University of Pierre et Marie Curie – Paris 6
UNIVERSITAS

THESIS

Specialization

Networks and Telecommunications

Submitted for the degree of
Doctor of Philosophy

By

Miss Zeinab Movahedi

Title:

An Autonomic Architecture for Wireless Networks: Proposition and Evaluation Methodology

Defence: 25 November 2011

Committee

Francine Krief	Reviewer	Professor, ENSEIRIB
Otto Carlos Muniz Bandeira Duarte	Reviewer	Professor, Brazil
Yacine Ghamri-Doudane	Examiner	HDR, ENSIIE
Marcelo Dias de Amorim	Examiner	HDR, CNRS Research Director, UPMC
Guy Pujolle	Supervisor	Professor, UPMC
Rami Langar	Advisor	Associate Professor, UPMC

To my beloved family and my beloved friends,

each of you has a special place in my heart and my memory forever

Abstract

Autonomic network management is a promising approach to reduce the cost and the complexity of managing network infrastructures. It relates to the capability of the network to fulfill self-configuration, self-optimization, self-protection and self-healing functionalities. To achieve these properties, a new network management solution based on autonomic architecture is paramount. The objective of this thesis consists in providing an autonomic architecture for wireless mobile networks which is able to self-adapt according to their dynamic context.

To achieve this, we proposed SADA, a self-adaptive autonomic architecture for wireless mobile networks. SADA presents two new autonomic features: first, it enriches the IBM reference autonomic model exploited by existing architectures with self-adaptation capabilities. Second, it defines a second autonomic control loop around the monitoring component, allowing it to self-adapt according to the network context. The proposed SADA architecture provides the self-adaptation features based on a learning mechanism which uses the experienced outcome of its decisions to enhance its operation. Moreover, the SADA architecture uses a self-adaptive monitoring approach which provides required information for adaptation algorithms with a minimum extra overhead. As a case of application, SADA has been employed in routing service, resulting in the proposal of a self-adaptive routing scheme. Simulation results showed that SADA improves significantly the network performance in terms of packet delivery ratio, average end-to-end delay and routing overhead compared to the classical IP-based architecture.

Moreover, we applied the SADA architecture to design a self-adaptive knowledge monitoring scheme for autonomic network trust management. The objective of this case study was to investigate the effectiveness of the autonomic monitoring mechanism proposed by the SADA architecture.

Lastly, we proposed a quantitative evaluation methodology to evaluate and compare the efficiency of autonomic architectures from the autonomicity standpoint. To show the applicability of our methodology, we considered two use cases in two different contexts: (i) a wireless ad-hoc network using SADA architecture, and (ii) an infrastructure-based wireless network. Results showed that the proposed SADA architecture constitutes an efficient autonomic solution, characterized by its distributed intelligence, learning capability, self-adaptation features, low complexity, light monitoring, high quality of knowledge, security support and evolvability.

Key Words:

Self-adaptation, Autonomic Architecture, Architecture Evaluation, Wireless Networks.

Table of contents

Chapter 1

Introduction

Contents

1.1 Context

Over the recent years, computing and communication technologies have significantly evolved in both wired and wireless environments, enabling a wide range of applications and services for hundreds of millions of users connected to the network. This Information Technology (IT) industry's exploitation of the technologies has led to the verge of a complexity crisis. Software developers have fully exploited a four to six orders of magnitude increase in computational power, producing ever more sophisticated software applications and environments. There has been an exponential growth in the number and variety of systems and components. The value of database technology and the Internet has fueled significant growth in storage subsystems to hold gigantesque structured and unstructured information. Networks have interconnected the distributed, heterogeneous systems of the IT industry. Today's information society creates unpredictable and highly variable workloads on those networked systems[1].

This evolution has complicated the management of current and emerging computing technologies, making them brittle regarding crashes caused by hardware failures, defective

[1]Extract from IBM System Journal [1]

design, or human errors. Crashes in the computer infrastructure can cause significant economic losses. In the year 2000, it was estimated that a one-hour downtime in a brokerage operations system could amount to up to 6.5 million dollars loss in revenues [2]. Not only the failures themselves may cause losses, but the time required to fix them is important. What is worse, due to the pressure to quickly restore operations, humans may introduce configuration errors that may lead to a future failure. As a reaction to this scenario, autonomic computing has been proposed as an emerging solution [1].

The essence of autonomic computing is to develop self-managing computing systems [3] that manage themselves given high-level objectives from administrators. That is, the system should be able to identify important events in its environment and react to them, without any sort of external or human intervention. Those systems, in general, also learn from past experiences, improving their reactions over time. According to this vision, IBM defined four self-management properties for autonomic systems: they should automatically configure, heal, protect and optimize themselves without external intervention [1,4,5]. Self-properties are achieved through key properties of automaticity, adaptivity and awareness, which consist of preemptive and proactive cross-approaches to several areas of computing systems. By removing humans from the decision-making process, changes would occur in most optimized manner and would be less prone to errors [4]. Even more, the reduced need for human intervention will optimize the operational costs of the systems. Autonomic computing has been recognized as a step forward in computing, and as such several computer societies and research agencies consider it as one of the greatest computing challenges for the years to come [6–8].

In this thesis, we propose an autonomic architecture able to self-adapt its operation according to network conditions. More specifically, we delve into the self-adapting aspects of wireless mobile networks. Self-adapting refers to the ability of the system to perform adaptation operations using its internal knowledge to decide why, when, where and how adaptation is performed, without any external intervention in the decision making process. Self-adaptation is a paramount property to enable the four aforementioned self-properties of an autonomic system in an optimized manner according to the network context.

Wireless mobile networks are characterized by dynamic topology, shared wireless medium, resource scarcity, low physical security of nodes, just to mention a few. Due to these characteristics, self-adaptation consists in a very important feature for wireless mobile networks, enabling to cope with the dynamicity of the underlying network and to optimize scarce networking resources.

Current network adaptation approaches mainly utilize configuration policies or rules [9–14]. Although the straightforwardness of policy-based approach, the need for low-level configuration details does not meet the ultimate objective of autonomic computing. Further,

static policy configurations built a-priori into network devices lack the flexibility and may not be sufficient to handle different changes in these underlying environments. In general, utilized knowledge in policy-based adaptation approaches is limited to the hard-coded conditions using state variables, which are prone to administrator errors leading to policies conflict, un-optimized or instable network. If the system internal or external context changes in a way that is not suitable for the pre-programmed adaptation, external control is required to adjust the existing policies. Recently, a goal-based approach to policy refinement has been introduced in [15, 16] where low-level actions are selected to satisfy a high-level goal using inference and concepts of the event-calculus theory. Although this approach removes some limitations of the policy-based adaptation, it only attempts to lead the system to a desirable state. However, when the system reaches a desirable state, it would not try to improve its performance anymore. Moreover, when a desirable state cannot be reached, the system does not know which among the undesirable states is least bad.

Within the Unity project [17], utility functions are used to arbitrate resource allocation among different application environments as a result of solving an optimization problem. One problem in viewing adaptation of autonomic architectures as an optimization problem or a utility function is that, in general, most mathematical programming approaches deal with static, non-changing, and well defined problems and lack the ability to continuously adapt the problem formulation to represent the changing environment. Furthermore, knowledge in such approaches is limited to either current configurations or time series of one or more variables. These approaches are limited in their coping range where sudden changes in any network component can cause the failure of the applied adaptation strategy. With the cognitive architecture proposed in [18], random normal distribution is used to allows the dynamic reconfiguration of main protocol stack parameters for achieving some performance goals driven by target quality metrics. The main limitation of this approach is that it is only applicable for adjusting parameters taking values in a continuous interval. Further, it relies on the use of random generators which incur higher computational costs [19].

To address the aforementioned issues of current autonomic self-adaptation mechanisms, we propose SADA, a Self-Adaptive Autonomic Architecture for wireless mobile networks. The main objective of SADA is to present an autonomic management architecture which is able to learn from past experiences to enhance future operations. This intelligent adaptation is mandatory to converge to an optimal network operation, as it is one of the key self-management properties.

1.2 Objectives of the thesis

The objective of this thesis consists in providing an autonomic architecture for wireless mobile networks which is able to self-adapt according to their dynamic context. Albeit there are some autonomic architectures for or applicable to wireless mobile networks, they are not self-adaptive and rely mainly on a set of predefined static policies which make no use of previous experiences to enhance future operations. Moreover, those works do not take into account the particularities of wireless mobile networks such as their requirements and characteristics in terms of resource constraints and lack of a central control. Particulary, they rely on static resource-consuming monitoring approaches to feed the adaptation algorithms.

This work presents a general overview of existing autonomic architectures, allowing us to identify issues existing in current autonomic solutions to derive a new autonomic architecture. Mainly, we have identified the lack of learning mechanisms, security and an appropriate and self-adaptive monitoring scheme for autonomic networks. Based on the survey, we design the SADA architecture, whose goal is to add self-adaptive features to the network management. As showcases, SADA is employed on routing and trust management of wireless mobile ad hoc networks (MANETs). Lastly, we propose a quantitative evaluation methodology to evaluate and compare the efficiency of autonomic architectures from the autonomicity standpoint. To show the applicability of our methodology, we consider two use cases in two different contexts: (i) a wireless ad-hoc network using the SADA architecture, and (ii) an infrastructure-based wireless network.

1.3 Contributions

We summarize the contributions of this thesis as follows:

- **A survey of autonomic network architectures.** The survey provides a coherent classification of the existing autonomic architectures and overviews these proposals. Moreover, it presents a qualitative comparison between those architectures, emphasizing the relevance of our contribution. As a result, the survey allows us to identify open issues and opportunities of current autonomic proposals to derive a new self-adaptive autonomic architecture according to the requirements of wireless mobile networks.

- **SADA: Self-Adaptive Autonomic Architecture for wireless mobile networks.** Assessing existing autonomic architectures, we design a self-adaptive architecture for mobile wireless networks, called SADA. SADA presents two new autonomic features: first, it enriches the IBM reference autonomic model exploited by existing

architectures with self-adaptation capabilities. Second, it defines a second autonomic control loop around the monitoring component, allowing it to be self-adapted according to the network context. Regarding the first contribution, the proposed SADA architecture uses a learning mechanism based on random neural networks with reinforcement learning inspired by biological neurones. For the second contribution, a self-adaptive monitoring approach is proposed which uses normal transiting packets in the network to provide required information for adaptation algorithms with a minimum extra overhead. Although proposed for wireless mobile networks, this architecture can be used in other types of networks due to its generic model and its meager resource requirements.

- **SRS: Self-adaptive Routing Scheme for mobile ad hoc networks.** SADA has been employed in routing service resulting in a self-adaptive routing scheme. The proposed scheme is independent of any routing protocol and consists in self-adapting the route discovery process of mobile ad hoc networks with a minimum routing overhead. The routing self-adaptation is achieved based on random neural networks which adapt the path selection process based on recent experienced network performance. The effectiveness of the SADA-based SRS scheme is demonstrated by its performance comparison against classical IP-based routing schemes in the context of MANETs.

- **ATMS: Autonomic Trust Monitoring Scheme for mobile ad hoc networks.** Based on the open issue of security identified in our survey contribution, we applied the SADA architecture to design a self-adaptive knowledge monitoring scheme for autonomic network trust management. The main goal of ATMS is to provide an uniform up-to-date trust knowledge throughout the network with a minimum monitoring overhead, reducing the impact of double-face attacks. ATMS is characterized by its protocol and trust framework independence, self-adaptation, simplicity and low computational intensiveness. The effectiveness of the SADA-based ATMS scheme is demonstrated by its comparison in terms of network performance and knowledge quality metrics with a non-adaptive trust monitoring scheme in the context of MANETs.

- **A methodology for quantitative evaluation and comparison of autonomic architectures.** This contribution consists in two propositions: first, we identified main quantitative evaluation criteria contributing to the efficiency of autonomic architectures. To allow the straightforward applicability of the identified evaluation criteria in different network contexts, a measurement technique has been proposed for each of the proposed evaluation criteria. Second, we proposed a quantitative methodology for evaluating and comparing autonomic network architectures in an

unified manner. Our methodology is based on fuzzy-logic and correlates the identified criteria to obtain an overall measure of autonomicity. This autonomicity score can be used to compare quantitatively different existing autonomic architectures and classifies them approximatively regarding to the network management efficiency and performance.

- **Evaluation of autonomicity of a wireless ad-hoc network and an infrastructure-based wireless network.** To show the applicability of the proposed methodology, two use cases have been considered in two different contexts: (i) a multi-hop wireless ad-hoc network, and (ii) an infrastructure-based wireless network. For the first use case, the proposed self-adaptive routing scheme based on SADA architecture is considered. As a case of application of our methodology in infrastructure-based networks, we considered AHOPS, which consists in an autonomic handover piloting system for wireless heterogenous networks. For each use case, we considered a set of reference scenarios, evaluating the efficiency of the architecture in terms of individual evaluation criteria as well as the overall degree of autonomicity in different network contexts.

1.4 Thesis Organization

The remaining of this thesis is organized as follows. Chapter 2 provides an overview of autonomic network architectures as well as a comparison of the existing autonomic proposals. Its objective is to draw the characteristics of existing autonomic network architectures, emphasizing the open issues unsolved in current solutions.

Chapter 3 presents SADA, a self-adaptive autonomic architecture for wireless mobile networks. Further, is details the modules of the SADA architecture and emphasizes the design of its components. As well, is describes our proposal of the self-adaptive routing scheme based on the SADA architecture. Finally, it presents the evaluation environment and discuss the achieved results.

Chapter 4 presents our proposal of the autonomic knowledge monitoring scheme for trust management in mobile ad hoc networks. First, it presents some motivations on the mutual impact of the autonomic architecture and the trust management. Moreover, it presents the components of our autonomic trust monitoring scheme, emphasizing its knowledge management strategy. Finally, it details the simulation environment, used metrics and results of the evaluation.

Chapter 5 presents our proposition of evaluation and comparison of autonomic network architectures. First, it presents the identified evaluation criteria characterizing the autonomicity of the architecture. Second, it describes our proposition of the quantitative

methodology for evaluating and comparing autonomic network architectures.

Chapter 6 describes the autonomicity evaluation of an infrastructure-based and a wireless ad hoc network as two case studies of application of our evaluation criteria and methodology. Further, it analyzes the characteristics and assets of the SADA architecture, compared to the current state of the research.

Finally, chapter 7 concludes the thesis. It recapitalizes the contributions of the thesis and point out perspectives and directions we envision for proceeding with the research.

Chapter 2

Autonomic Network Architectures

Contents

2.1 Introduction

This chapter provides some background on autonomic communication and its main properties. It presents a coherent classification of existing autonomic architectures and overviews these proposals, providing a holistic view on the current state of the research. Moreover, it identifies a set of qualitative evaluation metrics and uses them to compare qualitatively the surveyed architectures.

The remaining of this chapter is organized in four sections as follows. Section 2.2 provides a general introduction to autonomic computing and gives some background. Section 2.3 presents a coherent methodology to classify the autonomic network architectures and provides an overview of existing architectural solutions. Section 2.4 presents some qualitative evaluation metrics important for characterizing autonomic network architectures. It is followed by a comparison of surveyed architectures based on identified metrics. Finally, section 2.5 concludes the chapter.

2.2 Autonomic Computing

The autonomic paradigm proposes to tame the increasing complexity of managing computing systems by the incorporation of self-management functions [1]. According to this paradigm, systems should operate unattended, automatically adjusting their properties to accommodate any eventual change on the environment. This paradigm is inspired from the autonomous nervous system of the human body, which checks blood sugar level and maintains normal body temperature without any conscious effort [20].

The term of autonomicity has been the subject of misapplying in IT community. Above all, the automaticity has been amiss taken more often for autonomicity. Following, we provide the clear definition of each term:

- *Automaticity* [21] refers to the ability of performing one or more tasks without any manual intervention or external help. It does not include the performance optimization issues to better fit some sort of performance goals. As an example, a queue scheduling engine carries out the automatic execution of the scheduling algorithm for different types of packets.

- *Autonomicity* [20] is the art of self-managing given a set of high-level objectives from administrators. High-level objectives define for a system what are its goals and the system attempts to accomplish them in the best manner. Consequently, an autonomic system is able to monitor its own performance and adapt itself accordingly, optimize its use of resources and overcome occurred events. This is what distinguishes the autonomic system from its automatic counterpart.

The autonomicity implies a set of well-known properties and characteristics that are outlined in next sections.

2.2.1 Self-management properties

The main properties of self-management as portrayed by IBM are described in the following [3, 22]:

- **Self-configuring**: This property refers to the capacity of the system to configure and reconfigure itself in accordance with high-level objectives in a changing environment. It involves the ability of both the new component and the system to instal, configure and integrate when a new component is introduced to the system. The component would be able to incorporate itself seamlessly, and, the system to adapt itself to its presence. The end system could then make use of this component normally or modify its behavior accordingly.

- **Self-optimizing**: The objective of self-optimization is to enable efficient operation of the system even in unpredictable environments. An autonomic computing system will proactively seek opportunities to make itself more efficient in performance and cost. For this, the system should be aware of its ideal performance, measure its current performance against the ideal and have strategies for attempting improvements.

- **Self-healing**: This property consists in the capacity of discovering and repairing potential problems to ensure that the system runs smoothly. It may be achieved by predicting problems and taking proactive actions either to prevent failures or to reduce their impact.

- **Self-protecting**: Self-protection designates the ability of the system to protect itself from what compromises it from achieving its goals. It involves the protection from malicious attacks, intrusion tentative or inadvertent failures [9][13].

2.2.2 Autonomic networks: the conceptual architecture

Since network management becomes more and more distributed, the autonomic computing is essential to keep such systems manageable. The term of autonomic networking is used to denote the act of applying autonomic principles to the management of networks.

By definition, an autonomic network operates and serves its objectives optimally by managing its own self without any external intervention even in case of environmental changes [21]. Autonomicity may be achieved via a distributed architecture which is made of the interactive collection of a set of autonomic managers coupled with managed components. An autonomic manager is an individual self-managed element which contributes to

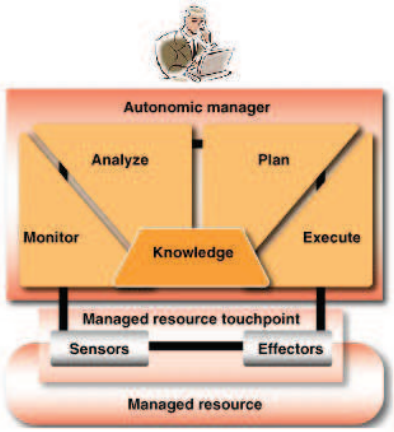


Figure 2.1: IBM's MAPE-K reference model for the ACL

the overall self-management of the network. It is able to percept its internal and external environment and act locally on its assigned managed components. An autonomic manager interacts with other pairs in order to keep the entire network performing in accordance with high-level objectives. The managed component represents any hardware or software resource that is given autonomic behavior by coupling it with an autonomic manager [5]. A storage disk, a CPU, a database, a wired or wireless network, a router queue, a fault localization service are examples of some managed components.

The autonomic manager achieves the self-management properties thanks to monitoring the managed component(s) and other autonomic managers, analyzing the collected data, planing adaptation tasks and executing the decided plans on the network. This process forms an *autonomic control loop* (ACL) [20] around network resources. IBM proposed a generic reference model for ACL called MAPE-K model (Monitor, Analyze, Plan, Execute and Knowledge). This model, as depicted in Figure 2.1, provides a generic vision of basic components of an autonomic manager. Hence, it is widely being used to communicate the architectural aspects of autonomic systems [5]. As illustrated in Figure 2.1, the administrator is situated out of the ACL, dealing only with defining the high-level objectives for ACL guidance. The software or hardware components used to perform monitoring and carrying out changes from/to the network are respectively called sensors and effectors.

For clarification reason, a detailed conceptual view of MAPE-K model is shown in Figure 2.2 which consists of:

- A *high-level objective base* which is a repository where high-level objectives are stored. This repository serves for initial configuration of the network and guides the operation of other ACL components.

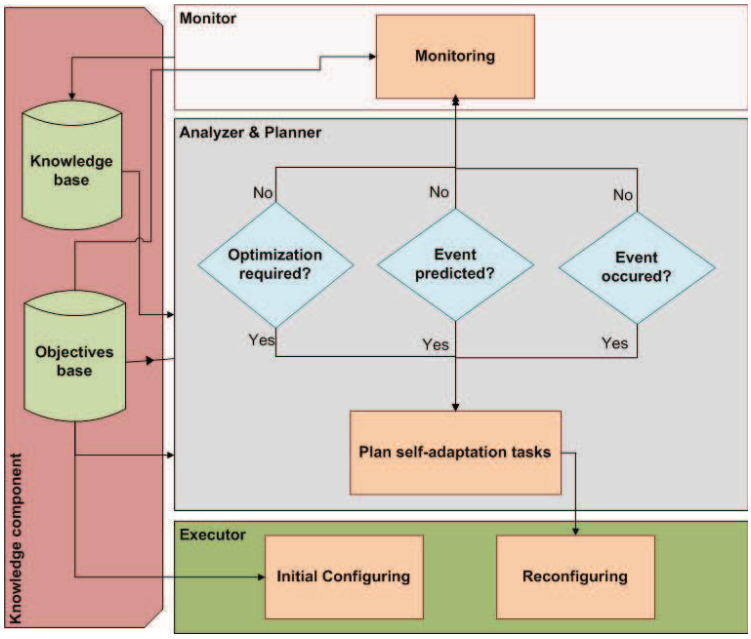


Figure 2.2: Detailed MAPE-K model for the ACL

- A *monitor* which is in charge of capturing necessary measurement of the environment that are of significance to the self-properties of the underlying network. This data is processed and filtered, producing relevant information that is used by the other components.

- A *knowledge base* where processed collected data are stored. The aggregated data are referred to as *Knowledge*.

- An *analyze component* which analyzes the knowledge based on high-level objectives. Accordingly, it verifies the current performance, predicts future state of the network and detects events.

- A *planner* which constructs adaptation plans based of the result of the analyze process. The adaptation may use techniques such as artificial intelligence, control theory, game theory or domain-specific algorithms. The actions of this component may be configured using abstractions such as policies.

- An *executor* which, based on the output of the plan process, reconfigures the managed component or communicates with other autonomic managers.

As illustrated in figure 2.2, the analyze and plan components are considered together

as their functionality are tightly linked. In the following, we use the *adaptation* term to refer to these two MAPE-K components.

An autonomic system can be closed-adaptive or open-adaptive [23]. An ANM architecture is closed-adaptive if the same adaptation strategy is applied repetitively in the same context regardless of whether the strategy was successful or not. A closed-adaptive system maintains a predefined number of adaptation strategies (e.g., predefined set of configuration policies for a router). Open-adaptive schemes continuously evolve and enhance their applied adaptation strategies. This latter implies the use of some artificial intelligence techniques, allowing the system to learn over time.

The adaptation may be triggered periodically or in response to external or internal events. No matter what approach is taken, a trade-off between reactivity and network resources consumption has to be found, taking into account the network state. In fact, more the network is dynamic, less reactive the adaptation should be, avoiding frequent adaptation oscillations. In a fairly dynamic environment, the network should be sufficiently sensitive to changes while maintaining minimum extra overhead. Moreover, the sensitivity to changes may depend to the period of time needed by adaptation engine to be able to resume an incoming situation.

The existing monitoring approaches used for autonomic networking rely either on a local view or on a global view of the network. The *local view* consists in a node-based local observation of the environment. It is generally obtained through a passive observation of the internal and local state or/and retrieved from a cross-layer engine [24–26]. However, a decision, even applied in the local environment, may influence the network-wide performances. Thus, autonomic managers should have a holistic view of the network. This network-wide view is referred to as the *network or global view*. It is built by collecting numerous samples of local views from various nodes within the network [27]. A node can utilize the network view to evaluate its own *relative* status [28]. This can be used for decision-making process as well as optimization of collaborative applications such as the choice of algorithms, load balancing, routing decisions, position estimation, energy management as well as self-management properties. However, collecting and maintaining network-wide global statistics can be expensive. Therefore, a monitoring approach should ensure a correct network view while maintaining a minimum extra overhead.

2.3 Autonomic network architectures

This section describes several initiatives on building autonomic network management (ANM) architectures. Despite many of surveyed proposals do not present all self-management properties, they have basic frameworks, characteristics and mechanisms enabling them to be

extended to those functionalities.

We categorize initiatives found in the literature into two main groups: *flat* and *hierarchical*. In the flat approach, autonomic managers are distributed in the network in a peer fashion. Hence, the architecture should define how these elements would cooperate to provide a convergent overall autonomicity. In hierarchical approach, a coordination management overlay is defined which aligns the operation of lower-levels autonomic managers. This classification describes two different visions of network management, each one having its particular characteristics and issues.

In the following, we first describe some important hierarchical autonomic architectures, followed by distributed flat approaches proposed in the literature. We describe each architecture regarding to the approach employed for each of its MAPE-K components. Moreover, we highlight and detail how its ACL performs and how the constituent autonomic managers interact and cooperate between each other since this considerably impacts the convergence of the solution and the overall system performance.

2.3.1 Hierarchical autonomic architectures

Compared to classical network management, the hierarchical approach defines several management levels, ranging from *manager level* to *managed one*. The manager level is responsible for decision making. The managed level executes the decided plan dictated by the manager level on network resources. In contrast with classical management approaches, hierarchical autonomic architectures consider a distributed manager level. They are more likely achieving the optimal network state since the network decisions are made in a level holding a holistic vision of network elements. In the following, we provide an overview of existing hierarchical network architectures.

2.3.1.1 AutoI project architecture

Autonomic Internet (AutoI) [29] is one of the FP7 European project dealing with the autonomic management of next-generation networks. The proposed AutoI architecture copes with two key aspects that differentiate it from other efforts [10]: First, it enables the cooperation of heterogeneous management domains. Second, it emphasizes on autonomic management of virtualized resources and services, allowing user mobility, service and component migration, programmability and extensibility.

AutoI proposed a two-level distributed hierarchical architecture: the lower-level deals with autonomic domain-wide management, i.e. a collection of entities subject to the same set of business goals, protocols and configurations. The high-level tier orchestrates autonomously between lower-level components. Also, this orchestration level enables autonomic management domains run by different operators or administrators to automatically

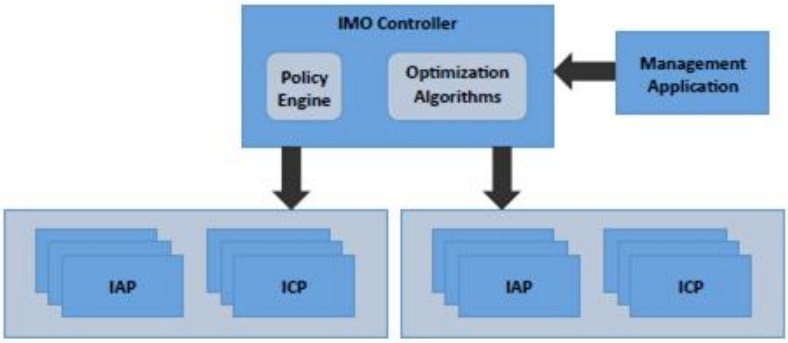


Figure 2.3: The AutoI IMO structure

adjust their configuration to accommodate the federation of networks [30].

Both orchestration and management levels use policy-based adaptation mechanisms to face with network changes. The orchestration level is planed to be enabled with some utility-based intelligence features.

Monitoring is performed by an adaptive *Information Management Overlay (IMO)* that acts as an enabler for self-management functionality. The IMO collects, processes, and disseminates information from/to the network entities and management applications [31]. As depicted in Figure 2.3, the IMO is composed of the *IMO controller* and a set of *IMO-nodes* placed dynamically in appropriate points of the network, forming a monitoring hierarchy. IMO-nodes are in charge of information retrieval and act as *information collection point (ICP)* or/and as *information aggregation point (IAP)*. The context information is collected by ICP and periodically sent to the nearest IAP node which processes, aggregates and transmits the data to a node that exploits this information (i.e. a management application). The IMO Controller is a policy-based centralized controller responsible for the setup and optimization of the overlay regarding the optimization requirements of management applications (e.g. CPU/memory, network resources, or response time). The optimization is performed based on policies applied on overlay optimization algorithms.

In order to simplify the analysis and design of management, the AutoI architecture is presented as a coordination of a set of planes [32, 33]. Each plane performs a set of management tasks, abstracting those management issues away from other planes. A conceptual view of AutoI plans is illustrated in Figure 2.4. They are defined as follows:

- The *Virtualization Plane* virtualizes physical resources, allowing the migration and on-the-fly reconfiguration of network resources. It abstracts all the virtualization issues away from other components of the architecture.

- The *Management Plane (MP)* deals with the maintenance and creation of individual

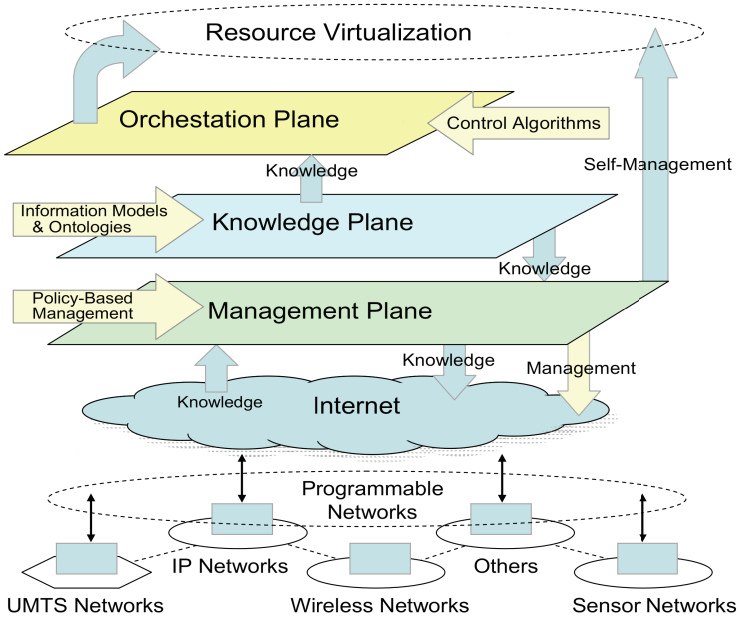


Figure 2.4: The AutoI Planes

ACLs. Those loops are realized by Autonomic Management Systems (AMSs), which perform the MAPE-K functions of an ACL. Each AMS represents an administrative and/or organizational boundary, called AMS domain, that manages a set of devices, subnetworks, or networks using a common set of policies and context.

- The *Service Enablers Plane* (SEP) is responsible for service deployment and composition. It uses the virtual resources of the virtualization plane to set up new services, such as a VPN, a file sharing service or a multimedia transport service.

- The *Knowledge Plane* implements a distributed information service, providing all the planes with their required information. It timely disseminates information for the other planes, and determines the three W's of information management: Which information is needed, from Who and When. This is performed following the Information Management Overlay described above. This information is modeled using a set of ontologies, specified in a common information model [34].

- The *Orchestration Plane* (OP) deals with the orchestration of multiple domains as well as the interaction of different AMSs and services. While the SEP and the MP deal with the management of a single service or control loop, respectively, the orchestration plane manages the interaction among several such components. Being intelligent components themselves, the orchestration plane acts as a mediator, detecting and

managing conflicts, determining SLAs that satisfy all involved parties and determining the need for (re-)deployment and re-location of AMSs and services. The proposed OP is implemented by the Distributed Orchestration Component (DOC), which acts as a middleware that provides interfaces to simplify the orchestration of AMSs. Federation, negotiation, distribution and governance behaviors define the interactions between DOCs and AMSs in a request/response manner.

The efficiency of the proposed architecture was tested under a fixed network testbed environment consisted of seven networked physical machines, carrying several virtual networks [35]. The results showed that the architecture performs well for creating and migrating on-demand virtual infrastructures. As well, the deployment of networking (eg. routing) and application (eg. Video-on-Demand) services over virtual AUTOI infrastructures have been successfully carried out. In addition, mid-scale and large-scale service deployment experiments have been performed. Regarding the mid-scale service deployment, the deployment of 90 Virtual Routers in 15 networks (interconnected by 75 Virtual Links), distributed over four physical machines were reliably supported by the testbed. Regarding large-scale service deployment, analysis are carried out over Grid5000 infrastructure[1]. Experiments showed that one hundred and ten service enabler middlewares (ANPI) were deployed on a virtual network built over one hundred and ten virtual routers on Grid5000 in an aggressive way (around 30 seconds). Prototypes and implementation of the AutoI architecture are already available online [36].

To summarize, the AutoI architecture constitutes a policy-based approach towards the autonomic networking. The main assets of the AutoI architecture are the heterogeneity management through the orchestration plane and the support for the virtualization. However, the scalability of its monitoring approach remains an issue since it uses a centralized IMO controller for optimizing the Information Management Overlay. Another limitation relies on using only policy-based adaptation which makes no use of learning mechanism and can lead to an un-optimized network. Moreover, the security aspect of the architecture in the underlying virtualized environment should be investigated.

2.3.1.2 DRAMA architecture

DRAMA (Dynamic Readdressing and Management for the Army) is a distributed policy-based network management for MANETs using intelligent agents [11]. The main purpose of DRAMA design is to set up a network that can autonomously adjust its behavior without human intervention [37]. This approach was especially designed for military networks consisting of wireless ad hoc routing domains connected by a wired backbone network.

[1]Grid5000 is an initiative from the French Ministry of Research through the ACI GRID incentive action, INRIA, CNRS and RENATER and other contributing partners (http://www.grid5000.fr).

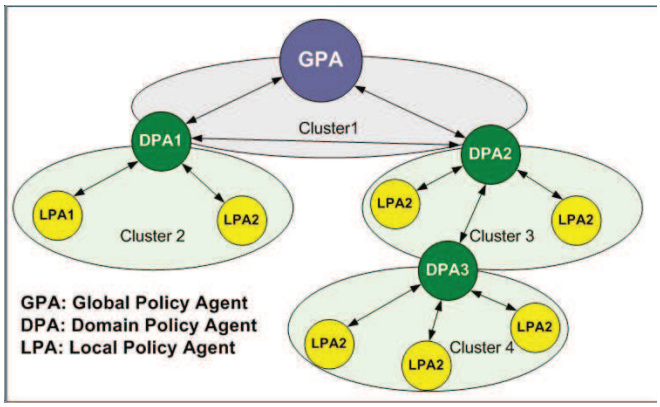


Figure 2.5: DRAMA management hierarchy

In such an environment, a hierarchical management approach is the most suitable since military systems are naturally organized into hierarchical authority domains. Besides, different domain managers are usually already known. The objective is to design a system which is dynamically organized into clusters or domains where a hierarchy of managers cooperate to manage the entire network. To achieve this, a set of Policy Agents with different roles are used.

As shown in Figure 2.5, the network is managed through a two tiers hierarchical architecture [11]. In the lowest level of the hierarchy, we find *LPA (Local Policy Agent)* which is responsible for individual node management. An instance of the LPA is placed on every node in the network. In the highest level of the hierarchy, we find *GPA (Global Policy Agent)* which is charged with the dissemination of policies and the management of multiple *DPAs (Domain Policy Agents)*. A DPA in turn, can manage multiple other DPAs and is responsible for domain management and LPAs aggregation.

The management hierarchy is especially used in order to disseminate policies and report monitoring information to other nodes in the hierarchy. Policies are specified by network administrators and reflect the commander's intention. *DRCP/DCDP protocols (Dynamic Registration and Configuration Protocol/Dynamic Configuration Distributed Protocol)* are conjointly used to distribute policies as well as new (or updated) decisions throughout the management hierarchy.

The *YAP (Yelp Announcement Protocol)* is used for reporting management information. Monitoring information is sent in a case of an abnormal observed situation and is reported in the form of events sent up to the management hierarchy. Each LPA reports events up to its DPA and so on. These events can be used to trigger other management actions.

The process of management and enforcing policies is performed by policy agents ac-

cording to a MAPE-K autonomic loop. Each policy agent is responsible for monitoring events, evaluating activated policy conditions and signalling and controlling the execution of policies to be enforced. The behavior of an agent can be adjusted according to the requirements of the network environment and high-level goals disseminated by the GPA. We note that in this research work, a particular attention has been focused on automated policy generation and control [38–40].

Although the successful development of DRAMA management tool [40,41], the proposed solution does not provide a complete automation of both network management and network planning. Management in many cases is locally performed. A distributed coordination across all the system instances (LPA, DPA and GPA) is required for an effective network management. Moreover, due to the centralized structure of the highest management level, the efficiency of the architecture regarding failures remains an issue. Besides, the use of several proprietary protocols (e.g. DRCP/DCDP) restricts an eventual wide adoption of this solution.

2.3.1.3 CA-MANET

Context-Awareness for the autonomic management of MANETs (CA-MANET) is a context-aware platform employing a hierarchical organizational model for MANET self-management [12, 42]. CA-MANET adopts a Policy-Based Network Management (PBNM) approach together with context-awareness, using policies triggered according to the context information.

The network is managed through a three-tier hierarchical architecture. The proposed policy-based model is based on a hybrid (hierarchical and distributed) organizational approach and hyper-cluster formation [12]. Two node modules are distinguished: *Cluster Manager (CM)* and *Terminate Node (TN)*. Three different roles are assigned to node modules: *Manager Node (MN)* and *Cluster Head (CH)* roles are assigned to CM nodes. *Cluster Node (CN)* role is assigned to TN.

As it is illustrated in Figure 2.6 [12], a *Context Repository (CR)* is deployed on every node of the MANET which stores diverse types of context, according to the role of each node (CN, CH or MN). The network is divided into two-level hierarchical clusters. A CH manages a set of CN belonging to its cluster. In turn, multiple MN manages groups of CHs. We note that each MN incorporates CH functionalities. In turn, each CH implements the MN element structure. Different platform components communicate between each other using XML-RPC protocol.

A CN consists of a *Policy Enforcement Point (PEP)*, a *Context Collection Point (CCP)* and a Context Repository. The PEP is responsible for enforcing activated policies. The CCP communicates with the sensors available to the device and collects context informa-

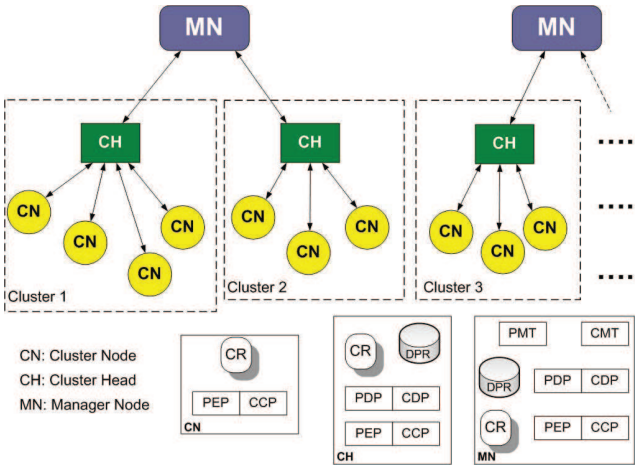


Figure 2.6: CA-MANET organizational model

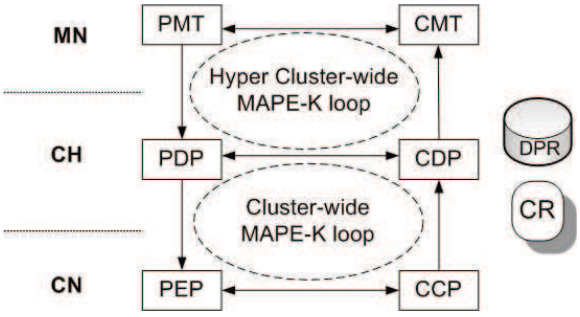


Figure 2.7: CA-MANET closed autonomic control loop

tion.

