A Survey of Autonomic Network Architectures and Evaluation Criteria

Zeinab Movahedi, Mouna Ayari, Rami Langar, and Guy Pujolle

***Abstract*—Autonomic network management is a promising approach to reduce the cost and the complexity of managing network infrastructures. It attempts to lead the human administrator out of the network control loop, leaving the management tasks to be performed by the network itself. Due to its important implication on automating management systems, this area has attracted a growing attention from both academia and industry. In this paper, we provide a holistic view of autonomic architecture proposals and the evaluation metrics existing so far. Based on this, we identify some new criteria important to the autonomic architectures. Finally, we compare different existing autonomic architectures and describe the pros and cons of each one regarding to the network management and performances.**

***Index Terms*—Autonomic Computing, Network Management, Autonomic Architecture, Architecture Evaluation.**

I. Introduction

OVER the recent years, communication technologies has significantly evolved in both wired and wireless environments, enabling a wide range of applications and services for hundreds of million of users connected to the network. This evolution has complicated the management of current and emerging network technologies.

To overcome this ever-increasing size and complexity of management, the autonomic computing initiative and in particular, autonomic network management (ANM), have been proposed as an emerging solution. The essence of autonomic computing is to develop self-managing computing systems [1] that manage themselves given high-level objectives from administrators. This paradigm is inspired from the autonomous nervous system of the human body [2], which checks blood sugar level and maintains normal body temperature without any conscious effort. IBM defined four self-properties an autonomic system should exhibit: self-configuring, self-healing, self-optimizing and self-protecting. 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.

Autonomic networking is a special case of autonomic computing. It relates to the capability of the network to operate and serve its purpose by managing its own self without external intervention even in case of environmental changes. The autonomicity can be achieved via a distributed management architecture where a set of autonomic elements cooperate with each other to provide a convergent autonomic management. The autonomic elements are composed of the monitoring, analyzing, planning and executing components, referred to as the MAPE-K model, to enable the self-adaptation to the environment changes.

Compared to autonomic system management, the autonomic management of networks is much more complex. This is due to the existence of several management standards, different protocols and different vendors that may conflict when two or more heterogenous management domains are interconnected.

Since IBM launched autonomic computing initiative in 2001 [3], a number of research has been carried out unifying recent advancements and trends in various areas of research applicable to autonomic computing. Recent surveys [4] [5] focused on work performed on each of the main building blocks of an autonomic element. To the best of our knowledge, a survey of complete architectural solutions has not been provided so far. To this end, this paper first provides a holistic view of existing complete autonomic networking architectures that tie together all the MAPE-K building blocks. Besides, rather than trying to enumerate an exhaustive list, we have chosen to review only architectures which by minor extension, meet all the management issues. Work carried out on one particular component of MAPE-K model is already addressed in the literature and, hence, is out of the scope of this survey. Moreover, we present an overview of the autonomic solutions' evaluation techniques and identify some innovative qualitative and quantitative evaluation metrics. Contrary to existing papers which rather only mention some important evaluation metrics, we describe how these criteria can be measured. Based on the identified qualitative criteria, we analyze and compare the existing autonomic architectures. This comparison may lead to a convergent autonomic architecture which takes advantage of the pros of the surveyed architectures while avoiding their limitations and drawbacks.

The rest of this article is organized as follows. Section II provides a general introduction to autonomic computing and reports the related work. Section III presents a coherent methodology to classify the autonomic networking architectures and provides an overview of existing complete architectural solutions. Section IV identifies and describes the main quantitative and qualitative evaluation criteria which can be used to evaluate the scope and the efficiency of existing solutions. Moreover, it presents a qualitative comparison of surveyed architectures. Finally, section V concludes this paper.

II. Background

The term of autonomicity has been the subject of misapplying in IT community. Above all, the automaticity has

Manuscript received 15 June 2010; revised 27 December 2010.

The authors are with the Department of Systems and Networks Engineering, Laboratoire d'Informatique de Paris 6, University Pierre et Marie Curie, 75005 Paris, France (e-mail: name.lastname@lip6.fr).

M. Ayari is also with CRISTAL lab, National School of Computer Sciences Manouba, 2010 Manouba, Tunisia (e-mail: mouna.ayari@cristal.rnu.tn).

Digital Object Identifier 10.1109/SURV.2011.042711.00078

been amiss taken more often for autonomicity. Following, we provide the clear definition of each term:

- *Automaticity* [6] 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 type of packets.
- *Autonomicity* [2] 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 focus of this paper is on autonomic communication. The autonomicity implies a set of well-known properties and characteristics that are outlined in next sections.

A. Self-management properties

The main properties of self-management [1] [7] as portrayed by IBM are described in the following:

- **Self-configuring**: This property refers to the capacity of the system to configure and reconfigure itself in accordance with high-level policies 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].

B. 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.

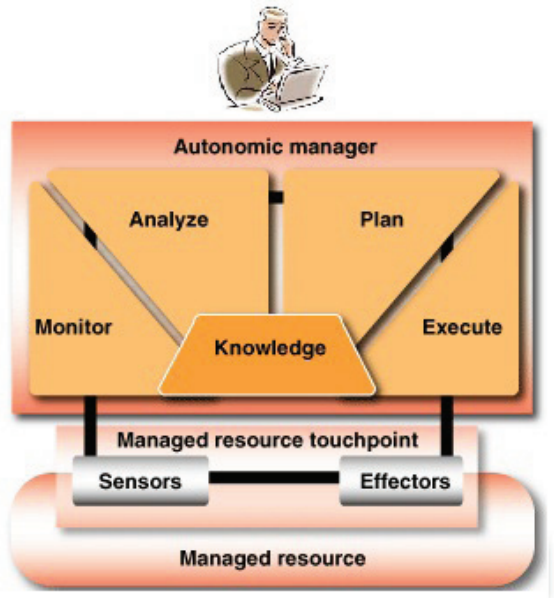

Fig. 1. IBM's MAPE-K reference model for the ACL

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 [6]. 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 the global self-management of 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 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) [2] 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 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 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 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.
- 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.
- A *knowledge base* where aggregated 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 network and detects events.
- A *planner* which constructs adaptation plans based of the result of the analyze process.
- An *executor* which, based on the output of the plan process, reconfigures the managed component and communicates with other autonomic managers.

As illustrated in Figure 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.

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 resource 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 powerful of adaptation approach is affected by its ability to continuously evolve and enhance the applied adaptation strategies. This takes advantage of learning from previous experiences to perform more optimally in the future.

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 [8], [9], [10] of network elements.

In networking, 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 [11]. A node can utilize the global view to evaluate its own *relative* status [12]. 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, monitoring approach should ensure minimum extra overhead while maintaining a correct network view.

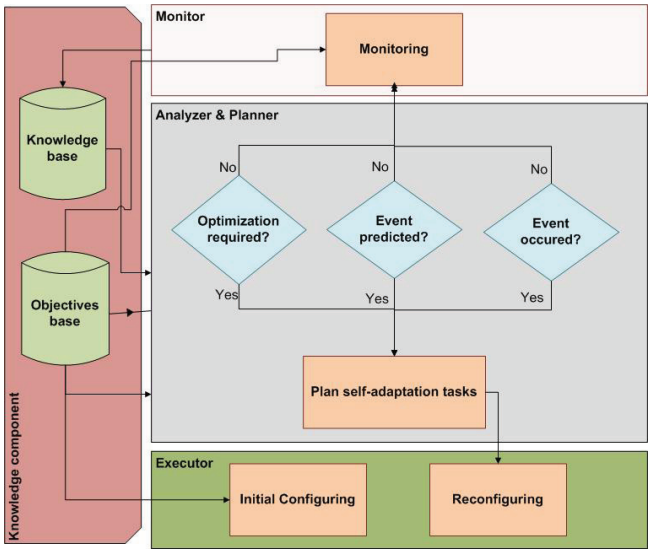

Fig. 2. Detailed MAPE-K model for the ACL

The following subsection provides an overview of previous works carried out in the literature on surveying autonomic network efforts.

C. Related work

A significant body of work has been dedicated to autonomic communication principles and characteristics. However, a literature search does not yield relevant progress in evaluating and comparing Autonomic Network Management (ANM) architectures. We believe that one of the major challenges is the dispersion of research in the architectural area of autonomic networking. In the following, we provide the current state-of-the art of the work carrying out on both classification and evaluation of autonomic computing solutions.

In [22], a review of the research done on autonomic system's frameworks and infrastructures is presented. In the first part of the survey, the authors classified existing solutions into seven areas: biologically inspired, large-scale distributed, agent-based, component-based, technique focused, self-managed service oriented and non autonomous system specific frameworks. In the second part, the authors enumerated and analyzed the techniques that can be used to achieve each of the self-management capabilities. However, some major issues still remain.

First, the proposed categorization is too generic to differentiate between existing architectures regarding specific relevant characteristics or autonomic techniques. Therefore, one architecture may belong to more than one category. For example, a biologically inspired approach may belong to other three areas: large scale, agent-based and artificial intelligent technique focused frameworks. In addition, the authors do not provide a clear description of definition and properties of each class, leaving the categorization ambiguous to an unspecialized reader. Moreover, the evaluation approach of each proposed class of autonomic architectures is not provided.

