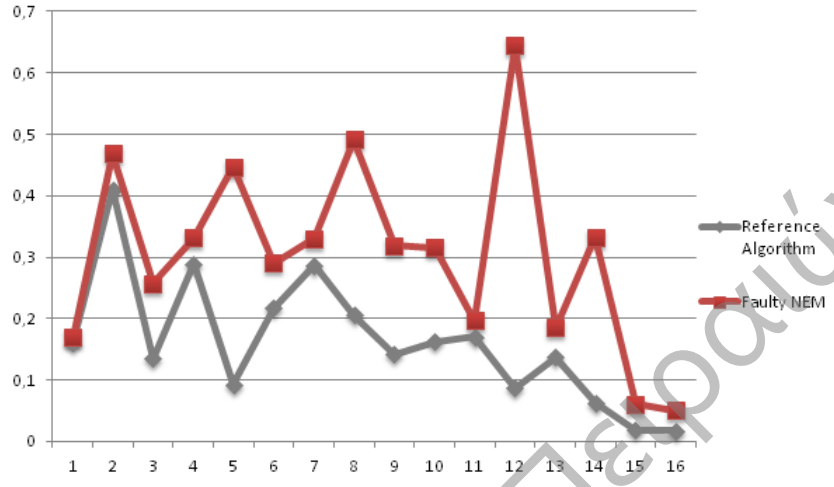

Figure 5.6: The trust evaluation value of the NEM in contrast to this of the reference algorithm.

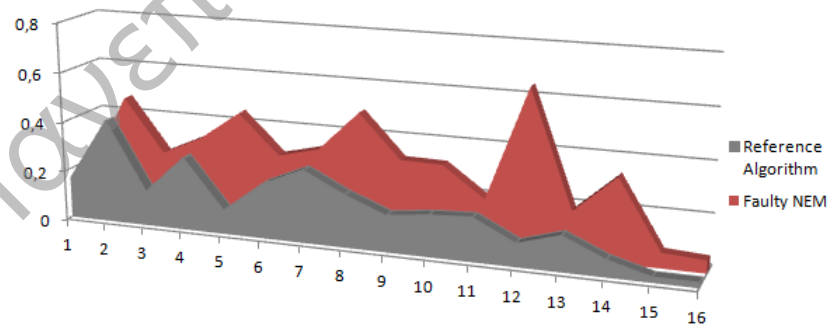

Figure 5.7: Comparison on the decisions of the NEM in contrast to these of the reference algorithm.

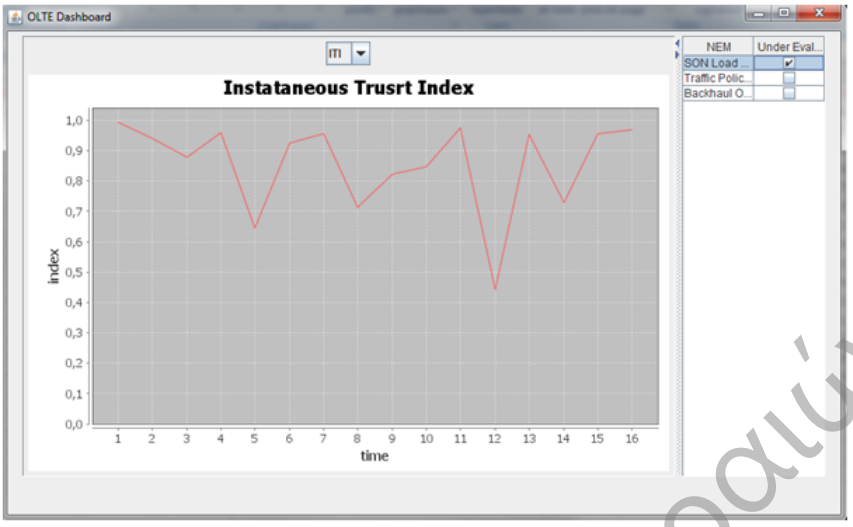

Figure 5.8: Trust index of the faulty NEM.

operator to detect such deterioration and then to diagnose the malfunctioning in order to decide on applying corrective actions or stopping the NEM.

Finally, figure 5.9 presents the distance between the ITI of the first and the second set of experiments, where we can see all these that are aforementioned such as the stability, the robustness and the efficiency that the first NEM presents in contrast to the faulty NEM. As supporting evidence, we computed some extra metrics to show the differences between the two graphs. The elements of the red graph have an average of 0.9326 and a standard deviation of 0.0484 in contrast to the blue graph, which in each one of these metrics has lower values (0.8567 and 0.1526 respectively). Moreover, the medians (which is the middle number of a group of numbers; that is, half the numbers have values that are greater than the median, and half the numbers have values that are less than the median) of the elements of the red and the blue graph are equal to 0.9330 and 0.9340 respectively. Taking into consideration that, the average and even the median (higher than the

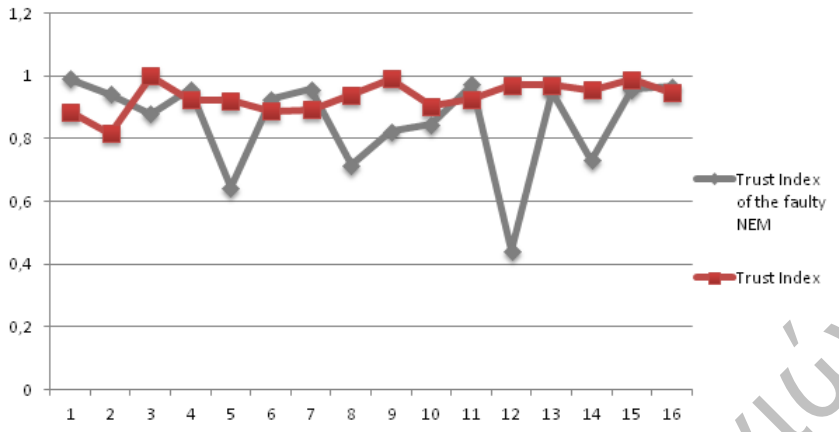

Figure 5.9: Comparison between the trust indices.

red graph) of the blue graph is quite high but the standard deviation of the elements is very high too, we can conclude that the number of elements that are low for both graphs are around the same but a percentage of these of the blue graph are really low creating some dangerous curves causing instability and lack of trustworthiness.