A *Policy Decision Point (PDP)*, a *Context Decision Point (CDP)* and a *Distributed Policy Repository (DPR)* are installed in every CH. The PDP is responsible for policy-decision making. The CDP monitors context and communicates with both the CCP modules of CN associated with its cluster and its corresponding MN. The DPR is a set of distributed and/or replicated LDAP directories configured to store policies.

A *Policy Manager Tool (PMT)* and a *Context Manager Tool (CMT)* are installed in each MN, situated at the highest level of the role hierarchy. The Context Manager Tool interacts with the Policy Manager Tool and with the PDP available at the MN in order to analyze monitored context and plan the appropriate decision according to policies stored in the DPR.

MN, CH and CN are supporting management responsibilities in a hierarchical manner. Each CN gathers locally context information. These Monitored context are transmitted from CN to their corresponding CH and then to MN. Cluster-wide decisions are taken by CH based on monitored context and defined policies. In order to establish if needed, MANET-wide configuration adaptations, CHs aggregate and process the network context and report it at regular intervals to the MNs.

As shown in Figure 2.7, the autonomic control loop is established between network nodes with different hierarchical roles (CN, CH and MN) rather than between internal components of each node. CR and DPR constitute the knowledge database.

The coordination between distributed CH (and especially between distributed PDP components) was simply limited to the maintenance and deployment of DPR [43, 44]. An automated replication mechanism, based on advanced replication and distribution features

of modern LDAP servers, ensures the consistency of introduced policies and offers a logically uniform view of autonomic management objectives.

The efficiency of CA-MANET have was evaluated using both testbed experimentation and network simulations [12,43]. Performance evaluations considered the dynamic selection of a proactive or reactive routing protocol (OLSR and AODV in this case) according to the mobility and context information. The results showed that the proposed policy-based mechanism enhances the QoS performances of running applications. However, the impact of the mobility and frequent topological changes were not studied.

To summarize, the most important contribution of CA-MANET is the exploiting of context-awareness for autonomic network management. Similar to other policy-based approaches, CA-MANET does not use any learning mechanism to enhance its policy-based adaptation process. Moreover, CA-MANET does not define any management mechanism to ensure the coordination between high-level manager nodes. Hence, The convergence of the entire network operation remains an issue.

2.3.1.4 Unity architecture

Unity architecture is an autonomic data prototype carried out at IBM's Thomas J. Watson Research Center. Unity was conceived to provide some needs of self-management of distributed computing systems [17,45,46]. Unity is an agent-based and service oriented architecture that employs three techniques: self-configuration, self-healing and self-optimization. The defined self-configuration technique was considered as a first step towards a full self-assembly goal. The proposed self-healing design employs sentinels and a simple cluster regeneration strategy. Finally, the self-optimization technique is performed using specific utility functions.

Unity system consists of a set of autonomic components which cooperate with each other in order to optimize the overall system performance and fulfil Service Level Agreements (SLAs). Each component is an autonomic element that manages itself and interacts with its environment. On one hand, an individual autonomic element is responsible for its own internal behavior and operations. On the other hand, it delivers services to other autonomic elements and exchanges monitoring information with them.

The distributed system is divided into clusters which consist of a set of application environments, providing application servers to clients. Each cluster is managed by a resource arbiter which arbitrates resource allocation among different application environments. As shown in Figure 2.8 [46], the main components that make up an application environment are:

- *Application manager element*: This element is responsible for the management of

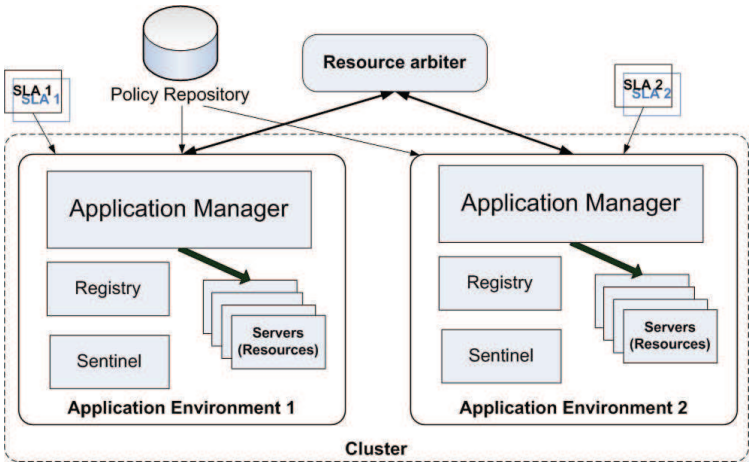


Figure 2.8: Unity scenario

available resources within an application environment. Moreover, it governs interactions of the application with its environment.

- *Server element*: It is responsible for the announcement of available resources and existing server's capabilities. In the described work, servers are considered as resources.

- *Registry element*: It is responsible for locating all other elements with which the application manager needs to communicate.

- *Sentinel element*: It plays the role of a monitor, providing monitoring services to other elements. It allows an application manager to ask about the monitoring functioning of other elements when needed.

Administrator high-level policies and utility functions are stored in the *policy repository*. An application environment is managed by an application manager which implements policies stored in the policy repository in order to adjust resources' usage. Initially, when the unity system is set up, the resource arbiter determines how many policy repositories are required in the cluster. A replication policy mechanism is established in order to insure a consistent view of policies. Whenever a policy data is modified or added in a policy repository, it is sent to all other policy repositories in the same cluster.

The resource arbiter is able to remove resources from one application manager to give them to another. If a policy may not be enforced because of the lack of resources, the resource arbiter asks for additional resources. Self-optimizing scenarios are based on the use of utility functions rather than action policies.

Testbed experimentations have been carried out on a realistic unity prototype implementation. A number of self-optimizing scenarios have been experimented. Results show that utility functions provide a powerful way to express high-level objectives and allow self-management properties in an autonomic system. However, the use of utility functions in Unity was limited to resource allocation for self-optimization purposes. Moreover, mathematical utility functions used were not detailed. The success of the system was measured based on the performance of each application regarding to the governing SLA.

Even Unity constitutes a valuable platform for studying and validating an autonomic system, autonomic aspects of the system are limited to self-configuration and self-optimization management properties. These latter constitute the goal to achieve in authors' future works.

2.3.2 Flat autonomic architectures

In this section, we describe some existing flat autonomic architectures. As outlined before, in this approach, nodes cooperate with each other in a peer fashion to provide the network with autonomic properties. In order to make adaptations made by different managers aligned with each other, nodes should have a correct view of other managers and of the network context.

2.3.2.1 ADMA architecture

Autonomous Decentralized Management Architecture (ADMA) is a distributed management architecture designed to provide MANETs with some autonomic principles [13]. The main objective of ADMA is to self-manage the process of distribution of high-level policies all over the MANET, enabling the self-configuration property. The proposed autonomic policy-based system operates in a peer to peer manner and does not require any centralized entity to perform network management functionalities [47]. Distributed nodes cooperate with each other in order to meet high-level policies predefined by the network administrator.

As it is depicted in Figure 2.9, the ADMA node structure consists of four basic components: *Local Policy Decision Point (LPDP)*, *Policy Enforcement Point (PEP)*, the *monitor* and the *local policy repository* [13, 47, 48]. The local policy repository is a local database where predefined policies are stored. These latter include configuration, reconfiguration, monitoring and meta-policies [13]. The LPDP analyzes monitoring information reported by the monitor and plans local decisions based on the predefined policies. It governs resource management, node configuration and reconfiguration. It is also responsible for interacting with other LPDPs in order to disseminate policies to non-configured ones, avoiding inconsistent decisions. The proposed LPDP does not refer to any higher-central decision making entity. It acts as a final authority for the decision that must be enforced by the PEP.

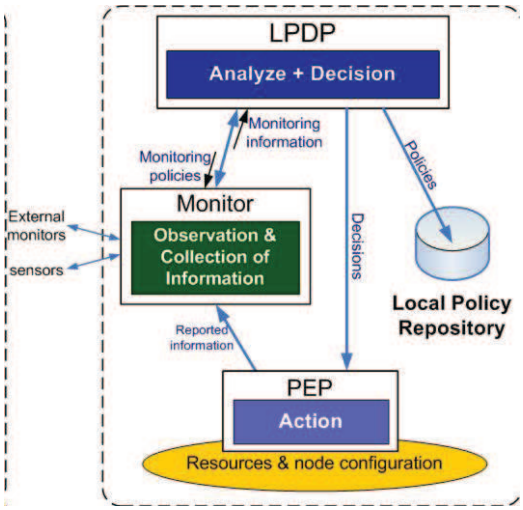


Figure 2.9: ADMA node structure

The PEP is the local entity responsible for enforcing policies and LPDP decisions. It acts as final effectors on node resources and configuration state. The PEP is reconfigured by the LPDP as a side-effect of events observed by the monitor. The monitor collects local and external monitoring information and reports it to the LPDP according to monitoring policies. It interacts with sensors and external monitors in order to accomplish its task.

The proposed management model is based on two basic operations: policy distribution, and reconfiguration and self-adaptation. A *Distributed Policy Management Protocol (DPMP)* was proposed in order to accomplish these tasks [47, 48]. DPMP allows policy distribution, information monitoring and reconfiguration. It supports intra-node communication between LPDP, PEP and monitor components as well as inter-nodes communication between different LPDP and monitors. Communication between internal ADMA components obeys to the well-known autonomic control loop.

The DPMP protocol operates as follows: Initially, the network administrator introduces the predefined policies to at least one node. Next, the DPMP protocol disseminates these policies hop by hop to all network nodes. Besides, a new node joining the network requests for predefined policies and receives them from one of its neighbors. Based on the collected monitoring information and predefined reconfiguration policies, each node is able to reconfigure and adapt its behavior by modifying the applied policies without stopping the system operation.

Performed simulation experiments show that DPMP protocol operates with a minimum generated overhead. Besides, its performance is not affected with high nodes mobility and unpredictable topology variations.

To summarize, ADMA combines policy-based management with autonomic principles

in a fully decentralized system. In ADMA, coordination between autonomous nodes is limited to the replication of predefined policies and the collect of required monitoring information. The consistency and optimization of the network is only based on the first definition of predefined policies by the network administrator, which is prone to human errors. Moreover, the authors focus more on the distribution of policies rather than the plan and analyze component that triggers the distribution mechanism. Further, ADMA does not specify any network-wide monitoring mechanism which may lead to an un-optimized network operation.

2.3.2.2 ANA project architecture

The Autonomic Network Architecture (ANA) is one of the FP6 EU-IST funded project on future and emerging technologies [49]. The objective of this project is to design and develop a clean-slate meta-management architecture with inherent autonomic behaviors to flexibly host, interconnect and federate multiple heterogenous networks [50].

ANA separates competing interests in the network into smaller and easier manageable realms, called *Compartments*. In order to allow different networking styles to be supported, ANA defines only some generic abstractions such that compartments are able to interwork [51]. It does not impose how network compartments should work internally, leaving addressing and naming up to the compartment. Consequently, each compartment implements the operational rules and administrative policies for its communication context. the functionalities defined by ANA consists in:

- How to join and leave a compartment: member registration, trust model, authentication, etc.
- How to reach (communicate with) another member: peer resolution, addressing, routing, forwarding, etc.
- The compartment-wide policies: interaction rules with "external world", the compartment boundaries (administrative or technical), peerings with other compartments, etc.

A compartment is composed of a set of *Node Compartments (NCs)* which are the conceptual view of the compartment members, representing the available networking resources. An ANA node compartment hosts following compartment abstractions:

- *Functional Block (FB)*: Any functionality required to communicate to a compartment or within a compartment is implemented by a corresponding functional block. As such, the FB can also be regarded as the processing elements or tasks hosted by an

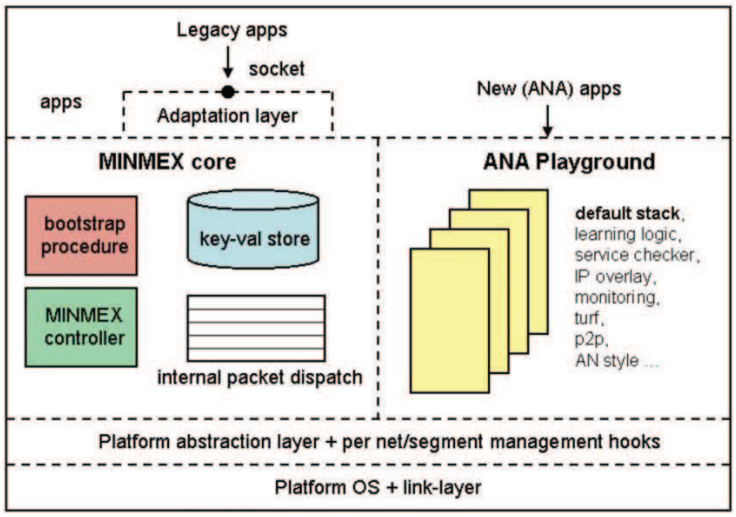


Figure 2.10: An ANA node

ANA node that constitute the compartment stack. Several FBs within an ANA node may be succeed to ensure a complete service. The composition of FBs is dynamic in the term that it may be dynamically build and dynamically re-composed at runtime in order to cope with context requirements [52].

- *Information Channel (IC)*: Communication inside a compartment is mediated via information channels which are the entry points to beforehand set up communication channels.

- *Information Dispatch Point (IDP)*: The Information dispatch point represents a start-point to which an information channel is bound. The IDP enables the network to reach the destination in an address agnostic way. A *Resolution process*, defined by each compartment, returns access to an IDP that can be used to reach the target NC member(s) via the bounded IC.

A conceptual representation of an ANA node is illustrated in Figure 2.10 [50]. The core functionality of ANA is provided by the *Minimal Infrastructure for Maximal Extendibility (Minmex)* which provides the basic low-level functionalities required to bootstrap and run an ANA node. Minmex acts as a message switching broker service to FBs. The *playground* is like an execution environment hosting FBs that exchange messages through MINMEX. This is where complex protocols and functions are implemented and where network-wide autonomicity is achieved [53].

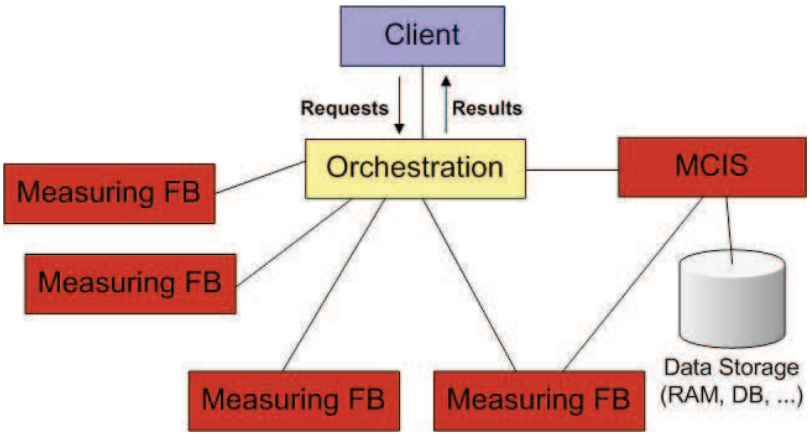


Figure 2.11: Conceptual view of ANA monitoring framework

Within the Minmex core, a *Minmex controller* performs a continuous assessment of the basic operation of an ANA node. The *Information Dispatch Table* maps IDPs to service providing FBs. The *Key Value Repository* maps service descriptions to service provider IDPs and thus enables the publish as well as the resolve and lookup functionality. Obviously, Minmex is the common denominator among ANA nodes, while playground implements compartment's specific functionalities.

As depicted in Figure 2.11 [52], monitoring is implemented through a set of *Monitoring FBs (MFBs)*. A *Measuring MFB* corresponds to various kinds of measurement tools used today. The *Client MFB* sets monitoring requests to Orchestration MFB in order to get specific monitoring information. The *Orchestration MFB* is the intelligent entity of monitoring which maps the request into appropriates Measuring MFB taking into account the context information. The *Multi-Compartment Information Sharing MFB (MCIS)* consists in a distributed system that provides lookup and store operations and supports multi-attribute range queries. It represents a context repository where some collected data are stored.

While each compartment may have its own self-adaptation mechanism, ANA defines a policy-driven self-adaptation mechanism to let intra-compartment self-optimization mechanisms to be inter-compartment aware.

A significant number of tests have been performed to evaluate the efficiency of the proposed architecture. The performed experiments include scalability tests of the proposed unified address management framework [54], reliability and QoS performance tests of the proposed inter and intra compartment-wide routing schemes [55], the stability and convergence of the managed networks [9], the efficiency of the proposed self-optimization

mechanism [9], just to mention a few. A real implementation of ANA software is available online [56].

The ANA architecture presents a policy-based meta-management architecture for interconnecting heterogenous networks with inherent inter-domain autonomic behaviors. Whereas, the intra-domain autonomicity is leaved to each compartment. Consequently, the efficiency of the entire autonomic network depends on the intra-domain-wide autonomic techniques used in constituent compartments.

2.3.2.3 INM architecture within 4WARD project

Wired and Wireless World Wide Architecture and Design for Future Internet (4WARD) [57] is one of the FP7 European project dealing with the design of a clean slate framework of innovative networking models, defining the direction towards a "Network of the Future". 4WARD proposes innovations to enhance the operation of both individual network architectures as well as the coexistence, inter-operability, and complementarity of several heterogeneous network architectures. In the scope of this thesis, we focus on the In-Network Management (INM) proposal, which consists in an evolutionary distributed autonomic architecture with embedded management intelligence in the network.

One of the strong point of the INM architecture is its gradual evolution and its support for migration from current Internet paradigms towards a purer INM system. This characteristic was enabled considering a *service-oriented architecture (SOA)*. Thanks to SOA paradigm, any future enhancement of the INM architecture will correspond on a novel service which is either composed from existing services or consists on a new enabled one.

As depicted in Figure 2.12, the INM defines a concise set of architectural elements from which any distributed and embedded management structure can be created [58, 59]:

- *Global Management Point (GMP)*: On the operator side, the global management point provides the high-level entry point into the management of a physical or virtual network. In the first hierarchical refinement, the global management point provides access to one or more management domains, each allowing access to a well-defined subset of the embedded management functions.

- *Self-Managing Entities (SEs)*: Self-managing entities are service-oriented elements of the architecture which encapsulate self-management functions of individual services. They are the logical constructs that encompass the properties necessary to achieve the autonomous operation of the network infrastructure. SEs provide the means for embedding a set of generic properties and abstract interfaces that enable network operation with only high-level intervention from the operator. In order to avoid the duplication of properties of different services, each SE is associated with basic and

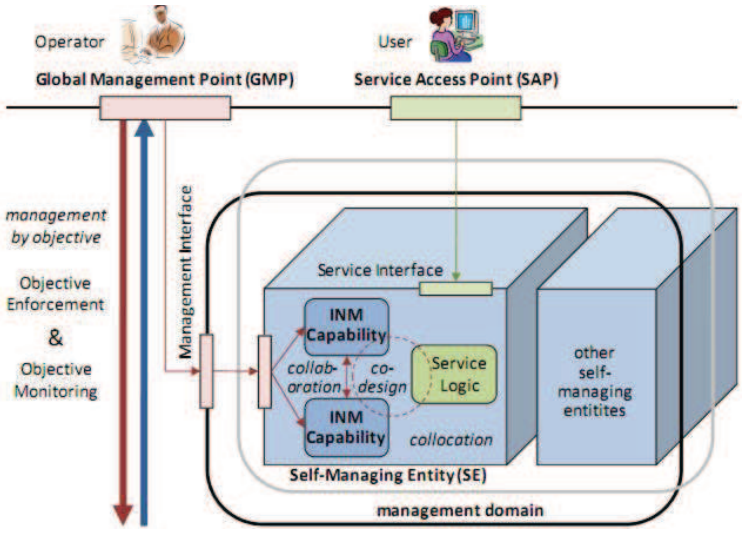


Figure 2.12: Conceptual view of the In-Network Management

generic service properties. More complex services may be formed by combining several such SEs. Self-managing entities are organized (either through a predefined order, or via self-organization) in a hierarchy according to the relationships between services. They collaborate with one another and enforce objectives on the service level in order to meet the service-specific objectives dictated by the operator's high-level objectives.

- *Management Capabilities (MCs)*: Each SE contains one or multiple Management Capabilities which are the fine-granular architectural elements implementing a specific function. MCs are the key enablers of self-management properties, implementing the actual INM self-property algorithms on a fine granularity. They can be composed to construct complex management functions (e.g. performance monitoring, situation awareness, self-adaptation schemes). The different MCs collaborate with each other in or between networks in order to achieve expected autonomous behaviors.

In such an environment, each MAPE-K components can be considered as a distributed network service. Following this logic, they consist of a set of SEs collaborating with each other via their corresponding MCs in order to achieve the desired functionality. Each composing MC implements a monitoring or self-adaptation functionality.

The enforcement and monitoring of objectives occur along the hierarchy of the previously described elements. Each objective is specified by the operator at the level of the GMP and split into sub-objectives via management domains and SEs until reaching individual MCs. Correspondingly, objectives and performance are monitored and aggre-

gated towards the GMP in the opposite sequence of elements. The proposed monitoring approach uses a set of complementary distributed algorithms that provide runtime views of the network state. This functionality includes real-time monitoring of network-wide metrics (called aggregates), group size estimation, topology discovery, data search, and anomaly detection. These algorithms provide the necessary input to the self-adaptation mechanisms [59]. Mainly, the INM architecture uses tree-based [60] and gossip-based [61] underlying protocols to achieve distributed monitoring of network-wide aggregates.

Three possible design options for the self-adaptation control loop are identified [59]. First, fully embedding the self-adaptation control loop within the MC in a monolithic manner. Second, designing the self-adaptation control loop as an identifiable entity inside the MC. Third, dedicating an MC for implementing the self-adaptation control loop, which enables self-adaptive behavior of other INM MCs that are otherwise not self-adaptive. While the proposed INM does not rely on the organizational design of the management, the distributed nature of INM make the first and the second options more appropriate to the management of future networks.

The proposed self-adaptation technique is based on chemical networking. This technique consists in modeling the system's dynamic behavior as a chemical reaction network, and making use of analytical tools developed in chemistry to predict the behavior of such systems. Based on this prediction, the reaction providing the most appropriate result will be selected. This methodology is applied for designing INM self-adapting algorithms which can continuously adapt to the network dynamics and enable resilient operation. In addition to previous self-adaptation mechanism, a basic policy-based self-adaptation scheme was proposed in the context of wireless multi-hop networks. The proposed mechanism dynamically select routing protocols and parameters based on networking conditions.

In order to enable the gradual evolution towards a purer INM system, two types of INM architectural elements are considered: *INM entities* and *non-INM entities*. Contrary to INM entities which are network elements supporting the INM architecture, the non-INM entities do not implement the INM features. While MCs are embedded in self-managing INM entities (SE), the non-INM entities are simply to be managed by MCs.

The performances of all proposed monitoring algorithms and adaptation mechanisms have been evaluated in both wired and wireless environments [59]. The performed tests were related especially to the overhead consumed by gossip-based and tree-based aggregation approaches, the message overhead of the proposed topology discovery mechanism, the QoS performances of the self-adaptive routing in wireless multi-hop networks, just to mention a few.

As described before, the INM architecture contributes to variety of components contributing to the autonomic management of a future architecture. Specially, a great deal

of interest has been focused on both the monitoring and self-adaptation processes. Moreover, thanks to its gradual evolution feature, INM architecture is able to manage both the INM and non-INM entities. However, similar to other autonomic efforts, the INM architecture does not consider any inherent security feature. Moreover, each proposed mechanism has been addressed and tested separately. Consequently, it is not clear how the proposed mechanisms collaborate to provide a network-wide autonomicity. Hence, an integrated evaluation use-case is required to validate the efficiency of all proposed mechanisms within the framework of the proposed architecture.

2.3.2.4 Cognitive Network architecture

Another trend aiming at proposing a self-adapting network architecture is addressed by cognitive networks(CogNet). Cognitive network is a recently emerged networking paradigm that combines cognitive algorithms, cooperative networking, and cross-layer design in order to provide real-time optimization of complex communication systems [18]. The cognitive network is defined as a network capable of perceiving current network conditions and then planning, learning, and acting according to end-to-end goals. The idea was introduced by Mitola [62], along with the concept of cognitive radio to provide efficient spectrum sharing by avoiding interference among communication systems. However, cognitive networking presents a much broader concept which considers network-wide goals [63] and cross-layer design [28]. From this definition, we can conclude that cognitive networking consists in a particular variant of autonomic communication with emphasis on self-optimization and self-configuration properties and inherent cross-layer design.

A CogNet consists in a set of cognitive nodes exchanging cognitive information with each other. As depicted in figure 2.13 [64], a cognitive node is made up three functional elements:

- *Cognitive Plane*: The cognitive plane is responsible for data analysis and decision making processes, leading to an optimal operational point given the network state. It heads monitoring information from the protocol stack as well as controlling them by issuing configuration commands. The decision making is based on local or network-wide information.

- *Cross-layer Coordination and Signaling Plane (CCSP)*: This plane provides an optimal signaling information delivery and acts as an intermediatory between protocol stack layers and the cognitive plane. It abstracts the task of layering information monitoring and configuring away from the cognitive plane. In order to operate with the standard protocol stack, each protocol layer is enhanced with a small software

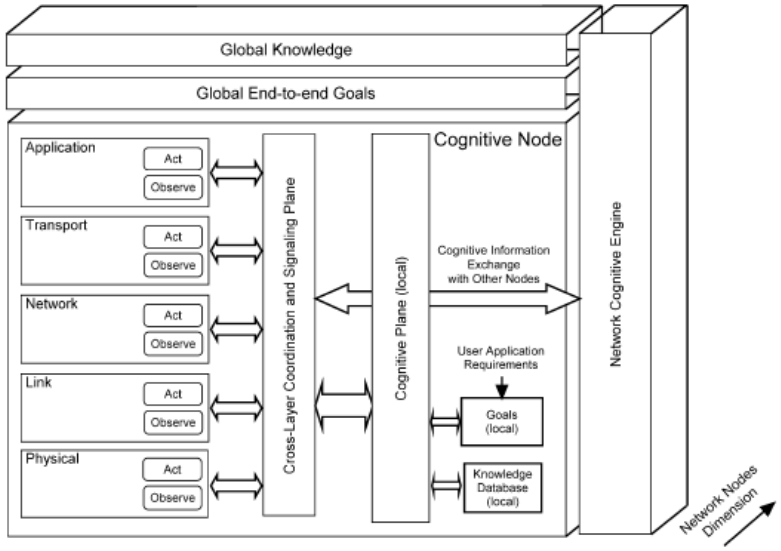


Figure 2.13: Cognitive network framework

module able either to obtain internal layer information (observation) or to tune layer parameters (action).

- *Network Cognitive Engine (NCE)*: The Network Cognitive Engine drives the coordination of cognitive planes of different cognitive nodes, enabling the network-wide scope of the architecture. The process include harvesting cognitive information available at cognitive nodes, analyzing monitored information, constructing global knowledge and goals, and reporting them back to cognitive nodes. These global information can be used to adjust nodes's local knowledge and, as a result, their behavior. The information collected by NCE could be node related (e.g. local goals of the node) or parameters associated with a particular flow transmission.

as illustrated in figure 2.14 [65], the cognitive adaptation process is performed according to a quality feedback loop which consists of three phases: data analysis, decision-making, and action. The cognitive plane monitors the overall operation of the network or the performance of current protocol parameters setup according to well defined target quality metrics. The quality metrics could be the overall packet delivery ratio for the overall network performance, the measured data rate for a physical layer parameter, the end-to-end delay for real-time multimedia applications at the application layer, juste to mention a few. The feedback loop iteration is completed with the enforcement of reconfiguration actions from the cognitive plane. These reconfiguration actions are the result of decision-making procedures performed by the cognitive plane.

In the proposed approach [65], each protocol parameter P is expressed in terms of its

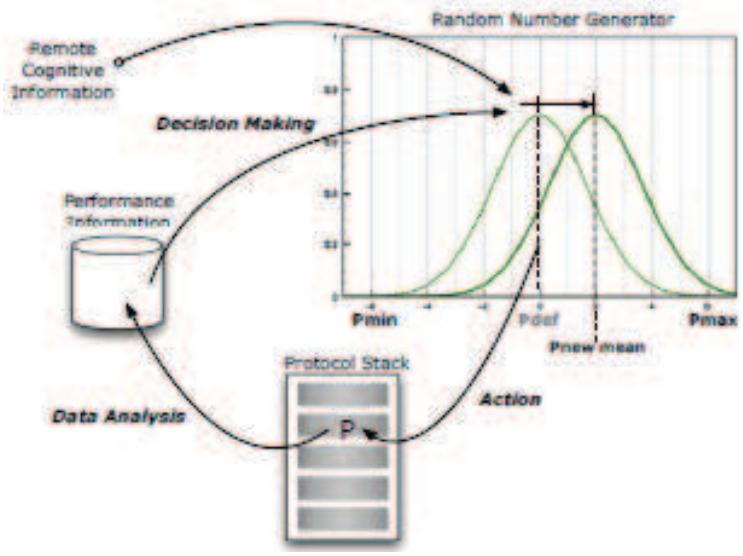


Figure 2.14: The quality feedback loop

default value Pdef and its operation range [Pmin, Pmax]. At the end of an interval I, the cognitive mechanism measures and stores the obtained performance from the current value of P in accordance with the defined quality metric. This data analysis phase allows the algorithm to build a history of operation with different settings and enables reconfiguring or re-adjusting future setups based on its past experience. Then, in the decision-making phase, the mechanism selects the value of P that provides the best performance according to the performance information base. That value is assigned to the mean of a random number generator that follows a normal distribution. Finally, in the action phase, a new value for P is chosen in the range [Pmin, Pmax] from the random number generator. The initial mean for the number generator is set to Pdef. This loop continuously adjusts the mean of the normal distribution to the value of P that provides the best performance under current network conditions.

As a proof of concept, the proposed CogNet approach was implemented into the NS-2 Simulator. The proposed approach was illustrated in the cognitive setting of TCP/IP reference model which brings cognitive reconfiguration into network management and protocol configuration [65]. Performance evaluation studied the application of the proposed approach for dynamic reconfiguration of the TCP congestion window. The window increase factor is adjusted during runtime based on the TCP throughput experience achieved in the immediate past. This is a sender side only modification which constrains proper configuration of window increase/decrease parameters during connection life time. It does not require any changes at the TCP receiver or any other network nodes and it is transparent

to other TCP modules.

Cognitive network presents an interesting feature for self-optimizing and self-configuring communication networks. The proposed cognitive framework allows dynamic reconfiguration of main protocol stack parameters at different layers for achieving performance goals driven by target quality metrics. However, the architecture, in this stage, does not specify any monitoring/coordination approach implemented by the Network Cognitive Engine. The proposed approach does not raise compatibility issues. Cognitive nodes can interoperate with ordinary nodes since the cognitive mechanism does not change protocol messages and operation. Another asset concerns the use of a learning mechanism which allows to converge to an optimal adaptation.

2.4 Evaluation of Autonomic Network Architectures

A relevant step for progressing the field of autonomic networking is to be able to evaluate and compare the performances of current solutions [66]. Two evaluation approaches can be considered:

- *Qualitative evaluation:* It consists in comparing different systems' characteristics or properties on a non-numeric and discrete scale (e.g. categories, levels, etc.).

- *Quantitative evaluation:* It designates comparing different systems' numerical measurements and quantities.

In the following, we first identify the main qualitative evaluation criteria contributing to the degree of autonomicity of ANM architectures. Moreover, we analyze and compare the characteristics of surveyed architectures based on the identified set of qualitative criteria. The quantitative evaluation approach which constitutes one of the main contributions of this thesis is separately addressed in chapter 5.

2.4.1 Qualitative evaluation approach

The qualitative evaluation provides a mean for comparing systems at a glance regarding non-numeric characteristics and properties. When applied to ANM architecture field, it enables a concise view of the management characteristics of the system. Moreover, it summarizes approaches used for each of its MAPE-K components.

In this section, we identify the main quantitative evaluation views applied to position an ANM architecture among other trends in the field. These qualitative metrics can be used in order to elaborate a taxonomy of autonomic network approaches or architectures.

2.4.1.1 Category of architecture

This metric specifies the general structuring of the architecture. In section 2.3, we divided ANM architectures into flat and hierarchical schemes. These categories can be used as an evaluation metric since they describe two different visions of network management, each one having its particular characteristics and issues. The hierarchical category operates like the classical management where a central manager supervises a set of managed objects. The network is more likely to be converged to an optimal solution since the decisions are made by a centralized manager with an overall view of the entire network. However, this approach suffers from scalability issues in large networks as well as the problem of a single point of failure. An alternative solution is to form a cluster of manager nodes handling the underlying managed network in a multi-level hierarchy. However, a mediation process is needed in order to converge the operation of the management cluster. Contrary with the hierarchical architectures, some ANM proposals are completely flat and distributed. The main issue would be to define a mechanism for converging the decisions and behavior of individual manager nodes. The efficiency of each category depends on the mechanisms taken to overcome their limitations.

2.4.1.2 Focus of Interest

As the design of an ideal autonomic architecture providing all self-management properties is a complex task, a step-by-step approach towards the full autonomicity has been taken by the research community. For this reason, each of the existing ANM architectures focus on a subset of self-management functionalities referred to as the *focus of interest*. Obviously, we can only compare the performance of those ANM architectures that address same self-management properties.

2.4.1.3 Target context

Each architecture is designed for a specific target environment, called *Target context*. Hence, it should be evaluated considering the characteristics of the network where it operates. As an example, a centralized architecture may be evaluated sufficient for a wired network composed of a few number of statistic nodes. Whereas, the same architecture is unacceptable for a network of thousand of mobile nodes.

2.4.1.4 Adaptation approach

In general, adaptation of an entity refers to changes that make such entity more fit to its environment [67]. In autonomic networking, adaptation refers to the ability of the network to autonomously perform adaptation operations using monitoring knowledge to

decide why, when, where and how adaptation should be performed [23]. Thus, adaptation involves analyze and plan components of MAPE-K model which are considered together in this paper as their functionality tightly linked. The adaptation approaches used in the surveyed architecture include policy-based adaptation [68], utility function based [46], chemical network based [69], just to mention a few.

2.4.1.5 Monitoring approach

As outlined before, the network monitoring involves capturing necessary measurements of the environment (either physical or virtual) that are of significance to the self-properties of the underlying network.

Different monitoring approaches for retrieving network views are used in the literature. These approaches can be classified into two main categories regarding to the granularity of the underlying management network:

- Monitoring approaches relying on disseminating knowledge to all network elements. Gossip-based aggregation [61], selective broadcasting [70] and situated view [71] approaches belongs to this category.

- Monitoring approaches which disseminate information to a subset of network elements. This category include distributed context repository [72] and tree-based aggregations [60], just to mention a few.

2.4.1.6 Network convergence mechanism

The issue of the convergence of the autonomic management is raised when a non centralized architecture is used. This is due to the existence of several MAPE-K control loops with the decision making processes. Hence, a cooperation between autonomic managers should be defined in order to converge to an optimal solution. This evaluation view indicates the mechanism taken to address this issue.

2.4.1.7 Learning ability

An extreme potential of autonomic networking is its ability to learn from past experiences to enhance future operations. This intelligent adaptation is mandatory to converge to the optimal adaptation. As an example, a policy-based solution without learning capability may consider any exceed of delay up to a predefined threshold as a sign of congestion in its policies description. Accordingly, each time this threshold is exceed, the manager decides to drop some non-priority packets to reduce the delay for QoS packets. In contrast, a learning-based solution would rethink the accuracy of previous congestion perception by

analyzing the success of past experiences. If learning mechanisms are used, the system could change this threshold if it learns that an increase or decrease of that value fits more with the reality of the underlying network. As well, the system could adjust its reaction once it learns that the further reaction provides better result. The qualitative learning ability evaluation view consists in determining if the architecture makes use of a learning mechanism in its decision process.

2.4.1.8 Security mechanism

The security is a prominent issue of autonomic networking since it impacts the smooth operation of all MAPE-K components. Aside commune security issues existing in any distributed architecture, autonomic networking should face its specific issues. To achieve this, the vulnerabilities of autonomic architectures should be identified and recovered. Mainly, these vulnerabilities include the reliance on managers cooperation, the reliance on network-wide information and the distribution of intelligence which can be exploited by different types of attacks menacing the integrity, confidentiality and authenticity of highly valuable autonomic managers' information exchanges as well as the availability of services (see chapter 5 for further details on security issues of autonomic architectures). As the result, cryptography and trust can be considered as two primordial security features of any autonomic network architecture. The qualitative security criterion determines if any security defense line is defined and if so, which mechanism is used.

2.4.1.9 Heterogeneity management

An autonomic network may be composed of a set of autonomously managed systems that interact with each other in order to enable end-to-end communications. However, the integration and mediation of several management systems may be challenging due to the problem of heterogeneity [30]. The issue is raised from the existence of several management standards, different protocols and different vendors. A heterogeneity management scheme is thus needed in order to spur the cooperation among heterogenous management systems. Some of the surveyed architectures address this issue while others ignore this aspect and focus on the self-management of a homogeneous network.

2.4.1.10 Openness

Some of the surveyed architectures provide their solution for public uses. This feature enables further application of the architecture in various networks and augments its chances to become a reference ANM architecture. As well, the open access of the solution can help discovering and diagnosis of its shortcomings and issues. Consequently, the proposal is

more likely to be enhanced.

2.4.1.11 Evolvability

The evolvability consists in the property of the architecture to support future add-on and enhancements. This characteristic opens the architecture for being improved by the ongoing advancement within the communication technologies. We qualify an architecture being evolvable when the core element is designed alike to a middleware. In this wise, a core element will act as a control framework which enables interfaces to smaller architectural components. In this way, any further architectural or algorithmic enhancement can be plugged into the system smoothly.

2.4.1.12 Validation

This criterion indicates if the pretended features of a given architecture is validated by appropriate evaluation alternatives (simulations, mathematical modeling, etc.). If so, we need also acknowledge the evaluation metrics used to validate the proposed solution. The numerical results may also help to judge about the scalability of the proposed architecture which is an important requirement of future and ongoing architectures.

2.4.2 Comparison of surveyed autonomic architectures

In section 2.3, we provided a description of major existing ANM architectures, highlighting the mechanisms used for each of their MAPE-K components. We have also described how autonomic elements of each architecture interact to each other in order to meet some autonomic objective properties. We observed that a significant number of architectures has been already proposed and different approaches has been taken by each of surveyed architectures. In order to progress to a convergent ANM architecture, there is a need to discuss how the different proposals compare, what limitations of one proposal are addressed by the others and what is the best proposal for a given case.

As stated before, the comparison can be done either by comparing architectures regarding to the results of quantitative measurements or with regard to their qualitative properties and to the approaches taken for each of its components. In order to compare ANM architecture quantitatively, different proposals have to be implemented under a same technology. In addition, their performances in terms of each proposed criteria should be measured under same scenarios and configurations.

Table 2.1 (page 41) summarizes the characteristics of existing autonomic architectures and evaluates them qualitatively regarding some management characteristics and evaluation criteria. The following points can be identified from the table:

- Except DRAMA which relies on a central point of management in the highest level of the hierarchy, other surveyed architectures are either flat distributed or hierarchical distributed. This is expected since a centralized architecture suffers from a serious scalability problem. It is very hard to a central node to maintain a runtime global view of the entire network status and to provide each node with the runtime suitable decision, especially in the case of a dynamic network.

- Self-configuring property is addressed by all architectures. This is due to the fact that other self-management functionalities require self-configuring in order to enforce performed functions. Hence, we can consider the self-configuration as an inherent part of any self-management property.

- We can observe that when the architecture targets a dynamic large scale network, a flat distributed architecture is often preferable.