TABLE I
BRIEF OVERVIEW OF SURVEYS CARRIED OUT IN THE FIELD OF AUTONOMIC COMMUNICATION

Subject covered	related papers
Categorization and regroupment of the existing research efforts	[13], [14], [15] and [16]
Overview of open problems and challenges	[17], [18], [1], [19] and [20]
Overview of academic and industry based autonomic projects	[14], [21]
Survey of works carried out on one or more self-management properties	[22], [23] and [24]
Survey of autonomic frameworks and architectures	[22], [5] and [4]
Survey of works carried on MAPE-K components	[5], [4]

A fairly comprehensive survey of the work focusing on each of the MAPE-K components and their implementation is provided in [5]. Contrary to [22], the authors were interested on the evaluation of the degree of autonomicity of each presented MAPE-K loop focus. They defined four elements of autonomicity as follows: Support, Core, Autonomous and Autonomic. The support element is when one particular aspect or component of an architecture affecting the performance of the complete architecture is considered. An autonomic monitoring tool is an example of 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 architecture is considered which reflects its own performance and adapts itself accordingly. In this level, the administrator participates only in initial parameterizing of autonomic manager(s) and it is up to the network to operate smoothly afterwards. The main limitation of this model is that it is too generic and does not describe how one can compare between two solutions belonging to a same category.

In [4], an overview of current approaches focusing on each of the management functionalities (monitoring, analysis, plan and execution) is presented. The authors provided a coherent classification methodology to classify existing research efforts. Furthermore, the authors identified six classes of properties mandatory to enable the architecture with autonomic principles. 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 do not describe how the proposed evaluation criteria can be measured. In contrast with our paper, the authors analyze the main contributions on each MAPE-K component separately and do not focus on complete ANM architectures. They only mention some of the major research projects targeting the development of such architectures, without describing approaches used for each of the autonomic architecture's building blocks.

Other works carried out in the field of autonomic communication exist. They are summarized in Table I.

So far, the above-mentioned surveys have focused mainly on providing a classification of existing autonomic architectures rather than presenting how to evaluate them. Few surveys [21] [4] have been discussed this issue.

First efforts to describe the degree of autonomicity of a system have been proposed by IBM [25] as a set of Autonomic Computing Adoption Model Levels. The proposed 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 which use monitoring tools that allow them to discover the network's state and make require changes manually. Managed level presents a system which performs monitoring in an intelligent way that reduces the system administration burden and simplifies for administrators to plan accordingly. In Predictive level, monitoring is performed in a way that enables the system to recognize its behavior and suggest actions approved and carried out by IT staff. The Adaptive level is where the system is able to suggest actions and carry out certain adaptation without human involvement. The Autonomic level refers to a full autonomic system where monitoring and adaptation are performed automatically. The problem with this model is that it is too narrow and the differentiation between levels is too vague to describe the diversity of self-management [5].

Table II 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-the-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.

In [21], a set of metrics for comparing autonomic computing solutions is provided. These metrics include Quality of Service (QoS), cost of autonomy, granularity/flexibility, degree of self-operation, just to mention a few.

The authors of [26] identified a set of evaluation views to capture various performance aspects of a distributed ANM architecture. Each view is described with regard to the MANET case study. The outlined evaluation views include 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.

The authors of [27] raised the issue of synthesizing multiple sub-metrics into an overall measure of autonomicity, and calibrating scores across different systems. However, they did not provide any further details on how these metrics can be correlated.

As one can understand from Table II, the proposed criteria do not spread over all aspects that an ANM architecture should cover. Moreover, we outline that the literature provides only a generic definition of the intended metrics, without any further description of its scope. Consequently, some metrics present

TABLE II
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.	[21], [26]
Cost of autonomy	The overhead of extra autonomic activity.	[21], [26] and [27]
Granularity/flexibility	The degree of decentralization of the architecture as a trade-off between the failure avoidance and overhead.	[21], [26]
Failure avoidance	The ability to cope with predicted and unpredicted failures.	[21]
Degree of self-operation	In which extend the architecture (or each MAPE-K components) performs in itself without administration involvement.	[21], [26], [4] and [27]
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.	[21], [26]
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.	[21], [26]
Memory strength	The ability of the system to maintain current state, behavior trend or historic of past actions in order to utilize them for management decisions.	[4]
Degree of activity	The reactive or proactive behavior 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.	[4]
Ability to learn	The ability of the architecture to continuously evolve and enhance their applied adaptation strategies.	[4]
Degree of awareness	The degree of self-awareness and environment-awareness of the architecture.	[4]
Granularity of intelligence	The degree of distribution of the intelligence in the network, i.e. the granularity of learning mechanisms.	[4]
benchmarking	A mean to bring all above-mentioned metrics together and observe their effective impact on the smooth operation of the real system.	[21]

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 qualitatively or quantitatively among ANM architectures.

III. AUTONOMIC NETWORK ARCHITECTURES

This section describes several initiatives on building autonomic network architectures. As opposed to [4] [5], which survey individual components with potentiality to contribute to the network autonomic management, this work focuses on complete architectural propositions aggregating more than one MAPE-K components. 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. The SOCRATES architecture for which the choice of the organizational structure of the architecture has not been yet finalized is addressed in a separate subsection. 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.

A. 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.

1) AutoI project architecture: Autonomic Internet (AutoI) [28] 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 [29]: 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 adjust their configuration to accommodate the federation of networks [30].

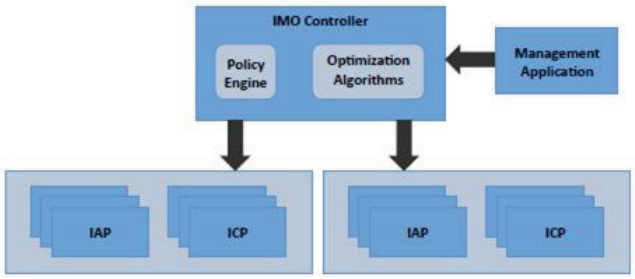

Fig. 3. The AutoI IMO structure

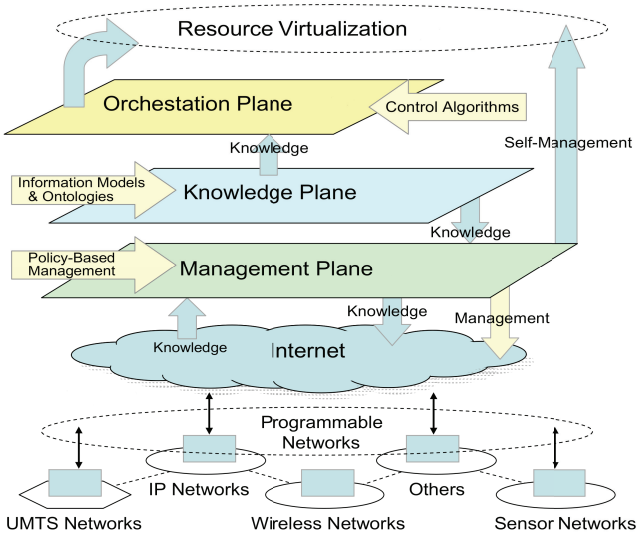

Fig. 4. The AutoI Planes

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 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 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 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, 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). 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). The deployment of one hundred and ten services over this virtual network (with one hundred and ten routers) has been executed in around 8 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. Moreover, the security aspect of the architecture should be further investigated.

2) DRAMA architecture: DRAMA (Dynamic Readdressing and Management for the Army) is a distributed policy-based network management for MANETs using intelligent agents [37]. The main purpose of DRAMA design is to set up a network that can autonomously adjust its behavior without human intervention [38]. This approach was especially designed for military networks consisting of wireless ad hoc routing domains connected by a wired backbone network. In such 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. In such context, the system 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 5, the network is managed through a two tier hierarchical architecture [37]. 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 LPA aggregation. A *Policy Agent Framework (PAF)* is integrated into the policy agent component architecture [39] [40]. It provides the following functionalities: policy enforcement, community service, asynchronous event transport and adaptive communications service.

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 they reflect the commander's intention.

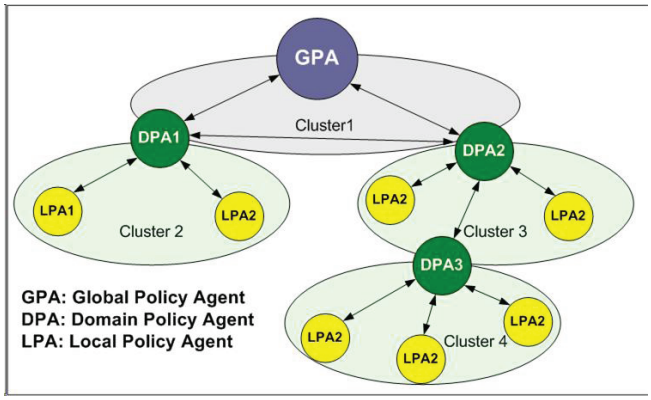

Fig. 5. DRAMA Management Hierarchy

DRCP/DCDP protocols (Dynamic Registration and Configuration Protocol/Dynamic Configuration Distributed Protocol) are conjointly used to distribute policies and 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 according 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 [39] [40] [41].

Although the successful development of DRAMA management tool [42] [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, AMPS (Ad hoc Mobility Protocol Suite), DRCP/DCDP restricts an eventual wide adoption of such solution.