To sum up, we made a demonstration, through two sets of experiments, showing that our proposed mechanism works properly. The presented experiments are just a slight percentage of what is needed to analyze it in full depth, because of the lack of time, but the results are quite promising. Moreover, we strongly believe that, some small anomalies in the linearity of the trust index, due to the malfunctioning of the reference algorithm, could be corrected if there was more available time for the simulations in order to enhance the solutions of the genetic algorithm with more repetitions.

6 Conclusion and Future Work

In this work, we presented an approach to dynamically compute a trust level based in reliability and performances to judge the behaviour of autonomic control loops in autonomic networking. Our model is based on the computation of a trust index that reflects the trustworthiness of an entity. We used a genetic algorithm, as a reference algorithm, which has been proved to be very efficient for optimization problems; we filled a static case base with similar cases to these of a NEM and compared them using a mechanism that is UMF compliant. Clearly, the reported work is limited in many respects, but we tried to implement a first approach of the online trust evaluation fully integrated to the UMF which is a novel research domain. We plan to apply our evaluation mechanism to more use cases than SON load balancing NEM only and for any topology of the antennas, and to continue this work in both simulation and prototyping directions. However we believe that this is a promising direction due to the following achieved benefits.

We demonstrated some cases concerning the evaluation of the SON load balancing NEM but we also gave some guidelines in which way our work can be used evaluating other types of NEMs too. We also showed clearly

that this mechanism can be easily located into the UMF and does not need any other information than this that UMF can provide.

In the future, it would be very fruitful if some procedures before and after the mechanism could be completed in order to transform all the evaluation part to automatic. Soon, it could be defined clearly how exactly NEMs are going to be chosen to be evaluated, which will be exactly the conditions that influence the duration of the evaluation, what should happen after the evaluation (call for Gov) and in which extent human operator should become involved. The successful translation of high level to low level policies is of high importance from the operator's point of view. A successful policy will lead to well controlled and efficient network operations, while an unsuccessful policy may lead to misconfigurations, QoS / QoE degradation and network instabilities.

These achievements we plan to bring to the design of a certification process that shall guide the entire life-cycle of future network equipment, and similar to ISO quality standards shall concentrate on process and on their unified descriptions. Firstly, a standardized classification of NEMs is needed in order to proceed to more automatic, generic, reliable and trustworthy evaluation in a greater range of automatic components that, in the end, could be certified.

Bibliography

[1] Anis Jallouz, S. G.-D. (2013). *A Case-Based Reasoning Approach to Trust Evaluation in Autonomic Networks* . France: Alcatel-Lucent, Bell Labs.

[2] Chervany, D. H. *Trust and Distrust Definitions: One Bite at a Time.*

[3] Golembiewski, R. T. (1975). The Centrality of Interpersonal Trust in Group Processes. *Theories of Group Processes.* London: John Wiley & Sons.

[4] I. Shuaib, R. A. (2011)The Seventh International Conference on Autonomic and Autonomous Systems. *ICAS.*

[5] Karmouch, N. S. (n.d.). Towards autonomic network management: an analysis of current and future research directions. *Communication Surveys Tutorials.*

[6] Lewis, J. D. (1985). Trust as a Social Reality. *Social Forces* .

[7] Luciano B., M. B. (2006). Towards Pervasive Supervision for Autonomic Systems. *Proceedings of the IEEE Workshop on Distributed Intelligent*

Systems: Collective Intelligence Its Applications.

[8] Tanja Azderska, B. J.-B. (2012). A Holistic Approach for Designing Human-Centric Trust Systems. *In Science + Bussiness Media.* New York: Springer.

[9] Tariq King, D. B. (2007). Towards Self-Testing in Autonomic Computing Systems. *Proceedings of the Eighth International Symposium on Autonomous Decentralized Systems.* Arizona, USA.

[10] Tariq M. King, A. R. (2008). A Reusable Object-Oriented Design to Support Self-Testable Autonomic Software. *Proceedings of the ACM Symposium on Applied Computing.* Fortaleza, Ceara, Brazil.

[11] Thaddeus O. Eze, R. J. (2011). The Challenge of Validation for Autonomic and Self-Managing Systems. *The Seventh International Conference on Autonomic and Autonomous Systems .*

[12] ALBLF. (2013). Unified Management Framework Specifications(D2.4). Paris.

[13] 4G Americas. (2013). Self-Optimizing Networks: The Benefits of SON in LTE.

Abbreviations

AC	Autonomic Computing
AN	Autonomic Networking
AS	Autonomic System
CB	Case Base
CBR	Case Based Reasoning Approach
GA	Genetic Algorithm
IT	Information Technology
ITI	Instantaneous Trust Index
KPI	Key Performance Indicator
LB	Load Balancing NEM
LoA	Level of Autonomicity
NEM	Network Empowerment Mechanism
OLTE	On Line Trust Evaluation
QoE	Quality of Experience
QoE	Quality of Service
SD	Security Demand
SON	Self-Organizing Network
TV	Trust Value
UMF	Unified Management Framework