- Policy-based adaptation is the most commonly used self-adaptation mechanism. The reason can be the straightforward application of this mechanism where adaptation plans are described by a simple event-condition-action. Another typical advantage of policy-based autonomic networks is their ability to reconfigure and self-adapt by modifying the applied policies at runtime without stopping the network operation [23].

- The Cognitive network is the only open-adaptive scheme, enhancing its adaptation process based on a learning mechanism which makes use of experiences gained from previous adaptation actions. This approach has the potentiality to be used in parallel to the policy-based management approach in order to overcome its shortcomings. Mainly, with policy-based adaptation, human operators are still faced with the burden of having to precisely foresee all network events, describe their desired behavior and develop necessary policies. Omitting the corresponding policy for a particular network behavior may lead to non optimized resource usage. Another main issue with policies is the problem of policies conflicts: an event might satisfy the conditions of two different ECA (Event-Condition-Action) rules, while one rule may dictate an action that conflicts with the other satisfying rule [73]. A learning approach may overcome these limitations.

- Most of the architectures rely on semi-distributed knowledge bases rather than full distributed ones. It signifies that a number of distributed replication policy repositories is dispersed in the network. This approach avoids abundant extra overhead of monitoring approach while ensuring a correct network view.

- Neither of these architectures does not make use of security mechanisms. This aspect

remains a prominent issue of autonomic networking since it impacts the smooth operation of all MAPE-K components.

- AutoI, ANA and INM proposals addressed the heterogeneity management issue while others ignore this aspect. Generally, it is performed by using a coordination mechanism which ensures the federation of several heterogeneous systems. AutoI uses the concept of Orchestration to address this issue. To achieve this, federation, distribution, negotiation and governance Orchestration Behaviors are defined. The ANA approach is based on a unified address management framework. The INM architecture enables the interoperation of INM and non-INM entities. This is done by managing non-INM entities using corresponding self-managing entities.

- Only AutoI and ANA solutions are open-source. Unity and INM proposals provide some open-source code while some software remain proprietary.

- The AutoI, ANA and INM proposals are designed considering evolvability features.

- All proposed architectures are scalable for the context for which they were designed initially.

- None of the surveyed architectures did not perform quantitative or qualitative comparison with already existing autonomic architectures. Performance results were related to the evaluation of the effectiveness of the proposed mechanisms and the overall system behavior regarding some Qos performance metrics. Moreover, comparison efforts were restricted to classic centralized network management approaches which do not support any of autonomic properties. For example, in [41], authors presented a study of the scalability and performance of their proposed DRAMA architecture. Besides, they compared DRAMA performances to the conventional SNMP-based management system which do not support any property of autonomic systems. Performance gain of DRAMA architecture compared to SNMP was expected mainly at large scale and with a dynamic environment such as MANETs.

2.5 Conclusion

This chapter provided a holistic view of research in the area of autonomic network architectures, aiming at identifying the pros and cons of each one to provide a roadmap forward the full autonomicity. We classified autonomic network architectures into hierarchical and flat ones. We identified four major efforts in hierarchical category: AutoI project, Unity, DRAMA and CA-MANET architectures. The main flat architectures that we managed to find in the literature were ADMA, ANA, INM and Cognitive Network architectures.

Moreover, we identified the main qualitative metrics that can be used to evaluate and compare the efficiency of autonomic architectures. These identified criteria are the most relevant architecturally for autonomic networking and could be refined progressively. Further, we raised the issue of lack of efficient tools and models for evaluating and comparing quantitatively autonomic network architectures regarding the efficiency of autonomic solution. Moreover, we noted that the existing architectures do not neither make use of learning mechanisms (except Cognitive Network architecture) nor security techniques which are prerequisite to any ideal autonomic solution. We outlined that the use of learning mechanisms can significantly improve the performance of policy-based adaptation schemes towards the optimal solution. Moreover, we highlighted the importance of evolvability feature to support the future progress in the field of autonomic communications.

Table 2.1: Comparison of Autonomic network architectures

	AutoI	DRAMA	CA-MANET	ADMA	ANA	Unity	INM	CogNet
Category	Distributed hierarchic	Centralized hierarchic	Distributed hierarchic	Flat	Flat	Distributed hierarchic	Flat	Flat
Tiers	3	3	3	1	1	1	1	1
Focus of Interest	Self-configuring	Self-configuring	Self-configuring	Self-configuring	Inter compartment self-optimizing, self-configuring	Self-optimizing, self-healing, self-configuring	Self-healing, self-optimizing, self-configuring	Self-optimizing, self-configuring
Target context	Static & dynamic networks	Dynamic centralized networks with group mobility pattern	Networks with low mobility	Highly dynamic networks	Large scale networks	Grid environment	Static & dynamic networks	Static & dynamic networks
Adaptation approach	Policy-based	Policy-based	Policy-based	Policy-based	Policy-based	Utility functions, Policy-based	Chemical networking, Policy-based	Learning-based cognitive process
Monitoring approach	IMO	YAP protocol	XML-RPC	DPMP protocol	Any measurement tool used by the monitoring MFB requested by the Orchestration MFB	Provided by the sentinel element of each application environment	Gossip & tree based, NATO!, H&S, etc.	Not defined
Monitoring granularity	Semi-network wide	Network wide	Semi-network wide	Network wide	Semi-network wide	Semi-network wide	Semi-network wide	Unknown
Network convergence approach	distribution & negotiation mechanisms	A centralized autonomic manager (GPA) disseminates decided policies using DRCP/DCDP	Distributed autonomic managers (MN nodes) disseminate decided policies based on uniform distributed DPRs	DPMP protocol	Distributed Multi-compartment Information Sharing (MCIS)	Distributed autonomic managers (Resource Arbiter) disseminate decided policies based on uniform distributed DPRs	Centralized GMP policies	The uniform network-wide view managed by the NCE
Learning ability	No	No	No	No	No	No	No	Yes
Security	No	No	No	No	No	No	No	No
heterogeneity management	Yes	No	No	No	Yes	No	Yes in the sense of supporting INM and non-INM entities	No
Openness	Yes	No	No	No	Yes	Both open-source & proprietary software	Both open-source & proprietary software	No
Evolvability	Yes	No	No	No	Yes	No	Yes	No
Validation								
Use-case	deployment & maintenance of end-user & networking services in fixed & wireless network environments	Military MANET networks	MANETs with low mobility	Generalized MANETs and highly dynamic networks	Various experiments in fixed and MANET environments	Grid service architectures, client application environment using web service interfaces	Static networks, mobile MANET	Cognitive configuration of TCP window size in a small wired topology
Metrics	Bandwidth, The sum of all durations of virtual router creation, Time to deploy	DRAMA traffic overhead	E2E delay, average control cost, average convergence time per node, average context dissemination cost	Latency: average self-configuration convergence time, generated control overhead	Mean mapping size per controller regarding Unified Address Management Framework, E2E reliability(PDR) & number of transmission per pkt received/send regarding the intra-compartment routing, etc.	Response time of self-optimizing, measured system utility	Detection delay, accuracy of threshold detection, protocol overhead (aggregation approaches), drop ratio (anomaly detection), Average E2E delay (self-adaptation), E2E path length, delay & jitter (INM support for MANET routing)	Average throughput
Scalability	Studied up to 90 Virtual Routers in 15 networks interconnected by 75 Virtual Links distributed over four physical machines, up 110 services over 110 routers in Grid5000	Studied up to 504 nodes (21 nodes per domain)	Studied up to 700 static nodes	Studied up to 300 nodes with various network density & random node mobility pattern	Studied for the Address Management framework (3000 VNs with an average of 80 VNs per federation, and 3 million end hosts with an average of 1000 end hosts per VN)	Not studied	Studied up to 300 nodes for a highly dynamic MANET (policy-based self-adaptation), up to 5232 nodes for a static environment (monitoring approach), up to 800 connections (anomaly detection)	Not studied

Chapter 3

The SADA Architecture

Contents

"The reasonable man adapts himself to the world;
the unreasonable one persists in trying to adapt the world to himself.
Therefore all progress depends on the unreasonable man."

George Bernard Shaw, Maxims for Revolutionists: Reason, Man and Superman (1903)

3.1 Introduction

Due to the dynamic nature of wireless mobile networks, the underlying network context changes continually over time. As such, those networks require an adequate architecture which adapts continuously network configurations according to the current condition of the underlying network. This context motivated us to propose a Self-Adaptive Autonomic architecture for self-managing wireless networks, called SADA. The main objective of SADA is to present an autonomic management architecture which is able to learn from recent experiences to enhance its future operations. This intelligent adaptation is mandatory to converge to an optimal network operation, as it is one of the key self-management properties. This chapter presents the proposed SADA architecture, its associated components and their design, as well as a simple case study. The proposed case study demonstrates the application of the SADA architecture for self-adapting the route discovery process of mobile ad hoc networks.

The chapter is organized in six sections as follows. Section 3.2 gives some motivation for applying self-adaptation principles in networking. Section 3.3 describes the SADA architecture and its components. Section 3.4 presents the design concepts we used to model the components of the proposed architecture and describes their correlation with our architecture. Section 3.5 proposes a routing case study and presents simulation results. Finally, section 3.6 concludes the chapter.

3.2 Motivation

As outlined in the previous chapter, current MAPE-K based autonomic network architectures rely in closed-adaptive techniques where a same pre-defined adaptation strategy is applied repetitively in the same context regardless of whether the strategy was successful or not. Closed-adaptive strategies suffer from major insufficiencies which make them inappropriate for autonomic communications:

- The administrator should foresee different network states and define appropriate actions to be taken regarding each potential state. The need for such low-level configuration details does not meet the self-property objective of autonomic computing.

- Pre-defined adaptation strategies deal with static, non-changing, and well defined problems and lack the ability to deal with a new situation not predicted in a-priori defined adaptation strategies. This limits the coping range of such approaches (the range of acceptable variations within which the system can adapt its behavior and outside which it may collapse [23]).
- Static adaptation strategies built a-priori into network devices lack the flexibility and may not be sufficient to handle different changes in these underlying environments. If the system internal or external context changes in a way that is not suitable for the pre-programmed adaptation, external control is required to adjust the existing policies.
- Another major limitation is that the hard-wired control makes no use of any experiences gained from previous adaptation actions. This learning ability is mandatory to converge to the optimal adaptation.
- Pre-defined adaptation strategies lack the ability to find the optimal solution for indeterministic networking configuration parameters. These parameters are function of some other network characteristics and parameters which makes it impossible to determine the behavior of such indeterministic parameters.
- Static adaptation strategies may result in further vulnerabilities regarding misbehavior attacks since it leads to a deterministic system behavior. As a consequence, a misbehavior attacker can anticipate the behavior of the system and use this knowledge to set up its attack.

To address the aforementioned issues of current MAPE-K based autonomic architectures, we propose a self-adaptive autonomic architecture (SADA) enriched with self-adaptation features. The self-adaptation is achieved through the add on of a cognitive adaptation component to the reference MAPE-K model. In next section, the proposed SADA architecture and its components are described.

3.3 SADA - Self-Adaptive Autonomic Architecture

SADA is an autonomic network architecture which provides the autonomic network with self-adaptation features. With regard to the reference MAPE-K model and autonomic architectures applying that model, SADA presents two supplementary characteristics: first, it enriches analyzing and planning components of MAPE-K model with self-adaptation capabilities. Second, it defines a second autonomic control loop which is formed around the monitoring component and allows the monitoring mechanisms to be self-adapted according

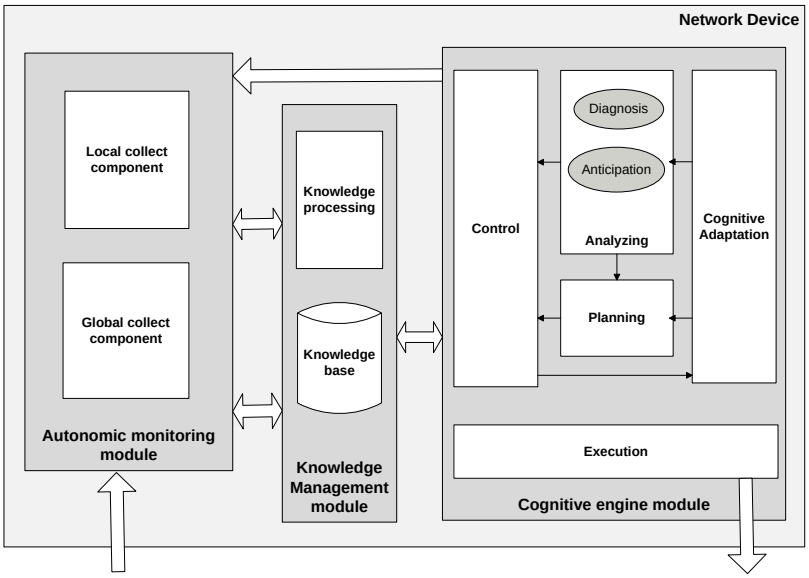


Figure 3.1: SADA architecture

to the network context. SADA characteristics result from the requirements and properties of autonomic communication and aims at overcoming the shortcoming of existing architectures. With SADA, autonomicity is achieved in a distributed manner where network devices collaborate to provide an overall autonomicity. Each network element represents an autonomic manager which manages directly its resources and implicitly the resources of other network elements.

The conceptual model of SADA architecture is illustrated in figure 3.1. It is composed of three major modules named **Cognitive engine**, **Autonomic monitoring** and **Knowledge management** modules. The main module is the cognitive engine employing our self-adaptation approaches, whereas autonomic monitoring and knowledge management modules provide support for the first one. These modules are respectively detailed in subsections 3.3.1, 3.3.2 and 3.3.3.

3.3.1 The cognitive engine module

The cognitive engine is the intelligent part of the architecture which enables the network with self-management properties. It holds five independent components, namely analyzing, planning, control, cognitive adaptation and execution components.

The analyzing component analyzes the knowledge in order to verify the current performance, predicts the future state of the network and detects events. It is composed of two main functionalities: network anticipation and network diagnosis. Network anticipation relates to the utilization of the system knowledge to further anticipate the future state of

the network. State anticipation is necessary to achieve the self-management functionalities as it is needed to avoid either reaching non-optimal or failing states (i.e., failing to achieve self-optimization and self-healing). Network diagnosis consists in interpreting the knowledge into a state description of the managed network, e.g. detection of an attack, a failure or a relevant network state of the underlying network (congestion, high dynamicity, etc.)

The planning component is responsible for planning adaptation strategies when the need for optimizing network operation or dealing with a detected or anticipated anomaly is detected by the analyzing component. This phase may use techniques such as artificial intelligence, control theory, game theory or domain-specific algorithms. The actions of this component may be configured using abstractions such as policies.

The cognitive adaptation component is responsible for adapting the analyzing and planning components according to the conditions of the underlying network as well as experiences gained from previous actions. This component makes use of learning mechanisms to enhance the future operation of the network based on past experiences which guides the adaptation strategies to take new adaptation decisions.

The control component supervises the operation of other components within the cognitive engine module to ensure the consistency of all decision-making processes. It verifies if the strategies used in planning, analyzing and cognitive adaptation components have been resulted in an enhancement of network operation in changing environment. If any anomaly is detected, the cognitive adaptation component will be activated. This latter finds self-adaptation strategies to adapt and optimize the mechanisms within other components according to the network conditions.

The execution component enforces the decided plans and self-adaptation strategies on networking resources, allowing the autonomic operation of the network. The enforcement includes the reconfiguration of local or remote networking devices, the reconfiguration of an adaptation strategy within the planning and analyzing components, just to mention a few.

3.3.2 The autonomic monitoring module

The autonomic monitoring module holds mechanisms to gather all data required by the cognitive engine module. The autonomic monitoring module is composed of a local collect component and a global collect component. The first one collects local data by monitoring local resources and/or cross-layer approaches. The second component is in charge of monitoring network-wide knowledge, allowing the node to adapt itself according to the conditions of the underlying network. Both components must consider the limitation and heterogeneity of the network resource capacities such as memory, bandwidth and processing and must be efficient in using these resources while providing a sufficient correct view

about the network conditions.

Comparing to the reference MAPE-K model, an important contribution to the autonomic monitoring module consists in its autonomic ability, i.e. it is considered as one management object that should be self-adapted during the operation of the network. To achieve this, some adaptation strategies should be defined within the cognitive engine module, treating the smooth and efficient operation of the autonomic monitoring module. As an example, when a periodic monitoring mechanism is used, the cognitive engine module infers continuously the efficient interval update which ensures a correct view on network conditions while generating a minimum extra overhead.

3.3.3 The knowledge management module

The knowledge management module is responsible for processing of the data before sending them to the cognitive engine module. Normalization, previous calculation and aggregation are examples of processing used for facilitating the analysis and the inference of the autonomic engine module. The knowledge management module holds two independent components, namely **knowledge processing** and **knowledge base**. The first component is responsible for the data processing process and representing the network, users and applications by a model that efficiently reflects the networking environment. The second component represents a knowledge base where processed collected data, denoted knowledge, are stored. In addition to the cognitive engine module, the autonomic monitoring module uses the knowledge stored in the knowledge base for its autonomic adaptation.

3.4 SADA design

In this section, we describe the design of the cognitive engine, autonomic monitoring and knowledge management modules of the proposed SADA architecture.

3.4.1 Cognitive engine design

The additional functionality of the cognitive engine module compared with existing autonomic architectures is its self-adaptation ability. In general, the adaptation of an entity refers to changes that make such an entity more fits more to its environment [67]. System adaptation refers to the enforcement of one or more strategies, algorithms, rules or configurations in order to change one or more aspects of the system with the objective of achieving a set of higher-level goals. Autonomic adaptation refers to the ability of the system to perform adaptation operations using its internal knowledge to decide why, when, where and how adaptation is performed, without any external intervention in the decision making process.

In this section, we propose to use random neural networks with reinforcement learning for to model the cognitive engine module. The choice of random neural networks is motivated by its strength for modeling the different states in which the network can be found, while reinforcement learning allows the cognitive engine to learn about the efficiency of its past decisions and infers their positive or negative impacts on the network performance.

3.4.1.1 Random Neural Networks

Inspired by the spiking behavior of biophysical neurons, The random neural network (RNN) is a recurrent network of N fully connected neurons which exchange positive and negative signals in the form of unit amplitude spikes [74].

A biological neuron can be either excited or quiescent according to its state. The state of a neuron is described by its signal potential which is a non-negative integer associated with the accumulation of positive signals at the neuron. Each neuron can fire only when its potential is strictly positive. When a biophysical neuron is excited, it fires a train of positive or negative signals, called action potentials or spikes, along its axon according to the exponential distribution with rate r_i. Positive signals have an excitatory effect in the sense that they increase the signal potential of the receiver neuron by 1. In contrast, negative signals have an inhibitory effect and cancel a positive spike. The combined effect of excitatory and inhibitory inputs changes the potential level of the receiving neuron and determines whether it will become excited. Positive and negative signals can also arrive from the outside world according to Poisson processes of rates Λ_i and λ_i, respectively.

A random neural network is represented in figure 3.2. The state q_i of a neuron i, which consists in the probability that the neuron i is excited can be described as follows:

$$q_i = \frac{\lambda^+(i)}{r(i) + \lambda^-(j)} \tag{3.1}$$

where $\lambda^+(i)$ and $\lambda^-(i)$ are respectively the total excitation and inhabitation signals that the neuron i receives from other neurons j. These parameters can be calculated as follows:

$$\lambda^+(i) = \sum_{j \in\ neurons, j \neq\ i} q_j w^+_{ji} + \Lambda_i \tag{3.2}$$

$$\lambda^-(i) = \sum_{j \in\ neurons, j \neq\ i} q_j w^-_{ji} + \lambda_i \tag{3.3}$$

Where w^+_{ji} and w^-_{ji} are weight metrics representing respectively the positive and negative firing rate from neuron j to neuron i and should be learned from input data. The total firing rate from the neuron i, $r(i)$, can be obtained by summing the total positive and

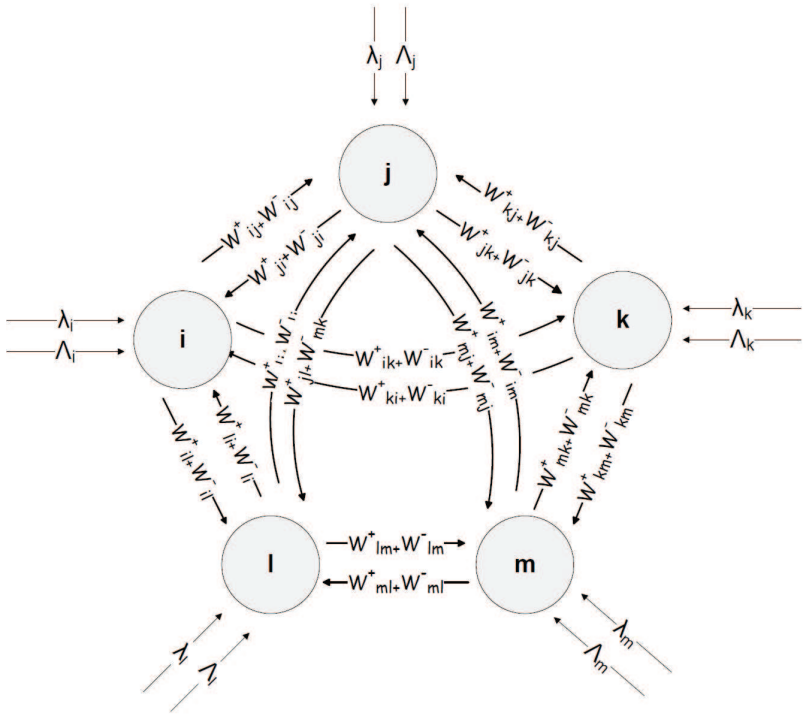


Figure 3.2: A random neural network

negative firing rate of a neuron i, as follows:

$$r(i) = \sum_{j \in \ neurons, j \neq \ i} [w_{ij}^{+} + w_{ij}^{-}] \tag{3.4}$$

The RNN has found widespread application in diverse areas of engineering and physical sciences e.g. modeling queueing and biological networks, image processing, pattern recognition, classification, combinatorial optimization and communication systems. They use either the original RNN or extended models with additional capabilities using a variant network model or additional interaction features [75]. The success of the RNN model can be attributed to its unique features which include the following [76]:

- It represents in a closer manner the signals transmitted in a biological neuronal network than other artificial neural networks (ANNs).

- It consists in a prominent modeling tool that can capture the behavior of interacting entities in different frameworks such as biological and queueing networks. Indeed, the stochastic excitatory and inhibitory interactions in the network make it an excellent modeling tool for various interacting entities.

- Although it is a recurrent neural network, its steady-state probability distribution is described by an analytical equation that can be easily and efficiently computed.
- The network stability has been proven by mathematical analysis.
- Its standard learning algorithm has low complexity and strong generalization capacity even for a relatively small training data set.
- It can be easily implemented in both software and hardware since its neurons can be represented by simple counters.
- There is a direct analogy between the RNN and the connectionist ANN.
- The neuron potential is represented as an integer rather than a binary variable resulting in a more detailed system-state description.
- It consists in a universal approximator for bounded continuous functions.
- Parallel organization permits solutions to problems where multiple constraints must be satisfied simultaneously.
- With RNNs, rules are implicit rather than explicit.
- One of the main assets of RNN is its ability to learn from examples.

3.4.1.2 Correlating random neural networks and the proposed approach

The spiking behavior of the biological neurones and its corresponding mathematical RNN model fit well with the requirements of an autonomic architecture in which a mechanism that reflects the success or failure of a performed adaptation is necessary. Hence, the presented RNN can be used to model the self-adaptation functionality of the cognitive engine module. An important advantage consists in its generalizable modeling which allows to describe various decision problems using the same model.

The adaptation of the described RNN to the cognitive engine module of the SADA architecture is fairly straightforward. We consider any indeterministic adaptation problem as a decision problem where the objective is to find a decision which results in best performance for the current network conditions. To achieve this, each adaptation problem is modeled by a RNN. Each neuron within a RNN represents a potential decision choice for that decision problem. The RNN continuously seeks the most appropriate choice according to the network conditions.

We consider two classes of adaptation problems according to their space of possible decisions:

- Close decision space: The first class involves adaptation problems with a close space of possible decisions, e.g. adaptation of the transmission range which can take a value among 100m, 200m and 150m. In this case, each potential choice is represented by a neuron, i.e. one different neuron for representing each case of 100m, 200m and 150m.

- Open decision space: This class consists in adaptation problems where the space of possible decisions are not predefined and can be infinite (more realistically with un-predefined bound), e.g. adaptation of the value of frequency of beaconing update which may be either every 20ms, 40ms, 60ms and so on. In this case, we consider a RNN with three neurons, each one representing a potential preference in increasing, decreasing or keeping the current beaconing interval.

In order to manage the inter influence of decisions made by inter-dependant decision problems, only one RNN is used representing conjointly those decision problems, where neurons correspond to possible combination of choices among them. As an example, we consider that the performance of a routing protocol depends on the value of two protocol parameters, $P1$ and $P2$, which should be adjusted according to the network conditions. $P1$ can select a choice between v_1, v_2 and v_3 while $P2$ could be either w_1 or w_2. The corresponding RNN represents the combination set of these choices which are (v_1, w_1), (v_1, w_2), (v_1, w_3), (v_2, w_1), (v_2, w_2), (v_2, w_3).

In our context, a decision corresponds to the selection of the most excited neuron in the RNN, i.e. the neuron i with the highest value of q_i. In order to reflect the impact of the underlying network conditions on the decision problem (self-adaptation), the weights of equations 3.2 and 3.3 should be learned from the outside network. Learning is defined as "a process by which the free parameters of a neural network are adapted through a process of stimulation by the environment in which the network is embedded. The type of learning is determined by the manner in which the parameter changes take place. In other words an effective learning process has a close relationship with the interaction between environment and neurons, which are ideally part of the medium [77]".

Although several learning mechanisms have been proposed for RNN [75], we are focusing particulary on a reinforcement learning (RL) mechanism with E-rule because of its lower computational complexity, unsupervised nature and adaptability with network changes. In RL, a system takes a sequence of cascaded decisions related to the perceived state of the environment and accordingly receives external reinforcement either positive (reward) or negative (punishment). The weight update is related to the attained environmental reinforcement so that 'good' decisions are rewarded. The reinforcement $R_\tau(i, a_\tau)$ that neuron i receives in this cascaded decision environment is a function of trial τ and the external reinforcement, a_τ, associated with the output neuron. In E-rule, the reinforcement relies on an

adaptive internal expectation of the reward: the weights are updated according to the difference between the actual reinforcement and the internal expectation. For example, if the difference is positive, the weights of all selected neurons in the decision path are reinforced proportionally to the difference, while their neighbor neurons are punished accordingly. Otherwise, all excitatory weights leading to all neurons except that corresponding to the previous decision are increased, and the inhibitory weights leading to the previous wining neuron are increased in the same manner. In this way, the algorithm is adaptive to changes in the environment and results in obtaining time-varying reinforcement. The mathematical representation of the employed RL algorithm is as follows:

- $If R^{+}_{\tau,\beta} \leq R^{+}_{\tau+1}(i, a_{\tau+1})$
  - $w^{+}_{\tau+1}(i,j) = w^{+}_{\tau}(i,j) + R^{+}_{\tau+1}(i, a_{\tau+1}),$
  - $w^{-}_{\tau+1}(i,k) = w^{-}_{\tau}(i,k) + \frac{R^{+}_{\tau+1}(i,a_{\tau+1})}{N-2}, \forall k \neq j.$
- Else
  - $w^{+}_{\tau+1}(i,k) = w^{+}_{\tau}(i,k) + \frac{R^{+}_{\tau+1}(i,a_{\tau+1})}{N-2}, \forall k \neq j,$
  - $w^{-}_{\tau+1}(i,j) = w^{-}_{\tau}(i,j) + R^{+}_{\tau+1}(i, a_{\tau+1}),$

where $w^{+}_{\tau+1}(i,j)$ and $w^{-}_{\tau+1}(i,j)$ are, respectively, the updated positive and negative firing rates the neuron i receives from the neuron j at the trial $\tau+1$. $w^{+}_{\tau}(i,j)$ and $w^{-}_{\tau}(i,j)$ are, respectively, the old positive and negative firing rates of the neuron i received from the neuron j at the trial τ. N represents the number of neurons in the network. $R^{+}_{\tau,\beta}$ denotes the internal expectation of the reward and $R_{\tau+1}(i, a_{\tau+1})$ represents the new reinforcement reached the neuron i. In the above equation, index j corresponds to the selected neuron of the network that resulted in the particular reward value. Numerical instability is avoided by re-normalizing the weights after each update.

In SADA, the new reinforcement reached by a neuron can be obtained by observing its outcome on the environment. The outcome is evaluated regarding the goal of the decision-making process. For instance, if the goal of a RNN is to optimize the QoS performance gained in the network, the outcome resulted by each decision should be evaluated regarding the associated QoS performance metrics, e.g. in terms of delay, throughput, loss ratio, just to mention a few. The autonomic monitoring module is in charge of monitoring the outcome of a new performed decision regarding the associated metrics. The monitored information is, then, processed and aggregated by the knowledge management module to represent the overall reinforcement obtained by a given decision.

3.4.1.3 Interaction with policy-based management

In networking, two classes of configuration parameters can be distinguished: The first class involves configuration parameters which have deterministic behavior, i.e. one can determine their behavior based on some information on the network conditions. For instance, when the network mobility is high, it is always preferable to use a reactive routing protocol rather than a proactive one. Policy-based management constitutes a valid mechanism to adjust the value of these deterministic parameters. The second class consists in parameters for which it is not always possible to anticipate the best setup due to the unpredictable state of the network. For instance, the choice of the default initial value of TCP congestion window may change over the time based on the available network bandwidth and the level of congestion at the end-to-end path between the sender and receiver which can not be determined in real-time. The use of the proposed random neural network can optimize significantly the configuration of such parameters.

Furthermore, the proposed RNN can be seen also as a complementary of policy-based management and be used as a methodology to self-adapt pre-defined policies. Indeed, a policies is represented by a condition-action clause which determines the action to be taken when a given condition is satisfied. The condition defines an important event, mainly represented by a threshold, for which an adaptation action is necessary. However, it is difficult to define the exact condition that characterizes an event. For example, a policy may consider any exceed of delay up to a predefined threshold as a sign of congestion. Accordingly, each time this threshold is exceed, the manager decides to drop some non-priority packets to reduce the delay for QoS packets. However, the delay threshold may be exceeded for other reasons than congestion, e.g. the nodes mobility may results in network changes which cause some supplementary delay before updating routing tables. Consequently, the reaction on the network congestion (packets dropping) has been taken without being on that state. The proposed random neural network can change this threshold if it learns that an increase or decrease of that value fits more with the reality of the underlying network. As well, the system could adjust its reaction once it learns that the further reaction provides a better result.

3.4.2 Autonomic monitoring module design

The autonomic monitoring module represents the second entity of the proposed SADA architecture. Each node implements an instance of the autonomic monitoring module which performs monitoring and constructs necessary information according to the requirements of the cognitive engine module (decision-making RNNs). As most of autonomic architectures, we consider two types of knowledge: internal (local) view and external (network) view. The

internal view is obtained from monitoring local resources or/and cross-layer information. The external view represents the knowledge of all or a subset of network resources required for decisions with a more global scope. Such a metric could be energy level, load or neighbor degree, just to mention a few.

The proposed design for the autonomic monitoring module consists in two contributions: first, we propose to make use of any existing network traffics to carry the monitoring information along their movement to their destination. This mechanism uses any data or signaling packet traveling in the network to update continuously visited nodes' global view. For instance, a packet can carry information on metrics such as energy level, load or lifetime of links, and update it continually when visiting more nodes. In this way, the external view of each intermediate node will be updated progressively with arrived packets. The main advantage of this technique is that it avoids the significant overhead for acquiring the external view, as it is the case in most of monitoring mechanisms applied to autonomic networks. Indeed, this approach economizes in the IP-header since already existing packets in the network are used. Further, as packets are the most mobile elements existing always in the network, we can assert with a high probability that all nodes receive the monitoring information. As such, an uniform and updated network-wide view can be ensured in a real-time, required for the real-time management problems.

The second contribution in the design of the autonomic monitoring module consists in its self-adaptation characteristic. Indeed, the proposed autonomic monitoring module is coupled with the cognitive engine module of the SADA architecture which ensures its self-management properties. Hence, the operation of the employed monitoring approach can be adjusted over time according to the conditions of the underlying network. Regarding the proposed packet-based monitoring technique, the cognitive engine module allows to adjust the rate of packets' piggybacking according to the state of the network. For instance, if the network is overloaded with a significant number of packets, the cognitive engine module can decide on the rate of applying the piggybacking technique to avoid generating the supplementary load to the already congestionned network.

3.4.3 Knowledge management module design

The knowledge management module is an entity of the SADA architecture which manages the processing of the data before sending them to the cognitive engine module. In the context of this thesis, we propose a simple data processing model to aggregate several types of information required by RNNs within the cognitive engine module. The knowledge heterogeneity management and the user and application modeling using ontologies [78, 79] and other modeling tools are out of the scope of this thesis.

As described before, the learning is performed by observing the performance of a de-

cision on its environment regarding the associated performance metrics to that decision. The objective of the proposed aggregating model is to combine all associated performance metrics characterizing the goal of a given RNN to derive an overall performance obtained by a decision (a neuron). To achieve this, the individual performance metrics should be combined using an appropriate goal function. The individual metrics may have different dimensions resulting in unfair proportion between those metrics in a linear combination function. For instance, the performance metrics for a RNN created to self-adapt a QoS routing protocol could be the delay and the throughput obtained by each decision. However, these QoS performance metrics can not be combined using a linear function. To solve this problem, we propose an additive utility-based goal function which correlates fairly all individual performance metrics into an unique goal metric, called global goal.

Each metric is expressed in terms of the utility of its different possible values for achieving the global goal. The contribution of each value to the global goal is calculated in terms of its utility. The resulted utility is then combined to a global goal using an additive goal function, as follows:

$$G(m_1, m_2, ..., m_M) = \sum_{k=1}^{M} U(m_k) \tag{3.5}$$

where $G(m_1, m_2, ..., m_M)$ denotes the goal function combing the values of the individual performance metrics and $U(m_k)$ represents the utility of measured kth metric of interest. We considered the positive contribution of each metric in the global goal (i.e the higher value is the better); e.g. we take the bandwidth directly but 1/delay should be used instead of delay.

The selected utility function should have low complexity and increase strictly monotonically when the value of a metric k increases. Moreover, it should converge to a bound value when the value is considerably satisfiable. Considering these requirements, we used $U(m_k) = 1 - e^{-\alpha_k m_k}$ as the utility function where α_k is the relative weight of the kth metric of interest.

3.5 SRS - Self-Adaptive Routing Scheme based on SADA architecture

Routing is a fundamental network functionality for mobile ad hoc networks (MANETs). Its main function is to support a correct delivery of data from one node to another. Routing service is composed of two phases: route discovery and route maintenance. Route discovery process finds reactively or proactively routes allowing packets delivery. The route maintenance strategy recovers broken paths. This case study demonstrates the application of the SADA architecture for self-adapting the route discovery process of MANET routing

protocols.

This case study handles the problem of selection of the most appropriate next hop in the route discovery process, giving any preferred routing goal (e.g. number of hop or QoS goals). To solve this problem, we proposed SRS, a Self-adaptive Routing Scheme based on SADA architecture. This problem is modeled as a RNN by the cognitive engine module of the SADA architecture based on the information provided by the autonomic monitoring module. A reinforcement learning mechanism is used allowing privileging or avoiding some current paths based on their recent experienced performance, monitored by the autonomic monitoring module.

The case study proceeds as follows. First, we report some basic routing protocols with an emphasis on their path adaptation process. Next, we detail the proposed self-adaptive routing scheme based on the SADA architecture. Afterwards, we describe the methodology of evaluation, simulation environment and results. Finally, we discuss achieved results and close the case study.

3.5.1 Related work

A significant number of MANET routing protocols has been already proposed in literature [80–83]. Whether the protocol uses a proactive or a reactive mechanism, the packet forwarding is either table driven (e.g. DSR) or source-initiated (e.g. AODV).

AODV [81] uses a table-driven approach, in which the next hop to any known destination is stored. Further, each table entry (any known destination) contains the destination sequence number and the number of hops towards that destination. A source node attempting to send data packets to a destination, broadcasts a Route Request (RREQ) packet if it has no entry for that destination in its routing table. When an intermediate node receives a RREQ, it rebroadcasts the RREQ except if it has a current entry for that destination in its table. That is, if it has a route with a sequence number that is greater than or equal to that contained in the RREQ packet. In the latter case, the node unicasts a route reply packet (RREP) back to its neighbor from which it received the RREQ packet until it reaches the source. The RREP packet establishes the forward path to the destination, updating the next hop and the destination sequence number for the corresponding table entry. If a node receives two routes for a same destination with a same sequence number, the one ensuring a shortest path (lower number of hops) is selected.

With DSR [80], nodes store the recent discovered paths in their cache. A source node desiring to send a data packet to some node, uses the path stored in its cache, if it has an entry for the desired destination. Otherwise, the source broadcasts a RREQ packet including its own address. As the RREQ packet arrives to any intermediate node, the node adds its address to the RREQ packet and rebroadcasts it if the destination asked does not

exist in its cache. Otherwise, the node sends back a RREP packet to the source. When the source receives the RREP packet, it transmits data packets on the discovered route. Moreover, an entry in the cache will be inserted for the future use. The node will also maintain the age information of the entry so as to know whether the cache is fresh or not.

Other table-driven and source-initiated routing protocols for MANETs use some similar principles than AODV and DSR. Both approaches employ a form of flooding (reactive routing) or network-wide periodic updates (proactive protocols) that produce a poor long-term utilization of the network. One issue with these protocols is that even when a node can learn from its recent communications to update the routing table, the broadcast mechanism is used to discover a newer path. Considering an example of AODV protocol, when a node receives a RREQ with a destination sequence number greater than the one it has in its routing table for the same destination, the node rebroadcasts the RREQ packet. However, the broadcasting mechanism generates extra unnecessary overhead to the network. Indeed, nodes can learn from recently incoming packets allowing them to decide about the more appropriate route to the destination in the changing MANET environment.

Another issue with the current routing protocols is their reliance on relative incomplete information about the up-to-dateness of the current path. For example, this latter is ensured by the destination sequence number with AODV protocol which consists in the largest sequence number known by the source for that destination. This latter information is obtained by the observation of traversing RREP and RREQ packets. However, using this mechanism, the information of a node regarding the largest destination sequence number can be incorrect in the large scale MANETs. Hence, the routing configurations would not be always self-adapted. However, the routing protocol in MANETs' changing environment should self-adapt permanently to ensure that the best route to the destination is always selected.

The only initiative with some route discovery self-adaptation features is addressed in [84]. The authors proposed a source-initiated routing protocol called AHCPN *(Ad Hoc Cognitive Packet Network)* which uses neural networks to select a best path satisfying the preferred routing goal. The RREP packets are used to inform a node about the quality of paths towards the destination. Moreover, the protocol uses source-routed data acknowledgement (DACK) packets to update the knowledge of intermediate nodes. Although the proposed approach is very original, its reliance on DACK packets generates a significant extra overhead to the network, resulting in its very poor network performance comparable to a flooding protocol [84].

To solve the self-adaptation issues of existing MANET routing protocols, we propose SRS, a Self-adaptive Routing Scheme based on the SADA architecture. The cognitive engine module of the SADA architecture is used to self-adapt the problem of selecting

the best next hop. The autonomic monitoring module feeds the cognitive engine module with required information using transiting routing packets arrived from or traversed a same destination. This monitoring technique together with the overhead gained thanks to the self-adaptation capability can reduce significantly the overall control traffic in the network. As an important asset, inherited from the SADA architecture, the SRS is a protocol-independent scheme, which can be integrated to any existing routing protocol to enhance its performance. This latter constitutes an important feature of the SADA architecture for managing the route discovery problem as well as any other self-adaptation problem. In next section, the proposed self-adaptive routing scheme is described.