3) Context-Awareness for the Autonomic Management of MANETs: A context-aware platform employing a hierarchical organizational model for MANET management was introduced in [43] [44]. The main purpose of the proposed solution was to establish a system that is capable of self-managing MANETs. In order to achieve this objective, authors adopted 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

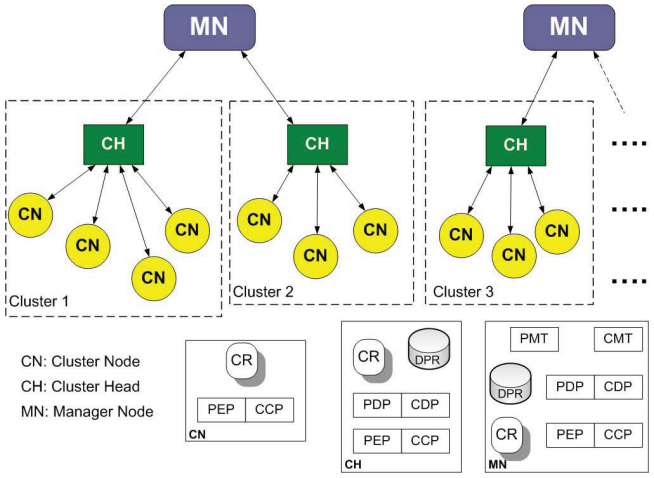

Fig. 6. Organizational Model of proposed Pavlou architecture

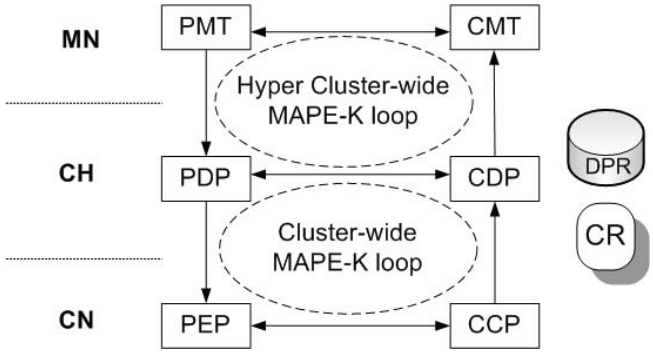

Fig. 7. Closed autonomic control loop of proposed Pavlou architecture

hybrid (hierarchical and distributed) organizational approach and hyper-cluster formation [44]. 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 6 [44], a *Context Repository (CR)* is deployed on every node of the MANET and is used to store 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 CH. We note that each MN incorporates CH functionalities. In turn, each CH implements 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. On one hand, the PEP is responsible for enforcing activated policies. On the other hand, the CCP is responsible for communicating with the sensors available to the device and collecting context information.

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 is responsible for monitoring context and communicating between both the CCP modules of CN associated with its cluster and the corresponding MN. The DPR is a set of distributed and/or replicated LDAP directories configured to store policies.

A *Policy Manager Tool* and *Context Manager Tool* components 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 to take 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 adaptation, CHs are responsible for the aggregation and processing of context and reporting it at regular intervals to the MNs.

The proposed platform introduces many interesting features. It combines hierarchical context gathering, processing and dissemination with distributed policy-based management. High level objectives expressed through policies implementing adaptability to context can guide configuration changes in the network, leading to a degree of autonomic-decision making.

The proposed architecture is qualified as autonomic since the network state can be configured and reconfigured dynamically without human intervention. This property does not deal to a fully autonomic solution but to a self-configuring one. Other self-management properties (self-optimizing, self-protecting and self-healing) are not addressed in the presented research work.

Moreover, the network is not formed by autonomous elements. Only CH and MN implement context analyzing, decision-making and planning actions functionalities. CN, which constitute the majority of network nodes, are simple sensors and effectors. They have the simple role of gathering monitoring information, processing it, send it to CH and enforce policies.

As shown in Figure 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 [45] [46]. 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 most important contribution of CA-MANET is the exploiting of context-awareness for autonomic network management. Authors evaluated the efficiency of their self-configuring approach using both testbed experimentation and network simulations [44] [45]. The studied scenarios 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

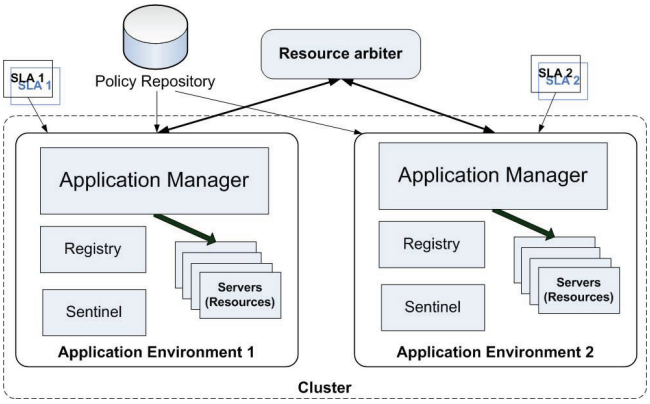

Fig. 8. Unity scenario

running applications. It is important to note that the impact of mobility and frequent topological changes was not studied.

Similar to DRAMA architecture, CA-MANET does not define any management mechanism to ensure the interoperability of the high-level clusters. Hence, the architecture, in its actual stage, does not support the integration of heterogenous networks.

4) Unity architecture: Unity architecture is an autonomic data prototype carried out at IBM's Thomas J. Watson Research Center. This earlier proposition was conceived in order to provide some needs of self-management of distributed computing systems. This work is described in [47] [48] [49]. 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 between each other in order to optimize the overall system performance and fulfil 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. An application environment provides application servers to clients. Each cluster is managed by a resource arbiter. As shown in Figure 8 [49], the main components that make up a Unity system are:

- *Application manager element*: This element is responsible for the management of available resources. Moreover, it governs interactions of the application with its environment.
- *Resource arbiter element*: It is responsible for the allocation of resources to each application environment.
- *Server element*: It is responsible for the announcement of available resources and existing server's capabilities. In the described work, authors considered servers as resources (they are identical).
- *Registry element*: It is responsible for locating all other elements with which the application manager needs to communicate.
- *Policy repository element*: Administrator high-level policies and utility functions are stored in the policy repository.
- *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.

An application environment is managed by an application manager. This later has the task of implementing policies stored in the policy repository in order to adjust resource 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 rather than self-* or self-assembly. These latter constitute the goal to achieve in authors' future works.

B. 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 aligned with each other, nodes should have a correct view of other managers and of the network context.

1) ADMA architecture: Autonomous Decentralized Management Architecture (ADMA) is a distributed management architecture designed to provide MANETs with some autonomic principles [50]. The main objective of ADMA is to introduce a degree of autonomicity to the network management process and mainly 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 [51]. All nodes

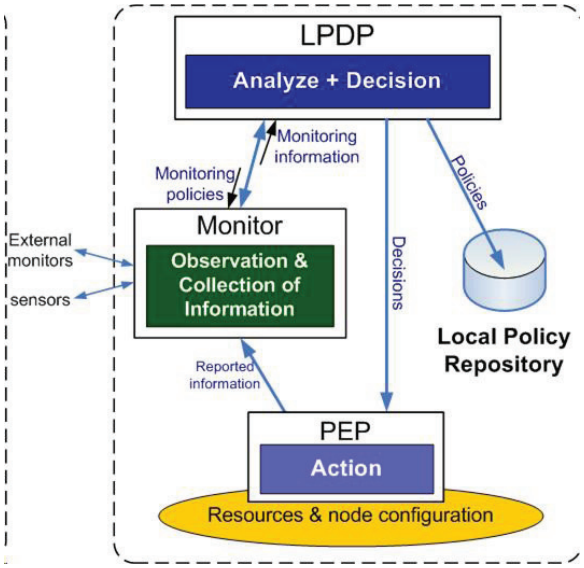

Fig. 9. ADMA node structure

are equivalent. Distributed nodes cooperate between each other in order to meet high level goals predefined by the network administrator.

The role of the network administrator is limited to defining goals and high-level policies in an early stage, before the network formation. He should be familiar with this task and qualified with a good level of experience allowing him to define the appropriate decisions that should be taken in certain well-defined circumstances and possible environment variations. He is also responsible for defining what information should be collected by monitors.

As it is depicted in Figure 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* [52] [51] [50].

The LPDP analyzes monitoring information reported by the monitor, makes and plans local decisions thanks to 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 and to avoid inconsistent decisions. The LPDP in ADMA architecture 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.

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 local policy repository is a local database where predefined policies are stored.

Predefined policies include configuration, reconfiguration, monitoring and meta-policies [50]. Policies are expressed as event-condition-action rules. Once these proactive policies are defined, the network administrator verifies their consistency. Then, it introduces them to at least one node. Policies will be disseminated 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 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.

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 [52] [51]. DPMP allows policy distribution, collecting monitoring information 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. 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.

ADMA presents interesting features. It combines policy-based management view with autonomic principles in a fully decentralized system. However, ADMA presents some insufficiencies. Coordination between autonomous nodes is limited to the replication of predefined policies and the collect of required monitoring information. The consistency 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.

2) ANA project architecture: The Autonomic Network Architecture (ANA) [53] is one of the FP6 EU-IST funded project on future and emerging technologies. 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 [54].

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 [55]. 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. It defines:

- 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

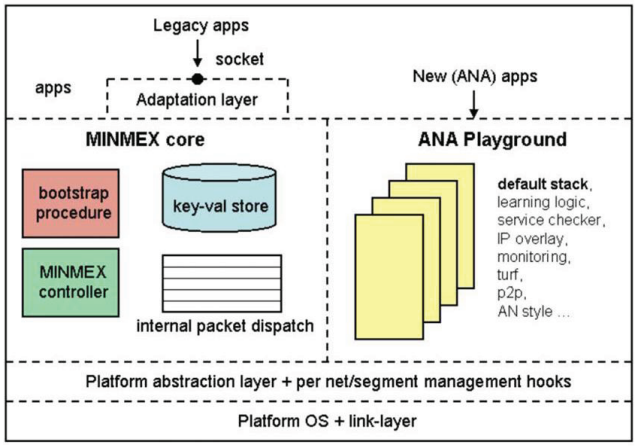

Fig. 10. An ANA node

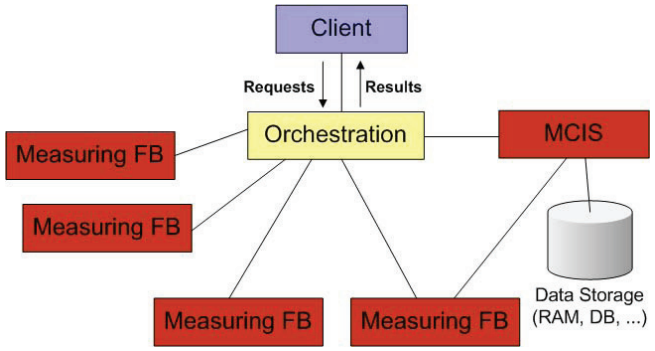

Fig. 11. Conceptual view of the monitoring framework

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 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 [56].
- *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 startpoint 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 10 [54].

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 [57].

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 11 [56], in the context of the ANA architecture, 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 [58], reliability and QoS performance tests of the proposed inter and intra compartment-wide routing schemes [59], the stability and convergence of the managed networks [60], the efficiency of the proposed self-optimization mechanism [60], just to mention a few.
A real implementation of ANA software is available online [61].

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.

3) In-Network Management architecture within 4WARD project: Wired and Wireless World Wide Architecture and Design for Future Internet (4WARD) [62] 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-

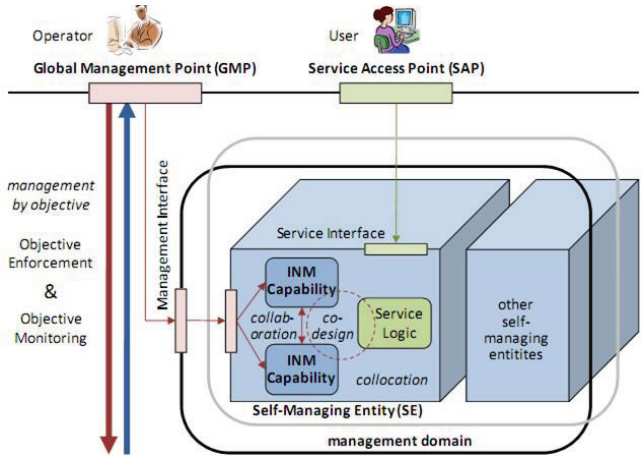

Fig. 12. Conceptual view of the In-Network Management

operability, and complementarity of several heterogeneous network architectures. In the scope of this paper, 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 nature 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. 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 management capabilities are embedded in self-managing INM entities (SE), the non-INM entities are simply to be managed by management capabilities.

As depicted in Figure 12, the INM defines a concise set of architectural elements [63] [64] from which any distributed and embedded management structure can be created:

- *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 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 operators 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 which implement the actual self-* (e.g. self-adaptive) INM 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 self-managing entities until reaching individual management capabilities. Correspondingly, objectives and performance are monitored and aggregated 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 [64].

The INM architecture makes use of tree-based and gossip-based underlying protocols to achieve distributed monitoring of network-wide aggregates. These approaches were also used to detect a threshold crossing of metrics, e.g. average or peak load. The tree-based approach, called *TCA-GAP* [65], involves creating and maintaining a spanning tree and aggregating state information along that tree, bottom-up from the leaves towards the root. Such a tree can be built in a decentralized, self-stabilizing manner, which guarantees robustness in the monitoring protocol. The aggregated information is available at the root of the spanning tree. In order to reduce the monitoring overhead, the TCA-GAP is equipped with local thresholds and a local hysteresis mechanism. This mechanism switches nodes to a passive state whenever their contribution to threshold detection is not needed. A local mechanism for dynamic re-computation of local thresholds, triggered by violation of local invariants, ensures the detection of threshold crossings.

The gossip approach relies on randomized communication to disseminate and process state information in the network. The proposed gossip protocol, called *G-GAP* [66], performs continuous monitoring of aggregates and is enhanced with a recovery mechanism from node failures. Using this approach, the aggregated data is available at all node.

In order to avoid the significant network and CPU resources usage consumed by monitoring and aggregating techniques, *Not All aT Once! (NATO!)* [67] scheme was proposed as an alternative. NATO! is an efficient statistical probability scheme for precisely estimating the size of a group of nodes affected by the same event. In addition, the proposed NATO! algorithm could assist INM to decide which management configuration is more appropriate for different network services. As an example, for the case of QoS routing, the QoS MC of the strategic node could interrogate all the devices within its domain with a specific request (i.e. what nodes are able to accommodate a flow with specific QoS requirements). According to the collected statistics, the QoS MC may decide to use either a QoS-aware routing or an alternative mechanism (e.g. network coding).

An important function required for autonomic management of highly dynamic networks is network topology and nodes discovery. To address this issue, the INM architecture developed *Hide and Seek (H&S)* [68] MC, a new algorithm for network discovery, and information propagation and synchronization. The H&S algorithm operates around two roles: seeker and hider. A seeker node sends directional contact messages to its neighborhood using a probabilistic eyesight direction (PED) function. This function aims to narrow the directions through which contact messages are sent. Once contacted, hider and seeker synchronize their knowledge, keeping track of each other. The hider becomes a new seeker and the process is repeated until all nodes have been contacted [64].

Another issue in network management is the search of users, network nodes, information, content or services of any kind residing on network resources. The INM architecture investigated the efficiency of random walks and flooding for exploring networks, based on case studies evaluated by simulation and transient analysis [69] [70].

In order to maintain critical functionality within the network, a distributed approach for autonomous anomaly detection and collaborative fault-localization was proposed. The proposed statistical method [71] [72] is based on parameter estimation of Gamma distributions obtained by measuring probe response delays via simple end-to-end transactions (performed by ICMP, etc.). The idea is to learn the expected probe response delay and packet drop rate of each connection in each node, and use it for parameter adaptation such that the manual configuration effort is minimized [64].

Three possible design options for the self-adaptation control loop are identified [64]. 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. Obviously, the first and second options are more appropriate for distributed network architectures and the third one is more suitable for centralized network topology. While the proposed INM does not rely on any specific design option, 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 systems 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.

The performances of all proposed monitoring algorithms and adaptation mechanisms have been evaluated in both wired and wireless environments [64]. The performed tests were related especially to the overhead consumed by gossip-based and tree-based aggregation approaches, the estimation error committed by the proposed group size estimation mechanism, 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.

4) Cognitive Network architecture: Another trend aiming at proposing a self-adapting network architecture is addressed by cognitive networks. Cognitive network (CogNet) 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 [73]. 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 [74], 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 [75] and cross-layer design [12].

From this definition, we can conclude that cognitive networking consists in a particular variant of autonomic com-

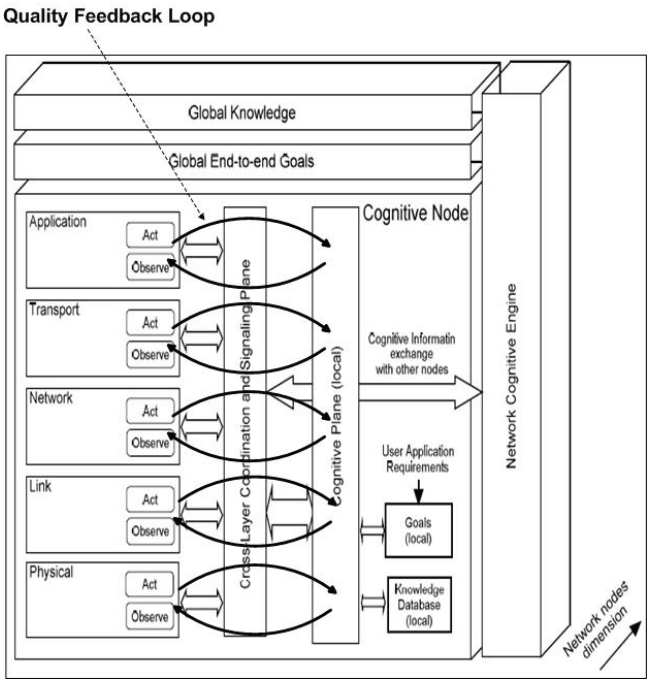

Fig. 13. Cognitive network framework

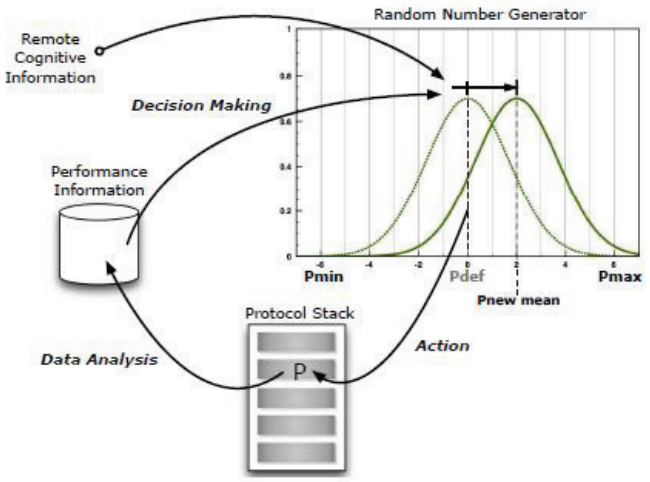

Fig. 14. The quality feedback loop

munication with emphasis on self-optimization and self-configuration properties and inherent cross-layer design.

As depicted in Figure 13 [76], a cognitive network is composed of a set of cognitive nodes exchanging cognitive information between each other. 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 standard protocol stack, each protocol layer is enhanced with a small software 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 information gathered, 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 (local goals of the node and applications demands, the knowledge obtained by the node, etc.) or parameters associated with a particular flow transmission.

The cognitive adaptation process is performed according to a quality feedback loop which consists of three phases: data analysis, decision-making, and action, as illustrated in Figure 14 [77]. 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. For example, the quality metric 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.