3.5.2 The proposed SRS Scheme for Mobile Ad hoc Networks

The proposed routing self-adaptation scheme is based on the SADA architecture. Each node of the ad hoc network is equipped with an instance of the cognitive engine module and the autonomic monitoring module of the SADA architecture. The objective is to self-adapt the route discovery process of MANETs so that a next hop resulting in a best path to the destination is always selected. This phase is ensured by the cognitive engine module based on the knowledge obtained by the autonomic monitoring module. In the following, we detail the function of each of these modules in the context of routing self-adaptation.

3.5.2.1 SRS - Cognitive engine module

Due to the dynamic nature of MANETs, the selection of a best path ensuring a preferred routing goal is an indeterministic decision problem. Indeed, it is impossible to determine exactly at any time which next hop will result in a best path (e.g. a path with minim number of hops) until the destination. This kind of adaptation problems are modeled as a random neural network by the cognitive engine module of the SADA architecture.

We highlight that the SRS scheme is independent of any routing protocol and aims to self-adapt the process of route discovery, regardless of the table-driven or source-initiated nature of the route management. As such, we do not describe the condition that trigger the route discovery process or the process of informing a source about the discovered route as they are highly dependent to the employed routing protocol. In the following, only the route self-adaptation and its highly related monitoring processes are described.

Inspired from [84], the cognitive engine module within each node holds a RNN per pair of destination and the preferred routing goal, called a category. The RNN within each node is responsible for determining the next hop that leads to a best path to the destination giving the preferred routing goal. Hence, each RNN is composed of N neurons, each one representing a neighbor, i.e. a potential next hop. The neurons' weights reflect the potentiality of each neuron (next hop) to represent the optimal decision (optimal next hop

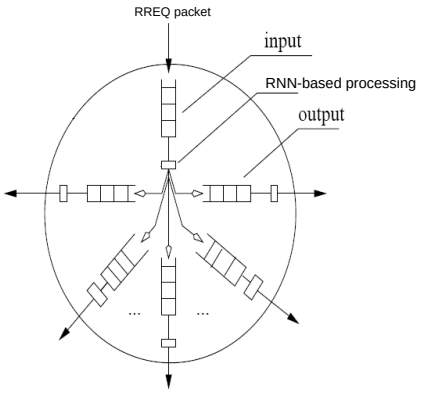


Figure 3.3: The next hop selection phase

to reach the destination). Hence, the weights are continually updated by the autonomic monitoring module to represent the current network state. This process is described in section 3.5.2.2. As depicted in figure 3.3, when a RREQ packet arrives at a node possessing a RNN for that category, the next hop is directly selected by executing the corresponding RNN. The execution of RNN consists in computing the state q_i of all neurons using the equation 3.1. The neighbor leading to a path optimizing the desired routing goal is selected as the next hop. This corresponds to the neuron which has the largest excitation probability. If the cognitive engine module does not possess sufficient information to execute the RNN, the RREQ packet will be broadcasted until the destination or an intermediate node with an updated RNN is reached.

3.5.2.2 SRS - Autonomic monitoring module

In SRS, the monitoring consists in collecting information allowing a node to recognize its current neighbors as well as to evaluate the efficiency of its preferred current next hop for a given destination. The first one is used to update the RNNs' neurons, while the second one updates the weights of current neurons. In order to minimize the monitoring overhead, the autonomic monitoring module collects this information using normal transiting packets in the network.

In contrast with the existing MANET routing protocols which rely on a promiscuous mode or a periodic beaconing for maintaining neighbors lists, SRS assumes that the neighbor is alive as long as it keeps transmitting packets. If the node does not receive any packet from a neighbor for a period of time, then it removes that neighbor from its neighbors list.

In order to update RNNs' weights, both RREQ and RREP packets are used. Each of these packets progressively measures the quality of the explored route regarding each of the metrics characterizing the desired routing goal and carries these information throughout

their path. Upon visiting nodes, RREQ and RREP packets update their information about the quality of the path between their originator and the current node (the quality of routes are considered similar in both directions). Based on the acquired information, the RREQ and RREP packets update the weights of the corresponding RNN in each intermediate node. When two or more metrics are used to characterize the preferred routing goal of a packet, these individual monitored information are aggregated to a global goal by the knowledge management engine as described in section 3.4.3. If the global goal corresponding to the same category is enhanced, the neuron corresponding to that decision will be rewarded, otherwise, that neuron will be punished according to the reinforcement learning process described in section 3.4.1.2. As the result of the reinforcement learning, the neuron weights are adjusted so that the future decision will meet more likely the desired routing goal.

3.5.3 Evaluation

We carried out a set of simulation experiments using NS-2 version 2.30 in order to show the effectiveness of the proposed approach. Our self-adaptive routing scheme was instantiated on DSR as routing protocol, simplifying the inspiration of the structure of RREQ and RREP packets from the source-initiated AHCPN protocol.

We implemented the cognitive engine, the autonomic monitoring and the knowledge management modules of SADA architecture as generic networking modules and provided necessary interfaces to link them to the routing agent. The DSR protocol was modified to execute the defined monitoring, knowledge management and cognitive engine modules. All protocol-specific modifications are performed in the routing agent, allowing the straightforward integration of the same SADA modules for any other routing protocol. Another advantage of integrating the SRS scheme to existing protocols rather than as a new protocol is to benefit from some already well-defined routing mechanisms of these existing protocols (e.g. route maintenance).

Analysis compare the results provided by the SRS with those yielded by DSR and AODV. Moreover, we performed some elementary tests regarding the AHCPN protocol using the implementation provided by [84]. We observed that the AHCPN generates a significant amount of routing overhead and lacks very ordinary routing mechanisms used in other protocols, degrading more and more its performance. Consequently, we avoided to compare our scheme with the AHCPN protocol in this case study.

3.5.3.1 Environment setting

The IEEE 802.11 distributed coordination function (DCF) is used as medium access control (MAC) protocol. The radio model takes into account similar characteristics to a commercial Lucent's WaveLAN radio interface with a nominal bit-rate of 11 Mb/s for the shared-media

Table 3.1: SRS - Simulation parameters

Parameter	Value
Transmission Rate	11 Mbps
Radio Proagation Model	TwoRay Ground
Mobility Model	Random waypoint
Network area	1300m x 500m
Transmission Range	250m
Interface queue size	64 packets
node number (nn)	30, 50, 70
CBR rate	4 pkt/s
CBR sources (nc)	10, 20, 30
Data packet size	512 bytes
Maximum speed (Ms)	1, 5, 10, 20 m/s
Pause time	5s
Simulation time	500s

radio and nominal radio ranges of 250 meters. The mobility model applied is the random way-point model, where node speeds are randomly chosen between 0m/s and a maximal speed of 1m/s, 5m/s, 10m/s or 20m/s, evaluating the network with different dynamicity. The pause time is fixed to 5s. The network area dimensions were fixed for all simulations to 1300×500 meters square. The total number of nodes (nn) placed randomly in this area is 30, 50 or 70 nodes, characterizing networks with different densities.

The data traffic used is CBR (Constant Bit Ratio) with a number of connections (nc) equal to 10, 20 or 30 defined randomly. Each source node generates 4 packets/sec with a packet size of 512 bytes. Data traffic sessions happen at a random time in the simulation. The total simulated time was 500 seconds and each plotted point is an average of 30 simulations with a 95% confidence interval.

In our tests, we considered two types of information monitored by RREQ and RREP packets: path lifetime and number of hops. Table 3.1 summarizes main simulation parameters.

We introduce three metrics to evaluate the efficiency of our routing self-adaptation scheme: **packet delivery ratio** (PDR), **average end-to-end delay (E2E delay) of data packets** and **routing overhead**. With the first metric, we compare the ability of the routing protocols to delivery successfully data packets. The second metric compares the delay achieved by the self-adaptive and the non self-adaptive schemes. With the third metric, we evaluate the amount of control traffic generated by each protocol. Metric definitions are as follows:

- Packet Delivery Ratio (PDR) - number of data packets successfully delivered at the destination divided by the number of data packets sent from the source.
- Average End-to-end delay (E2E delay) - transmission delay of data packets delivered

successfully. The delay consists of propagation delays, queuing delays at interfaces, retransmissions delays at the MAC layer, as well as buffering delays during the route discovery.

- Control Overhead - the total control overhead bytes sent and forwarded to ensure the delivery of data packets divided by the total delivered data packet bytes. The control overhead consists of the amount of all non data packets, including route request, route reply and route error packets.

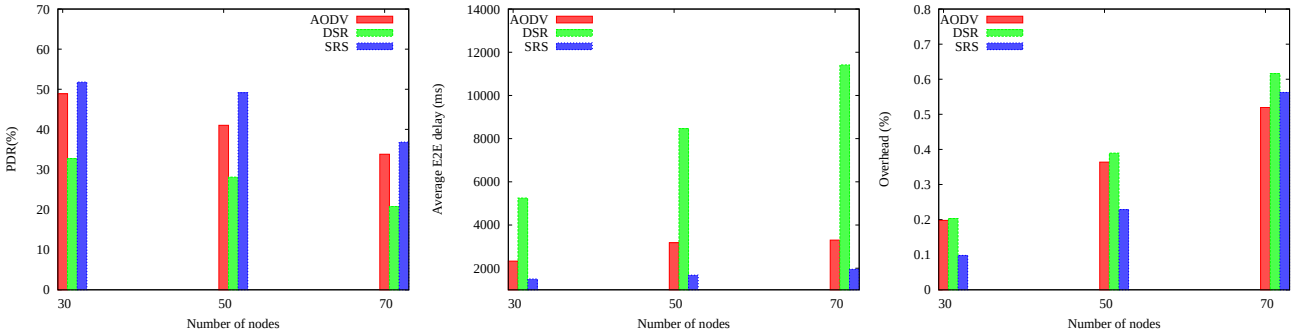
3.5.3.2 Results and analysis

Results of our self-adaptive routing scheme are presented on three perspectives, the packet delivery ratio, the average E2E delay and the control overhead. For all perspectives, we compare SRS scheme with DSR and AODV protocols considering the impact of different parameters, such as number of nodes, nodes speed and number of connections. Initially, we evaluate the impact of number of nodes on the efficiency of SRS compared with DSR and AODV. Next, we compare the SRS scheme with DSR and AODV protocols regarding different network dynamicities. Finally, we compare the performance of SRS with DSR and AODV protocols considering different number of connections.

Impact of the number of nodes The results of different network densities are illustrated in Fig. 3.4 and Fig. 3.5. For these comparisons, the number of connections are set to 30, evaluating a network with high traffic. We evaluated the impact of different network densities for nodes speed of 10m/s and 20m/s, examining the correlated impact of number of nodes and network dynamicity.

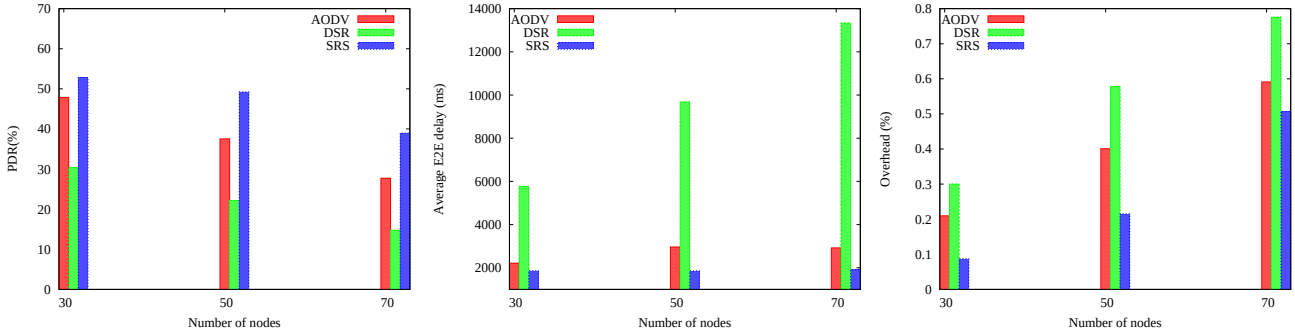
The PDR graphics in Fig. 3.4(a) and Fig. 3.5(a) show that the PDR decreases when the number of nodes increases independently of scenario used. This is due to further cached terminal problem and collisions for higher densities. As expected, higher nodes speed results in lower PDR for AODV and DSR protocols. The PDR of SRS scheme is not impacted negatively by higher nodes speed due to the reception of information carried by RREQ and RREP packets by further nodes. For all scenarios, the PDR achieved by our scheme is significantly better than the PDR of DSR and AODV protocol. However, this effect is more elevated for nodes speed of 20m/s when SRS is compared with AODV protocol.

Fig. 3.4(b) and Fig. 3.5(b) plot the average E2E delay for different network densities. The delay increases for higher number of nodes independently of scheme and scenario used due to further cached terminal problem, medium access delay and retransmissions. DSR results in higher delays than AODV and SRS due to the higher network overhead (Fig. 3.4(c) and 3.5(c)) as well as the later reaction to network changes. This effect is avoided for SRS



(a) PDR (b) Average E2E delay (c) Overhead

Figure 3.4: Impact of number of nodes - Ms = 10, nc = 30



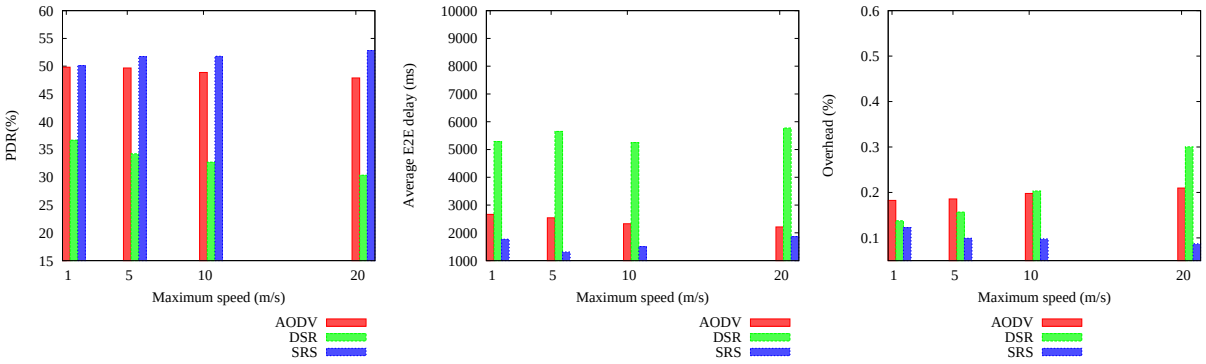
(a) PDR (b) Average E2E delay (c) Overhead

Figure 3.5: Impact of number of nodes - Ms = 20, nc = 30

scheme using the self-adaptation mechanism. These figures show that SRS scheme results in lower delay than DSR and AODV protocols independently of scenario used. As expected, average E2E delay increases for higher nodes speed for all scenarios.

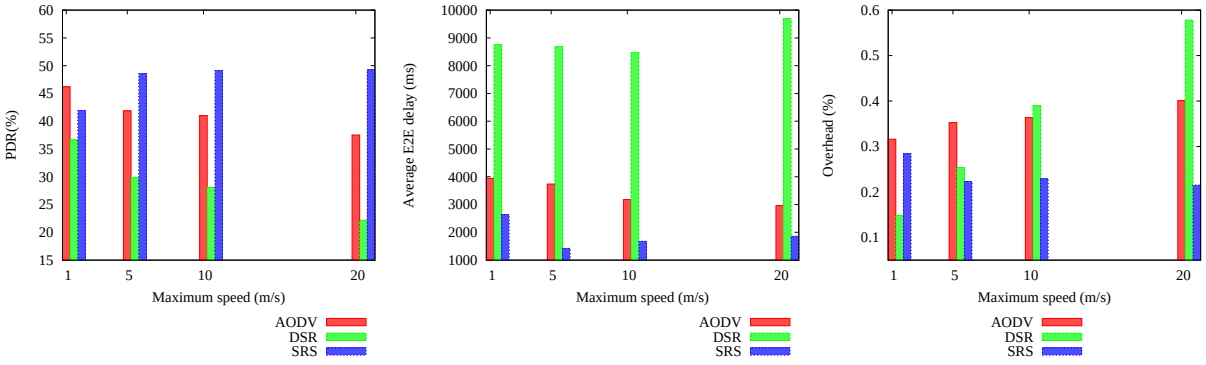
The overhead graphics in Fig. 3.4(c) and Fig. 3.5(c) show that the overhead increases for higher network densities for all schemes independently of scenario used. This is due to further cached terminal problems and medium access collisions. For all scenarios, our scheme generates s significantly lower amount of overhead compared to AODV and DSR protocols. As expected, the overhead is higher for nodes speed of 20m/s than nodes speed of 10m/s for DSR and AODV protocols. Similar to the result for PDR, higher nodes speed results in slightly lower overhead with SRS scheme which is due to the better propagation of knowledge among nodes.

Impact of the network dynamicity The impact of different nodes speeds is plotted in Fig. 3.6 and Fig. 3.7. For these comparisons, we set the *nc* to 30 connections. Results are plotted for 30 and 50 nodes, evaluating the correlated impact of network density and



(a) PDR (b) Average E2E delay (c) Overhead

Figure 3.6: Impact of nodes speed - nn = 30, nc = 30



(a) PDR (b) Average E2E delay (c) Overhead

Figure 3.7: Impact of nodes speed - nn = 50, nc = 30

dynamicity.

The PDR graphics in Fig. 3.6(a) and Fig. 3.7(a) show that SRS ensures higher PDR than DSR and AODV for almost all scenarios. As expected, when the number of nodes increases from 30 to 50 nodes, the PDR decreases independently of nodes speed. For higher number of nodes, the PDR achieved by our scheme is more significantly higher than the one obtained by DSR and AODV. For *nn* of 50 nodes and nodes speed of 1m/s, AODV is slightly better than our scheme (around 3%). The nodes speed does not impact negatively the PDR of our scheme, contrary to DSR and AODV protocols for which the PDR decreases when nodes speed increases. Indeed, nodes speed can sometimes be considered as a positive factor for SRS scheme, allowing the reception of RREQ and RREP packets by more nodes.

Fig. 3.6(b) and Fig. 3.7(b) plot the average E2E delay for different nodes speeds. As expected, the average E2E delay increases for higher number of nodes. For both densities, the delay of AODV protocol decreases when nodes speed increases. The average E2E delay resulted by SRS scheme decreases for nodes speed of 5m/s, followed by an increase for

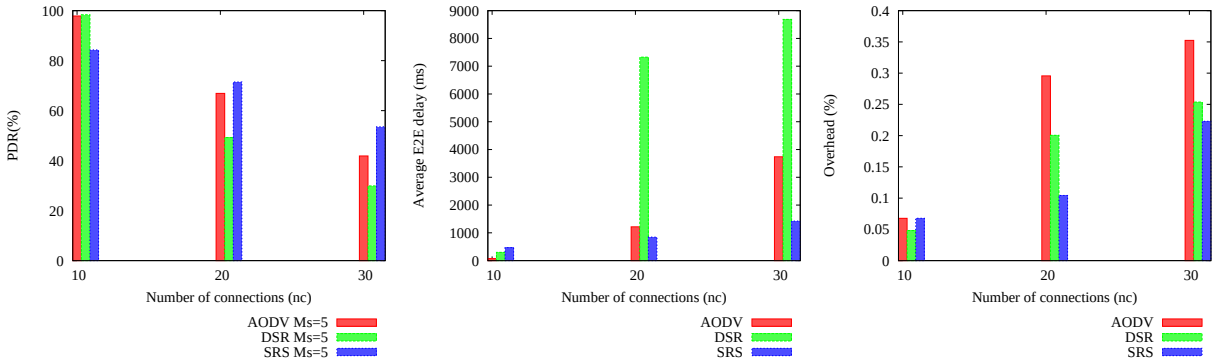
nodes speed of 10m/s and 20m/s. The E2E delay of DSR protocol increases when nodes speed increases, except for nodes speed of 10m/s for which the delay slightly decreases. The average E2E delay resulted by the SRS scheme is significantly better than DSR and AODV protocols independently of scenario used. For all scenarios, DSR protocol has always a delay significantly higher than SRS and AODV which shows the slow adaptation of the DSR protocol. This effect is elevated for nn of 50 nodes.

The overhead graphics in Fig. 3.6(c) and Fig. 3.7(c) show that the overhead generated by our scheme is significantly lower than the one resulted by DSR and AODV. DSR results always in a higher overhead than AODV as it is source-initiated. For DSR and AODV protocol, the overhead increases when nodes speed increase which is due to the failure of current routes and retransmissions. With SRS scheme, the nodes speed can improve the nodes knowledge about path lifetimes, avoiding fragile paths and reducing overall overhead. As expected, higher number of nodes results in higher overhead for all routing schemes and nodes speeds.

Impact of the number of connections Fig. 3.8 and Fig. 3.9 evaluate the impact of different number of connections on performance of SRS scheme compared with DSR and AODV protocols. For these comparison, the number of nodes is set to 50 nodes. Results are presented for nodes speed of 5m/s and 10m/s, evaluating the correlated impact of different network dynamicities.

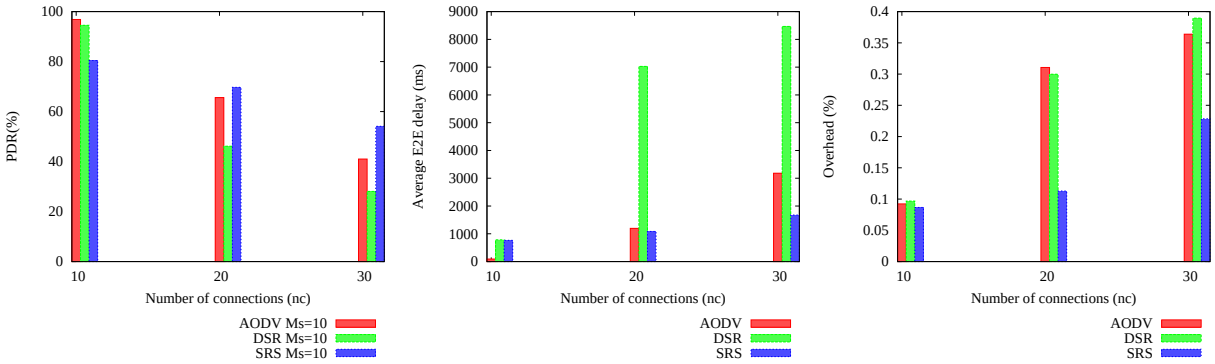
The PDR graphics in Fig. 3.8(a) and Fig. 3.9(a) show that the PDR decreases with higher network traffics for all schemes independently of scenario used. This latter is due to the further medium access attempts and collisions. As expected, the PDR is higher for nodes speed of 5m/s than nodes speed of 10m/s. These figures show that the PDR achieved by SRS scheme is better than the PDR of DSR and AODV for nc of 20 and 30 connections independently of network dynamicity. This effect is more elevated for higher network traffic. The PDR of AODV and DSR are higher than the PDR achieved by our scheme for low number of connections. Indeed, the well performance of the learning-based SRS scheme depends on the quality of the knowledge provided to it. As the autonomic monitoring module of SRS is based on usual packets, the quantity of samples monitored by nodes may be low when the network traffic is light, reducing the network throughput. As a result, the autonomic monitoring module need to be enhanced by other monitoring mechanism when network traffic is low.

Fig. 3.8(b) and Fig. 3.9(b) illustrate the average E2E delay of SRS scheme and DSR and AODV protocols for different network traffics and nodes speeds. As expected, the delay increases for higher number od connections independently of scenario used. The delay achieved by our scheme is significantly lower than the one achieved by DSR and AODV



(a) PDR (b) Average E2E delay (c) Overhead

Figure 3.8: Impact of number of connections - nn = 50, Ms = 5



(a) PDR (b) Average E2E delay (c) Overhead

Figure 3.9: Impact of number of connections - nn = 50, Ms = 10

protocols (up to 200 ms in the worse case and up to 7000 ms in the best case). Similar to results for PDR, the delay of DSR and AODV is better for *nc* of 10 connections and nodes speed of 5m/s. For nodes speed of 10m/s and *nc* of 10 connections, the delay achieved by our scheme is slightly better than the delay of DSR protocol.

The overhead graphics in Fig. 3.8(c) and Fig. 3.9(c) show that the overhead generated by our scheme is significantly lower than the one generated by DSR and AODV protocols for *nc* of 20 and 30 connections. As expected, the overhead increases for higher number of connections independently of routing scheme and scenario used. For all scenarios, the overhead increases when nodes speed increases which is due to frequently link failures for higher mobilities. Similar to results illustrated in Fig. 3.6, and Fig. 3.7, this effect is alleviated for our scheme for which the overhead is very slightly influenced from nodes speed.

3.5.4 Discussion

Results demonstrates that our self-adaptive routing scheme can improve significantly the network performance. When the network traffic is as high as allowing nodes to acquire the required knowledge, the percentage of packet delivery is higher for our scheme than DSR and AODV protocols independently of other network parameters, such as nodes speed and number of nodes. Similarly, the average E2E delay and the overhead obtained by our scheme is lower than DSR and AODV protocol for medium and high network traffic. For low network traffic, results are better for AODV protocol compared to our scheme. However, as our scheme is integrated to DSR routing protocol, we can expect that our scheme will outperform the AODV protocol for all cases when SRS is integrated to AODV protocol. Parameters as node speed, number of connections and number of nodes influence the average E2E delay and overhead, where their higher values increases the delay and network overhead. Our scheme manages better to face the network dynamicity compared to DSR and AODV protocols, for which performances significantly decrease for higher nodes speeds. In general, our scheme outperforms DSR and AODV protocols for all network densities. This latter determines that when the number of nodes' neighbors and hence RNNs' neurons increase, the complexity of the non-linear RNN calculations remains as low as it does not affect negatively network performance.

3.6 Conclusion

In this chapter, we presented SADA, a self-adaptive autonomic architecture which enriches existing autonomic architectures with self-adaptation capabilities. Compared to the existing autonomic architectures applying the reference MAPE-K model, SADA adds two new features: First, it enables the self-adaptation of the analyzing and planning components based on a learning mechanism. Second, it defines an autonomic control loop around the monitoring module which ensures the autonomic functionality of that module.

Inspired from the biological neurones, the proposed self-adaptation process is based on random neural networks with a reinforcement learning which model the efficiency of a past decision using positive and negative spikes. The use of random neural networks enables the network to converge to the optimal configuration using its experiences to enhance future operations. In order to enable the real-time monitoring of the underlying network required by the self-adaptation process, the normal transiting packets in the network are used to disseminate the necessary global views with a minimum extra overhead.

We investigated the efficiency of the SADA architecture by the proposal of a self-adaptive routing scheme, called SRS, for mobile ad hoc networks. This scheme is protocol-independent and its goal is to self-adapt the routing discovery process, with a minimum

extra overhead. SRS is based on random neural networks with reinforcement learning, modeling the expected paths performance via any next hop.

The proposed self-adaptive routing scheme was instantiated on DSR routing protocol and compared with DSR and AODV protocols. We evaluated the efficiency of the proposed scheme by simulations considering different network densities, nodes speeds and number of connections. Three metrics evaluated the efficiency of the SRS scheme, measuring its ability to deliver data, latency and overhead. Simulation results showed a significant improvement in the generated overhead. Further, the packet delivery ratio has been enhanced with medium and high network traffics. The use of self-adaptive routing scheme yielded delays mainly lower than those produced by other evaluated protocols.

Chapter 4

Autonomic Trust Monitoring Scheme

Contents

4.1 Introduction

In the area of autonomic computing, self-protection is considered as one of the main self-management properties. It is defined as the ability of the system to protect itself from what compromises it from achieving its goals. It involves the protection from malicious attacks, intrusion tentative or inadvertent failures. Security must be considered as a cornerstone requirement and not as an added value service. The experience has shown that it is difficult to add security to a protocol suite unless it is in-built into the architecture from the very beginning. This chapter holds the problem of trust management required by autonomic architectures in the context of mobile ad hoc networks (MANETs).

MANET is a self-organized wireless network formed by a set of mobile devices communicating among themselves without any established infrastructure. Because of the distributed nature of MANETs, nodes must collaborate with each other to support the functions of the network. However, due to the self-organized nature and insufficient resources, nodes in a network can behave selfishly or maliciously for individual interests, for example, refuse to forward packets. Trusting on a misbehavior node can lead to unforeseen pitfalls [85], such as slow network efficiency, high resource consumption and vulnerability to attacks. From the autonomic architecture standpoint, the operation of the architecture depends on the cooperation among individual autonomic managers. Although the collaboration of autonomic managers is an usual assumption in autonomic networks, it is not so obvious that the collaboration exists in practical networks. Hence, a trust management mechanism becomes necessary [86] to allow nodes infer how much they can trust on other nodes' behavior.

Trust management consists of three main components: *knowledge collection*, *trust level computation* and *trust establishment*. The knowledge collection component provides the information about nodes' behavior. The computation component calculates a trust level for each entity based on the collected behavioral data or trust evidence. The result is the assignment of a trust level, representing how much a node can trust in others. The trustworthiness establishment component infers if a node can be trusted based on its trust level.

This chapter demonstrates the application of the proposed autonomic monitoring control loop proposed by the SADA architecture by the proposal of ATMS, an Autonomic Trust Knowledge Monitoring Scheme for mobile ad hoc networks. The proposed solution uses both local and remote information, providing a uniform view about nodes' trustworthiness with a minimum extra overhead. ATMS adapts itself according to the network conditions, based on the proposed autonomic monitoring control loop defined by the SADA architecture.

This chapter proceeds as follows. Section 4.2 gives some motivation on the mutual

impact of the autonomic architecture and the trust management. Section 4.3 reports related works, highlighting existing monitoring schemes for trust management in the context of MANETs. Section 4.4 presents ATMS, emphasizing its knowledge management strategy. Section 4.5 details the simulation environment, used metrics and results of the ATMS evaluation. We discuss achieved results in section 4.6. Finally, section 4.7 concludes the chapter.

4.2 Motivation

The autonomic computing and trust management could be considered as complementary approaches, i.e. in one hand, autonomic computing can help optimizing a trust management solution, and in the other hand, the trust management can optimize the efficiency of autonomic network architectures, considering trust relationships in communications among autonomic elements.

We highlight that the trust management among autonomic elements can be considered as an important feature of autonomic communication. Indeed, the operation of network depends on the cooperation among individual autonomic managers. The penetration of an attacker in the cooperation cycle could damage hardly the overall performance of the network. A malicious attacker may execute damaging adaptation to the network. Hence, the performance of the network decreases locally. In an ultimate stage, this suboptimal performance causes abnormal changes and slant input to decision and learning algorithms of other managers. Therefore, the global view becomes slant and the overall network performance will become sub-optimal.

Furthermore, the accuracy of the adaptation decisions depends on the accuracy of the perception of the surrounding environment. The malicious node may either report incorrect information to managers or alter a passing message in order to tamper other manager's perception. In addition, it can penetrate to the management overlay and report forged knowledge to other managers and distort their perception. Hence, the manager may decide incorrectly which could alter step by step other managers' view and lead to cascade failures. Consequently, the performance of network decreases and the service availability could be menaced by Denial of Service (DoS) attacks, for example. In addition, an attacker may simply declare itself as an autonomic manager without any further malicious action. Thus, other managers will communicate with this infiltrated attacker for cooperation purpose. Also, the attacker may communicate to other managers to avoid being suspected. This generates supplementary overhead menacing the service availability.

4.3 Related work

Existing trust management frameworks for MANETs can be separated into two groups according to the kind of information collected by the knowledge collection component [85]. In the first group, trust is established only based on local information, namely, the information nodes have collected by themselves on the behavior of their neighbors. The other group comprises trust management frameworks that use both, local and remote information. The remote information is referred to as recommendations, consisting in the opinion of other nodes about the trustworthiness of a given node.

Most of existing trust management frameworks for MANETs are based on local information [87, 88]. Generally, nodes evaluate the trust level of their neighbors by a *watchdog* mechanism. The watchdog mechanism is implemented by comparing sent packets with overheard packets to verify if there is a match. That is, one node will buffer all packets it has sent, then overhear its neighbors's action. If the node hears that one sent packet was correctly forwarded by a neighbor, it considers a normal behavior for that neighbor. Otherwise, one misbehavior has been observed. Although the overhead of the local-based trust management frameworks could be negligible, they are prone to double-face attackers [89], i.e. nodes that behave selfishly in a position (or regarding a node) and normally in another position (or regarding another node) in order to remain undetected. These attacks cannot be detected only based on local information. Hence, in order to enforce the knowledge a node has about a neighbor, some existing proposals in literature rely also on a second-hand information transmitted by other neighbors [85, 89, 90] trying to minimize the vulnerability to double-face attacks. In general, they employ trust estimation approaches to demotivate such double-face conduct.

In [91], Buchegger and Le Boudec investigated the trade-off between robustness and efficiency of reputation systems in mobile ad hoc networks. They showed that taking into account the recommendations of other nodes can speed up the maliciousness detection.

Li et al. [89] also stated that using only local information gives an incomplete partial solution for trust management. They proposed an objective trust management framework (OTMF) for MANETs based on both direct and indirect information. The direct observation is obtained by a watchdog mechanism.The second-hand information can be formed periodically between neighbors and then used by other nodes. After the formation of second-hand information, it is flooded in the network. The nodes receiving this information will check it by a deviation test and use the trustworthiness of the information provider in the recommendation framework as the weight for this information when necessary. Although this mechanism ensures a uniform distribution of trust values over the network, it generates a significant amount of overhead to MANETs.

In [85], Velloso et al. proposed a human-based trust model, which builds a trust relationship between nodes in an ad hoc network based on previous individual experiences and on the recommendations of others. A Recommendation Exchange Protocol (REP) was proposed allowing nodes to exchange recommendations about their neighbors. The REP only considers interactions with neighbors, which makes the protocol scaling well for large networks. Recommendation can be obtained by sending a Trust Request (TREQ) or by receiving a Trust Advertisement (TA) message from other neighbors. TA messages are unsolicited recommendations. A node only sends a TA message when the recommendation about a particular neighbor varies more than a certain threshold value. Thus, all messages are transmitted by one hop broadcasts avoiding flooding in multi-hop communications. However, since the nodes are only aware of neighbors trust values, the detection of mobile malicious nodes can be time-consuming, especially regarding double-face attackers.

G. Bella et al. [92] presented a protocol to allow a node to evaluate the reputation of another one by means of both direct observation and recommendations received from other nodes. A Global Reputation Table (GRT), which contains the node's view on neighbors and far nodes, is periodically exchanged with the others. Table information is not broadcasted all over the net with a flooding procedure, but when a node receives a table from one neighbor, it calculates new values, and then, with a prefixed schedule, it sends the new GRT to its neighbors. This kind of sharing limits the traffic in the net, avoiding the overload and limiting the use of energy. However, it can involve a non-uniform distribution of the same value.

4.4 Autonomic Trust Knowledge Monitoring Scheme (ATMS)

A mechanism for trust value monitoring should provide an uniform view about trustworthiness of nodes on the network, allowing nodes to adapt their communication with another node according to its trustworthiness. However, due to the dynamic nature of MANETs and their vulnerabilities, the trust values of nodes may change frequently. Each node should be aware of updated trust value of other nodes as it may enter to a communication with the other. Thus, the information should be distributed to nodes in a way that all nodes have always a same view on other nodes. The uniform view requires the real-time monitoring of trust values all over the network, allowing nodes to take decisions based on up-to-date trust values.

In order to inhibit double-face conduct attack on a trust establishment framework, we propose an autonomic trust monitoring scheme (ATMS) to establish trust relations between pairs of nodes in autonomic MANETs. This work takes into account autonomic principles of the SADA architecture in order to propose a new self-adaptive and protocol-independent

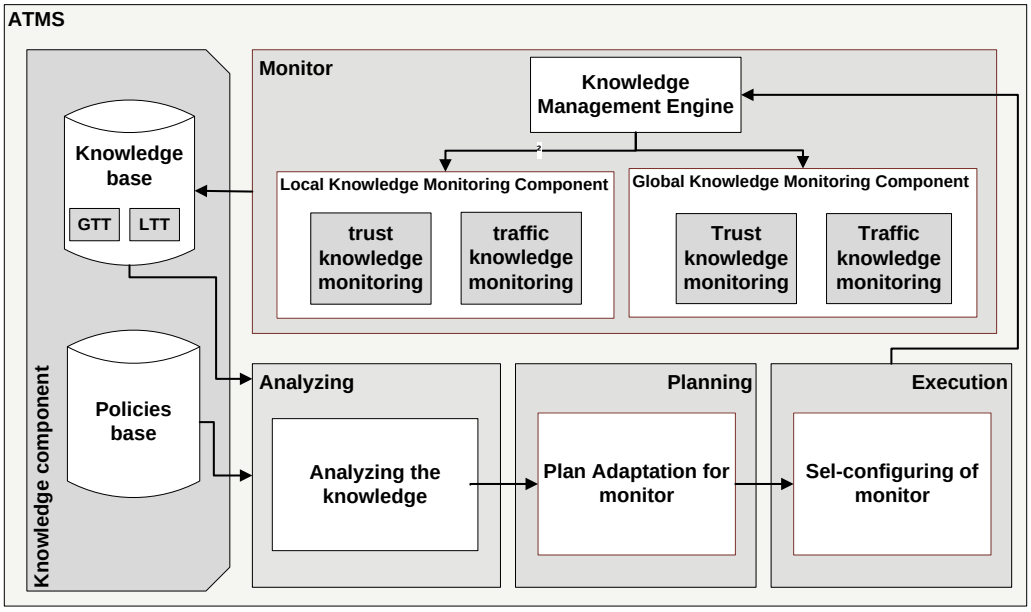


Figure 4.1: Proposed autonomic trust management framework

trust management scheme, improving the trust establishment between nodes and promoting the collaboration in MANETs. The proposed scheme ensures a uniform distribution of trust values among nodes while maintaining a minimum generated overhead. As illustrated in Fig. 4.1, the proposed approach is comprised of five main components: *monitor*, *knowledge*, *analyzing*, *planning* and *execution*.

4.4.1 The Monitor Component

The monitor component is in charge of collecting local and global information required for establishing trust relationship among nodes. The monitor is composed of a knowledge management engine, a local and a global trust monitoring modules. The local trust monitoring module performs a local observation of neighbor nodes. In order to avoid the aforementioned drawbacks of promiscuous mode, a node obtains information on trustworthiness of a neighbor based on amount of traffic it receives from that neighbor. When a node i receives a packet from a neighbor j, it checks if the packet was generated or forwarded by that neighbor j. The node i can obtain this information just by observation of the IP header, which contains the source address of the packet.

A node i evaluates the trustworthiness of a neighbor j considering the amount of traffic it forwarded compared to the amount of traffic it generated. From the point of view of node i, this ratio represents the degree of unselfishness in the use of the link $j-i$. The basic idea is that a node reputation grows only if it forwards packets of another sender source. The node under analysis has no interest in that connection: on the contrary, it uses its energy

only to forward other packets. The information acquired by local observation is stored in a table, called Local Trust Table (LTT). This table contains one entry by neighbor for which it stores the data amount generated and forwarded by that neighbor as well as the local trustworthiness estimated for that neighbor.

The global trust monitoring module is in charge of exchanging information with other nodes. Based on the table LTT, every node of the network constructs a Global Trust Table (GTT) which is gradually completed by trust values of all network nodes. To populate the GTTs with information on all participants to the MANET, nodes should exchange their GTT tables by some communication procedures.

In order to avoid the significant overhead generated by flooding procedure, we propose a knowledge management engine which uses data or signaling packets traveling in the network to exchange such information. A packet can carry information on nodes, and update it continually when visiting other nodes. The global view of each intermediate node will be updated based on information carried by arriving packets. The knowledge management process decides on when it piggybacks trust information on transiting packets based on the network state. Indeed, when the network is not overloaded, the knowledge management engine activates the *normal mode* in which all transiting packets are used to distribute trust level information in a timely manner.

However, when the network is congestionned, the *safe mode* it activated which uses only some transiting packets to carry trust information, reducing the trust distribution overhead. The state of network is determined by analyze component described in section 4.4.3 based on the local traffic rate. This latter is monitored by *traffic knowledge monitoring* component and denotes the number of packets received over an interval of time. If analyze component detects a congestion, it activates the plan component which switches the knowledge management engine to the safe mode, defining the rate of generating monitoring packets.

4.4.2 The Knowledge Component

The knowledge component consists in two elements: a knowledge base, in which LTT and GTT tables are stored, and a policy base, in which a set of predefined condition-event rules are stocked. The policies allow the framework to analyze its current state and adjust its monitoring rate accordingly. For instance, we employs a policy which *reduces the generation rate of knowledge monitoring* when *the observed local traffic rate exceeds a congestion threshold* α. This adaptation reduces the overhead of monitoring and is mandatory to avoid serious congestion collapse and increase network performance.

4.4.3 The Analyzing Component

The analyzing component periodically verifies if any predefined policies threshold is exceeded considering the knowledge provided by the monitor component. If this is the case, it activates the planning component in order to react to that event. In our case, if the traffic rate exceeds a congestion threshold α, the planning component will be activated to adapt the generation rate of the monitoring component according to the network condition.

4.4.4 The Planning Component

The planning component executes the knowledge monitoring optimization algorithm (KMO) when the traffic rate exceeds a congestion threshold α. In this case, the global knowledge monitoring process will be executed with a certain probability p_i, denoting whether transiting packets should be piggybacked or not. This probability is inversely proportional to the network average trust level retrieved from the node GTT table to ensure that nodes obtain sufficient information of nodes trustworthiness in a case of poor network trustworthiness. That is, if the trust level of the network is low, the p_i is set up to to a high value to use most transiting packets and enforcing trust knowledge. Every time intervals of t seconds, the KMO switches the knowledge management engine to the normal mode if it detects that the congestion is finished.

4.4.5 The Execution Component

The execution component is responsible for enforcing decisions taken by planning component. That is, when the planning component decides a new value for monitoring generation rate p_i, the execution component configures the knowledge management engine with this new value.

4.5 Evaluation

In order to show the effectiveness of the proposed approach, we carried out a set of simulation experiments using NS-2 version 2.32. Our trust knowledge monitoring scheme was instantiated on the trust management framework proposed in [92] which uses Dynamic Source Routing (DSR) as routing protocol. This original framework was modified to execute our proposed monitoring and knowledge components. The protocol presented in [92] is referred in this paper as Bella framework in honor of its author. Hence, analysis compare results provided by the ATMS with those yielded by the original Bella, in the presence of double conduct attacks.

Table 4.1: ATMS - Simulation parameters

Parameter	Value
Transmission Rate	11 Mbps
Radio Proagation Model	TwoRay Ground
Mobility Model	Random waypoint
Network area	500m x 500m
Transmission Range	100m
Interface queue size	64 packets
node number (nn)	20, 30, 50
CBR rate	4 pkt/s
CBR sources (nc)	5, 10
Data packet size	512 bytes
Maximum speed (Ms)	2, 10, 20 m/s
Pause time	5s
Simulation time	1000s

4.5.1 Environment settings

The IEEE 802.11 distributed coordination function (DCF) is used as medium access control (MAC) protocol. The radio model takes into account similar characteristics to a commercial Lucentï¿½s WaveLAN radio interface with a nominal bit-rate of 11 Mb/s for the shared-media radio and nominal radio ranges of 100 meters. The mobility model applied is the random way-point model, where node speeds are randomly chosen between 0m/s and a maximal speed of 2m/s, 10m/s or 15m/s evaluating the network with different dynamicity. The pause time is fixed to 5s. The network area dimensions were fixed for all simulations to 500 meters square. The total number of nodes (nn) placed randomly in this area is 20, 30 or 50 nodes, characterizing networks with different densities. 10% of double-face attacker nodes are chosen randomly from the total number of nodes in the beginning of each simulation. An double-face attacker node is a source node which makes use of network but starts dropping received packets after a period of time of normal behavior.

The data traffic used is CBR (Constant Bit Ratio) with a number of connections (nc) equal to 5 or 10 defined randomly. Each source node generates 4 packets/sec with a packet size of 512 bytes. Data traffic sessions happen at a random time in the simulation. The total simulated time was 1000 seconds and each plotted point is an average of 30 simulations with a 95% confidence interval. Table 4.1 summarizes main simulation parameters.

We investigate two aspects on our knowledge monitoring scheme: its impact on the **network performance** and the **knowledge quality**. Hence, we employ two metrics related to network performance and three metrics related to the knowledge quality in our evaluations. As network performance metrics, we have the *Packet Delivery Ratio (PDR)* and the *Average E2E delay*. The former consists in the number of data packets successfully

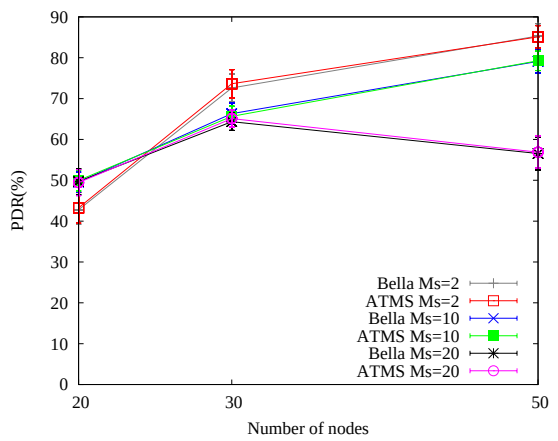
delivered at the destination divided by the number of data packets sent from the source. The latter lies in transmission delay of data packets delivered successfully, consisting of propagation delays, queuing delays at interfaces, retransmissions delays at the MAC layer, as well as buffering delays during the route discovery.

As knowledge quality metrics, we employ the *average trust*, the *trust standard deviation* and the *correctness*. The first metric consists in the average trustworthiness of a node estimated by all other nodes in the network. This metric shows the trend of the trust estimated for a normal or a misbehavior node. Average trust should converge to a high reference value for a normal node and to a low reference value for an attacker. The reference value should ideally converge to 100% for a normal node integrated to the network without generating any packets. For a source attacker which makes use of network resources without collaborating to other nodes transmission, the reference value should ideally converge to 0%. The second metric lies in the standard deviation of average trust of nodes estimated by all other nodes in the network. This metric shows the uniformity of the trust knowledge estimation among all nodes in the network. Ideally, the standard deviation should converge to 0, describing an uniform trust estimation about all nodes through the network. Finally, the third metric consists in the number of nodes having a correct perception on other nodes trustworthiness. This metric allows comparing the accuracy of global trust knowledge. Ideally, the correctness metric should converge to 1, showing a correct view about other node trustworthiness through the network. In our analysis, we considered a node as trustworthy when its trust value is higher than 70%. A node with trust value lower than 30% is evaluated as a misbehavior node.

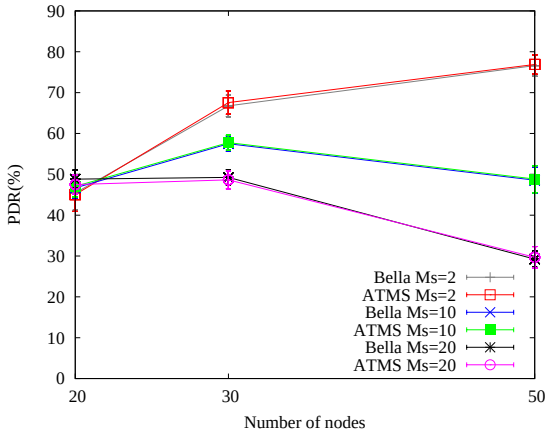
4.5.2 Results and analysis

Results of the trust monitoring scheme are presented on two perspectives, **network performance** and **knowledge quality**. For network performance perspectives, we compare ATMS and Bella schemes considering different percentage of nodes in the network and the impact of different parameters, such as nodes speed and number of connections. For knowledge quality metrics, the efficiency of ATMS and Bella schemes are compared over time. In this paper, knowledge quality results are presented only for 50 nodes.

Both of perspectives present one case with *nc* of 5 (low traffic) and one case with *nc* of 10 (higher traffic). The low number of connections may have double contradictory impacts on our scheme. On one hand, due to lower number of connections and thus lower generated data packets, lower total supplementary trust related traffic will be piggybacked reducing overhead. On the other hand, estimated trust values will converge later since there are less packets updating these values. The former impact can be observed on performance results and the latter on knowledge quality results.

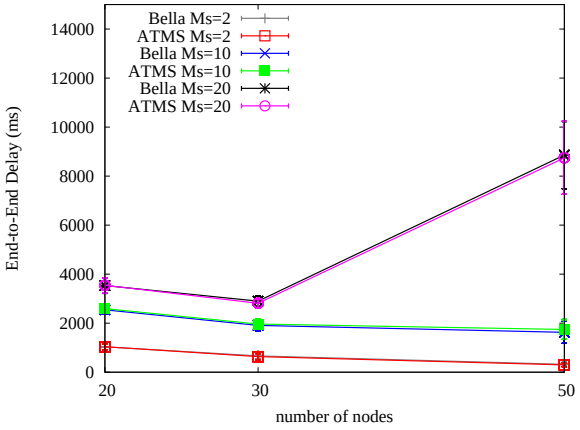


(a) PDR - nc = 5

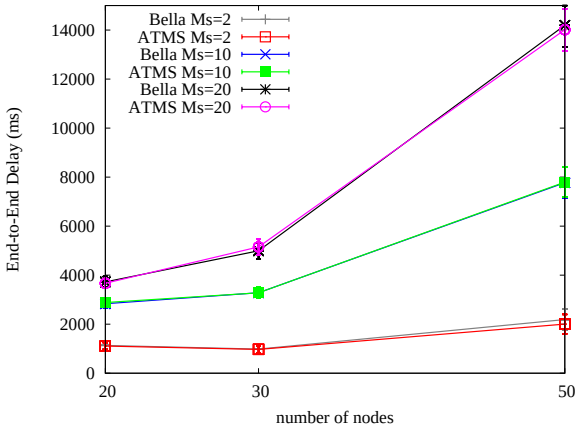


(b) PDR - nc = 10

Figure 4.2: Impact of number of nodes on PDR



(a) Average E2E delay - nc = 5



(b) Average E2E delay - nc = 10

Figure 4.3: Impact of number of nodes on average E2E delay

4.5.2.1 Performance results

The PDR graphics in Fig. 4.2 show that the trust management framework with our scheme (ATMS) presents minor differences than Bella. This behavior was observed on different scenarios. As expected, higher number of nodes results in lower PDR when the maximum speed of nodes or number of connections is high. Node speed higher than 10m/s decreases the PDR and its effect its more elevated for *nc* of 10 connections. Moreover, higher traffic results in lower PDR for all cases. Node speed and number of connections affect both schemes similarly as shown in these figures. With speed of 10 et 20m/s and *nc* of 10 connections, the PDR achieved by ATMS is slightly better than the PDR of Bella in the same scenario.

The E2E delay graphics in Fig. 4.3 show that both schemes increments the delay when

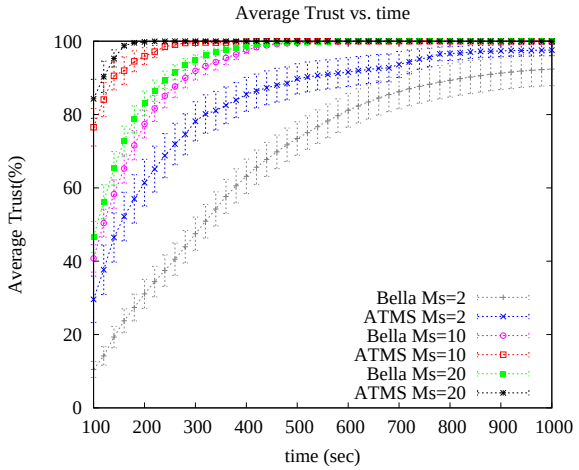
network traffic increases. Higher network density increases the delay for *nc* of 10 connections. This effect can also be observed for 5 connections when number of nodes is 50 and node speed is 20m/s. E2E delay resulted by ATMS is better than Bella for all node speeds. For 5 connections, the delay is always slightly improved for our scheme, except for node speed of 10m/s. When 10 connections are used, ATMS slightly improves the average E2E delay except with 30 nodes moving with a maximum speed of 20m/s.

These results can be considered as satisfactory since all knowledge quality results presented in Section 4.5.2.2 show improvements. Moreover, further improvements in performance metrics are expected when a trust establishment component will be integrated to schemes, establishing (or not) trust relationships with other nodes based on their estimated trust value. In this case, our scheme is more likely to trust (or not) correctly as it holds a better knowledge quality resulting in performance improvements.

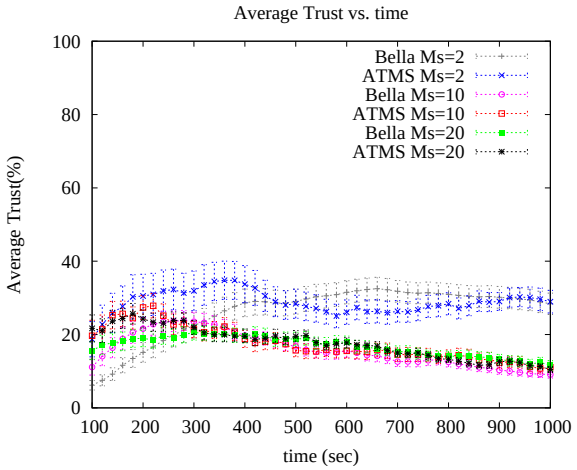
4.5.2.2 Knowledge quality results

The results of knowledge quality metrics are illustrated in Fig.4.4 and Fig. 4.5. The average trust graphics in Fig. 4.4(a) and Fig. 4.5(a) show that our scheme manages to get always a higher average trust than Bella for a normal node independently of scenario used. As expected, this improvement is more relevant for higher number of connections. For a same scenario, the estimated trust value is higher compared to Bella up to 30% in some cases. As expected, average trust value increases over time for all scenarios since nodes trust knowledge of nodes increases over time. For the same reason, average trust of different scenarios trends to converge to a maximum trust value over time. Our scheme manages to converge faster than Bella independently of scenario used. As expected, the time to converge to a perfect trust value is lower with higher number of connections. Both schemes result in higher average trust when nodes speed increases independently of the number of connections which is due to further contacts with other nodes.

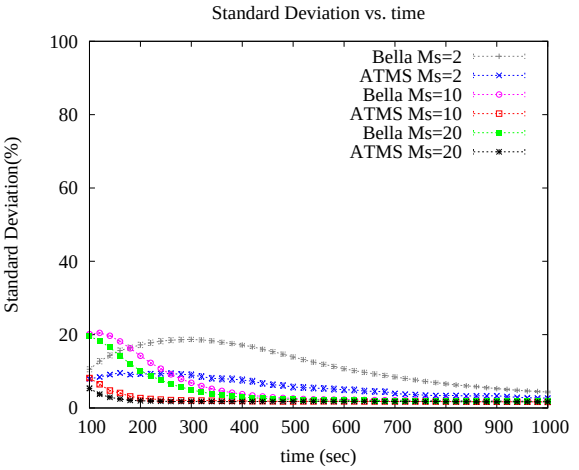
Fig. 4.4(b) and Fig. 4.5(b) plot the trend of the average trust over time for a double-face attacker node. The double-face attacker node under observation initiates its attacks after around 120 seconds of simulation time. As expected, the average trust increases in the beginning of simulation followed by a decrease after a certain time for all scenarios. The decrease starts when nodes detects some misbehavior. With higher *nc*, the average trust decreases earlier and faster in time for all scenarios. As expected, *nc* of 10 results in higher average trust at the beginning phase (before the attack starts) than *nc* of 5. Hence, it takes a bit more time to reach the minimum value for scenarios with *nc* of 10 than scenarios with *nc* of 5. As shown by these figures, our scheme detects the attack earlier than Bella for a same scenario independently of number of connections and maximum speed used. For Ms of 2m/s, both schemes takes more time to detect the problem independently of number



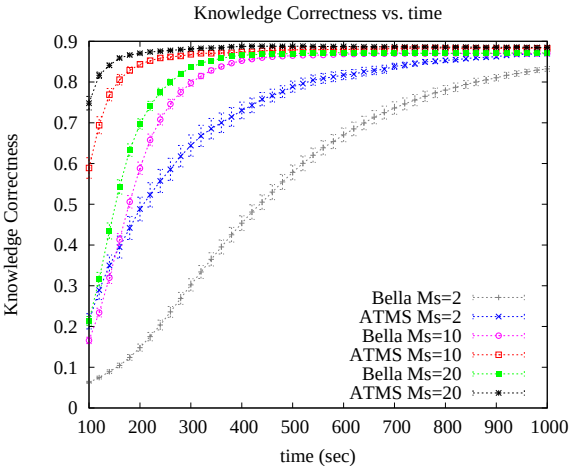
(a) Average trust of a normal node



(b) Average trust of an attacker node



(c) Standard deviation

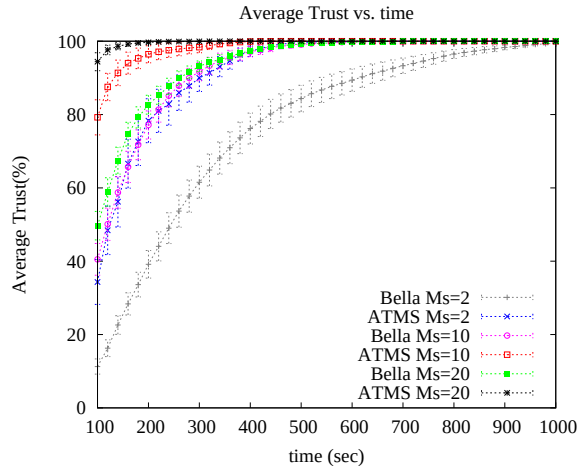


(d) Correctness

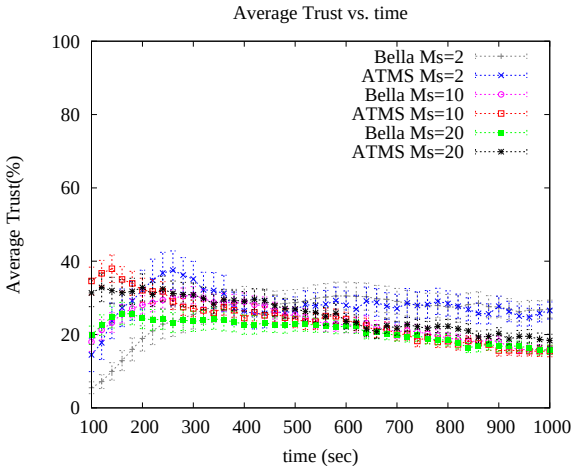
Figure 4.4: Quality of Knowledge trend over time - nc = 5

of connections. This problem is elevated for lower traffic. However, our scheme detects the attack earlier than Bella in these situations (Bella detects the attack only after 650s of simulation time).

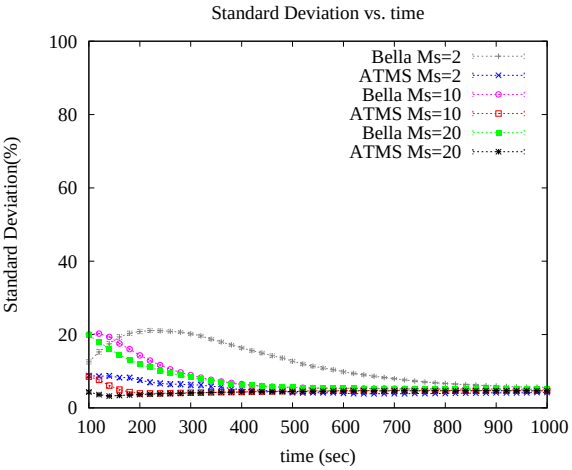
Fig. 4.4(c) and Fig. 4.5(c) illustrate the trust standard deviation among all nodes. As expected, the trust standard deviation decreases for all scenarios over time independently of number of connections and maximum speed. The standard deviation decreases when maximum speed increases which is due to further contacts between nodes resulting in a more uniform knowledge. The minimum trust standard deviation obtained by *nc* of 10 is slightly higher than the one with *nc* of 5 which can be due to congestion problem in case of high traffic. The trust standard deviation is lower with *nc* of 5 before the



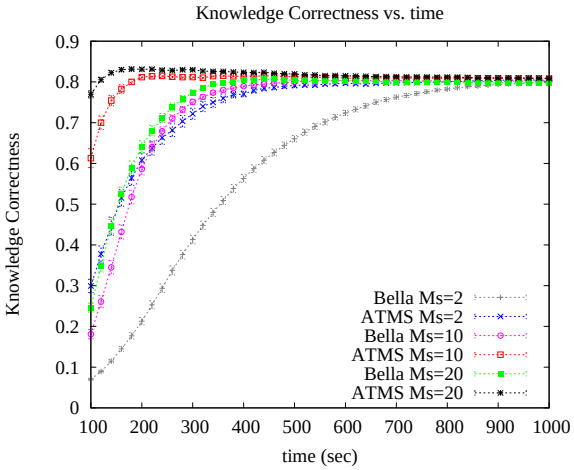
(a) Average trust of a normal node



(b) Average trust of an attacker node



(c) Standard deviation



(d) Correctness

Figure 4.5: Quality of Knowledge trend over time - nc = 10

convergence. Independently of number of connections, the standard deviation with Bella scheme increases significantly under node speed of 2m/s for an important period of time. This effect is negligibly (less than 2%) observed for our scheme for node speed of 2m/s and *nc* of 5 connections. As shown by these figures, our scheme presents significant lower standard deviation compared to Bella scheme which reveals that ATMS provides nodes with a more uniform trust knowledge.

Fig. 4.4(d) and Fig. 4.5(d) show the correctness of trust knowledge. As expected, the correctness increases over time for all scenarios independently of number of connections and maximum speed. Similarly to previous results, the correctness increases with higher speeds for a same scenario. As expected, more number of connections results in faster convergence

to the maximum achievable bound independently of nodes speed. This maximum bound is lower with nc of 5 connections due to the congestion problem. This effect has been also confirmed by comparing results on trust standard deviation with nc of 5 and nc of 10. As expected, we observe higher correctness for nc of 10 connections for a same nodes speed before the convergence time. These figures demonstrate that the correctness is significantly better for our scheme than Bella scheme independently of the value of M and nc.

4.6 Discussion

Results demonstrate that our trust knowledge monitoring scheme can improve significantly the trustworthiness estimation of nodes. For all cases, the trust standard deviation is lower for our scheme than Bella which proves the higher uniformity of trust knowledge hold by nodes. Similarly, its correctness is better than the correctness achieved by Bella scheme for all cases. The average trust on a double-face attacker and on a normal node converge faster to the correct value for our scheme than for Bella scheme. The impact of our scheme in terms of overhead is implicitly presented by performance metrics. For all cases, its end-to-end delay is lower or equal to the delay of Bella scheme. In general, the percentage of packet delivery is the same achieved by Bella scheme. Parameters as node speed and number of connections influence the packet delivery ratio, where their higher values decrease the delivery. The relevant knowledge quality obtained is a promising asset for our scheme allowing ATMS to decide more correctly about the trustworthy of nodes.

4.7 Conclusion

This work described an initiative of application of autonomic principles on trust management of mobile ad hoc networks. In this context, we proposed ATMS, an autonomic monitoring scheme for trust management, based on the SADA architecture. Its main goal is to provide an uniform up-to-date trust knowledge throughout the network with a minimum monitoring overhead and reduce the impact of double-face attacks. The autonomic monitoring control loop of the SADA architecture enabled the scheme to self-adapt to network conditions. This latter is composed of knowledge monitoring and analyzing phases, followed by adaptation planning which self-adapts the monitoring processes according to the network context. ATMS is characterized by its protocol and trust framework independence, self-adaptation, simplicity and low computational intensiveness.

The trust monitoring scheme was instantiated on Bella framework and compared with it. Two classes of metrics evaluated the effectiveness of the scheme, measuring its impact on network performance and on trust knowledge quality. Simulation results showed a

significant improvement in trust knowledge quality with satisfiable network performance in terms of delay and packet delivery ratio.

Chapter 5

A Methodology for Evaluation of Autonomic Network Architectures

Contents

5.1 Introduction

Autonomic networking is a promising approach towards the optimal management of network resources. It aims at self-managing communication infrastructures, reducing the complexity and the cost of network management. An enormous yet dispersed autonomic network management (ANM) architectures have been already proposed, each one using a different approach to overcome the management limitations of current models. However, a literature search does not yield relevant progress in evaluating and comparing ANM architectures. On one hand, the evaluation scope used by each autonomic architecture was only related to some separate QoS metrics. On the other hand, the existing architectures have been considered in a stand-alone manner without being compared to any other autonomic proposal. We believe that instead of overflowing the field of autonomic architectures with a huge number of management approaches, we should concentrate them together and draw the approach(es) which is(are) more appropriate for a given case. In order to converge to such a reference model, a comparative study between autonomic network architectures is thus required. This context motivated us to delve into the evaluation aspect of autonomic network architectures.

This chapter identifies the main quantitative evaluation criteria important to evaluate the efficiency of autonomic architectures. Based on the identified evaluation criteria, it describes a quantitative methodology for evaluating and comparing autonomic network architectures in an unified manner. Our methodology is based on fuzzy-logic and correlates the identified evaluation criteria to obtain an overall measure of autonomicity. This autonomicity score can be used to compare quantitatively different autonomic architectures and classifies them regarding to the network management efficiency and performance.

The remainder of this chapter is organized in five sections as follows. Section 5.2 reports existing initiatives on evaluating autonomic architectures. Section 5.3 identifies the main evaluation criteria and describes how they can be measured. Section 5.4 presents the proposed methodology of autonomicity evaluation based on fuzzy-logic and its phases. Section 5.5 describes how the proposed fuzzy-logic evaluation methodology can be used to compare autonomic network architectures. Finally, section 5.6 concludes the chapter.

5.2 Related Work

IBM proposed a set of Autonomic Computing Adoption Model Levels [93] to describe qualitatively the degree of autonomicity of a system. The classification is based on how much the administrator is involved in management tasks. The *Basic level* is where the system is managed by high skilled staffs who uses monitoring tools that allow them to discover the network's state and make required changes manually. The *Managed level*

presents a system which performs the monitoring in an intelligent way that reduces the system administration burden and simplifies for administrators to plan accordingly. In the *Predictive level*, the monitoring is performed in a way that enables the system to recognize its behaviour and suggest actions approved and carried out by the IT staff. The *Adaptive level* is where the system is able to suggest actions and carry out certain adaptation without human involvement. Finally, the *Autonomic level* refers to a full autonomic system where monitoring and adaptation are performed automatically.

In [5], the authors defined four qualitative elements of autonomicity as follows: Support, Core, Autonomous and Autonomic. The *support element* is when one particular component affecting the performance of a complete autonomic architecture is considered. An autonomic monitoring tool is an example of the support class. The *Core element* describes an end-to-end self-management solution for a core application. An end-to-end IP-mobile solution to support the mobility of terminals could be considered in this class. The *autonomous element* is where a full end-to-end emergent intelligent solution is produced, taking into account more than one application. In this level, the system is not measuring its own performance and adapting how it carries out its task to better fit some sort of performance goals. At *autonomic level*, a full generic autonomic architecture is considered which reflects its own performance and adapts itself accordingly. In this level, the administrator participates only in initial parameterizing of the autonomic manager(s) and it is up to the network to operate smoothly afterwards.

In [94], we presented a coherent classification methodology to qualitatively compare ANM architectures. A set of qualitative evaluation views has been proposed which consists in the category of architecture, adaptation approach, monitoring approach, learning ability, security, etc, just to mention a few. The proposed evaluation views were used to compare existing ANM architectures (see chapter 3 for further details).

In [95], a set of metrics for evaluating autonomic computing solutions is provided. These metrics include QoS, cost of autonomy, granularity/flexibility, failure avoidance, etc, just to mention a few. However, the authors did not specify the concrete scope of each metric and how one can measure the identified metrics.

In [66], the authors identified a set of evaluation views to capture various performance aspects of a distributed ANM architecture. Each view is described for the context of Mobile Ad Hoc Networks (MANETs). The outlined evaluation views include the degree of self-operation, performance of autonomic response (QoS), cost of autonomy, speed of autonomic response, sensitivity to change and granularity. The authors proposed the use of *radar charts* as a graphical display for comparing between different views of several systems. However, each proposed evaluation view is considered separately and without further description on how it can be quantitatively measured.

Table 5.1: Evaluation metrics identified in the literature

Metric	Definition	References
performance of autonomic response (QoS)	A mean to conclude the degree to which the system is reaching its primary goal to improve some aspect of a service.	[95], [66]
Cost of autonomy	The overhead of extra autonomic activity.	[95], [66] and [96]
Granularity/flexibility	The degree of decentralization of the architecture as a trade-off between the failure avoidance and overhead.	[95], [66]
Failure avoidance	The ability to cope with predicted and unpredicted failures.	[95]
Degree of self-operation	In which extend the architecture (or each MAPE-K components) performs in itself without administration involvement.	[95], [66], [23] and [96]
Speed of autonomic response	How fast the system adapts itself to a change. It includes the total time to adapt, the reaction time and the stabilization time.	[95], [66]
sensitivity to change	The reaction time, taken between an event is occurred until executing an adaptation response to cope with the event. Ideally, a reasonable observation time, not too short neither too long, is needed to avoid oscillations.	[95], [66]
Memory strength	The ability of the system to maintain current state, behaviour trend or historic of past actions in order to utilize them for management decisions.	[23]
Degree of activity	Reactive or a proactive behaviour of the system to satisfy the self-management properties. Proactivity is critical in computing systems where consequences of slight performance degradation or sudden failures are at higher costs than that of computing future states. Obviously, a favorable ANM architecture is highly proactive rather than reactive.	[23]
Ability to learn	The ability of the architecture to continuously evolve and enhance their applied adaptation strategies.	[23]
Degree of awareness	The degree of self-awareness and environment-awareness of the architecture.	[23]
Granularity of intelligence	The degree of distribution of the intelligence in the network, i.e. the granularity of learning mechanisms.	[23]
Benchmarking	A mean to bring all above-mentioned metrics together and observe their effective impact on the smooth operation of the real system.	[95]

Six classes of properties for enabling autonomic principles are presented in [23]. These properties are degree of activity, ability to learn, granularity of the intelligence, degree of awareness, memory strength and degree of self-operation. Nevertheless, the authors did not describe how the proposed evaluation criteria can be measured.

The authors of [96] stated the need for synthesizing multiple sub-metrics into an overall measure of autonomicity, and calibrating scores across different systems.

Table 5.1 provides a concise definition of the metrics outlined in the literature. We merged some metrics in order to present a consistent view of the state-of-art in the field. This is done when several representations are used for two or more metrics referring to a same ability or functionality of the system. As shown in the table, the existing criteria do not spread over all aspects that an AMN architecture should cover. Moreover, the

literature provides only a generic definition of the intended metrics, without any further description of its scope. Consequently, some metrics present too generic aspect to be able to be measured. In addition, they do not describe how one can measure the mentioned metrics or apply them to compare quantitatively the ANM architectures.

In this chapter, we present a coherent classification of the main evaluation criteria that contribute to the autonomicity of the architecture. We advance the state-of-art in the evaluation field by refining some existing metrics and defining some new evaluation criteria to cover all aspects of autonomicity. In addition, we propose a fuzzy-logic based approach to correlate individual sub-metrics to provide a unique score of autonomicity per architecture. This score can be used to compare ANM architectures from the overall degree of autonomicity viewpoint. To the best of our knowledge, we are the first to provide a quantitative study on the evaluation of ANM architectures. Specifically, we propose a coherent methodology for comparing holistically among ANM architectures regarding the efficiency of the autonomic solution.

5.3 Quantitative Evaluation Criteria

In this section, we identify main evaluation criteria characterizing the efficiency of an ANM architecture. These criteria are identified based on inherent requirements of MAPE-K components for optimally performing the autonomic control loop. In the other word, each presented evaluation criteria is somehow important for the global efficiency of the architecture.

5.3.1 Accuracy of awareness

A key factor for achieving autonomic management is the ability to feed the reasoning algorithms with required knowledge. Generally, two types of knowledge are collected by the monitoring component: internal (local) view and external (network) view. The internal view consists in node perception about its local operating context. It is obtained from monitoring local resources or/and cross-layer information. The external view represents the knowledge of entire or a subset of network resources required for decisions with a more global scope. Such a parameter could be node position, energy level, load or neighbor degree, just to mention a few. The external view can be used for the optimization of collaborative applications such as the choice of algorithms, load balancing, routing decisions, position estimation, energy management as well as self-management properties. A node's external view is constructed by collecting numerous samples of local views received from various nodes within the network [28].

Depending on the intended utilization of each information, the collected data is either

used as it is received or aggregated somehow, providing the node perception about the network status. For instance, if the architecture supports different services which require all nodes' position (e.g. a geographic routing protocol), the node's external view is composed of all received nodes position without any aggregation. In contrast, if each node requires to evaluate its own relative status compared to the network-wide one, for example regarding the load, any received sample is gradually averaged, representing the network state for that parameter.

Intuitively, more frequently the global statistics are updated, better is the quality of network awareness. Nevertheless, frequent network view updates may produce extra overhead for not so much benefit. Therefore, a reasonable frequency of update should be taken, considering the requirements of decision-making algorithms regarding each information and the dynamicity of the underlying network.

The collected internal and external views provide the necessary awareness of the operating context. The quality of network awareness which is a prerequisite factor for the well-performance of the entire ANM architecture, is referred to as the *accuracy of awareness*. The accuracy of awareness is usually only affected by the quality of the external view since we can assume that the internal view is always true. The accuracy of awareness can be expressed using three metrics:

- *Accessibility of knowledge:* This metric denotes the ratio of nodes receiving a given external view versus total target nodes for that external view, i.e. all nodes requiring that information for their decision-making. In order to calculate the overall accessibly of knowledge, this ratio should be computed for each external view dissemination, giving the accessibly metric for that dissemination. The overall accessibly metric can be obtained by averaging all such measures.

- *Quality of knowledge/relative standard deviation (sd):* The average standard deviation between the collected external view and the ideal external view can be used to evaluate the quality of network-wide awareness. In order to unify the unity of standard deviation for different external view parameters, the deviation should be normalized, for example in terms of percentage, called the *relative standard deviation*. Two cases can be considered depending on the type of each external view parameter:

  - *Aggregated parameter:* For an external view consisting in an aggregated parameter, the relative standard deviation is computed comparing each node's external view with the exact value for a perfect external view, obtained by averaging all local views within the network. For an aggregated external view parameter, the

relative standard deviation represents the degree of uniformity of the external view in the network.

- *Un-aggregated parameter:* For an external view consisting in an un-aggregated parameter, the relative standard deviation is computed comparing each node's external view with the exact real value for that parameter. For an un-aggregated external view parameter, we consider the *average relative standard deviation* since each external view is composed of one entry by originator of the external view. For example, if the external view parameter is the position, the relative standard deviation between the position of each node i stored in the external view of each node j versus the real position of the node i is computed. All these values are then averaged over all nodes j giving the average relative standard deviation for the position parameter.

Obviously, a lower (relative) standard deviation indicates a better quality of the external views. Ideally, the standard deviation should converge to zero. However, a small enough value for standard deviation does not affect the performance of decision-making algorithms. This threshold, which is generally specific to the knowledge and its use by the decision-making algorithm, is denoted as $thrsd$.

- *Correctness (corr):* The correctness metric consists in the fraction of nodes able to evaluate their relative state correctly. Two cases can be considered depending on the type of the external view:
  - *Aggregated parameter:* For an external view consisting in an aggregated parameter, the correctness metric is obtained by comparing the ratio of nodes which have a same result (below or above the network-wide average) when their local views is compared respectively against the node's external view and the perfect network-wide average. This latter can be obtained by averaging all local views within the network.
  - *Un-aggregated parameter:* For an external view consisting in an un-aggregated parameter, the correctness metric is the ratio of nodes which have the same perception when each of their external view entries is compared with its exact value in underlying network.

  Obviously, a higher value of the correctness factor signifies a more convergent view among network nodes. Related to the previous point, the best value of correctness metric is obtained when the relative standard deviation is equal to zero.

We note that the accessibility of knowledge metric is somehow reflected in the standard deviation of knowledge. Indeed, when the knowledge is not reached some of its target nodes,

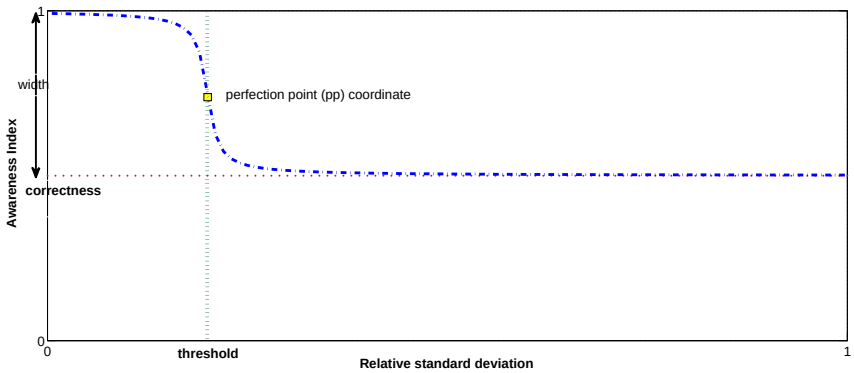


Figure 5.1: Awareness index

the standard deviation for that piece of knowledge increases. However, we considered the accessibility of knowledge as a separate metric, allowing to study specially the covering range of monitoring mechanisms.

The above mentioned contributing factors should be somehow combined, giving the overall accuracy of awareness of the architecture, referred to as as the *awareness index*. Since the accessibility of awareness is already reflected in the standard deviation, the awareness index represents only the combination of the standard deviation and the correctness metrics. The awareness index of an architecture should present two characteristics: First, it should be bounded between its correctness value and the maximum possible value of the awareness index. Second, it should converge progressively to the maximum value of the awareness index when the value of standard deviation is lower than $thrsd$. Giving these characteristics, we approximated the awareness index by an $arctan$ function which can acceptably reflect the required behavior. The proposed function looks like the one illustrated in figure 5.1 and considers a range between 0 and 1. The mathematical representation of the awareness index, denoted $index_{awareness}$, is as follows:

$$Index_{awareness} = -(arctan((sd - thrsd) * 100)/pi) * width + ycoordinate_{pp} \tag{5.1}$$

where variable $width$ denotes the width of the curve, defining the distance between upper and lower bounds of $arctan$ function. The $ycoordinate_{pp}$ reflects the position of the perfection point of the $arctan$ curve which is approximated by the y coordinate of the $thrsd$ value. These parameters can be obtained as follows:

$$width = 1 - corr \tag{5.2}$$

$$ycoordinate_{pp} = 1 - \frac{1}{2} * width \tag{5.3}$$

Obviously, higher the value of the awareness index is, more efficiently the architecture operates regarding the monitoring mechanism.