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 [77], each protocol parameter P is expressed in terms of its 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 Network Simulator (ns-2). The proposed approach was illustrated in the cognitive setting of TCP/IP

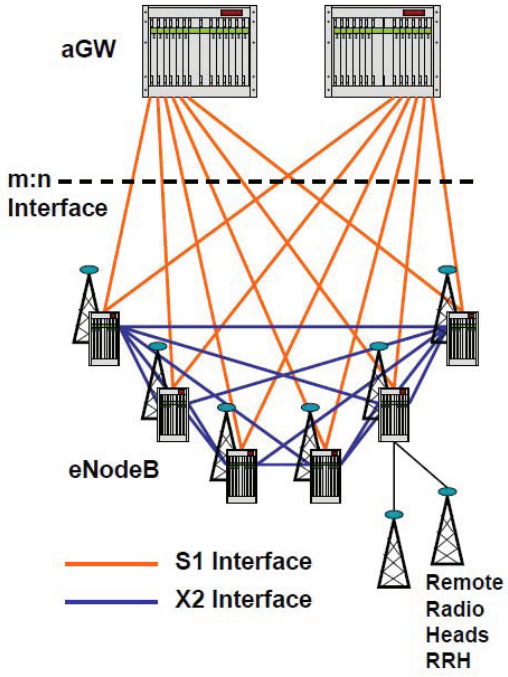

Fig. 15. 3G LTE Network Architecture [78]

reference model which brings cognitive reconfiguration into network management and protocol configuration [77]. 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.

C. SOCRATES project

SOCRATES (Self-Optimization and Self-ConfiguRaTion in wirelEss networkS) [79] is an FP7 European project dealing with the development of self-organization methods for LTE networks (Long Term Evolution of the 3^{rd} generation mobile networks). It concentrates on Self-Organizing Networks (SON). In particular, SOCRATES focuses on the development, evaluation and demonstration of methods and algorithms for the self-configuration, self-optimization and self-healing of LTE radio networks. Until the redaction of this survey, the work on this project is in progress.

Figure 15 shows the evolved 3G LTE architecture considered by the SOCRATES project. The evolved Radio Access Network (RAN) architecture (LTE) consists of a set of evolved Base Stations (eNodeBs) which are directly connected by the X2 interface. This interface is also used to exchange handover information between neighbor nodes. We note that SOCRATES is interesting on the LTE architecture including the X2 interface between eNodeBs and the S1 interface allowing connection towards the core network (System Architecture Evolution Gateway, aGW) [78].

In this project, three high-level solutions with different level of distribution have been described and discussed as possible functional architectures which integrate SOCRATES SON functionalities [78], [80], [81]. Figure 16 shows theses three solution approaches: distributed solution, centralized solution and hybrid solution.

As shown in Figure 16(a), the distributed solution is a flat architecture where all eNodeBs are equivalent. Decisions are made in a distributed manner and are based on information available in the eNodeB. Besides, self-optimization algorithms are running locally in each eNodeB. eNodeBs communicate with each other via the X2 interface.

Figure 16(b) illustrates the centralized solution where decisions are made in a central node. Moreover, self-organizing mechanisms are executed only in this node. eNodeBs are not able to take any independent decision or action. They only exchange monitoring information with the central node and execute its decisions.

The hybrid solution is depicted in Figure 16(c). This solution combines both distributed and centralized ones. Some of the self-organization algorithms are running locally in the eNodeB (related to tasks with local scope or on few neighbors cells). On the other hand, tasks with a wide scope, where many cells or the whole system will be affected, need to be managed from a central node.

Table III summarizes a comparison made by SOCRATES project between distributed and centralized solutions. Hybrid solution benefits from centralized and distributed mechanisms. However, an hybrid solution requires a proper division of the responsibilities and management tasks between the central node and the distributed SON entities.

Until the publication of the deliverable D5.10 [82], in order to lead to a common architecture for SOCRATES, no final decision regarding the selection of one of these architectures has been yet made. At this stage, authors have only presented and argued architectural preferences for each use case. Table IV summarizes SOCRATES use case architectural preferences [82].

Different use cases are detailed in [3] [4] including triggers, input sources, parameters and measurement list, architectural aspects, etc. In particular, a main interest has been focused on the definition of self-optimization metrics and algorithms including load balance algorithm in LTE mobile communication system, self-optimization of home eNodeB and especially radio and handover parameters optimization [80].

Figure 17 shows the functional architecture of the SON feedback loop defined in SOCRATES project. It refers to

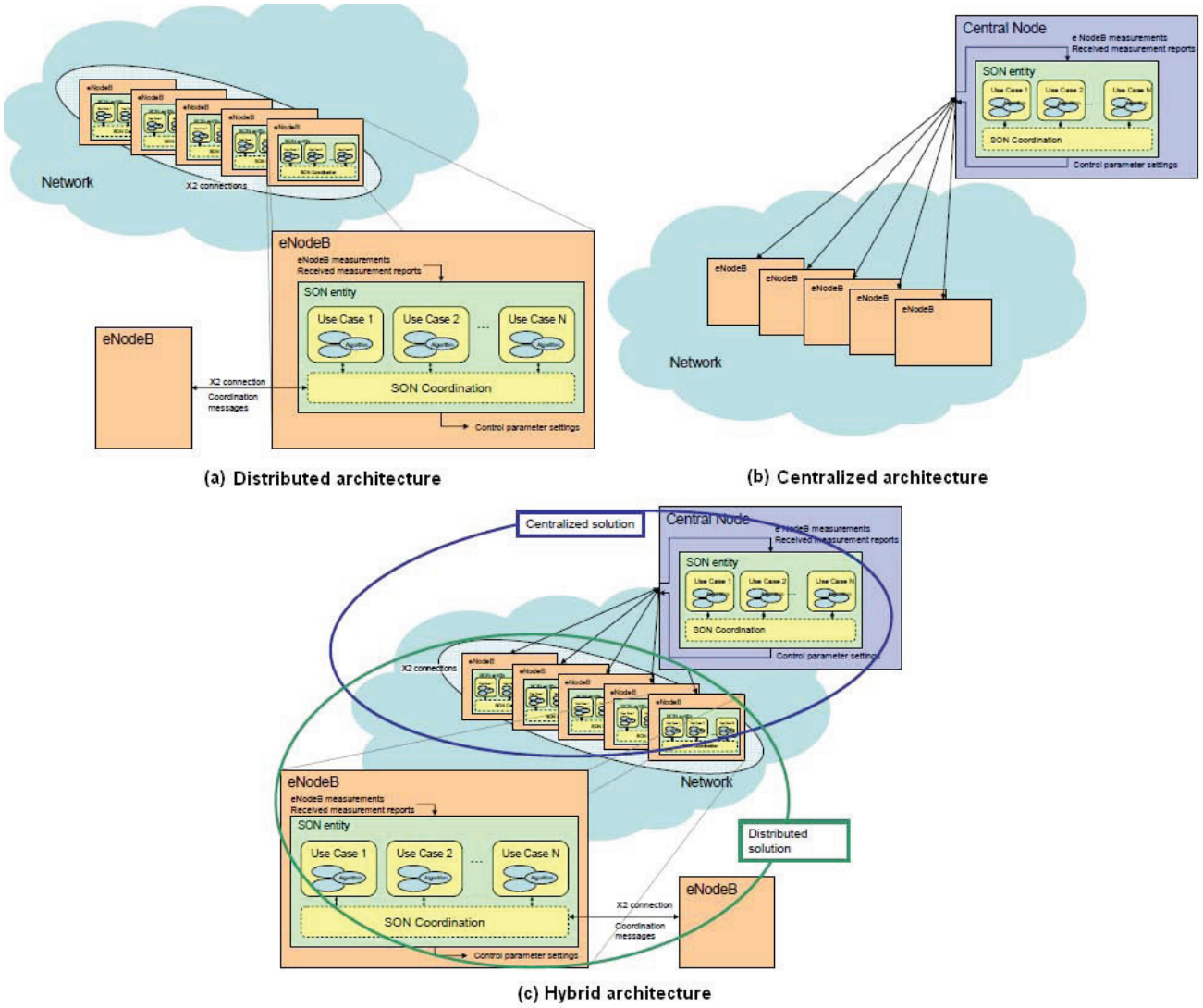

Fig. 16. Proposed SOCRATES SON architectures for LTE networks [78]

the well-known MAPE-K loop. The ”System” represents the managed resource: the functionality of the node to be configured, optimized or healed. The ”Sensor” acting as a monitor is responsible for acquiring, aggregating or processing information from the ”System”. Monitoring information may be measurements, status, configuration information or alarms. The ”SON Functionality” combines the analyzer and planner in the autonomic MAPE-K loop. It represents an algorithm which analyzes reported information from the ”Sensor”, plans the appropriate decision and triggers the ”Actuator” with necessary information to change the ”System” behavior if necessary. Finally, the ”Actuator” playing the role of executor translates ”SON functionality” information and decisions to parameters supported by the ”System” in order to change its behavior in the desired way. The functions of the feedback loop may be implemented locally into an eNodeB or in a distributed way to several eNodeBs or into entities with several management levels.

We note that SOCRATES project tackles an important topic ”self-organization of LTE mobile networks”. At this stage, the project proposes a set of good ideas, rules, principles and boundary conditions to achieve self-configuration, self-optimization and self-healing of radio LTE networks goals. A detailed description and specification of the project objective and its selected use cases is provided. However, we remark that unless handover optimization and load balancing goals [83], [84], [85], no explicit methods of each defined self-organization goal is given. Presented self-optimization methods are based on several statistics and measures. They are also based on predefined policies where actions are triggered by the exceeding of given estimated thresholds.

The work on this project is yet in progress. Many details related to the SOCRATES framework are not yet accomplished. For example, the choice of the level of distribution of the associated SON architecture is not yet made and so, explicit interactions between system elements and their cooperation in order to meet a given desired goal are not yet defined. This is very important in order to carefully judge the effectiveness of the proposed self-organization framework. Moreover, proposed evaluation metrics are closely related to the evaluation of the overall performances of the proposed methods related to the achievement of specific self-optimization goals such

TABLE III
COMPARISON BETWEEN DISTRIBUTED AND CENTRALIZED SON ARCHITECTURES

Criteria	Distributed Solution	Centralized Solution
Scalability	scalable for numerous eNodeBs	not scalable
Complexity of management tasks	very complex, especially tasks requiring coordination and information exchange between many eNodeBs.	management tasks are performed locally
Adjustment of imperfections	has a local scope	has an entire network scope
Consistency of decisions making	difficulty of detection of inconsistent decisions: eNodeBs may continously exchange conflicting messages and no proper action will be performed unless a conflict handling is implemented.	The detection of inconsistent decisions is performed locally in the central node. No conflicting messages will be exchanged in the network.
Use cases	suitable for localized SON solution where the optimization scope is only one cell or few cells and where no information exchange for coordination purpose is necessary	suitable where there is a need to manage and survey the interaction between different cells in the network

TABLE IV
SOCRATES USE CASE ARCHITECTURAL PREFERENCES

	Preferred Architecture		
Use Case	**Distributed**	**Centralized**	**Hybrid**
Admission Control	x		
Cell Outage Management	x		
Handover Optimization	x		
Home eNodeB	x		
Packet Scheduling	x		
Automatic Generation of default Parameters	x (provides a higher degree of autonomy)	x (simple implementation)	
Inference Coordination		x	
Load Balancing			x

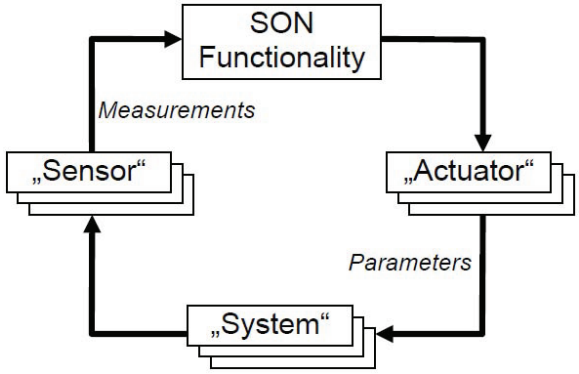

Fig. 17. SON feedback loop [78]

as handover or radio parameter optimization in LTE mobile networks. No metric or methodology of evaluation is provided for the evaluation of the overall autonomic architecture. Moreover, we note that the term of ”degree of autonomy” was also introduced as a criteria to be used, among others, in order to select the level of distribution of the final architecture. But no related definition was provided.

In this section, we provided a description of the majority of autonomic network architectures that we have found in the literature. For instance, we have described each proposal focus of interest, tiers, components structure, monitoring and self-adaptation approaches, just to mention a few. Besides the need for understanding how each architecture operates, its benefits and drawbacks, it is very important to discuss how to evaluate an autonomic architecture and how the different proposals compare, which is the focus of the following section.

IV. 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 [26]. 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 and quantitative 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.

A. Quantitative evaluation approach

As outlined before, the quantitative evaluation consists in using numeric measurements to compare ANM architectures. The quantitative evaluation is more useful for comparing ANM architectures which focus on same objectives. As an example, the performance of two architectures addressing the same self-management properties for a same network type under same topology and network configurations can be compared using quantitative evaluation. Otherwise, the comparison is meaningless.

Two kinds of quantitative comparison can be considered. On one hand, we can compare architectures performances 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 [27]. To best to our knowledge, the second alternative has not been addressed in the existing literature and hence, is out of the scope of this survey. However, this

evaluation view could be considered as an important future direction to progress the field of autonomic networking.

In this section, we identify main quantitative evaluation criteria and describe how they can be measured.

1) Learning Index: 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 fit 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 initial administrator (re)configuration plans.
- Facing with unprediction: Learning allows the network to cope with unpredicted situations. In this way, if any situation is omitted in 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 behavior. As a consequence, it becomes more difficult to verify behavior 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 behavior of the manager node. Malicious attackers will need to continuously look after manager nodes to find how they would operate thereafter and how their updated operations can be compromised.

Similar to [4], we define two extremes of the learning capability as follows: closed-adaptive and open-adaptive. A system is called closed-adaptive when it is not enabled with learning ability. Hence, a set of predefined adaptation strategies is applied without any enhancement in adaptation schemes in runtime. Open-adaptive architectures learn from past experiences to continuously evolve and enhance their applied adaptation strategies or, even, to cope with a condition unpredicted by administrators. This could be achieved by applying some artificial intelligence techniques such as reinforcement learning.

A prerequisite for the well-performance of learning algorithms is that they should be fed by correct knowledge. This aspect is denoted as the accuracy of awareness and is examined separately. In the scope of this subsection, we focus on the ability to learn of the architecture, supposing that the accuracy of awareness is ensured.

We introduce an efficient method for evaluating different architectures from the learning capability point of view. It consists of computing the degree of open-adaptability of the architecture. We defined a unified index called the *learning index* composed from contributive factors characterizing the learning capability of the architecture. These factors are:

- The ratio of the number of learnable management objects (policies, weights, parameters), $|L|$, versus the total number of network management parameters, $|M|$.
- The coefficient of learning ability c_ℓ, which is the relative number of well-performing learning-based decisions. This factor is inspired from [86] and can be computed based on the analysis of the evolution of learnable parameters:

$$c_\ell = \frac{\max\{1 - \frac{E\{D^{'}\}}{E\{ND^{'}\}}, 0\} + \min\{\frac{E\{D\}}{E\{PD\}}, 1\}}{2} \quad (1)$$

  where $E\{D\}$ is the average number of total correct learning-based decisions. $E\{D^{'}\}$ is the average number of total incorrect learning-based decisions. $E\{PD\}$ is the total number of correct decisions of "Target Achievers" (learning objects with positive evolution towards the goal). $E\{ND^{'}\}$ is the total number of incorrect decisions of "Target Damagers" (learning objects with negative evolution against the goal).
- The efficiency of learning e_ℓ, which is the overall degree of well-performance of the learning. This factor can be calculated based on the amount of progress towards the target:

$$e_\ell = \frac{K\{PD\}}{K\{PD\} + K\{ND^{'}\}} \quad (2)$$

  where $K\{PD\}$ is the average of total percent of learning object's enhancement of "Target Achievers". $K\{ND^{'}\}$ is the average of total percent of learning object's declination of "Target Damagers".

The learning index is denoted as $Index_\ell$ and can be computed by correlating these three contributing factors as follows:

$$Index_\ell = \frac{|L|}{|M|}.c_\ell.e_\ell \quad (3)$$

The value of learning index can be obtained by replacing the parameters of the equation (3) with their corresponding expressions presented in equations (1) and (2). 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 $|L|$ is zero. A complete open-adaptive system gives the value 1.0 for the learning index.

2) Accuracy of awareness: The quality of the network view has to be assured since learning and reasoning algorithms rely on this knowledge for decision making. The accuracy of awareness should be considered as a prerequisite for the autonomicity of the architecture rather than a metric for comparing ANM solutions. However, we describe it as an evaluation criteria due to its impact on the well-performance

TABLE V
DELAY GUIDELINE FOR VOIP

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

TABLE VI
DELAY GUIDELINE FOR VOIP

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

TABLE VII
EFFECT OF PACKET LOSS ON VOIP QUALITY

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

of adaptation mechanisms. We can observe that this criteria is included somehow indirectly in QoS performance index described hereafter. Obviously, each solution may need more or less degree of awareness according to approaches used for adaptation.

Intuitively, more frequently the view is updated, more the element disposes of the correct awareness of the network state. 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 type of knowledge and the dynamicity of the underlying network context. The collected data have to be aggregated somehow, providing a correct perception of the network status (called the node's network view). Hence, the network view can be constructed considering the following factors:

- *Temporal factor*: It reflects the up-to-dateness of the data sample. A temporal weighting metric can be used in the node network view to give more importance to up-to-date information while avoiding the sharp impact of transitory network changes. The insertion time into the network view can be used as a temporal weighting metric in order to give more weight to the most recently entered sample.
- *Spatial factor*: The relevance and accuracy of information reduce with the spatial distance between information source and destination node. The topological distance in hops can be used as the spatial weighting metric.
- *Trust factor*: It designates the confidence given to an incoming sample. We suggest to use trust weighting metric to give more weight to sample received from well reputed nodes. This assumes the existence of a reputation system in the network, where values between 0.0 and 1.1 are generated to indicate the node behavior.

When a distributed monitoring approach is used, the accuracy of awareness of network nodes can be expressed using the two metrics proposed in [12]. These metrics are:

- *Standard deviation of global views*: The standard deviation of global views can be used to evaluate the quality of network-wide awareness, showing the degree of uniformity of the global view in the network. Obviously, a lower standard deviation indicates a better quality of the network views. Ideally, the standard deviation should converge to zero.
- *Correctness*: The correctness consists in the fraction of nodes able to evaluate their relative state correctly. This metric is calculated based on the average of all local views within the network, giving the exact value for a perfect global view. The correctness metric is then 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 network-wide average and the node's global view. Obviously, a higher value of the correctness factor signifies a more convergent view among network nodes.

3) Quality of Services (QoS): Autonomic networking can be seen as a way to increase the overall performances of the network in terms of quality of service. 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. It is important to note that the desired QoS metrics may be different regarding to the objectives and applications of the target 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, in the case of rescue operation, the delay, loss rate and jitter are more important to the application.