5.3.2 Accuracy of adaptation

A key characteristic of an autonomic architecture is its ability to self-adapt according to the network context, minimizing the administrator involvement for carrying out management tasks. To achieve this, an autonomic architecture requires efficient decision algorithms, enabling it to plan accurate adaptation strategies according to the network conditions. The capacity of the autonomic architecture to make correct adaptation decisions is denoted as the *accuracy of adaptation*. The accuracy of adaptation of an architecture is calculated using the *adaptation index* which consists in the number of correct adaptation decisions divided by the total number of adaptation decisions. In order to evaluate exclusively the adaptation ability of the architecture, the efficiency of decisions should be considered under a parfait knowledge quality.

5.3.3 Ability to learn

An extreme potential of autonomic networking is its ability to learn from past experiences to enhance future operations. This intelligent adaptation is mandatory to converge to the optimal adaptation, as it is one of the key self-management properties. As an example, a policy-based solution without learning capability may use a policy which considers any exceed of delay up to a predefined threshold as a sign of congestion. Accordingly, each time this threshold is exceed, the manager decides to drop some non-priority packets to reduce the delay for QoS packets. In contrast, a learning-based solution would rethink the accuracy of a previous congestion perception by analyzing the efficiency of past experiences. If learning mechanisms are used, the system could change this threshold if it learns that an increase or decrease of that value fits more with the reality of the underlying network. As well, the system could adjust its reaction once it learns that the further reaction provides better result.

As a summary, learning capability presents three major advantages that make it mandatory for autonomic networking:

- Optimization of predefined plans: Learning allows the network to refine its decision over time. Without this capability, the system continues to react sub-optimally according to the initial adaptation strategies.

- Facing with unprediction: Learning allows the network to cope with unpredicted situations. In this way, if any situation is omitted in the initial list of policies, the

network learns progressively how to react to that event. Thus, this feature reduces the human administration involvement.

- Increasing security: Learning leads to a non-deterministic system behaviour. As a consequence, it becomes more difficult to verify the behaviour and operation of the system since there is no clear optimal state towards which the system converges. This property makes the network more secure: as the reaction to an event evolves based on learning, it is difficult for an exterior attacker to predict the behaviour of the manager node. Misbehavior attackers will need to continuously look after manager nodes to find how they would operate thereafter and how their updated operations can be compromised.

Although the importance of the ability to learn, there is a few number of autonomic architectures enabling this feature. The "cognitive network" proposal uses a primary mathematical learning mechanism to adjust some protocol stack parameters. Another potential approach consists in applying some artificial intelligence techniques such as reinforcement learning. This latter is the basis of the learning mechanism employed by our SADA architecture.

In order to evaluate the architecture from the learning ability standpoint, we introduce the learning index, denoted as $Index_{learning}$. The learning index deals with the number of learning-aware adaptation strategies, while the quality of such strategies are somehow considered in the accuracy of adaptation criterion. The learning index consists in the ratio of the number of learnable management objects (policies, parameters, decisions), $|L|$, versus the total number of network management objects, $|M|$, as follows:

$$Index_{learning} = \frac{|L|}{|M|} \tag{5.4}$$

Obviously, the number $|M|$ does not involve deterministic management objects for which the behavior can be easily understood in the design phase. For example, a policy which sends a message to new discovered neighbors does not require learning ability since the discovery of a new neighbor is deterministic. Note that the learning index is a number between 0.0 and 1.0 which indicates the overall learning ability of the architecture. The minimum value of 0.0 corresponds to a closed-adaptive system where no learning mechanism exists. A complete open-adaptive system gives the value 1.0 for the learning index.

5.3.4 Quality of services (QoS)

Autonomic networking could be seen as a way to increase the overall performances of the network in terms of quality of services. Therefore, the QoS can be used as a mean to

measure the efficiency of the autonomic solution. It can be measured based on quantitative measurements of domain-specific metrics, more usually packet delivery ratio (PDR) or throughput, loss rate, end-to-end delay, jitter, etc. The desired QoS metrics may be different according to the objectives and applications provided by the network. For example, when a MANET is installed for transferring files with the attendee in a conference scenario, the prominent required QoS is the packet delivery ratio. Whereas, for the case of a MANET established for the rescue operation, the architecture seeks to optimize the average end-to-end (E2E) delay, jitter and loss rate, enabling the VoIP application.

5.3.5 Cost of autonomicity and services

In autonomic networking, the network spends extra resources to support the MAPE-K functions of autonomic control loops which is called *cost of autoonmicity*. The autonomicity, in turn, helps the network to reduce the amount of control traffic generated to support network services which is referred to as the *cost of services*. This improvement is due to the optimal configuration of network services which reduces the amount of control traffic.

The cost of autonomicity includes internal cost (e.g. CPU usage, storage, etc.) as well as external cost (e.g. overhead of the knowledge monitoring, the distributed intelligent adaptation, etc.) [66, 96]. Similarly, the cost of services can be divided into internal and external costs. For the case of a routing protocol service, the internal service cost is the CPU usage and the storage memory used to support the protocol. The external cost is the amount of control traffic generated to assist the well-performance of the protocol.

Consequently, we define a composite cost performance metric, called the *cost index*. The cost index is defined as the average of the costs required for autonomicity and services assistance. Obviously, the cost should be viewed as a relative metric versus the degree of autonomicity provided to the system. For example, a solution which provides a perfect degree of autonomicity with a higher cost may be globally better than an architecture providing a poor autonomicity with lower cost. For an ideal autonomic architecture, lower the cost index is, better the solution optimizes the network resource usage.

The overall internal cost of autonomicity and services can be computed easily by the average relative amount of internal resources (percentage) consumed for the purpose of autonomicity and services. The external cost can be obtained by the relative total amount of extra traffic overhead (for autonomicity and services), $E\{Overhead\}$, comparing to the total network traffic, $E\{U\}$. Intuitively, the cost index, $Cost_{int}$, is the average of the internal and external costs:

$$Index_{Cost} = \frac{Cost_{int} + \frac{E\{Overhead\}}{E\{U\}}}{2} \tag{5.5}$$

5.3.6 Degree of distribution

An important evaluation metric is the granularity or the degree of distribution of the intelligence. The distribution impacts the management responsiveness. Moreover, a full distributed network management is likely more robust, scalable and provides better flexibility and fault-tolerance. However the distribution may cause extra overhead due to the communications between individual managers required for the adaptation convergence. Hence, the cooperation and interaction between autonomic managers should be defined in a way that minimizes the cost of autonomicity.

Intuitively, the *granularity index* of an architecture is the ratio of the number of autonomic managers to the total number of managed elements. The desirable granularity boundary for ensuring flexibility and scalability is not deterministic and depends on a set of network-specific parameters. For example, a highly dynamic MANET environment requires a larger management distribution than a wire stable network. However, one can generalize that it is always desirable to provide a higher granularity while maintaining minimum extra costs.

5.3.7 Security

The security is a prominent issue of autonomic networking since it impacts the smooth operation of all MAPE-K components. Aside commune security issues existing in any distributed architecture, autonomic networking should face its specific issues. To achieve this, the vulnerabilities of autonomic architectures should be identified and recovered. Some common vulnerabilities are:

- Cooperative intelligence: The operation of the network depends on cooperation among individual autonomic managers. The penetration of an attacker in the cooperation cycle could damage hardly the overall performance of the network. A malicious attacker may execute damaging adaptation to the network. Hence, the performance of the network decreases locally. In an ultimate stage, this suboptimal performance causes abnormal changes and slant input to decision and learning algorithms of other managers. Therefore, the global view becomes slant and the overall network performance will become sub-optimal.

- Dependance on local and global knowledge: The accuracy of the adaptation decisions depends on the accuracy of the perception of the surrounding environment. an attacker may either report incorrect information to managers or alter a message in order to tamper other managers' perception. In addition, it can penetrate to the management overlay and report forged knowledge to other managers and distort their perception. Hence, the manager may decide incorrectly which could alter step by step

other managers' views and lead to cascade failures. Consequently, the performance of network decreases and the service availability could be menaced by Denial of Service (DoS) attacks, for example. Similarly, learning capability as an important property towards full autonomicity is also high vulnerable to the accuracy of knowledge. Learning algorithms may change predefined network policies and plans. They, if feeded by falsified knowledge, may lead the network to a critical state.

- Distribution of intelligence: An attacker may simply declare itself as an autonomic manager without any further malicious action. Thus, other managers will communicate with this infiltrated attacker for cooperation purpose. Also, the attacker may communicate to other managers to avoid being suspected. This generates supplementary overhead menacing the service availability.

We can conclude that cryptography and trust are two primordial security features of any autonomic network architecture. Trust can be discussed from two points of view: among autonomic elements and between an element and its user for performing some management tasks [95]. One approach to enable trust between autonomic elements is to apply existing trust models. Authors in [97] propose to use a reputation digit between zero and one to each execution of an autonomic manager (AM) and assign the same digit to a group of autonomic managers based on the long term performance of each AM or group of AMs. A digit close to 1 indicates a high level of trust. Trust between an autonomic manager element and its administrator can be achieved by applying an authentication mechanism.

To compute the security level of ANM architectures, we assume the existence of a reputation scheme and a cryptographic-key distribution system. Based on this assumption, two major security metrics are identified: cryptographic key length and trust. Longer key lengths make cryptographic mechanisms more resistant to attacks. Thus, the security level is directly proportional to the key length. Regarding to the trust metric, a reputation system should be established in the network where values between 0.0 and 1.0 are generated to indicate the managers' behaviour. The overall trust can be obtained taking the lowest average reputation value of autonomic managers when various types of attack are executed. Since the managers reputation may vary during the network runtime, the average reputation of each manager over time should be taken.

5.4 Autonomicity Evaluation Methodology

A relevant step for progressing the field of autonomic networking is to enable the efficiency evaluation and comparison of current solutions [66].

Two kinds of comparison can be considered. On one hand, we can compare architectures performance regarding to one specific evaluation view. On the other hand, we can consider

a holistic comparison view, considering an overall performance measurement metric and calibrating scores across different systems [96].

The ultimate objective of this section is to propose a coherent evaluation methodology to correlate the identified evaluation metrics and provide an overall measure of autonomicity. This overall score of autonomicity is called the *degree of autonomicity*. The proposed evaluation methodology is based on fuzzy logic (FL) [98]. The use of fuzzy logic does not only combine and evaluate multiple criteria simultaneously, but also copes with imprecision and nonstatistical uncertainty. Since the desirable value for each evaluation criteria is a fuzzy value with uncertain boundaries, FL concept provides a robust mathematical framework to correlate these values into an unique score of autonomicity. Obviously, in the context of ANM architectures' evaluation, the individual evaluation criterion constitutes the inputs of the fuzzy inference system. The output is a score value obtained from the correlation of all contributing evaluation criteria.

The degree of autonomicity estimated by fuzzy inference process follows the steps of fuzzification, inference and defuzzification, as described in the following:

- *Fuzzification:* Fuzzification is the process of converting input values into fuzzy sets. Each input has a set of linguistic terms which are mapped to fuzzy sets by membership functions. As mentioned before, evaluation criteria are used as input values and represented by linguistic terms as "low", "high", "long", "medium", among others. Trapezoidal functions are chosen as membership functions since they have been extremely used for performance representation due to their simple formula and computational efficiency.

- *Inference:* Fuzzy inference is the process of mapping inputs to outputs by fuzzy logic. The mapping is realized by rules that determine the outputs associated to the inputs. Fuzzy rules follow the form of if-then and their inputs and outputs are values in fuzzy sets. These values are in the interval [0.0,1.0], with 0.0 representing absolute falseness and 1.0 representing absolute truth. For our case, this phase is based on Mamdani inference mechanism which uses min-max composition operators [99].

- *Defuzzification:* At the final stage, the resultant decision sets have to be converted into a precise value through a defuzzification method. In our context, the centralized defuzzification method is used to obtain the score of the architecture. This value score marks the degree of autonomicity of the architecture and can be used to compare between autonomic solutions. Obviously, higher the architecture score is, more autonomically it performs.

In the following subsections, we describe the step of fuzzification of evaluation criteria, followed by the inference and defuzzification phases.

5.4.1 Evaluation criteria and fuzzification of inputs

Since we have seven evaluation criteria, seven class of fuzzy inputs can be considered: accuracy of awareness, accuracy of adaptation, ability to learn, cost, granularity of intelligence, security and QoS. Hereafter, we describe how the fuzzification process is carried out on the identified criteria.

Accuracy of awareness The awareness index, as it was described in section 5.3.1, is a number between 0.0 and 1.0 which indicates the overall efficiency of the exploited monitoring mechanism. The membership function of the awareness index is illustrated in figure 5.2(a). Two fuzzy sets are defined according to the effect of the awareness index observed on several ANM architectures.

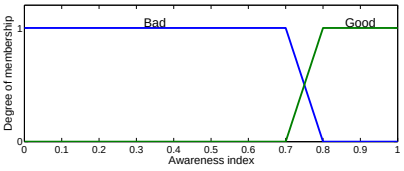
Accuracy of adaptation The accuracy of adaptation, as defined in section 5.3.2, is a number between 0.0 and 1.0 which indicates the percentage of correct adaptation strategies. The membership function of the adaptation index is presented in figure 5.2(b). Different boundaries are chosen considering comparative studies carried out for agents evaluation.

Ability to learn The learning index, as it was described in section 5.3.3, is a number between 0.0 and 1.0 which indicates the overall learning ability of the architecture. The membership function of the learning index is illustrated in figure 5.2(c). Different boundaries has been relaxed since the learning constitutes a novice feature addressed by ANM architectures.

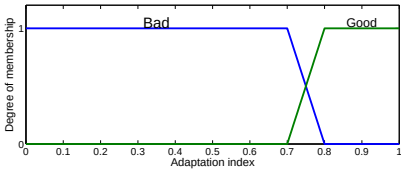
Cost of autonomicity and services The cost index, as defined in equation (5.5), is a value between 0.0 and 1.0. The membership function of the cost index is represented in figure 5.2(d). The fuzzy sets are chosen based on the common view-point of the research community regarding to the cost of services.

Granularity Regarding to the granularity, the membership function is monotonically increasing since it is always desirable, specially for high scale networks, to have more distribution while maintaining minimum overhead.

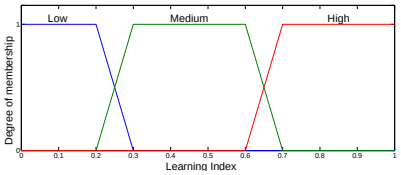
Security For cryptographic key length, two fuzzy sets are defined, short and long, as in [100]. If the secret key is 40 bits or less, the key is considered as short, and it is long with 128 bits or more. The membership function of cryptography key length is shown in figure 5.2(e). The network reputation linguistic variable has two fuzzy sets, good or bad.



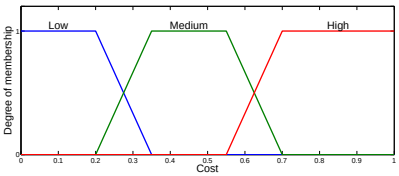
(a) Membership function of the awareness index



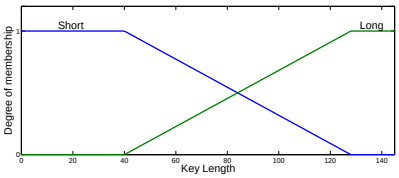
(b) Membership function of the adaptation index



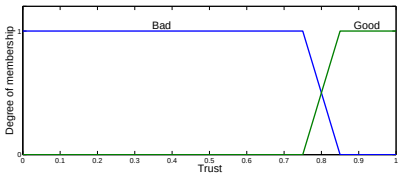
(c) Membership function of the learning index



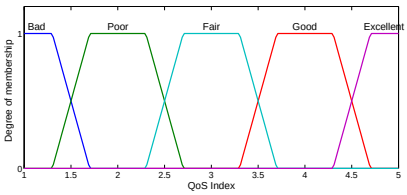
(d) Membership function of the cost index



(e) Membership function of cryptographic key length



(f) Membership function of trust

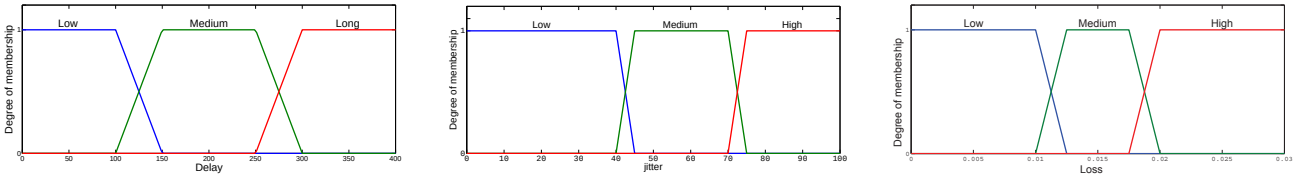


(g) Membership function of the QoS (MOS) index

Figure 5.2: Membership functions of fuzzy inputs (a) accuracy of awareness, (b) accuracy of adaptation, (c) learning ability, (d) cost, (e) cryptographic key length, (f) trust and (g) QoS (MOS)

Good reputations are those with values equal or higher than 0.8 [101]. The membership function of the network reputation is depicted in figure 5.2(f).

Quality of Services As outlined in section 5.3.4, each application may have its particular QoS requirements. To have a guideline, we consider an example of a VoIP application. Three common QoS metrics usually used to evaluate the performance of VoIP application are considered hereafter, namely loss rate, E2E delay and jitter. The QoS requirements for each of these fuzzy variables are taken from [102] and are illustrated in Tables 5.1(a), 5.1(b) and 5.1(c). The average overall E2E delay can be represented by the following linguistic terms: low, medium and long. Fuzzy inference considers the average overall E2E delay, estimated by the average E2E delay of all VoIP flows on the network. Based on the evaluation performed by IUT [102], the delay below 150 ms is considered as low, between 150 ms and 250 ms is considered as medium, and the delay longer than 250 ms is considered as long. Figure 5.3(a) presents the membership function for the average E2E delay. The



(a) Membership function of average E2E delay (b) Membership function of average E2E jitter (c) Membership function of average E2E loss

Figure 5.3: Membership functions of fuzzy inputs for QoS (MOS) (a) E2E delay, (b) jitter and (c) loss

average jitter and the average loss are represented by the following fuzzy sets: low, medium and high. Figure 5.3(b) and 5.3(c) present respectively the membership function for the average jitter and the average loss-rate.

(a)

E2E delay	Effect on perceived quality
[0,100-150] ms	Delay not detectable
[150,250] ms	Still acceptable quality
[250-300,..] ms	Unacceptable delay

(b)

Jitter	Effect on perceived quality
[0,40] ms	Jitter not detectable
[40,75] ms	good quality, but occasional delay or jumble noticeable
[75,..] ms	too much

(c)

Codec	Latency	Packet Loss (%)	MOS
G.711 w/o PLC	150	1	3.55
G.711 w PLC	150	1	4.31
G.711 w/o PLC	150	2	3.05
G.711 w PLC	150	2	4.26
G.729A+VAD	150	1	3.99
G.729A+VAD	150	2	3.82
G.723.1A+VAD	150	1	3.82
G.723.1A+VAD	150	2	3.60
G.711 w PLC	400	0	3.6

Table 5.2: Effects of QoS metrics on VoIP quality (a) delay, (b) jitter and (c) loss

In order to represent the quality of delivered service with an overall measure of QoS, the individual QoS metrics can, in turn, be correlated into an unique reference *QoS index*. For our example of the VoIP application, we carried out this process using a dedicated QoS fuzzy system which correlates loss rate, delay and jitter to a MOS (Mean Score Opinion) value based on estimations outlined by IUT [102]. As the result, the obtained MOS measure is used as an input of autonomicity fuzzy system instead of individual QoS metrics which does not separately reflect the delivered QoS. The membership function of MOS metric is represented in figure 5.2(g). Different boundaries are defined based on the definition and evaluation of MOS reported by IUT [102].

Table 5.3: Examples of fuzzy rules

Rule	Ability to learn	QoS	Cost	Accuracy of adaptation	Accuracy of awareness	Security	Autonomicity level
1	low	bad	high	bad	bad	bad	bad
...	...	...	...	...	...	...	...
i	low	fair	medium	good	good	medium	acceptable
...	...	...	...	...	...	...	...
j	high	good	medium	good	good	good	good
...	...	...	...	...	...	...	...
n	high	excellent	low	good	good	good	excellent

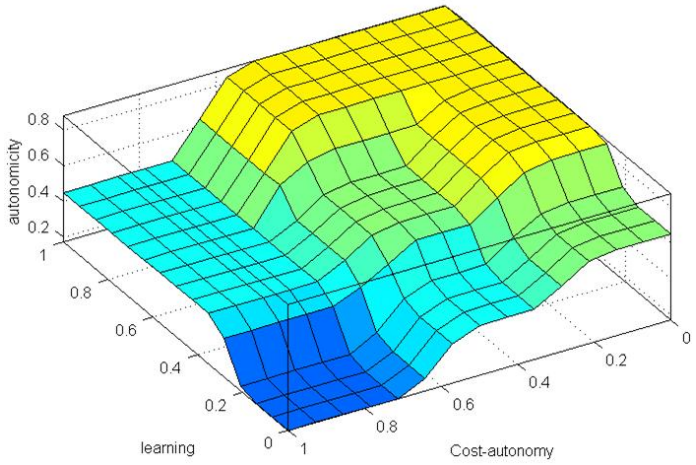


Figure 5.4: Correlating learning ability and cost criteria

5.4.2 Fuzzy inference process and defuzzification

Fuzzy inference is the step of mapping input values to output sets using fuzzy rules. Fuzzy rules are a set of if-then rules and the result is either bad, acceptable, good or excellent. In our case, the fuzzy inference consists in applying fuzzy rules on the fuzzified evaluation criteria in order to obtain fuzzy sets corresponding to the autonomicity level of the architecture. The considered fuzzified evaluation criteria are learning index, adaptation index, awareness index, QoS index, cost index, degree of distribution index and security index. As the efficiency of the architecture is a function of its global performance regarding each evaluation criteria, the total index representing each evaluation criteria is considered as input of the fuzzy system rather than the individual contributing metrics. For instance, the total QoS index is considered instead of the separate QoS metrics in terms of delay, jitter, etc. Table 5.3 presents some examples of applied fuzzy rules on our fuzzified evaluation criteria.

At the final stage, the resultant decision sets have to be converted intro precise value

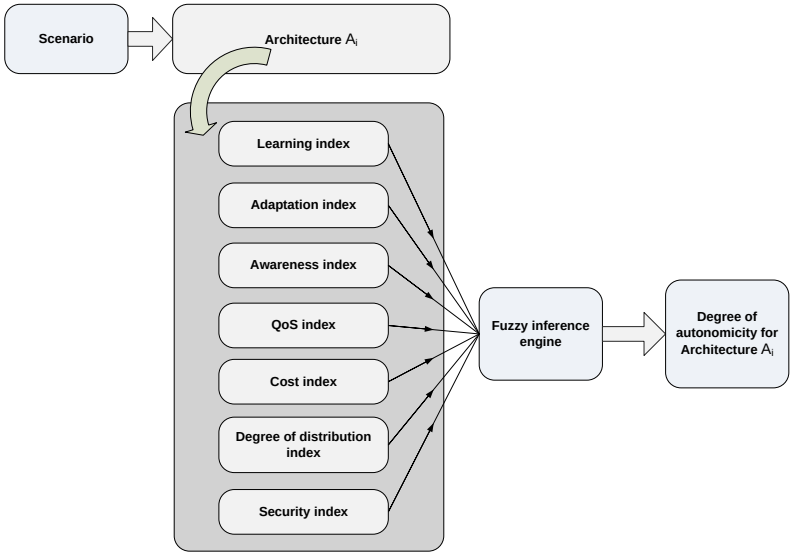


Figure 5.5: Degree of autonomicity obtention phases

through defuzzification method. To this end, the centralized defuzzification method [98] should be used to obtain the score of the architecture. This score marks the degree of autonomicity of the architecture and can be used to compare between autonomic solutions. Obviously, higher the architecture score is, more autonomically it performs.

In order to exemplify the impact of learning ability and the cost criteria on the degree of autonomicity of the architecture, Figure 5.4 correlates these two criteria. Note that with the cost up to 0.7, the learning ability is not an important factor to improve the degree of autonomicity.

5.5 Comparison of autonomic network architectures

The autonomicity score attributed to each architecture can be used to order the efficiency of existing ANM solutions. Obviously, the autonomicity comparison is meaningful for comparing architectures designed for a same context, since the effectiveness of an architecture depends on the employed use-cases. As an example, an autonomic architecture designed for cellular networks can not be compared to the one designed for multi-hop networks as their constraints and characteristics differ.

In order to compare architectures designed for a same context, different scenarios of test should be considered, each one evaluating the impact of one contributing factor on the degree of autonomicity of the architecture. The defined scenarios should be representative, allowing to draw conclusions for the efficiency of the architecture on similar networking contexts.

Figure 5.5 summarizes the process of evaluation and comparison of autonomic network architectures. For each reference scenario, the results obtained for individual evaluation criteria are fed as inputs to the fuzzy inference engine. This latter produces the overall degree of autonomicity as the output. The degree of autonomicity obtained for each architecture can be used to compare different autonomic network architectures designed for a same context.

5.6 Conclusion

This chapter presented the main evaluation criteria important for evaluating the efficiency of autonomic architectures from different autonomicity views. The identified criteria consider the requirements of the components of the MAPE-K model for optimally performing the autonomic control loop. These identified criteria include learning ability, accuracy of adaptation, accuracy of awareness, quality of service, cost of autonomicity and services, degree of distribution and security.

Further, we proposed a coherent quantitative methodology for evaluating and comparing autonomic architectures in an unified manner. Our methodology is based on fuzzy-logic and correlates the identified evaluation criteria to obtain an overall measure of autonomicity. This autonomicity score can be used to compare quantitatively different existing autonomic architectures and classifies them regarding to the network management efficiency and performance.

Chapter 6

Use cases - Evaluation of Autonomic Network Architectures

Contents

6.1 Introduction

In the previous chapter, we presented the main evaluation criteria characterizing the efficiency of autonomic network architectures. Further, we proposed a coherent fuzzy-based methodology to evaluate and compare autonomic architectures in an unified manner. In this chapter, we apply the identified evaluation criteria as well as the proposed evaluation methodology for evaluating the efficiency of autonomic network architectures.

To show the applicability of our methodology, two use cases have been considered in two different contexts: (i) an infrastructure-based wireless network, and (ii) a wireless mobile ad-hoc network (MANET). The AHOPS [103] architecture has been considered as our reference infrastructure-based wireless network. As a reference infrastructure-less architecture, the proposed SADA architecture described in chapter 3 has been considered. Although, the SADA autonomic architecture can be applied in both infrastructure-based and infrastructure-less contexts, it was applied on MANETs in the context of this thesis. Hence, we used SADA architecture as a case of application of our methodology for infrastructure-less architectures.

Different representative scenarios have been considered to evaluate the degree of autonomicity of AHOPS and SADA architectures regarding various network states. We considered three main parameters characterizing the network state: network traffic, network dynamicity and network density (number of users for cellular networks and number of nodes for multi-hop networks). The values *low*, *medium* and *high* have been considered for each of these parameters. Based on these contributing parameters, we defined an ideal scenario which describes a network with traffic load, mobility and density parameters set to values for which the architecture presents its ideal results. This ideal scenario is then used to create other reference scenarios, each one varying one of this basic scenario parameters to "low", "medium" and "high" values while other parameters are set as the basic scenario. Further details on the reference scenarios used for AHOPS and SADA architectures are described in the corresponding sections.

The objectives of the investigated use cases are twofold. First, the use cases allow to demonstrate the applicability of our evaluation work for autonomicity evaluation of existing autonomic architectures designed for different contexts. These use cases describe how the proposed methodology can be applied to obtain the degree of autonomicity in different scenarios for a cellular and a multi-hop network. Second, the use cases provide the quantitative measurements that allow to evaluate the investigated use cases regarding the efficiency of employed autonomic approaches.

The remaining of this chapter is organized as follows. Section 6.2 describes the application of the proposed evaluation criteria and methodology for the SADA architecture.

Further, it presents the evaluation environment and analyzes the obtained results. Section 6.3 provides a concise description of the AHOPS architecture. Next, it describes the evaluation environment and the obtained results for the AHOPS architecture. Section 6.4 discusses the achieved results for the SADA architecture and the AHOPS architecture. Section 6.5 analyzes the characteristics and assets of the SADA architecture, compared to the current state of the research. Finally, section 6.6 concludes this chapter.

6.2 SADA architecture applied in wireless ad hoc networks

This section evaluates the autonomicity of the SADA architecture, applied in the routing of MANETs. This latter resulted in the SRS, a self-adaptive routing scheme for MANETs, as presented in chapter 3. The cognitive engine of the SADA architecture enabled the optimization of the route discovery process in MANETs based on random neural networks with reinforcement learning. The autonomic monitoring module of the SADA architecture provided the cognitive engine with the required information based on piggybacking normal routing packets in the network, minimizing the autonomicity overhead.

6.2.1 Evaluation environment

In order to show the autonomicity efficiency of the SADA architecture, we carried out a set of simulation experiments using NS-2 version 2.30. The defined SRS scheme has been instantiated on the DSR as routing protocol.

The IEEE 802.11 distributed coordination function (DCF) is used as medium access control (MAC) protocol. The radio model takes into account similar characteristics to a commercial Lucent's WaveLAN radio interface with a nominal bit-rate of 11 Mb/s for the shared-media radio and a nominal radio range of 200 meters. The mobility model applied is the random way-point (RWP) model, where node speeds are randomly chosen according to the scenario with a pause time fixed to 5s. The initial position of nodes are selected randomly at the beginning of the simulation. The network area dimensions were fixed to 500 meters square for all simulations.

The data traffic used is CBR (Constant Bit Ratio) with number of connections selected according to the used reference scenario. Each source node generates 4 packets/sec with a packet size of 256 bytes. Data traffic sessions happen at a random time in the 10 first seconds of the simulation. The total simulated time was 300 seconds and each plotted result is an average of 30 simulations with a 95% confidence interval.

In these tests, we considered two types of decision parameters monitored by RREQ and RREP routing packets: path lifetime and number of hops. The path lifetime adds 4 supplementary bytes to the IP header. The hop count information is stored in the already

Table 6.1: SADA - Fixed simulation parameters

Parameter	Value
Transmission Rate	11 Mbps
Radio Proagation Model	TwoRay Ground
Mobility Model	Random waypoint
Network area	500m x 500m
Transmission Range	200m
Interface queue size	64 packets
CBR rate	4 pkt/s
Data packet size	256 bytes
Pause time	5s
Simulation time	300s

existing TTL field of the IP header and does not cause any extra overhead. Table 6.1 summarizes the main fixed simulation parameters.

Seven reference evaluation scenarios have been considered to evaluate the autonomicity of SADA architecture. These scenarios are described as follows:

- Scenario 1 - Ideal scenario (Low traffic, Low mobility, Low density): The ideal scenario consists in a scenario with low number of nodes (10 nodes), moving with a low random speed between 0 and 2m/s according to the RWP model. Moreover, a low number of source nodes (30% of nodes) is considered.

- Scenario 2 - Average traffic load: This scenario evaluates the impact of an average traffic load on the degree of autonomicity. Compared to the ideal scenario, the number of connections is increased to 50% of nodes representing a network with an average load.

- Scenario 3 - High traffic load: This scenario evaluates the impact of a high traffic load on the degree of autonomicity. Compared to the ideal scenario, this scenario considers 80% of source nodes in the network.

- Scenario 4 - Average mobility: This scenario evaluates the impact of a medium network dynamicity on the degree of autonomicity. Compared to the ideal scenario, the maximum speed of nodes is set to 10m/s.

- Scenario 5 - High mobility: In this scenario, the nodes speeds are further increased to represent a high dynamic network. Accordingly, we considered a maximum speed of 15m/s.

- Scenario 6 - Average density: This scenario considers the impact of a medium density

Table 6.2: SADA - Configuration of evaluation scenarios

Scenario	Number of connections (%)	Number of nodes	Maximum speed (m/s)
Scenario 1 - ideal	30	10	2
Scenario 2 - average traffic	50	10	2
Scenario 3 - high traffic	80	10	2
Scenario 4 - average mobility	30	10	10
Scenario 5 - high mobility	30	10	15
Scenario 6 - average density	30	30	2
Scenario 7 - high density	30	45	2

(number of nodes) on the degree of autonomicity. Compared to the ideal scenario, the number of nodes is set to 30 nodes.

- Scenario 7 - High density: This scenario considers 45 nodes, evaluating the impact of a high density on the degree of autonomicity.

Table 6.2 summarizes the configuration of each reference scenario. The results are presented on two perspectives: **individual autonomicity criteria** and **overall degree of autonomicity**. With the first perspective, we evaluate the efficiency of the SADA architecture regarding the identified evaluation criteria presented in chapter 5. These criteria are implemented using the NS-2 simulator. The implemented criteria include standard deviation, correctness, cost of autonomicity, cost of services and QoS-related metrics. The second perspective allows to compare the overall degree of autonomicity of the SADA architecture for different reference scenarios based on the fuzzy-logic methodology proposed in section 5.4. The proposed fuzzy engine is implemented using the fuzzy-logic toolbox of MATLAB version 7.11.0.

The application used in simulations is Telnet with a delay and loss rate constraints of 2 seconds and 2% respectively [104]. The overall measure of the QoS for Telnet application is obtained using a dedicated fuzzy system which correlates the achieved delay and loss rate to a overall measure of QoS based on estimations outlined by IUT [102].

For the knowledge-related criteria (standard deviation and correctness), we considered the comparison between the monitored decision parameters obtained by packets and the exact value for those parameters. The implemented correctness metric consists in the percentage of times that nodes possess a correct estimation on decision parameters before taking a new decision. For the standard deviation metric, we considered the normalized standard deviation between the monitored and exact value of the decision parameters before taking a new decision.

With the SRS scheme, the cost of autonomicity consists in the overhead generated for

monitoring the path lifetime and hop count, which consists in 4 supplementary bytes per a RREQ or a RREP packet. The cost of service corresponds to the overhead generated by RERR, RREQ and RREP packets, minus the amount of autonomicity overhead carried by RREQ and RREP packets.

6.2.2 Evaluation of individual autonomicity criteria

This section evaluates the autonomicity efficiency of the SADA architecture regarding the identified autonomicity criteria and describes the results obtained for each of these perspectives.

6.2.2.1 Learning ability

In the SADA architecture, decisions are carried out using the random neural network with reinforcement learning. This latter allows the network to learn from its past experiences to optimize its future operations. The learning index is a value in range of 0.0 to 1.0 which can be obtained by dividing the number of learnable objects by the total number of management objects. As the proposed random neural network-based learning mechanism is generic so that it can model any adaptation decision problem, the learning index of SRS is, hence, 1.0.

6.2.2.2 Accuracy of awareness

The accuracy of awareness of the SADA architecture is evaluated regarding the standard deviation and correctness metrics proposed in chapter 5.

Standard deviation The standard deviation for different decision parameters (path's lifetime and path's hop count) are illustrated in figure 6.1. These graphics show that the quality of monitored parameters varies from one scenario to another. As expected, the standard deviations between the monitored and the exact value of decision parameters are low for the ideal scenario, independently of the decision parameter. These graphics show that the standard deviation decreases for higher traffic loads, reaching its minimum value for the high traffic scenario. This can be explained by the network traffic reliance of the employed monitoring mechanism. Indeed, the decision parameters are monitored thanks to already existing RREP and RREQ packets. Hence, more the number of connections is, more there are RREP and RREQ packets updating nodes information and more the nodes information on decision parameters, hence, match with their exact values at anytime. As expected, the standard deviation is increased for higher network dynamicity, independently of the decision parameters. Indeed, higher nodes speed results in further changes in nodes

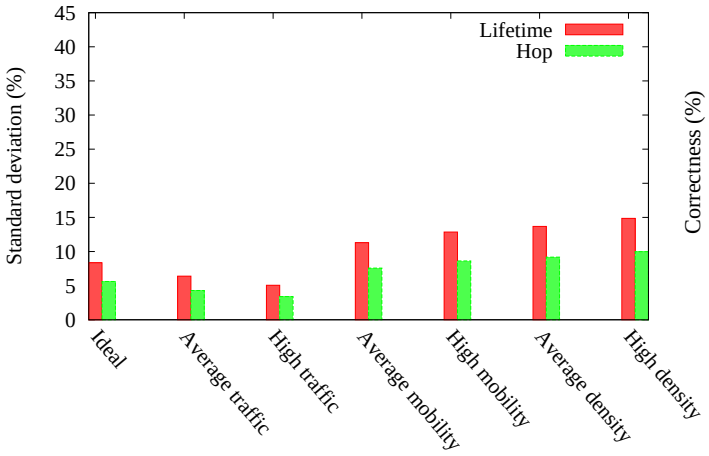


Figure 6.1: SADA - Standard deviation

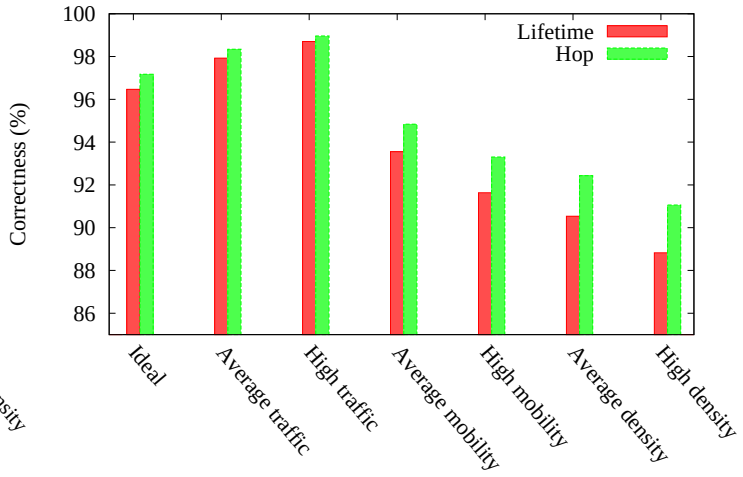


Figure 6.2: SADA - Correctness

positions and more frequent link failures or appearances, increasing or decreasing the paths' number of hops or lifetime over time. Hence, the gap between the nodes' estimation on decision parameters and the corresponding exact values increases. Graphics show that the network density is the most important factor increasing the nodes knowledge's standard deviation. This can be explained by further medium access retries and collisions which can delay/drop packets. Hence, the monitored and exact value of decision parameters may differ. For all scenarios, the standard deviation of the lifetime is higher than the one for the hop count parameter. Similarly, the standard deviation of the lifetime is more impacted by the different scenarios. Indeed, the lifetime expressed in terms of milliseconds is more sensitive to the network variations and depends to any network changes, while the deviation between a monitored and an exact value for the hop count happens proportionally to the occurrence of link failures and appearances.

Correctness Figure 6.2 illustrates the correctness of the knowledge hold by nodes. The results of correctness evaluation show that SADA ensures a high knowledge quality for decision algorithms, independently of scenario used (between 88.82% and 98.96%). The results show similar observations than for the standard deviation metric. The best correctness value is obtained for the high traffic scenario, while the worse result is for the high density scenario. As expected, the higher the traffic load is, the higher the nodes estimations are correct, resulting to a higher correctness value. Similar to the result of the standard deviation, higher network densities and nodes speeds decreases the quality of knowledge estimated by nodes. Although we observed a standard deviation between 10% and 15% for

Table 6.3: SADA - Results of the awareness index

Scenario	Index of awareness for hop	Index of awareness for lifetime	Total index of awareness
Ideal scenario	0,980981809	0,967959515	0,974470662
Average traffic	0,994962338	0,983409199	0,989185769
High traffic	0,998160057	0,993252016	0,995706036
Average mobility	0,954494895	0,938742047	0,946618471
High mobility	0,938839443	0,919675772	0,929257607
Average density	0,929971334	0,908857859	0,919414597
High density	0,916261386	0,891800562	0,904030974

Table 6.4: SADA - Quality of service results

Scenario	Delay (ms)	Loss rate (%)	Telnet QoS index
Ideal scenario	263.81	1.33	4.1
Average traffic	755.73	1.73	2.9
High traffic	1421.63	2.20	1.9
Average mobility	685.76	1.67	3.1
High mobility	695.58	1.60	3.1
Average density	275.14	1.21	4.2
High density	534.26	1.42	3.7

the high density scenario, figure 6.2 shows that its correctness remains very good (up to 90%). This approves that the SADA architecture manages always a very excellent quality of knowledge. Similar to the results of the standard deviation, the correctness of the hop count parameter is always better than for the lifetime parameter.