For the case of a MANET established for a rescue operation, the architecture seeks to optimize the average end-to-end (E2E) delay, jitter and loss rate, enabling the VoIP application. To provide a guideline, the QoS requirements for VoIP application taken from [87] are illustrated in Tables V, VI and VII. The average overall E2E delay can be estimated by the average E2E delay of all VoIP flows on the network. Based on the evaluation performed by IUT [87], the delay below 150 ms is considered as good quality, between 150 ms and 250 ms is considered as acceptable, and the delay longer than 250 ms is considered as too long to be comprehensible. The required quality for jitter and packet loss are outlined in corresponding tables. Figure 18 illustrates how the correlation of the QoS metrics may influence the quality of the perceived voice.

4) Cost of autonomicity and services: In autonomic networking, the network spends extra resources to support the MAPE-K functions of the autonomic control loops. 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. Therefore, we distinguish two types of cost:

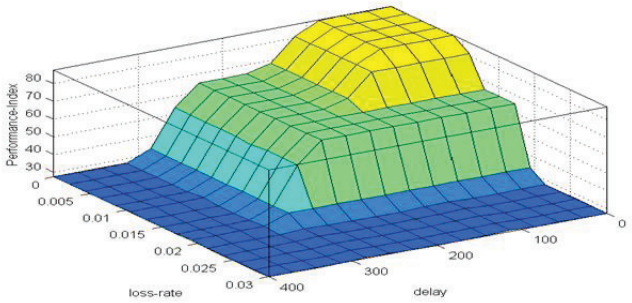

Fig. 18. Correlating QoS metrics and Performance Index of the VoIP

- *Cost of autonomicity:* The cost of autonomicity is defined as the overhead of extra autonomic activity which includes internal cost (e.g. CPU usage, storage, etc.) as well as external cost (e.g. overhead of knowledge monitoring, distributed intelligent adaptation, etc.) [26] [27].
- *Cost of services:* The cost of services is defined as the amount of control traffic generated to support prerequisite requirements of network services. As well, the cost of services can be divided into internal and external cost. For the case of a routing protocol service, the internal cost is the CPU usage and 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 consumed 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.

The overall internal cost of autonomicity and services, $Cost_{int}$, 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 useful network traffic, $E\{U\}$. Intuitively, the cost index, denoted as $Index_{Cost}$, is the average of internal and external cost:

$$Index_{Cost} = \frac{Cost_{int} + min\{\frac{E\{Overhead\}}{E\{U\}}, 1\}}{2} \qquad (4)$$

5) Granularity of intelligence: 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 scalable and provides better flexibility and fault-tolerance. However, the distribution may cause extra overall 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 minimize the cost of autonomicity.

Intuitively, the granularity of an architecture is the ratio of the number of autonomic managers to the total number of network 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.

6) 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 network depends on cooperation among individual autonomic managers. 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. Consequently, 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 suboptimal.
- Dependance on local and global knowledge: 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 managers perception. In addition, it can penetrate to the management overlay and report forged knowledge to other managers and distort their perception. Consequently, 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 (Denial of Service).
  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. Consequently, 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 and increases costs. Therefore, this vulnerability is abused to menace the service availability (Denial of Service).

We can conclude that cryptography and trust are two primordial security features of any autonomic network architecture.

Trust can be discussed from two point of views: among autonomic elements and between an element and its user for performing some management tasks [21]. One approach to enable trust between autonomic elements is to apply existing trust models. Authors in [88] 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 reputation.

With respect to cryptography, longer key lengths make cryptographic mechanisms more resistant to attacks. Thus, the security level is directly proportional to the key length.

The trust of a network is the lowest average reputation value of autonomic managers. The reputation system in the network generates values between 0.0 and 1.0 to indicate the behavior of each manager. The managers reputation may vary during the network runtime. Therefore, the average reputation of each manager should be taken.

B. 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.

1) Category of architecture: This metric specifies the general structuring of the architecture. In section III, 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 and 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) 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.

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.

4) Adaptation approach: In general, adaptation of an entity refers to changes that make such entity more fit to its environment [89]. 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 [4]. 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 [90], utility function based [49], chemical network based [91], just to mention a few.

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 [66], selective broadcasting [92] and situated view [93] approaches belongs to this category.
- Monitoring approaches which disseminate information to a subset of network elements. This category include distributed context repository [94] and tree-based aggregations [65], just to mention a few.

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 process. 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.

7) Learning ability: The qualitative learning ability evaluation view consists in determining if the architecture makes use of a learning mechanism in its decision process. This feature is primary to converge to the optimal configuration as discussed in section IV-A1. While the qualitative ability to learn criterion highlights the existence of a learning mechanism, the quantitative counterpart defines the strength of the applied solution.

8) Security mechanism: We have already discussed in section IV-A6 the relevance of security mechanisms to ensure the smooth operation of autonomic networks. The qualitative security criterion determines if any security defense line is defined and if so, which mechanism is used.

9) Heterogeneity management: An autonomic network can 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.

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.

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.

12) Validation: This criterion indicates if the pretended features of a given architecture is validated by appropriate evaluation alternatives (simulation, 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.

C. Comparison of surveyed autonomic architectures

In section III, 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.

It is important to note that a quantitative comparison of architectures using our methodology is not presented in this paper. This is related to the fact that any existing ANM architecture was initially evaluated for a particular self-management property and under a specific context of application. Consequently, any quantitative comparison between the different autonomic architectures using the existing experiment results will be unfair. Moreover, each architecture was developed originally by different groups and their source code is 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 special aspects of architectures (i.e. detail of the communication protocol used for autonomic management tasks, etc.). For these reasons, the measurement of architectures performances were not possible.

In the scope of this paper, we use only the proposed set of evaluation criteria to compare qualitatively between architectures. Table VIII summarizes the characteristics of existing autonomic architectures and evaluates them qualitatively regarding some management characteristics and evaluation criteria. We note that the SOCRATES architecture is not considered in this table since the architectural choices are not still finalized.

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-

TABLE VIII
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

adaptation mechanism. The reason can be the straightforward structure of this mechanism where adaptation plan is 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 [4].

- The Cognitive network succeeded to enhance the adaptation process using 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 conflicts between policies: an event might satisfy the conditions of two different ECA rules, while one rule may dictate an action that conflicts with the other satisfying rule [95]. 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 architectures 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 defined performance metrics. Moreover, comparison efforts were restricted to classic centralized network management approaches which do not support any of autonomic system properties. For example, in [42], 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.

V. Conclusion

While autonomic networking has attracted considerable attention from research community, the way towards a complete architecture that provides optimally all self-property functionalities is long. This work provides a holistic view of research in the area of autonomic network management (ANM) architectures, aiming at identifying the pros and cons of each existing architecture to go forward the full autonomicity. We classified ANM 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 another autonomic trend, called SOCRATES, for which the choice of the organizational structure of autonomic managers has not been yet finalized. Consequently, the architecture was neither categorized as flat nor hierarchical at its actual stage.

Moreover, we identified the main metrics that can be used to evaluate and compare the efficiency of autonomic architectures. We highlighted that the identified criteria are the most relevant architecturally for autonomic networking. They could be refined and their measurement can be adjusted by other metrics. We arose the fact that the comparison of ANM architectures make only sense when they are designed and applied for a same set of management functionalities under same environments. 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.

Acknowledgment

This work is carried out in the framework of Autonomic Internet (AutoI) project, which is funded by the European Union under the Framework Program 7 (FP7).

References

[1] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003.
[2] I. W. Paper, "An architectural blueprint for autonomic computing," IBM, Tech. Rep., June.
[3] P. Horn, "Autonomic computing: IBM's Perspective on the State of Information Technology," 2001.
[4] N. Samaan and A. Karmouch, "Towards autonomic network management: an analysis of current and future research directions," *IEEE Commun. Surveys Tuts.*, vol. 11, no. 3, pp. 22–36, Quarter 2009.
[5] M. C. Huebscher and J. A. McCann, "A survey of autonomic computing - degrees, models, and applications," *ACM Comput. Surv.*, vol. 40, no. 3, 2008.
[6] S. Schmid, M. Sifalakis, and D. Hutchison, "Towards autonomic networks," in *Autonomic Networking*, 2006, pp. 1–11.
[7] 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, pp. 165–176, 2003.
[8] M. A. Razzaque, S. Dobson, and P. Nixon, "Cross-layer architectures for autonomic communications," *J. Netw. Syst. Manage.*, vol. 15, no. 1, pp. 13–27, 2007.
[9] M. A. Razzaque, S. A. Dobson, and P. Nixon, "A cross-layer architecture for autonomic communications," in *Autonomic Networking*, 2006, pp. 25–35.
[10] 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: Proc. 2009 International Conference on Wireless Communications and Mobile Computing*. New York, NY, USA: ACM, 2009, pp. 170–175.

[11] 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.

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

[13] M. Parashar and S. Hariri, "Autonomic computing: An overview," in *Unconventional Programming Paradigms*. Springer Verlag, 2005, pp. 247–259.

[14] M. Salehie and L. Tahvildari, "Autonomic computing: emerging trends and open problems," in *DEAS '05: Proc. 2005 workshop on Design and evolution of autonomic application software*. New York, NY, USA: ACM, 2005, pp. 1–7.

[15] R. Sterritt, M. Parashar, H. Tianfield, and R. Unland, "A concise introduction to autonomic computing," *Adv. Eng. Inform.*, vol. 19, no. 3, pp. 181–187, 2005.

[16] 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, pp. 223–259, 2006.

[17] M. R. Nami and K. Bertels, "A survey of autonomic computing systems," in *ICAS '07: Proc. Third International Conference on Autonomic and Autonomous Systems*. Washington, DC, USA: IEEE Computer Society, 2007, p. 26.