For each decision parameter, the correctness and the standard deviation results of each reference scenario are correlated based on the equation 5.1, giving the awareness index regarding to that parameter as explained in the previous chapter. For the calculation of the awareness index, an standard deviation of 5% (threshold = 5% equation 5.1) has been tolerated. The overall accuracy of awareness is, then, calculated averaging the individual awareness index obtained for each of decision parameters. The results of the awareness index is presented in table 6.3.

6.2.2.3 Quality of Service

Table 6.4 summarizes the obtained results for the main Telnet QoS requirements, namely the End-to-End (E2E) delay and the loss rate. As well, it presents the overall Telnet quality obtained by correlating the delay and loss rate results.

As expected, the best E2E delay is obtained by the ideal scenario. Results show that

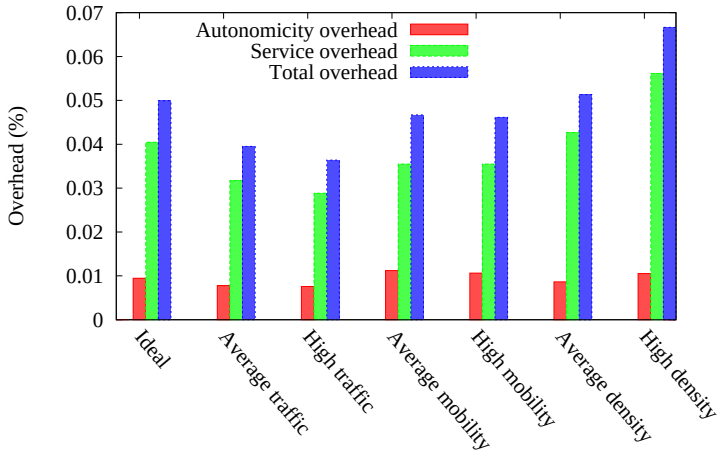


Figure 6.3: SADA - Overhead

the overall Telnet QoS is equal to 4.1 over 5, which represents a very good service quality for Telnet application. Compared to the ideal scenario, the delay and the loss rate are most impacted by the traffic load in the network. This can be explained by the further medium access collisions and retries as well as further cached terminal problems occurred in higher traffic loads, which can result in higher E2E delay and loss rate. As higher traffic scenarios represent the highest delay and lost rate, their correlated Telnet QoS is lower compared with other scenarios. As expected, mobility and density increase the E2E delay and loss rate of data packets except for the the average density scenario. Indeed, the average density scenario represents a better loss rate than in the ideal scenario which is due to the better network coverage, decreasing the packets loss rate.

6.2.2.4 Cost of autonomicity and services

The graphics in figure 6.3 plot the autonomicity, service and total overhead. The autonomicity overhead consists of the percentage of overhead generated to ensure the autonomic operation of the SADA architecture. This involves the cost generated by the lifetime and hop count information carried by RREQ and RREP packets. The service overhead consists in control overhead generated to ensure the routing service. This latter includes the overhead of RREQ, RREP and RRER packets, minus the autonomicity overhead carried by these packets.

Results show that the SADA architecture holds all types of overhead very low, independently of scenario used. As graphics show, the autonomicity generates less overhead than the routing service (the autonomicity overhead is only 1% in the worse case). The

advantage of this light generated overhead becomes more significant in a more elaborated scenario, where the information retrieved by autonomicity-related exchanges can be used by other applications and services, reducing more and more the overall control overhead. Graphics show that higher network densities result in higher overhead. This latter is due to further collisions and cached terminal problems which result in retransmissions and increases the overhead. For the average and high traffic scenarios, the overhead decreases compared to the ideal scenario. This latter is related to two factors: First, the network facilities offered thanks to the autonomicity and services overhead are used by more data traffics in the network. Second, the monitoring mechanism employed by the SADA architecture is enriched by more number of RREQ and RREP packets, resulting in better network awareness and reducing the overhead consequently. As graphics show, the mobility does not impact the generated overhead. This latter demonstrates that the SADA architecture manages to self-adapt the current paths in a high dynamic environment by selecting always a path offering a long lifetime.

As graphics show, the service overhead is more sensitive to the network scenario than the autonomicity overhead. Indeed, the size of autonomicity fields carried by packets is as low as their further number do not impact significantly their generated overhead.

6.2.2.5 Granularity of intelligence

The distribution of intelligence of the SADA architecture follows two axis: First, the knowledge management (knowledge monitoring and storing) is fully distributed in network nodes. Second, The decision strategy is completely distributed in all nodes throughout the network. As both of these processes are completely distributed, the SADA architecture can be considered as robust. This means that the failure of a node does not impact neither on the information recovery nor on the decision-making processes. Consequently, the score of the SADA architecture for this metric is 1 (in a range of 0.0 to 1.0).

6.2.2.6 Security

An autonomic trust monitoring scheme based on the SADA architecture has been proposed in chapter 4. The proposed scheme enabled a relevant quality of trustworthiness knowledge distributed all over the network. This latter should be fed to trust management frameworks to allow trust establishment decisions with manager nodes based on their estimated trustworthiness. In the current version of the proposed scheme, the monitored trust information is not used by managers nodes to decide on establishing or not trust relationship with others. As the security level of our architecture can not be measured in the current stage, we considered a security level equal to 0.5 (in a range of 0.0 to 1.0), neutralizing the effect of the security input required for the calculation of overall degree of autonomicity.

Table 6.5: SADA - Fuzzy inputs

Scenario	Learning index	QoS index	Cost index	Awareness index	Granularity index	Security index
Ideal scenario	1	4.1	0.049	0.974470662	1	0.5
Average traffic	1	2.9	0.039	0.989185769	1	0.5
High traffic	1	1.9	0.036	0.995706036	1	0.5
Average mobility	1	3.1	0.047	0.946618471	1	0.5
High mobility	1	3.1	0.046	0.929257607	1	0.5
Average density	1	4.2	0.051	0.919414597	1	0.5
High density	1	3.7	0.066	0.904030974	1	0.5

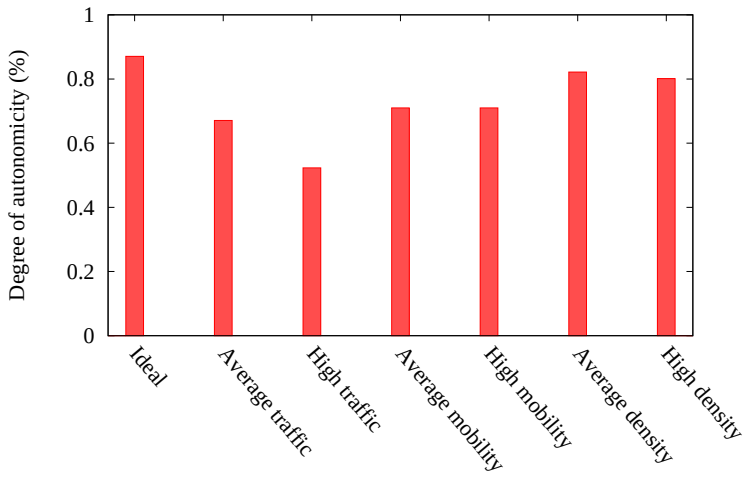


Figure 6.4: SADA - degree of autonomicity

6.2.3 Evaluation of the overall degree of autonomicity

The overall degree of autonomicity of the SADA architecture is obtained by combining all the individual evaluation criteria using the fuzzy inference engine described in chapter 5. This process is straightforward since the individual evaluation criteria represent normalized values. This process proceeds as follows: For each reference scenario, the results obtained for the individual evaluation criteria are fed as inputs to the fuzzy inference engine. This latter produces the overall degree of autonomicity as the output. Table 6.5 recapitulates the inputs' value obtained in previous subsection 6.2.2 for each of the reference scenarios.

Figure 6.4 plots the degree of autonomicity for each reference scenario. This figure shows that SADA provides a high degree of autonomicity for almost all scenarios. The

degree of autonomicity varies from one scenario to another. The highest autonomicity is achieved by the ideal scenario (up to 82%) for which the obtained QoS and quality of knowledge are high with a low cost. Graphics show that the degree of autonomicity is most impacted by the traffic load, decreasing it down to 52% for the high traffic scenario. This is mainly related to the low QoS level obtained with higher traffics (see table 6.5). As expected, the degree of autonomicity is decreased for higher mobilities compared to the ideal scenario. Graphics show that higher densities decrease the degree of autonomicity associated to the architecture. However, since the achieved QoS is better than with higher traffic loads, the obtained degree of autonomicity is higher than the one for average and high traffic. As expected, higher densities decrease slightly the degree of autonomicity of the architecture. Indeed, the negative impact of the higher cost and the lower accuracy of awareness in the average and high density scenarios are compensated by their good service delivery. Moreover, the obtained cost and accuracy of awareness for these scenarios are sufficiently suitable to satisfactory deliver services.

6.3 AHOPS: The Autonomic HandOver Piloting System

AHOPS [103] is an Autonomic HandOver Piloting System designed to overcome handover (HO) decision issues in heterogeneous wireless networks. The proposed autonomic solution consists in a set of distributed and cooperative autonomous agents capable to act autonomously on their environments. These agents are implemented in access points as well as terminal nodes using the Ginkgo Piloting Agent (GPA) technology [105].

The AHOPS functional architecture is presented in figure 6.5. The AHOPS is based on two main planes: a *knowledge* plane and a *piloting* plane [103]. The knowledge plane is instantiated into an information tier which monitors the required information for handover decision process. The required information are expressed in terms of agents neighboring list, network performance, application constraints and user preferences. The information is represented in an unified fashion using a simple ontology language developed by Ginkgo.

The piloting plane is responsible for optimizing the handover decision algorithm based on the knowledge provided by the knowledge plane. It uses a context-aware utility-based function to optimize the handover selection process. Moreover, the piloting plan employs a stability mechanism, based on an hysteresis margin [106], to avoid association to an point of attachment (PoA), which represents only a few better performance than the current associated PoA. The proposed stability mechanism allows to avoid the ping-pong effect of association and disassociation to fairly equivalent PoAs, which may result in degradation of quality of experience of mobile users. The handover decision strategy tier implements the functionalities of the piloting plane as depicted in figure 6.5. A third tier responsible for

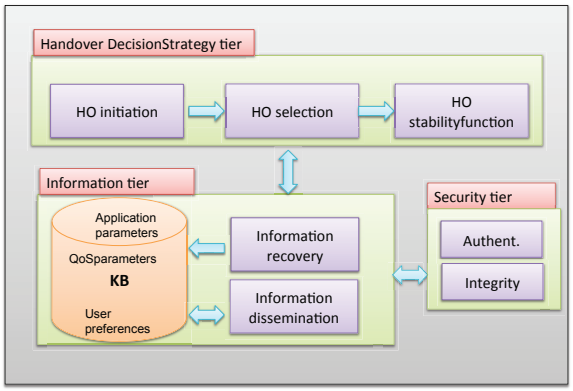


Figure 6.5: AHOPS functional architecture

AHOPS security is also recommended, which is mainly responsible for guaranteeing secure communications among neighboring agents and verifying the identity of the communicating agents.

The handover decision process is initiated by terminals' GPA. Hence, each terminal should be aware of the performance of any captured associated or non-associated PoAs to decide if a handover may be benefic. The required information is provided by the knowledge plan as follows: terminals' GPAs send frequently their local views to their associated PoA. The terminal's local view involves user performance parameters in terms of QoS. The GPA of the associated PoA constructs its local-network view, averaging the local views received from all the associated terminals. Obviously, terminals can only obtain decision parameters from the associated PoA. The only information that terminals can get from the non-associated PoAs is parameters related to the received signal (RSS, SNR, SINR). To enable terminals with non-associated PoAs' information required for the handover decision, neighboring PoAs exchange frequently their local-network views. The local-network view of neighboring PoAs is periodically transmitted to terminals, allowing them to estimate the state of any captured PoAs. Finally, terminals use an utility-function to decide the necessity of an handover initiation based on the retrieved information.

6.3.1 Evaluation environment

In order to evaluate the autonomicity of the AHOPS architecture, we carried out a set of simulation experiments using the SimVHO simulator. SimVHO is an extension of the property Ginkgo simulator [105], developed in Java 1.5 [107], which integrates the GPA technology and AHOPS implementation in a wireless network environment composed of several type of PoAs such as IEEE 802.11g access points, WiMAX base stations, LTE

Table 6.6: AHOPS - Fixed simulation parameters

Simulation Parameter	Value
Simulation time	100s
Simulation Area	500m x 500m
Number of PoA	17
Min_Pause	0s
Max_Pause	10s
Time Step	0.1s
Application	VoIP
Payload packet size	256 bytes
Control packet size	64 bytes

eNodeB and UMTS NodeB. The developed simulator is mono-process and runs in a discrete time. Mainly, SimVHO recovers the following information: Network topology and its changes (e.g. due to terminal mobility), the flows, network performance metrics and user preferences.

Table 6.6 summarizes main fixed simulation parameters. The network area dimension was fixed to 500 meters square, covered by a total number of 17 PoA for all simulations. The total simulated time was 100 seconds. The time step of the discrete SimVHO is set to 0.1 seconds and each evaluation criteria is an average of measurements taken after each time step with a confidence interval of 95%. The user mobility trajectories have been modeled by the RWP model with a random pause time between 0 and 10 seconds. The initial position of nodes are selected randomly at the beginning of the simulation.

The application traffic used in simulations is VoIP with a packet size of 256 bytes. Different agents within PoAs and terminals exchange their local views every 0.1 seconds. The size of these control messages is set to 64 bytes.

Seven reference evaluation scenarios have been considered to evaluate the autonomicity of AHOPS. These scenarios are described as follows:

- Scenario 1 - Ideal scenario (Low traffic, Low mobility, Low density): In the ideal scenario, a low number of users (60 terminals) moves with a low speed (walking speed of pedestrians) in the network. Moreover, between 5% and 15% of terminals communicate with their PoA, representing a network with low traffic load.

- Scenario 2 - Average traffic load: This scenario evaluates the impact of an average traffic load on the degree of autonomicity. Compared to the ideal scenario, the traffic load is increased to represent a network with an average load (between 40% and 60% of terminals).

- Scenario 3 - High traffic load: This scenario evaluates the impact of a high traffic

Table 6.7: AHOPS - Configuration of evaluation scenarios

Scenario	Number of connections (%)	Terminal number	Terminal speed (m/s)
Scenario 1 - ideal	5 - 15	60	0.4-2
Scenario 2 - average traffic	40 - 60	60	0.4-2
Scenario 3 - high traffic	70 - 90	60	0.4-2
Scenario 4 - average mobility	5 - 15	60	4- 8
Scenario 5 - high mobility	5 - 15	60	10-15
Scenario 6 - average density	5 - 15	75	0.4-2
Scenario 7 - high density	5 - 15	90	0.4-2

load on the degree of autonomicity. Compared to the ideal scenario, this scenario considers between 70% and 90% of terminals communicating with their base.

- Scenario 4 - Average mobility: This scenario evaluates the impact of a medium mobile users' movement on the degree of autonomicity. Compared to the ideal scenario, the mobiles speed is a random value between 4 and 8m/s.

- Scenario 5 - High mobility: In this scenario, the mobiles speeds are randomly selected between 10 and 15m/s, characterizing a network with high dynamicity.

- Scenario 6 - Average density: This scenario considers the impact of a medium density (number of terminals) on the degree of autonomicity. Compared to the ideal scenario, the number of terminals is set to 75 terminals.

- Scenario 7 - High density: This scenario considers 90 terminals, evaluating the impact of a high density on the degree of autonomicity of the AHOPS architecture.

Table 6.7 summarizes the configuration of each reference scenario. The results are presented on two perspectives: **individual autonomicity criteria** and **overall degree of autonomicity**. With the first perspective, we evaluate the efficiency of the AHOPS architecture regarding each of the identified evaluation criteria presented in the previous chapter. These criteria are implemented using the SimVHO simulator[1]. The implemented criteria include standard deviation, correctness, cost of autonomicity, cost of services and QoS-related metrics. The second perspective allows to compare the overall degree of autonomicity of AHOPS architecture for different reference scenarios based on the fuzzy-logic methodology proposed in the previous section. The proposed fuzzy engine is implemented using the fuzzy-logic toolbox of MATLAB version 7.11.0.

[1]The integration of the individual evaluation criteria in the SimVHO simulator was carried out in collaboration with M. Abid, a engineer researcher at Ginkgo networks - France

For knowledge-related criteria (standard deviation and correctness), we considered the comparison between the estimated values for the QoS performance parameters of each PoA hold by each terminal and the real QoS performance measured for that PoA. The used QoS parameters consist in VoIP QoS requirements in terms of experienced delay, throughput, loss rate and jitter. For the correctness metric, the impact of the knowledge quality on the decision process was considered. Indeed, we took into account the results of the decision algorithm with the estimated and the real measured input information. Next, we computed the number of decisions that yield the same result (initiate handover or not) whatever estimated or measured information is used. The number of similar decisions is incremented in two cases: First, when the handover is not initiated for both estimated and measured parameters. Second, when the handover is initiated and the same network selected for both inputs. The correctness metric is then calculated dividing the total number of correct decisions by the total number of decisions. In our simulations, the real measurement of the decision parameters are obtained by emulating an association between the terminal and captured PoA.

Similarly, the standard deviation is computed for the estimated versus measured QoS parameters. The obtained value is then normalized according to equation 6.1.

$$\mathrm{SD} = \frac{\sigma_x}{x_{max} - x_{min}} \tag{6.1}$$

where σ_x is the standard deviation of the parameter x and, x_{min} and x_{max} are the minimum and maximum possible values for that parameter x.

6.3.2 Evaluation of individual autonomicity criteria

This section evaluates the autonomicity efficiency of the AHOPS architecture regarding the identified autonomicity criteria and describes the results obtained for each of these perspectives.

6.3.2.1 Learning ability

In AHOPS, decisions are only based on utility functions which make no use of learning mechanisms.

6.3.2.2 Accuracy of awareness

The accuracy of awareness of AHOPS architecture is evaluated regarding the standard deviation and correctness metrics as described in the previous chapter.

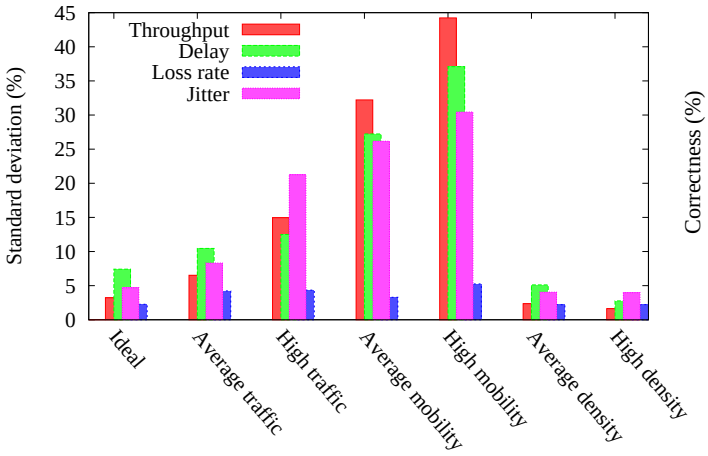


Figure 6.6: AHOPS - Standard deviation

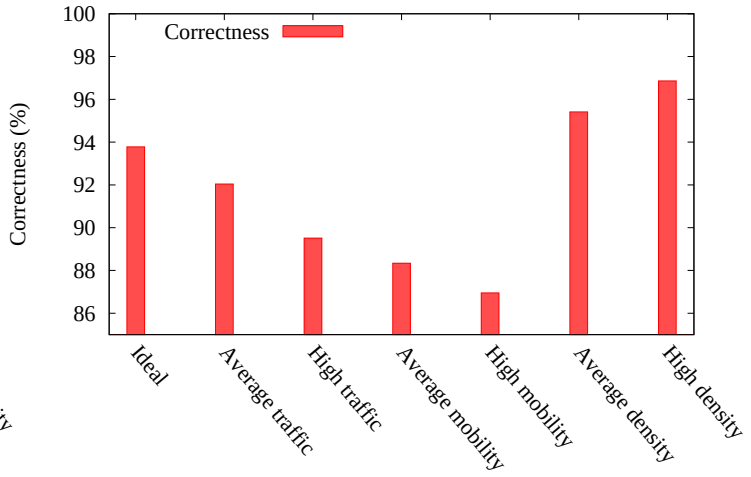


Figure 6.7: AHOPS - Correctness

Standard deviation The standard deviation for the different QoS decision parameters are illustrated in figure 6.6. These graphics show that the quality of estimated parameters varies from one scenario to another. As expected, the standard deviation between estimated and measured parameters are low for the ideal scenario, independently of the decision parameter. These graphics show that the standard deviation increases with higher traffic loads for all decision parameters. This can be explained by further medium access retries and collisions which can delay/drop the message exchanges between GPAs. Hence, the value of the estimated parameters may differ from the measured value. This effect is considerably elevated for the high traffic scenario. As expected, the increase of traffic load most impact on the throughput and jitter parameters with respectively up to 10% and 16% of increase. For the high traffic scenario, the standard deviation of delay is almost doubled compared to the ideal scenario. Indeed, the high traffic results in higher medium access collisions, which results in loss rate fluctuations over time. Hence, the estimated value of the loss rate parameter hold by the terminal may be not sufficiently updated, increasing its standard deviation for this scenario. Graphics show that the mobility is the most important factor impacting the GPAs' knowledge standard deviation. This is explained by network dynamicity with the arrival and departure of nodes fairly quickly causing rapid changes in network performance. Therefore, the value of decision parameters hold by a terminal for neighboring PoAs may not be updated according to the latest changes in that network. For high mobility scenario, the standard deviation increases up to 45% for the throughput parameter which can be due to the further handover events which results in frequent changes in the throughput of terminals. Graphics show that higher densities have

Table 6.8: AHOPS - Results of the awareness index

Scenario	index of awareness
Ideal scenario	0.94888
Average traffic	0.925052
High traffic	0.898056
Average mobility	0.885234
High mobility	0.871023
Average density	0.963016
High density	0.978574

positive impact on the quality of knowledge hold by terminals' GPA. Indeed, since PoAs exchange their average performance computed over the total number of their associated terminals, the estimation reflects better the network performance when higher number of values are considered in the average performance estimation.

Correctness Figure 6.7 illustrates the correctness of the knowledge hold by terminals. The results of correctness evaluation show that AHOPS ensures a sufficiently high knowledge quality for decision algorithms, independently of scenario used (between 86.95% and 96.86%). The results show similar observations than for the standard deviation metric. The best correctness value is obtained for high density scenario, while the worse result is for the high mobility scenario. As expected, higher the density is, higher the terminals estimations are correct, resulting to a higher correctness value. Similar to the result of the standard deviation, higher traffic loads and terminal speeds decrease the quality of knowledge estimated by terminals. Although we observed a very poor standard deviation for the high mobility scenario, figure 6.7 shows that its correctness remains pretty good. This is due to the reliance of the decision algorithms on only the RSS when the terminal is highly mobile, which results in a relatively high correctness between the estimated and measured based decisions.

Based on the correctness and standard deviation results, the accuracy of awareness of AHOPS regarding each of the decision parameters is calculated using the awareness index described in the previous chapter (see equation 5.1). For these calculations, an standard deviation of 5% (threshold = 5% equation 5.1) has been tolerated. For each reference scenario, the awareness index of individual decision parameters are then averaged, giving the overall awareness index for that scenario. The results of the awareness index for the AHOPS architecture are presented in table 6.8.

Table 6.9: AHOPS - Quality of service results

Scenario	Delay (ms)	Error	Jitter (ms)	MOS estimation
Ideal scenario	178.56	0.016	38.8	3.3
Average traffic	211.38	0.024	55.4	2.2
High traffic	223.71	0.027	62	2.1
Average mobility	177.84	0.016	41.9	3.1
High mobility	182.07	0.016	58.6	2
Average density	184.11	0.0185	41.2	2.8
High density	190.77	0.02	42.8	2.3

6.3.2.3 Quality of Service

Table 6.9 summarizes the obtained results for the main VoIP QoS requirements, namely delay, loss rate and jitter. The QoS results characterize the performance of the associated access networks, selected by the handover decision strategy. This strategy takes also into account monetary cost to decide to initiate the handover and to select the most appropriate target network. As expected, higher traffic load and density decrease the QoS performance for all QoS metrics. This is due to the complexity of handling a large amount of users and traffics, which may result in further medium access delays and collisions. The speed does not impact significantly the delay and loss rate, because of the adopted decision which takes into account the QoS rather than the monetary cost in order to maintain terminals connectivity. However, the jitter is impacted by high terminals speed, which is due to delay variations resulted by frequent handover association and dissociation. These individual QoS measurements have been used to estimate the MOS value experienced by users based on estimations outlined by IUT (see chapter 5). The obtained MOS value for each reference scenario is presented in table 6.9. As expected, the best results are obtained for the ideal scenario with low density, dynamicity and traffic load. High density and high traffic scenarios result in the worse MOS estimations.

6.3.2.4 Cost of autonomicity and services

The graphics in figure 6.8 plot the autonomicity, service and total overhead. The autonomicity overhead consists of the percentage of overhead generated to ensure the autonomic operation of the AHOPS architecture. This involves the messages exchanged between PoAs as well as the traffic sent by terminal to its associated PoA, enabling the local-network view. The service overhead consists in control overhead generated to ensure the particular handover service.

Results show that the autonomicity generates more overhead than the handover service.

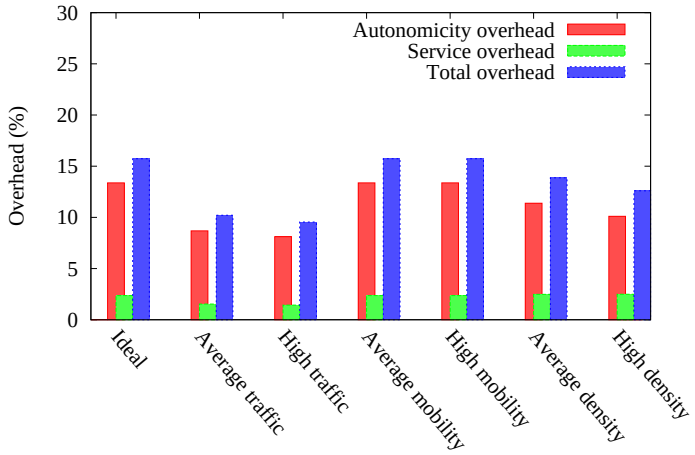


Figure 6.8: AHOPS - Overhead

However, in a more elaborated scenario, the information retrieved by autonomicity-related exchanges can be used by other applications and services, reducing the overall control overhead. Hence, the obtained autonomicity overhead can be considered negligible compared with its provided gain in the total overhead. As expected, higher traffic loads result in lower overhead percentage, since the payload increases. For higher densities, the amount of autonomicity overhead in bytes increase proportionally to the number of nodes. However, as the amount of payload increases, for which the packet size is greater than the control packets size, the percentage of autonomicity overhead (versus total traffic load) decreases compared to the ideal case. As expected, terminals speed does not impact the traffic load.

6.3.2.5 Granularity of intelligence

The distribution of intelligence in AHOPS follows two axis: First, the knowledge management (knowledge fetching and sharing) has been mainly managed by the PoA. Second, The decision strategy is mainly distributed in terminals. As both of these processes are completely distributed, AHOPS architecture can be considered as robust. This means that the failure of an access point does not impact on the information recovery. The same statement is made for decision-making process. Consequently, the score of AHOPS for this metric is 1.

6.3.2.6 Security

No security mechanism is implemented in AHOPS. However, some recommendations related to AHOPS needs in terms of authentication and integrity of communications have been

Table 6.10: AHOPS - Fuzzy inputs

Scenario	Learning index	QoS index	Cost index	Awareness index	Granularity index	Security index
Ideal scenario	0	3.3	0.1573	0.94888	1	0
Average traffic	0	2.2	0.1021	0.925052	1	0
High traffic	0	2.1	0.0955	0.898056	1	0
Average mobility	0	3.1	0.1573	0.885234	1	0
High mobility	0	2	0.1573	0.871023	1	0
Average density	0	2.8	0.1386	0.963016	1	0
High density	0	2.3	0.1259	0.978574	1	0

highlighted. In the current stage, the security index of the AHOPS architecture is 0.

6.3.3 Evaluation of the overall degree of autonomicity

The degree of autonomicity of the AHOPS architecture is obtained by combining all the evaluation criteria using the fuzzy inference engine described in the previous chapter. Table 6.10 recapitulates the results of each evaluation criteria contributing as the input of the proposed fuzzy engine.

Degree of autonomicity graphics presented in figure 6.9 show that the degree of autonomicity varies from one scenario to another. As expected, the best autonomicity is reached for the ideal scenario (up to 64%). The lowest level is obtained for the high mobility scenario with 42.30%.

Graphics show that the traffic load decreases the overall degree of autonomicity. This latter is mainly related to the delivered QoS degradation in such context. Moreover, higher traffics result in a lower knowledge quality hold by agents (see table 6.10) which leads to the decrease in the autonomicity level of the architecture compared to the ideal scenario. As expected, the high mobility scenario decreases the autonomicity of AHOPS. Indeed, the QoS delivered in high mobility scenario is significantly poor (see table 6.10) which results in lower level of autonomicity. This latter is due to the poor performance estimation of candidate access networks and frequent fluctuation in the performance of different access networks resulted by frequent arrivals and departures of highly mobile terminals. These graphics show that the degree of autonomicity decreases for high density scenario. However, its impact is less elevated than the case of high traffic load or high speed due to the good quality of knowledge provided to decision algorithms.

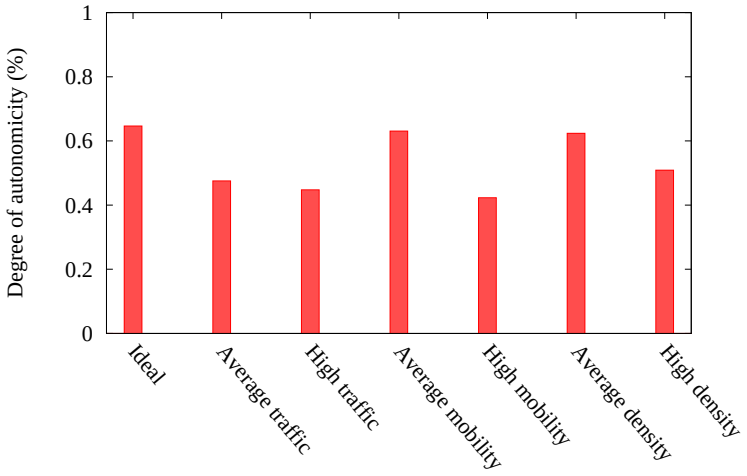


Figure 6.9: AHOPS - degree of autonomicity

6.4 Discussion

We highlight that the results obtained by each of the SADA and AHOPS architectures should be regarded at separately, as each of these architectures have been applied in different network contexts. The comparison of the SADA and AHOPS architectures would be possible when they are applied in a same case study with a same network context.

Results demonstrate that the SADA architecture provides a high degree of autonomicity to the highly dynamic and multi-hop ad hoc networks in most cases. It manages to provide a high quality of knowledge (between 90% et 99%) for decision-making algorithms, independently of used scenarios. The quality of knowledge is improved by further traffic load in the network. Although the knowledge quality of the SADA architecture decreases for higher network densities and mobilities, they remain always significantly high independently of used scenario (up to 90%). The SADA architecture uses random neural network with reinforcement learning, allowing the network to enhance its operation over time. The proposed learning mechanism is completely generic and can model any management adaptation problem. The SADA architecture reaches a good QoS performance for the employed Telnet application. The QoS achieved by the SADA architecture reaches up to 4.2 (in a scale of 5) for several reference scenarios. The overall overhead generated to the network is very low independently of the scenario used (maximum 7%). SADA provides the network with autonomic properties with a very slight autonomicity overhead (maximum 1%). The autonomicity overhead has been employed in the benefit of service overhead, decreasing its

value compared to a non-autonomic network. The generated autonomicity overhead can be used by several applications in a more elaborated scenario, increasing more and more the network performance. The SADA architecture is completely distributed and robust regarding both the knowledge management and the self-adaptation decision processes. The actual implementation of the SADA architecture provides some elementary security components for trust management frameworks which, when integrated to a trust establishment engine, may improves significantly network performance face to internal and external attackers. The results of the overall autonomicity degree of SADA architecture demonstrates that SADA constitutes an efficient autonomic architecture in the complex multi-hop wireless mobile context.

The obtained results for the standard deviation and correctness indicate that the accuracy of awareness of the AHOPS architecture is good despite the difficulty to have a good view of the network in such heterogeneous and distributed environments. The quality of knowledge decreases with lower number of associated terminals. The knowledge reliance to the number of terminals may lead to a poor knowledge quality for low associated number of clients. A more accurate network performance estimation model (than the simple averaging model) may improve this shortcoming. We observed that the quality of knowledge decreases significantly for high mobility scenario which can be improved by more frequent knowledge update in such a context. The AHOPS architecture reaches a sufficiently good QoS performance for high QoS constrained VoIP application in most scenarios. The autonomicity overhead generated to the network is higher (between 3 to 6 times more) than the service overhead independently of the scenario used. This latter, if used by several applications, can be considered as negligible. The service overhead is hold very low (maximum 3%) which demonstrates the gain of generated autonomicity overhead. The AHOPS architecture is completely distributed regarding the knowledge management as well as the handover decision and, hence, robust face to failures. A security tier is planned for AHOPS architecture which will integrate some security features to the architecture in terms of authenticity of exchanged messages and the identity of the corresponding terminals. The AHOPS architecture does not make use of any learning mechanism in the decision-making process, which can lead to non-optimized network performance when applied to indeterministic management decision problems.

6.5 Comparison of the SADA architecture

Although it was highly desirable to compare the results of our proposed SADA architecture with an existing autonomic architecture, this quantitative comparison has not been made in the context of this thesis. This was mainly related to the fact that the existing autonomic

architectures applied in multi-hop wireless mobile networks were developed originally by different groups and their source codes are either not open source or under development. Furthermore, the redevelopment of the existing architectures demands a huge amount of time and requires well detailed information about their design considerations (which is often not available). For these reasons, the quantitative comparison of our architecture with the existing autonomic architectures were not possible.

With regard to the qualitative comparison of the SADA architecture with the existing autonomic architectures, we used the qualitative evaluation criteria proposed in chapter 2. The following assets and characteristics have been identified for the SADA architecture compared with other autonomic proposals:

- The SADA architecture is a full distributed (flat) architecture in which the intelligence regarding the both decision-making and information monitoring processes is distributed in the network.

- Compared to the other autonomic proposals, the SADA architecture is a self-adaptive autonomic architecture. This property is the key enabler for all self-management properties, namely self-configuration, self-optimization, self-healing and self-protection.

- The SADA architecture is completely a generic architecture that can be applied for the both infrastructure-based and self-organizing multi-hop wireless networks. Although the SADA architecture has been applied in the context of wireless mobile networks in this thesis, SADA is network and protocol-independent.

- The SADA architecture is one of the few learning-based autonomic architectures. The learning ability is a paramount requirement for providing the network with self-optimization management property.

- The proposed learning approach is based on random neural networks with a reinforcement learning. The self-adaptation decision problems are modeled by a few number of neurons to ensure a low complexity for decision-making processes.

- The monitoring in the SADA architecture performs according to the autonomic control loop, resulting to an autonomic monitoring approach. As such, the monitoring mechanism can self-adapt to provide the network with the required information according to the network condition.

- The proposed monitoring mechanism uses both local and network view to provide the distributed autonomic control loop (self-adaptation algorithms) with the required information.

- As demonstrated by the quantitative measurements, the proposed monitoring mechanism generates a very low amount of overhead to the network. This meager monitoring mechanism has resulted in a significant gain in network performance which proves the efficiency of the proposed monitoring mechanism. Further, the relevance of the proposed autonomic monitoring approach is elevated when the retrieved information is used by several applications in the network, reducing the overall network overhead.
- As demonstrated by the quantitative results, the proposed monitoring mechanism provides an excellent quality of knowledge, accommodating uniform and up-to-date information to self-adaptation algorithms. This high quality of knowledge allows the convergence of the distributed self-adaptation processes.
- The proposed SADA architecture provides some primitives for security support. For instance, it provides the network elements with trustworthiness knowledge about others network elements. This knowledge allows to establish trust relationships only with trustee network elements, avoiding a range of security issues in the network.
- The results showed that the SADA architecture manages to cope with high network dynamicity. As such, a well performance of the SADA architecture in the context of Vehicular Ad Hoc Network (VANET) is expected.
- The proposed SADA architecture can be characterized as evolvable. The components of the SADA architecture are modeled in a way that they do not depend to any technology or any particular protocol. As such, the architecture is open to any feature advancement in information technology communities.

6.6 Conclusion

This chapter demonstrated the application of the proposed evaluation criteria and fuzzy-logic methodology to evaluate the autonomicity efficiency of autonomic architectures. The proposed evaluation criteria were adapted to the context of infrastructure-based networks (cellular networks) and wireless mobile ad hoc networks. For each of these contexts, we considered a different management problem in order to show the applicability of the proposed evaluation work for different contexts. For the case of wireless mobile ad hoc networks, we considered the SADA architecture applied in the self-adaptive routing of mobile ad hoc networks. Regarding to the infrastructure-based use case, the AHOPS architecture was considered which is an autonomic piloting architecture applied for the handover management in cellular networks.

In order to evaluate the autonomicity efficiency of these architectures in different networking contexts, different reference scenarios were considered, characterizing networks

with different densities, traffic loads and dynamicities. The autonomicity efficiency of the SADA and AHOPS architectures regarding each of the evaluation criteria were evaluated based on network simulations. The fuzzy-logic evaluation methodology used these results to draw the overall degree of autonomicity of each architecture for each reference scenario. The fuzzy-logic engine was implemented using the fuzzy-logic toolbox of MATLAB.

Results showed that the SADA architecture provides a significant quality of knowledge (up to 99%) with a very light overhead (maximum 1%) in the dynamic and resource-constrained multi-hop ad hoc networks. The achieved quality of service is satisfactory (up to 4.1 in a range of 5). The SADA architecture provides a completely distributed intelligence in the network. It uses learning mechanism to optimize its operation and presents some primitives for security support. The obtained overall degree of autonomicity of the SADA architecture demonstrates that SADA constitutes a very efficient autonomic solution for almost all scenarios. The AHOPS architecture achieved a high quality of knowledge (up to 97%) with a medium amount of autonomicity overhead (maximum 13%) generated to the network. The quality of knowledge was sensitive to the number of terminals in the network. The obtained quality of service is satisfactory for the demanding voice over IP application (a mean opinion score up to 3.3 in a range of 5). The AHOPS architecture is distributed regarding the both monitoring and decision-making processes. It obtained an acceptable level of autonomicity. However, this latter architecture requires learning, security and enhanced monitoring features to exhibit better autonomicity performance.

Finally, we compared qualitatively our proposed SADA architecture with other autonomic trends. The main identified assets of the SADA architecture consisted in distributed intelligence, learning capability, self-adaptation features, low complexity, autonomic monitoring, light monitoring, high quality of knowledge, security support and evolvability.

Chapter 7

Conclusion

Contents

7.1 Organization

This chapter summarizes the thesis conclusions and future directions. The objective is to reinforce the contributions we have achieved and point out some directions we envision for proceeding with the research. Hence, we first highlight the main contributions of the thesis in section 7.2. Then, we present in section 7.3 the future directions on the topics studied in this thesis.

7.2 Summary of contributions

Wireless mobile networks are characterized by their dynamic nature, mobility and scarce shared resources. Due to these characteristics, they require an autonomic architecture to self-adapt, enabling them to optimize the use of wireless resources and to cope with the network context dynamicity. The aim of the study carried out during the thesis was to improve the current state-of-the-art of autonomic architectures, especially for wireless mobile networks. This work presented six contributions in this subject, as described in the following:

1. A survey of autonomic network architectures In this contribution, we provided a coherent classification of existing autonomic architectures and surveyed these proposals. Next, we proposed a qualitative evaluation methodology to draw the characteristics of autonomic network architectures, emphasizing the open issues unsolved in current solutions. Mainly, we identified the lack of self-adaptive features, learning mechanisms, security and efficient monitoring approaches in existing architectures. Moreover, we discovered the lack of an appropriate quantitative methodology for evaluating and comparing autonomic architectures from the autonomicity standpoint.

2. SADA: Self-Adaptive Autonomic Architecture for wireless mobile networks Based on the raised open issues, we proposed SADA, a self-adaptive autonomic architecture for wireless mobile networks. Compared to the current state-of-the-art, SADA presents two new autonomic features. First, it enriches the IBM reference MAPE-K model exploited by existing architectures with self-adaptation capabilities. Second, it defines an autonomic control loop around the monitoring module which ensures the autonomic functionality of that module. The SADA architecture is composed of three main modules: the cognitive engine, the autonomic monitoring and the knowledge management modules. The *cognitive engine* is responsible for the self-adaptation of management tasks, allowing them to adjust the configuration parameters to values that fit better with the underlying network. The *autonomic monitoring module* supervises the monitoring of the information required for the

smooth operation of the cognitive engine. The *knowledge management module* manages the processing and representation of the knowledge before sending them to the cognitive engine module. The design of each of these modules was performed as follows:

- The **cognitive engine module** was designed based on random neural networks with reinforcement learning, inspired by biological neurones. The proposed model enables the network to self-adapt based on the experiences gained from the outcome of latest decisions. The proposed model is generic and can model any self-adaptation problem in a straightforward manner.

- The **autonomic monitoring module** was designed using an autonomic monitoring approach which uses normal transiting packets in the network to provide required information for self-adaptation algorithms. The proposed monitoring approach ensures the real-time monitoring of the underlying network with a minimum extra overhead.

- The **knowledge management module** was designed using utility functions to aggregate several information required for the self-adaptation algorithms to an overall representative information. This latter abstracts the knowledge process away from the cognitive engine module.

The proposed SADA architecture is characterized by its distributed intelligence, learning capability, self-adaptation features, low complexity, autonomic monitoring, light monitoring, high quality of knowledge, security support and evolvability.

3. SRS: Self-adaptive Routing Scheme for mobile ad hoc networks The application of the SADA architecture was demonstrated by the proposal of a self-adaptive routing scheme, called SRS, for mobile ad hoc networks. SRS uses the learning features of the SADA architecture to self-adapt the route discovery process of existing mobile ad hoc routing protocols. The routing self-adaptation was achieved instantiating the cognitive engine module of the SADA architecture. It is based on random neural networks which adapt the path selection process based on the recent experienced network performance. The autonomic monitoring module of the SADA architecture was employed to provide the used random neural network-based learning algorithms with required information, generating a negligible extra overhead.

In order to show the effectiveness of the proposed self-adaptive scheme, SRS was instantiated on DSR protocol and compared with DSR and AODV protocols. We evaluated the efficiency of the proposed scheme by simulations considering different network densities, nodes speeds and number of connections. Three metrics evaluated the efficiency of the SRS scheme, measuring its ability to deliver data, its latency and overhead. Simulation

results showed that our self-adaptive routing scheme can improve significantly the network performance in terms of packet delivery ratio (up to 28%), average end-to-end delay (up to 12s) and routing overhead (up to 37%). The proposed scheme is independent of any routing protocol and can be exploited by any existing routing protocol to enhance its performance.

4. ATMS: Autonomic Trust Monitoring Scheme for mobile ad hoc networks Based on the open issue of security identified in our survey contribution, we applied the SADA architecture to propose ATMS, an autonomic knowledge monitoring scheme for trust management in mobile ad hoc networks. The proposal of ATMS followed two main objectives. First, it ensured an uniform up-to-date trust knowledge throughout the network with a minimum monitoring overhead, providing a prerequisite primitive for securing the autonomic architecture. Second, ATMS demonstrated the efficiency of the autonomic monitoring approach of the SADA architecture, enabling the scheme to self-adapt to network conditions. ATMS is characterized by its protocol and trust framework independence, self-adaptation, simplicity and low computational intensiveness.

The proposed trust monitoring scheme was instantiated on Bella framework and compared with it. Two classes of metrics evaluated the effectiveness of the scheme, measuring its impact on network performance and on trust knowledge quality. For the first class, we compared the efficiency of schemes regarding the packet delivery ratio and the end-to-end delay. For the second perspective, we considered the average trust estimation, trust standard deviation and correctness. Simulation results showed a significant improvement in the trust knowledge quality (up to 25% for the average trust, up to 35% for the correctness and up to 17% for the standard deviation) with satisfiable network performance in terms of delay and packet delivery ratio.

5. A methodology for quantitative evaluation and comparison of autonomic architectures To address the raised autonomicity evaluation issue, we proposed two approaches for evaluating and comparing autonomic architectures from the autonomicity efficiency standpoint. The first approach consisted in evaluating autonomic proposals regarding the individual evaluation criteria characterizing the autonomicity of the architectures. The second approach consisted in evaluating and comparing autonomic architectures regarding their overall degree of autonomicity. These approaches were performed as follows:

- For the **first approach**, we identified the main quantitative evaluation criteria contributing to the efficiency of autonomic architectures. To allow the straightforward applicability of the identified evaluation criteria in different network contexts, a measurement technique was proposed for each of the proposed evaluation criteria. These criteria were identified based on the requirements of the components of the MAPE-K

model for optimally performing the autonomic control loop. The identified criteria include learning ability, accuracy of adaptation, accuracy of awareness, quality of service, cost of autonomicity and services, degree of distribution and security.

- For the **second approach**, we proposed a coherent quantitative methodology for evaluating and comparing autonomic architectures in an unified manner, based on the identified evaluation criteria. Our methodology is based on fuzzy-logic and correlates the identified criteria to obtain an overall measure of autonomicity. This autonomicity score can be used to compare quantitatively different autonomic architectures and classifies them approximatively regarding to the network management efficiency and performance.

6. Autonomicity evaluation of autonomic network architectures To show the applicability of the proposed evaluation techniques, two use cases were considered in two different contexts: (i) a multi-hop wireless ad-hoc network, and (ii) an infrastructure-based wireless network. For the first case, we considered the SADA architecture applied in the self-adaptive routing of mobile ad hoc networks. As a case of application of our methodology in infrastructure-based networks, we considered AHOPS, which consists in an autonomic handover piloting system for wireless heterogenous networks.

In order to evaluate the autonomicity efficiency of these architectures in different networking contexts, different reference scenarios were considered, characterizing networks with different densities, traffic loads and dynamicities. The autonomicity efficiency of the SADA and AHOPS architectures were evaluated regarding to both the individual evaluation criteria as well as the overall degree of autonomicity.

Results showed that the SADA architecture provides a significant quality of knowledge (up to 99%) with a very light overhead (maximum 1%). The achieved quality of service is satisfactory (up to 4.1 in a range of 5). The SADA architecture provides a completely distributed intelligence in the network. It uses learning mechanism to optimize its operation and presents some primitives for security support. The obtained overall degree of autonomicity of the SADA architecture demonstrated that SADA constitutes a very efficient autonomic solution for almost all scenarios. The AHOPS architecture achieved a high quality of knowledge (up to 97%) with a medium amount of autonomicity overhead (maximum 13%) generated to the network. The quality of knowledge was sensitive to the number of terminals in the network. The obtained quality of service is satisfactory for the demanding voice over IP application (a mean opinion score up to 3.3 in a range of 5). The AHOPS architecture is distributed regarding the both monitoring and decision-making processes. It obtained an acceptable level of autonomicity. However, this latter architecture requires learning, security and enhanced monitoring features to exhibit better

autonomicity performance.

7.3 Perspectives

The autonomic computing presented a novice problematic at the beginning of the thesis, for which the research community has not yet reached a common point. Due to this novelty, during the making of this thesis, we identified several new areas of study closely related to the topics treated in this thesis. This section presents the future work as well as the identified research perspectives regarding each class of contributions.

With regard to the autonomic architecture presented in chapter 3, several perspectives have been identified:

- A **short term** perspective consists in applying the proposed architecture for self-adapting simultaneously several applications and services. The objective of a such study is twofold: First, it allows to draw mechanisms to optimize the overall performance of running applications and services rather than by an application or a service. Although such a mechanism may lead to refrain the optimization of a given application in accordance to other applications, however, the overall network operation will be optimized. The second objective is to demonstrate the gain of the autonomicity monitoring overhead when the offered knowledge is used by several applications and services. A part of this overhead is offset by the reduced overhead to support services as well as by the gain in services performance. Moreover, when the same monitored information is used for enhancing two or more applications, the gain of autonomic networking becomes more and more relevant.

- As a **medium term** perspective, we plan to implement our autonomic network architecture in a test bed. Indeed, as the ultimate aim of autonomic networking trend is to cope with the management complexities of current and emerging IT communities, an autonomic architecture should be deployed in a real world. In this context, some new issues may be identified, leading to an adjustment of the approaches taken by our proposed architecture.

As a **short term** future work of the proposed routing case study, we intend to investigate its performance when integrated into the high dynamic vehicular ad hoc networks. Moreover, we intend to apply the proposed architecture for self-adapting other routing problems. For instance, we identified a very relevant case of application which consists in self-adapting the frequency of neighbor monitoring (e.g. beaconing interval in GPSR, HELLO interval in AODV) in mobile ad hoc routing protocols.

As a **medium term** perspective for the proposed trust management work elaborated in chapter 4, we intend to define more dedicated policies or even apply the learning algorithm to self-adapt the operation of the autonomic trust monitoring module. As well, we plan to take advantage of the relevant knowledge quality provided by our autonomic scheme as input of a routing decision process. Based on the excellent trust knowledge quality provided by our scheme, the framework is more likely to establish (or not) correct trust relationships with other nodes. Hence, we expect to observe a high network collaboration encouraged by this accurate trust establishment decision.

The proposed evaluation criteria and fuzzy-logic evaluation methodology elaborated in chapter 5 consisted in a first trend in comparison and evaluation of autonomic proposals with regard to their achieved autonomicity. Therefore, the design of such a methodology was very complex. In this context, some future works have been identified:

- A **medium term** perspective consists in comparing the performance of the proposed SADA architecture with several other autonomic architectures using the proposed methodology. This latter allows to evaluate our architecture compared with the current state of the research and to identify the pros and cons of each proposal. Particularly, we will focus on efficiency comparison regarding to the "cognitive network" which is the only architecture providing some degree of learning ability.

- As an important **long term** future work, we will investigate the use of such evaluation metrics and fuzzy-logic correlation methodology as an integrated run-time evaluation into our autonomic architecture. To achieve this, the employed evaluation criteria should be adapted in a way that their measurement becomes possible with the partial information monitored by the architecture. We believe that the run-time evaluation of network state consists in a relevant advance in autonomic architecture capabilities which enables decision-making algorithms, especially learning mechanisms, to converge to an optimal solution.

Appendix A

Publications

This appendix lists the publications produced during the thesis. The publications below resulted from the content presented in this manuscript or produced in the context of the FP7 EU financed Autonomic Internet (AutoI) project:

A.1 Journal papers

- A fuzzy-logic based methodology for comparative evaluation of autonomic network architectures. Zeinab Movahedi, Meriem Abid, Wafa Berrayana, Rami Langar, Guy Pujolle. *International Journal of Network Management (IJNM)*, August 2011 (Submitted).
- A learning-based Autonomic Network Architecture. Zeinab Movahedi, Rami Langar, Guy Pujolle. *Technical report*, LIP6, September 2011 (To be submitted to a journal).
- A Service-centric Orchestration Protocol for Self-organizing Autonomic Management Systems. Javier Rubio-Loyola, Joan Serrat, Daniel Macedo, Steven Davy, Zeinab Movahedi, Guy Pujolle, *IEEE Networks, Special Issue on Managing an Autonomic Future Internet*, 2011.
- Quantitative Evaluation of Autonomic Network Architectures: a case study, Zeinab Movahedi, Wafa Berrayana, Rami Langar, Guy Pujolle, *IEEE Communications Magazine*, June 2011 (Submitted).
- A survey of autonomic network architectures and evaluation criteria, Zeinab Movahedi, Mouna Ayari, Rami Langar, Guy Pujolle. *IEEE Communications Surveys and Tutorials*, 2011.

- The Autonomic Internet Approach for the Orchestration of Next-Generation Autonomic Networks. Daniel Macedo, Zeinab Movahedi, Javier Rubio-Loyola, Antonio Astorga, Giannis Koumoutsos, and Guy Pujolle. *Annals of Telecommunication*, 2011.

A.2 International conference papers

- An Autonomic Knowledge Monitoring Scheme for Trust Management on Mobile Ad Hoc Networks, Zeinab Movahedi, Michele Nogueira, Guy Pujolle, *Wireless Communications and Networking Conference (WCNC 2012)*, September 2011 (Submitted).

- A Policy-based Orchestration Plane For The Autonomic Management Of Future Networks. Zeinab Movahedi, Meriem Abid, Daniel F. Macedo, and Guy Pujolle. *In 6th International Workshop on Next Generation Networking Middleware (NGNM) As part of ManWeek 2009*, 2009.

- ADMA: Autonomous Decentralized Management Architecture for MANETs - A Simple Self-Configuring Case Study. Mouna Ayari, Zeinab Movahedi, Farouk Kamoun, Guy Pujolle. *International Wireless Communications and Mobile Computing Conference (IWCMC)*, 2009.

- A Pilot-based Autonomic Architecture for the Control of Future Networks. Guy Pujolle, Meriem Abid, Daniel Macedo, Zeinab Movahedi, *International conference on telecommunications and multimedia (TEMU2008)*, Greece, 2008.

- The Impact of Byzantine Attacks on Survivability of MANET Routing Protocols. Michele Nogueira, Zeinab Movahedi, Helbert da Silva, Aldri Luiz dos Santos. *Networking and Eletronic Commerce Research Conference (NAEC)*, Riva del Garda, Italy, October 2007.

- Manageability of Future Internet Virtual Networks from a Practical Viewpoint. J. Rubio-Loyola, A. Astorga, J. Serrat, L. Lefevre, A. Cheniour, D. Muldowney, S. Davy, A. Galis, L. Mamatas, S. Clayman, D. Macedo, Z. Movahedi, G. Pujolle, A. Fischer, H. de Meer. *FIA Book : Towards the Future Internet - Emerging Trends from European Research*, page 105-114, Valencia, 2010.

- Open Source for Future Internet Systems. W. Chai, L. Mamatas, A. Galis, J. Rubio-Loyola, J. Serrat, A. Fischer, A. Paler, Y. Al-Hazmi, A. Berl, H. de Meer, A. Cheniour, L. Lefèvre, S. Davy, D. Muldowney, G. Koumoutsos, A. Bassi, and Z. Movahedi. *FIA2009 : Future Internet Assembly in Stockholm - Poster*, November 2009.

- Self-organising Management Overlays for Future Internet Services. L. Cheng, A. Galis, B. Mathieu, K. Jean, R. Ocampo, L. Mamatas, J. Rubio-Loyola, J. Serrat, A. Berl, H. de Meer, S. Davy, Z. Movahedi, L. Lefèvre. MACE 2008, page 74-89, 2008.

A.3 Project deliverables

Moreover, during my thesis I have participated in the preparation of the following deliverables in the context of the AutoI project:

- Orchestration Plane and Interfaces. Deliverable D2.2 of AutoI project, June 2010
- Initial Orchestration Plane Design. Deliverable D2.1 of AutoI project, January 2009
- Autonomic Internet Initial Framework. Deliverable D6.1 of AutoI project, August 2008

List of figures

List of tables

References

[1] A. G. Ganek and T. A. Corbi, "The dawning of the autonomic computing era," *IBM System Journal*, vol. 42, pp. 5–18, January 2003.

[2] D. A. Patterson, "A simple way to estimate the cost of downtime," in *Proceedings of the 16th USENIX conference on System administration*, 2002.

[3] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, no. 1, pp. 41–50, 2003.

[4] S. Dobson, S. Denazis, A. Fernández, D. Gaïti, E. Gelenbe, F. Massacci, P. Nixon, F. Saffre, N. Schmidt, and F. Zambonelli, "A survey of autonomic communications," *ACM Trans. Auton. Adapt. Syst.*, vol. 1, no. 2, no. 2, pp. 223–259, 2006.

[5] M. C. Huebscher and J. A. McCann, "A survey of autonomic computing - degrees, models, and applications," *ACM Comput. Surv.*, vol. 40, no. 3, no. 3, 2008.

[6] The NSF Wireless Mobile Planning Group, "New architectures and disruptive technologies for the future internet: The wireless, mobile and sensor network perspective," tech. rep., NSF Wireless Mobile Planning Group Workshop, August 2005.

[7] Jon Crowcroft, "Gc4: scalable ubiquitous computing systems," *Grand Challenges in Computing, The British Computer Society*, 2004.

[8] Community Research & Development Information Service, "Information society technologies framework programme 6 - towards a global dependability and security framework,"

[9] F. Cantin, V. Goebel, B. Gueye, D. Kaafar, G. Leduc, M. Siekkinen, J. Xiao, and M. Young, "Self-optimization mechanisms, Deliverable d.3.8," tech. rep., ANA project, January 2009.

[10] A. Bassi, S. Denazis, A. Galis, C. Fahy, M. Serrano, and J. Serrat, "Autonomic internet: A perspective for future internet services based on autonomic principles," in *in Proc. of 2nd IEEE International Workshop on Modelling Autonomic Communications (MACE)*, 2007.

[11] R. Chadha, Y.-H. Cheng, J. Chiang, G. Levin, S.-W. Li, and A. Poylisher, "Policy-based mobile ad hoc network management for drama," *MILCOM Journal*, vol. 3, pp. 1317–1323, December 2004.

[12] A. Malatras, A. M. Hadjiantonis, and G. Pavlou, "Exploiting context-awareness for the autonomic management of mobile ad hoc networks," *Journal of Network and Systems Management*, vol. 15, pp. 29–55, March 2007.

[13] M. Ayari, Z. Movahedi, F. Kamoun, and G. Pujolle, "Adma: Autonomous decentralized management architecture for manets - a simple self-configuring case study," in *Prodeedings of the International Wireless Communications and Mobile Computing Conference (IWCMC), Autonomic Wireless Networking Workshop*, pp. Leipzig, Germany, Leipzig, Germany, June 2000.

[14] L. Z. Granville, G. A. F. de Sá Coelho, M. J. B. Almeida, and L. M. R. Tarouco, "An architecture for automated replacement of qos policies," in *Proceedings of the Seventh International Symposium on Computers and Communications (ISCC'02)*, ISCC '02, (Washington, DC, USA), pp. 796–, IEEE Computer Society, 2002.

[15] A. K. Bandara, E. Lupu, J. D. Moffett, and A. Russo, "A goal-based approach to policy refinement," in *POLICY*, pp. 229–239, 2004.

[16] S. Uttamchandani, C. L. Talcott, and D. Pease, "Eos: An approach of using behavior implications for policy-based self-management," in *DSOM*, pp. 16–27, 2003.

[17] D. M. Chess, A. Segal, and I. Whalley, "Unity: Experiences with a prototype autonomic computing system," in *ICAC '04: Proceedings of the First International Conference on Autonomic Computing*, (Washington, DC, USA), pp. 140–147, IEEE Computer Society, 2004.

[18] R. W. Thomas, L. A. Dasilva, and A. B. Mackenzie, "Cognitive networks," in *Proceedings of IEEE DySPAN 2005*, pp. 352–360, November 2005.

[19] J. Wainwright and M. Mulligan, *Environmental Modelling: Finding Simplicity in Complexity*. John Wiley and Sons, 2004.

[20] I. W. Paper, "An architectural blueprint for autonomic computing," tech. rep., IBM, June.

[21] S. Schmid, M. Sifalakis, and D. Hutchison, "Towards autonomic networks," in *Autonomic Networking*, pp. 1–11, 2006.

[22] D. F. Bantz, C. Bisdikian, D. Challener, J. P. Karidis, S. Mastrianni, A. Mohindra, D. G. Shea, and M. Vanover, "Autonomic personal computing," *IBM Syst. J.*, vol. 42, no. 1, no. 1, pp. 165–176, 2003.

[23] N. Samaan and A. Karmouch, "Towards autonomic network management: an analysis of current and future research directions," *Communications Surveys and Tutorials, IEEE*, vol. 11, no. 3, pp. 22–36, Quarter 2009.

[24] M. A. Razzaque, S. Dobson, and P. Nixon, "Cross-layer architectures for autonomic communications," *J. Netw. Syst. Manage.*, vol. 15, no. 1, no. 1, pp. 13–27, 2007.

[25] M. A. Razzaque, S. A. Dobson, and P. Nixon, "A cross-layer architecture for autonomic communications," in *Autonomic Networking*, pp. 25–35, 2006.

[26] W. Berrayana, G. Pujolle, and H. Youssef, "Xlengine: a cross-layer autonomic architecture with network wide knowledge for qos support in wireless networks," in *IWCMC '09: Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing*, (New York, NY, USA), pp. 170–175, ACM, 2009.

[27] R. Winter, J. Schiller, N. Nikaein, and C. Bonnet, "Crosstalk: a data dissemination-based crosslayer architecture for mobile ad-hoc networks," in *ASWN 2005, 5th Workshop on Applications and Services in Wireless Networks, June 29 - July 1, 2005, Paris, France*, 06 2005.

[28] R. Winter, J. H. Schiller, N. Nikaein, and C. Bonnet, "Crosstalk: Cross-layer decision support based on global knowledge," *Communications Magazine, IEEE*, vol. 44, no. 1, pp. 93 – 99, January 2006.

[29] "Autonomic Internet (AutoI) FP7 project,"

[30] Z. Movahedi, M. Abid, D. Fernandes Macedo, and G. Pujolle, "A policy-based orchestration plane for the autonomic management of future networks," in *6th International Workshop on Next-Generation Networking Middleware (NGNM) as part of Manweek 2009*, October 2009.

[31] L. Mamatas, "Towards an information management overlay for the future internet," in *Joint EMANICS, AutoI, Self-Net Workshop on Autonomic Management*, April 2009.

[32] A. Galis, S. Denazis, A. Bassi, P. Giacomin, A. Berl, A. Fischer, H. de Meer, J. Srassner, S. Davy, D. Macedo, G. Pujolle, J. R. Loyola, J. Serrat, L. Lefèvre, and A. Cheniour, "Management architecture and systems for future internet networks," in *FIA Book: "Towards the Future Internet - A European Research Perspective"*, (Prague), pp. 112–122, IOS Press, May 2009. ISBN 978-1-60750-007-0.

[33] J. Rubio-Loyola, J. Serrat, A. Astorga, A. Fischer, A. Berl, H. de Meer, and G. Koumoutsos, "A viewpoint of the network management paradigm for future internet networks," in *Proceedings of 1st IFIP/IEEE International Workshop on Management of the Future Internet (ManFI 2009)*, June 2009.

[34] C. Fahy, S. Davy, Z. Boudjemil, S. Meer, J. R. Loyola, J. Serrat, J. Strassner, A. Berl, H. Meer, and D. Macedo, "Towards an information model that supports service-aware, self-managing virtual resources," in *MACE '08: Proceedings of the 3rd IEEE international workshop on Modelling Autonomic Communications Environments*, (Berlin, Heidelberg), pp. 102–107, Springer-Verlag, 2008.

[35] J. Rubio-Loyola, J. Serrat, and et al., "Deliverable D6.3: Final results of the autonomic internet approach," 2010. project deliverable AutoI.

[36] "Open source software of AutoI FP7 project,"

[37] R. Chadha and C.-Y. Chiang, "Drama: Distributed policy management for manets," in *Proceedings of the IEEE Workshop on Policies for Distributed Systems and Networks, 2008. POLICY 2008*, pp. 235–237, Policy 2008, June 2008.

[38] C.-Y. Chiang, R. Chadha, S. Newman, and R. Lo, "Towards automation of management and planning for future military tactical networks," in *Military Communications Conference, 2006. MILCOM 2006. IEEE*, pp. 1 –7, 2006.

[39] C.-Y. Chiang, R. Chadha, Y.-H. Cheng, G. Levin, S. Li, and A. Poylisher, "Novel software agent framework with embedded policy control," in *Proceedings of MILCOM 2005, IEEE Military Communications Conference*, pp. 2863–2869, MILCOM 2005, October 2005.

[40] C.-Y. Chiang, G. Levin, Y. Gottlieb, R. Chadha, S. Li, A. Poylisher, S. Newman, R. Lo, and W. Izzo, "An automated policy generation system for mobile ad hoc

networks," in *Proceedings of MILCOM 2007, IEEE Military Communications Conference*, MILCOM 2007, October 2007.

[41] C.-Y. Chiang, S. Demers, P. Gopalakrishnan, L. Kant, A. Poylisher, Y.-H. Cheng, R. Chadha, G. Levin, S. Li, Y. Ling, S. Newman, L. LaVergne, and R. Lo, "Performance analysis of drama: A distributed policy-based system for manet management," in *Military Communications Conference, 2006. MILCOM 2006. IEEE*, pp. 1 –8, 2006.

[42] A. Hadjiantonis, A. Malatras, and G. Pavlou, "A context-aware, policy-based framework for the management of manets," in *The Seventh IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'06) Proceedings*, pp. 23–34, POLICY 2006, June 2006.

[43] A. Malatras, G. Pavlou, and S. Sivavakeesar, "A programmable framework for the deployment of services and protocols in mobile ad hoc networks," *IEEE Transactions on Network and Service Management*, vol. 4, no. 3, pp. 12–24, December 2007.

[44] A. M. Hadjiantonis and G. Pavlou, "Policy-based self-management of hybrid ad hoc networks for dynamic channel configuration," in *Prodeedings of the 11th IEEE/IFIP Network Operations and Management Symposium (NOMS)*, pp. 433–440, Salvador - Bahia - Brazil, April 2008.

[45] G. Tesauro, D. M. Chess, W. E. Walsh, R. Das, A. Segal, I. Whalley, J. O. Kephart, and S. R. White, "A multi-agent systems approach to autonomic computing," in *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, (Washington, DC, USA), pp. 464–471, IEEE Computer Society, 2004.

[46] J. O. Kephart and R. Das, "Achieving self-management via utility functions," *IEEE Internet Computing*, vol. 11, no. 1, no. 1, pp. 40–48, 2007.

[47] M. Ayari, F. Kamoun, and G. Pujolle, "Towards autonomous mobile ad hoc networks: A distributed policy-based management approach," in *Prodeedings of the The Fourth International Conference on Wireless and Mobile Communications (ICWMC)*, pp. 201–206, Athens - Greece, July 2008.

[48] M. Ayari, F. Kamoun, and Pujolle, "Distributed policy management protocol for self-configuring mobile ad-hoc networks," in *IFIP international Federation for Information Processing, Volume 265, Advances in Ad Hoc Networking*, pp. 73–84, Spain, July 2008.

[49] "Autonomic Network Architecture (ANA) FP7 project,"

[50] M. Sifalakis, A. Louca, L. Peluso, A. Mauthe, and T. Zseby, "A functional composition framework for autonomic network architectures," in *2nd IEEE International Workshop on Autonomic Communications and Network Management (IEEE NOMS/ACNM '08)*, IEEE, April 2008.

[51] C. Jelger, C. F. Tschudin, S. Schmid, and G. Leduc, "Basic abstractions for an autonomic network architecture," in *WOWMOM*, pp. 1–6, 2007.

[52] V. Goebel, E. Munthe-Kaas, T. Plagemann, B. Gueye, G. Leduc, S. Martin, C. Mertz, D. Witaszek, and T. Hossmann, "Integrated monitoring support in ANA, Deliverable D.3.7," tech. rep., ANA project, February 2009.

[53] G. Bouabene, C. Jelger, and S. Schmid, "ANA blueprint, Deliverable d1.4," tech. rep., ANA project, February 2009.

[54] G. Bouabene, Q. Zhang, and M. Scholler, "Final design, evaluation and prototype implementation of content-based routing mechanisms, Deliverable d.2.14," tech. rep., ANA project, January 2009.

[55] Y. Barouni, M. Bicudo, P. Spathis, K. Oikonomou, J. Xiao, and Q. Zhang, "Final design, evaluation and prototype implementation of content-based routing mechanisms, Deliverable d.2.16," tech. rep., ANA project, January 2009.

[56] "Open source software of AutoI FP7 project,"

[57] "4WARD FP7 project,"

[58] M. Franzke and G. H. et al., "Deliverable D4.2: In-network management concept," 2009. project deliverable 4WARD.

[59] A. Gonzalez and et al., "Deliverable D4.3: In-network management design," 2010. project deliverable 4WARD.

[60] F. Wuhib, M. Dam, and R. Stadler, "Decentralized detection of global threshold crossings using aggregation trees," *Comput. Netw.*, vol. 52, pp. 1745–1761, June 2008.

[61] F. Wuhib, M. Dam, and R. Stadler, "Robust monitoring of network-wide aggregates through gossiping," in *In Proc. Tenth IFIP/IEEE International Symposium on Integrated Management (IM 2007*, pp. 21–25, 2007.

[62] J. Mitola III, "Cognitive radio for flexible mobile multimedia communications," *Mobile Networks and Applications*, vol. 6, pp. 435–441, 2001. 10.1023/A:1011426600077.

[63] R. W. Thomas, D. H. Friend, L. A. Dasilva, and A. B. Mackenzie, "Cognitive networks: adaptation and learning to achieve end-to-end performance objectives," *Communications Magazine, IEEE*, vol. 44, no. 12, no. 12, pp. 51 –57, 2006.

[64] D. Kliazovich, F. Granelli, and N. F. R. Piesiewicz, "Architectures and cross-layer design for cognitive networks," in *World Scientific Publishing Co, Inc. River Edge, Handbook on Sensor Networks*, May 2009.

[65] D. Kliazovich, N. Malheiros, N. Fonseca, F. Granelli, R. Piesiewicz, and E. Madeira, "Cogprot: A framework for cognitive configuration and optimization of communication protocols," in *2nd International Conference on Mobile Lightweight Wireless Systems (MOBILIGHT)*, 2010.

[66] T. De Wolf and T. Holvoet, "Evaluation and comparison of decentralised autonomic computing systems," in *In Department of Computer Science, K.U.Leuven*, (Leuven, Belgium), Report CW 437, March 2006.

[67] P. Oreizy, M. M. Gorlick, R. N. Taylor, D. Heimbigner, G. Johnson, N. Medvidovic, A. Quilici, D. S. Rosenblum, and A. L. Wolf, "An architecture-based approach to self-adaptive software," *IEEE Intelligent Systems*, vol. 14, no. 3, no. 3, pp. 54–62, 1999.

[68] S. Calo and M. Sloman, "Guest editorial: Policy-based management of networks and services," *J. Netw. Syst. Manage.*, vol. 11, no. 3, no. 3, pp. 249–252, 2003.

[69] P. Dittrich and P. Speroni Di Fenizio, "Chemical organisation theory," in *Systems Biology* (H. Hauser, M. Betenbaugh, N. Jenkins, C. MacDonald, O.-W. Merten, M. Al-Rubeai, and M. Fussenegger, eds.), vol. 5 of *Cell Engineering*, pp. 361–393, Springer Netherlands.

[70] W. Berrayana, S. Lohier, G. Pujolle, and H. Youssef, "Local view's dissemination for a network wide optimization to improve qos in ad hoc networks," *Global Information Infrastructure Symposium (GIIS 2007), IEEE*, vol. 44, no. 1, pp. 207 – 210, July 2007.

[71] T. Bullot, R. Khatoun, L. Hugues, D. Gaîti, and L. Merghem-Boulahia, "A situatedness-based knowledge plane for autonomic networking," *Int. J. Netw. Manag.*, vol. 18, no. 2, no. 2, pp. 171–193, 2008.

[72] A. M. Hadjiantonis, A. Malatras, and G. Pavlou, "A context-aware, policy-based framework for the management of manets," in *POLICY '06: Proceedings of the Sev-*

enth IEEE International Workshop on Policies for Distributed Systems and Networks, (Washington, DC, USA), pp. 23–34, IEEE Computer Society, 2006.

[73] E. Lupu and M. Sloman, "Conflict analysis for management policies," in *Proceedings of the fifth IFIP/IEEE international symposium on Integrated network management V : integrated management in a virtual world*, (London, UK, UK), pp. 430–443, Chapman & Hall, Ltd., 1997.

[74] E. Gelenbe, "Random neural networks with negative and positive signals and product form solution," *Neural Comput.*, vol. 1, pp. 502–510, December 1989.

[75] S. Timotheou, "The random neural network: A survey," *Comput. J.*, vol. 53, no. 3, no. 3, pp. 251–267, 2010.

[76] E. Gelenbe, "Theory of the random neural network.," *Gelenbe, E. (ed.), Neural Networks: Advances and Applications, Elsevier Science Publishers B.V.*, vol. 53, no. 3, no. 3, pp. 1–20, 2010.

[77] M. A. Fischler and O. Firschein, *Intelligence: the eye, the brain, and the computer*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1987.

[78] A. Wong, P. Ray, N. Parameswaran, and J. Strassner, "Ontology mapping for the interoperability problem in network management," *Selected Areas in Communications, IEEE Journal on*, vol. 23, no. 10, pp. 2058 – 2068, oct. 2005.

[79] L. Stojanovic, J. Schneider, A. Maedche, S. Libischer, R. Studer, T. Lumpp, A. Abecker, G. Breiter, and J. Dinger, "The role of ontologies in autonomic computing systems," *IBM Syst. J.*, vol. 43, pp. 598–616, July 2004.

[80] D. Johnson, Y. Hu, and D. Maltz, "The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4." RFC 4728 (Experimental), Feb. 2007.

[81] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing." RFC 3561 (Experimental), July 2003.

[82] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)." RFC 3626 (Experimental), Oct. 2003.

[83] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers," in *Proceedings of the conference on Communications architectures, protocols and applications*, SIGCOMM '94, (New York, NY, USA), pp. 234–244, ACM, 1994.

[84] E. Gelenbe and R. Lent, "Power-aware ad hoc cognitive packet networks," *Ad Hoc Networks Journal*, vol. 2, no. 3, pp. 205–216, July 2004.

[85] P. Velloso, R. Laufer, D. Cunha, O. Duarte, and G. Pujolle, "Trust management in mobile ad hoc networks using a scalable maturity-based model.," *IEEE Transactions on Network and Service Management*, vol. 7, no. 3, no. 3, pp. 172–185, 2010.

[86] J. Cho, A. Swami, and I. Chen, "A survey on trust management for mobile ad hoc networks," *IEEE Communications Surveysand Tutorials*, vol. PP, no. 99, no. 99, pp. 1–22, 2010.

[87] J. Hu and M. Burmester, "Cooperation in mobile ad hoc networks," in *Guide to Wireless Ad Hoc Networks*, Computer Communications and Networks, pp. 43–57, Springer London, 2009.

[88] Y. Lindsay Sun, Z. Han, W. Yu, and L. K. Ray, "A trust evaluation framework in distributed networks: Vulnerability analysis and defense against attacks," in *IEEE INFOCOM*, pp. 230–236, 2006.

[89] L. J., L. R., and K. J., "Future trust management framework for mobile ad hoc networks," *IEEE Communications Magazine*, vol. 46, no. 4, pp. 108 –114, April 2008.

[90] M. Morvan and S. Sene, "A distributed trust diffusion protocol for ad hoc networks," in *Proceedings of the International Multi-Conference on Computing in the Global Information Technology*, pp. 87–, 2006.

[91] S. Buchegger and J. Le Boudec, "Performance analysis of the confidant protocol," in *Proceedings of the ACM international symposium on Mobile ad hoc networking & computing*, MobiHoc '02, (New York, NY, USA), pp. 226–236, ACM, 2002.

[92] G. Bella, G. Costantino, and S. Riccobene, "Managing reputation over manets," in *Proceedings of the Fourth International Conference on Information Assurance and Security*, (Washington, DC, USA), pp. 255–260, IEEE Computer Society, 2008.

[93] IBM, "http://www-03.ibm.com/autonomic/about-get-model.html,"

[94] Z. Movahedi, M. Ayari, R. Langar, and G. Pujolle, "A survey of autonomic network architectures and evaluation criteria," *Communications Surveys Tutorials, IEEE*, vol. PP, no. 99, no. 99, pp. 1 –27, 2011.

[95] J. A. McCann and M. C. Huebscher, "Evaluation issues in autonomic computing," in *GCC Workshops*, pp. 597–608, 2004.

[96] A. B. Brown, J. Hellerstein, M. Hogstrom, T. Lau, P. Shum, and M. P. Yost, "Benchmarking autonomic capabilities: Promises and pitfalls," in *ICAC '04: Proceedings of the First International Conference on Autonomic Computing*, (Washington, DC, USA), pp. 266–267, IEEE Computer Society, 2004.

[97] H. Chan, A. Segal, B. Arnold, and I. Whalley, "How can we trust an autonomic system to make the best decision?," in *ICAC '05: Proceedings of the Second International Conference on Automatic Computing*, (Washington, DC, USA), pp. 351–352, IEEE Computer Society, 2005.

[98] A. Lotfi Zadeh, "Fuzzy logic," *IEEE Computer*, vol. 21, no. 4, no. 4, pp. 83–93, 1988.

[99] L. A. Zadeh, "Fuzzy logic," *IEEE Computer*, vol. 21, no. 4, no. 4, pp. 83–93, 1988.

[100] J. Nie, J. Wen, J. Luo, X. He, and Z. Zhou, "An adaptive fuzzy logic based secure routing protocol in mobile ad hoc networks," *Fuzzy Sets and Systems*, vol. 157, no. 12, no. 12, pp. 1704–1712, 2006.

[101] M. Lima, H. da Silva, A. dos Santos, and G. Pujolle, "Survival multipath routing for manets," pp. 425 –432, April 2008.

[102] B. Tsetsgee and T. Lkhagvasuren, "The study on quality requirements of interactive voice in ip network," *Strategic Technologies, 2008. IFOST 2008. Third International Forum on*, pp. 314 – 318, June 2008.

[103] M. Abid, I. Fejjari, and G. Pujolle, "An autonomic piloting plane for the handover decision optimization.," in *NTMS* (K. A. Agha, M. Badra, and G. B. Newby, eds.), pp. 1–5, IEEE, 2009.

[104] Y. Chen, T. Farley, and N. Ye, "Qos requirements of network applications on the internet," *Inf. Knowl. Syst. Manag.*, vol. 4, pp. 55–76, January 2004.

[105] "Ginkgo-networks, white paper - ginkgo distributed network piloting system.," 2008.

[106] M. Abid, T. Ali-Yahiya, and G. Pujolle, "Hodstat: a handover decision stability technique for minimizing instability in wireless local area networks," in *IWCMC*, pp. 376–380, 2010.

[107] "Java 1.5 by sun microsystems,"