[18] J. O. Kephart, "Research challenges of autonomic computing," in *ICSE '05: Proc. 27th international conference on Software engineering*. New York, NY, USA: ACM, 2005, pp. 15–22.

[19] N. Agoulmine, S. Balasubramaniam, D. Botvich, J. Strassner, E. Lehtihet, and W. Donnelly, "Challenges for autonomic network management," *1st IEEE International Workshop on Modelling Autonomic Communications Environments (MACE)*, 2006.

[20] H. Schemeck, "Autonomic computing - vision and challenge for system design," *Proc. International Conference on Parallel Computing in Electrical. Engineering (PARELEC'04)*, pp. 3–3, September 2004.

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

[22] A. Khalid, M. A. Haye, M. J. Khan, and S. Shamail, "Survey of frameworks, architectures and techniques in autonomic computing," in *ICAS '09: Proc. 2009 Fifth International Conference on Autonomic and Autonomous Systems*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 220–225.

[23] M. J. Khan, M. M. Awais, and S. Shamail, "Achieving self-configuration capability in autonomic systems using case-based reasoning with a new similarity measure." in *ICIC (3)*, ser. Communications in Computer and Information Science, D.-S. Huang, L. Heutte, and M. Loog, Eds., vol. 2. Springer, 2007, pp. 97–106.

[24] B. Claudel, N. De palma, R. Lachaize, and D. Hagimont, "Self-protection for distributed component-based applications," in *8th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS 2006)*, 2006.

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

[26] 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.

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

[28] "Autonomic Internet (AutoI) FP7 project." [Online]. Available: http://ist-autoi.eu/

[29] 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. 2nd IEEE International Workshop on Modelling Autonomic Communications (MACE)*, 2007.

[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: IOS Press, May 2009, pp. 112–122, 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 *Proc. 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: Proc. 3rd IEEE international workshop on Modelling Autonomic Communications Environments*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 102–107.

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

[36] "Open source software of AutoI FP7 project." [Online]. Available: http://ist-autoi.eu/autoi/index.php

[37] 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.

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

[39] 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*, 2006, pp. 1 –7.

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

[41] 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 *Proc. MILCOM 2007, IEEE Military Communications Conference*. MILCOM 2007, October 2007.

[42] 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*, 2006, pp. 1 –8.

[43] 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*. POLICY 2006, June 2006, pp. 23–34.

[44] 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.

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

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

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

[48] 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: Proc. Third International Joint Conference on Autonomous Agents and Multiagent Systems*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 464–471.

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

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

[51] M. Ayari, F. Kamoun, and G. Pujolle, "Towards autonomous mobile ad hoc networks: A distributed policy-based management approach,"

in *Proc. The Fourth International Conference on Wireless and Mobile Communications (ICWMC)*. Athens - Greece, July 2008, pp. 201–206.

[52] 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*. Spain, July 2008, pp. 73–84.

[53] "Autonomic Network Architecture (ANA) FP7 project." [Online]. Available: www.ana-project.org/

[54] 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.

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

[56] 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," ANA project, Tech. Rep., February 2009.

[57] G. Bouabene, C. Jelger, and S. Schmid, "ANA blueprint, Deliverable d1.4," ANA project, Tech. Rep., February 2009.

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

[59] 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," ANA project, Tech. Rep., January 2009.

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

[61] "Open source software of AutoI FP7 project." [Online]. Available: http://www.ana-project.org/web/software/start

[62] "4WARD FP7 project." [Online]. Available: http://www.4ward-project.eu/

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

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

[65] 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. [Online]. Available: http://portal.acm.org/citation.cfm?id=1371259.1371336

[66] ——, "Robust monitoring of network-wide aggregates through gossiping," in *In Proc. Tenth IFIP/IEEE International Symposium on Integrated Management (IM 2007*, 2007, pp. 21–25.

[67] R. Cohen and A. Landau, ""not all at once!" - a generic scheme for estimating the number of affected nodes while avoiding feedback implosion," in *INFOCOM*, 2009, pp. 2641–2645.

[68] L. Guardalben, S. Sargento, P. Salvador, and V. Mirones, "A cooperative hide and seek discovery over in network management," in *Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/IFIP*, april 2010, pp. 217 –224.

[69] G. Haßlinger and T. Kunz, "Challenges for routing and search in dynamic and self-organizing networks," in *ADHOC-NOW*, 2009, pp. 42–54.

[70] G. Hasslinger and S. Kempken, "Applying random walks in structured and selforganizing networks," in *PIK'2008, Special Issue on Self-organizing networks*, vol. 31, 2008, pp. 17 –23.

[71] R. Steinert and D. Gillblad, "An initial approach to distributed adaptive fault-handling in networked systems," SODA, Tech. Rep.

[72] ——, "Distributed detection of latency shifts in networks," SODA, Tech. Rep.

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

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

[75] 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," *IEEE Commun. Mag.*, vol. 44, no. 12, pp. 51 –57, 2006.

[76] 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.

[77] 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.

[78] L. Schmelz et al., "Deliverable D2.4: Framework for the development of self-organisation methods," 2008, project deliverable 4WARD.

[79] "SOCRATES project." [Online]. Available: http://www.fp7-socrates.org/

[80] N. Scully et al., "Deliverable D2.5: Review of use cases and framework," 2009, project deliverable 4WARD.

[81] ——, "Deliverable D2.6: Review of use cases and framework ii," 2009, project deliverable 4WARD.

[82] ——, "Deliverable D5.10: Measurements, architecture and interfaces for self-organising networks," 2010, project deliverable 4WARD.

[83] A. Lobinger, S. Stefanski, T. Jansen, and I. Balan, "Load balancing in downlink lte self-optimizing networks," in *VTC Spring*, 2010, pp. 1–5.

[84] T. Jansen, I.-M. Balan, I. Moerman, and T. Kürner, "Handover parameter optimization in LTE self-organizing networks," in *Joint Workshop COST 2100 SWG 3.1 and FP7-ICT-SOCRATES, Papers*. European Cooperation in the Field of Scientific an Technical Research (Euro-COST), 2010.

[85] D. Dimitrova, H. B. van den, R. Litjens, and G. Heijenk, "Scheduling strategies for lte uplink with flow behaviour analysis," in *Fourth ERCIM Workshop on eMobility*, marc Brogle, E. Osipov, and G. Heijenk, Eds. Lulea, Sweden: Lulea Tekniska Universitet, May 2010, pp. 15–26. [Online]. Available: http://doc.utwente.nl/72116/

[86] G. N. Yannakakis, J. Levine, J. Hallam, and M. Papageorgiou, "Performance, robustness and effort cost comparison of machine learning mechanisms in flatland," in *in Proc. 11th Mediterranean Conference on Control and Automation MED03. IEEE*, 2003.

[87] 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.

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

[89] 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 Intell. Syst.*, vol. 14, no. 3, pp. 54–62, 1999.

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

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

[92] 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.

[93] 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, pp. 171–193, 2008.

[94] 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 Seventh IEEE International Workshop on Policies for Distributed Systems and Networks*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 23–34.

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

Zeinab Movahedi is a Ph.D. candidate at University of Pierre and Marie Curie (Paris 6), Laboratoire dInformatique de Paris 6 (LIP6), France. She received her M.Sc in Computer Science from University of Pierre and Marie curie, France, 2007. She has worked at autonomic networking area for many years; her interest domain is autonomic computing, intelligent wireless networks and security.

Mouna Ayari received her PhD in Computer Science conjointly from ENSI, University of Manouba, Tunisia and University of Paris 6, France in 2009, her M.Sc in network and computer science and her B.S. degree in Computer Science and network engineering from ENSI,Tunisia, in 2004 and 2003 respectively. Her research interests include autonomic networking, policy-based networking, network virtualization and quality of service management in wireless networks. She is currently an assistant professor at ISIM, University of Monastir, Tunisia and member of the PHARE research team of LIP6 Laboratory at the University of Paris 6, France, and RAMSIS research team of CRISTAL Laboratory at ENSI, Tunisia.

Rami Langar received the B.S. degree in telecommunications engineering from the Ecole Superieure des Communications, Tunis, Tunisia, in 2001; the M.S. degree in network and computer science from the University of Pierre and Marie Curie-Paris 6, Paris, France, in 2002; and the Ph.D. degree in network and computer science from the Ecole Nationale Superieure des Telecommunications de Paris, Paris, France, in 2006. In 2007 and 2008, he was with the School of Computer Science, University of Waterloo, Waterloo, ON, Canada, as a Post-Doctoral Research Fellow. He is currently an Associate Professor of computer science with the University of Pierre and Marie Curie-Paris 6. His research interests include post-Internet protocol architecture, mobility and resource management in Femtocell, wireless mesh and vehicular ad hoc networks, performance evaluation, as well as quality-of-service support. He has served as a technical program committee (TPC) Co-Chair of the IEEE ICC'12 Ad Hoc and Sensor Networks Symposium, a Posters Co-Chair of the IEEE GIIS'11, and a Tutorial Chair of the IEEE GIIS09. He has also served as the TPC member for many international conferences, including IEEE ICC, GLOBECOM, PIMRC and VTC. Dr. Langar is a member of IEEE and IEEE Communication Society.

Guy Pujolle is currently a professor at the Pierre and Marie Curie University (Paris 6) and a member of the Scientific Advisory Board of Orange/France Telecom Group. He was appointed by the Education Ministry to found the Department of Computer Science at the University of Versailles, where he spent the period 1994-2000 as Professor and Head. He was Head of the MASI Laboratory (University of Pierre and Marie Curie - Paris 6), 1981-1993, Professor at ENST (Ecole Nationale Superieure des Telecommunications), 1979-1981, and member of the scientific staff of INRIA (Institut National de la Recherche en Informatique et Automatique), 1974-1979.