Towards Autonomic Network Management: an Analysis of Current and Future Research Directions

Nancy Samaan and Ahmed Karmouch

***Abstract*—Autonomic network management is an innovative vision promising new horizons of efficient networking systems free from human control. This promise has, thus far, ushered in enormous yet dispersed research contributions in both industry and academia. The work presented in this article aims at putting these efforts into perspective deriving a more holistic view of the literature in this area. We analyze the requirements and the main contributions for the building blocks of any autonomic network management system (ANMS). We then describe a coherent classification methodology to compare existing ANMS architectures. Based on this analysis, we suggest a reference framework and highlight some open challenges and describe new research opportunities.**

***Index Terms*—Autonomic computing, network management, self-monitoring, self-adaptation.**

I. Introduction

OVER the past decades, network technologies have tremendously evolved with respect to their conception, coverage, capabilities and capacities. The traditional data-only networks are now replaced with sophisticated systems comprised of multi-vendor equipments, supporting multi-technologies and capable of providing a wide range of real-time media applications at extremely high speeds. Unfortunately, this evolution in networking technologies was accompanied by much slower advancements in network management solutions.

Early network management systems (NMSs) were limited to simple command line interfaces (CLI) provided to network administrators who were responsible for querying and manually configuring every network component. Next, came more advanced approaches that relied on different technologies such as software agents [1], active networks [2] and policy-based systems [3] aiming at partially automating the management tasks in order to achieve higher response times and lower management costs. Nonetheless, NMSs kept struggling to cope with newly emerging issues due to the ever-increasing size and complexity of the underlying network components and services.

Even with partial automation, a main limiting property of such NMSs is their embedded static nature in taking management decisions that lack the ability for situation and context adaptability. While these systems configurations may be optimal for particular settings, they may not be suitable for continuously increasing requirements and dynamics of users and applications particularly in wireless environments. The result is that the management of current and emerging network technologies is becoming the main bottleneck to any further advancements.

Autonomic computing paradigms, and, in particular, autonomic network management, represent an emerging solution for the aforementioned problem. Autonomic network management systems (ANMSs) hold the promise of anticipating, diagnosing and circumventing any impairment in the functionalities of the underlying network in an independent and autonomic manner, driven by a set of higher-level business-oriented goals, with minute human intervention or supervision. Unfortunately, although, in theory, autonomy seems to provide the ultimate solution for the complex network management problem, in general, research efforts towards ANMSs are still in their infancy and are still faced with many challenges before being realized as a successful solution.

One of the major challenges is the dispersion of research in this area and the lack of a clear formulation for the requirements of the main building blocks of ANMSs. This article seeks to clarify the autonomic network management concepts and to map the scientific efforts made thus far into a more holistic view. We show that ANMSs do not constitute an explicitly new alternative for communication management but is rather a concept which presents a unification of recent advancements and trends in various areas of research.

The remainder of this article proceeds as follows. In Section II, an overview of the network management problem and a discussion of approaches for traditional and automated network management are presented. Section III introduces the concepts of autonomic computing and autonomic network management. Sections IV through VII discuss requirements and contributions to the various functionalities within an ANMS. We then propose a coherent classification methodology to measure the degree of autonomicity in any ANMS and apply it to compare major research projects in the literature. A discussion of current challenges and future research trends is presented in Section IX. Finally, Section X presents some concluding remarks.

II. Network Management in a Nutshell

The management of any communication network is broadly concerned with the control and orchestration of the functionalities and behavior of various network components in order to reach a desired system state, given a set of pre-specified high-level business objectives and goals (e.g., maximizing resource utilization and minimizing service response time). In the following, a brief historical view of the development of management approaches is provided.

Manuscript received September 2007; revised March 2008.

The authors are with School of Information Technology & Engineering (SITE), University of Ottawa (e-mail: nsamaan@site.uottawa.ca).

Digital Object Identifier 10.1109/SURV.2009.090303.

1553-877X/09/$25.00 © 2009 IEEE

A. Traditional management approaches

From a historical perspective, management functionalities were defined as five main tasks, collectively referred to as FCAPS, which included: network fault-diagnosis, configuration, accounting, performance management, and security management. Earlier research efforts focused on developing different protocols such as the the simple network management (SNMP) protocol and data models (e.g., management information base (MIB) [4]), in order to automate the tasks of collecting monitoring information and reporting back to human administrators who were still responsible for manually interpreting and analyzing the collected data and then performing various service and component configuration using a set of interfaces. Web-based network management [5] and the common object request broker architecture (CORBA) [6] represent other milestones that contributed in further facilitating some of the management operations.

The main limitation of these approaches was that even simple operations, such as the introduction of a new network variable, required the overhead of taking the system down and manually introducing changes such as defining a new MIB entry in different tables and reconfiguring its relation to other variables and monitoring programs. Consequently, the performance of the managed networks using these paradigms was strongly constrained by the expertise of human operators.

B. Automated management approaches

The increased complexity and sizes of networking systems necessitated the development of efficient tools to alleviate the task of network management from human operators. The most notable of these tools are: software agents, active networks and policy languages.

The introduction of the agent technology [1], and in particular mobile agents, to NMSs brought along new operational capabilities and achieved asynchronous control and management decentralization. The use of agents offered the means to migrate the management code from the manager to the managed network elements. This, in turn, provided an on-demand and customized distribution of configuration and management programs.

The active networks technology is used to dynamically control network components through the injection of active programs at runtime [2]. A major limitation with this technology was that more programmability into routers necessitated the redesign of existing legacy systems. It also implied adding more complexity to the routers' management functionalities, and, consequently, introducing processing overheads and delays. It is worth noting here that both the mobile agents and active networks technologies are based on executing code on remote devices. Hence, they were not adopted in commercial solutions because of their security vulnerability.

In order to simplify the administration of large distributed systems, policies [3] have been introduced to formalize the description of the (un)authorized and (non)obligatory operations of various network elements in response to changes in the environment. Policies prescribe a set of rules, defined by network administrators, that guide the behavior of network components. A typical advantage of policy-based NMSs is their ability to reconfigure and adapt the behavior of the system by modifying the applied policies at runtime without stopping the system operation.

While technologies such as mobile agents, active networks and policy languages may automate the deployment of different management strategies for performing the necessary management operations, human operators using them were still faced with the burden of having to precisely describe and program their desired behavior, develop necessary policies (or code) and, even more, to continuously modify this behavior in response to changes in the environment. Another main limitation of these automated-management approaches was their reactive nature where human operators mostly rely on indications of service failure or performance degradation as signs to readjust the management and configuration strategies. One additional problem of these approaches was the inability to evolve and to adapt with changes in either supplied business objectives or users' requirements. As will be discussed in the next section, the introduction of autonomic computing paradigms provided a promising solution to the aforementioned problems.

III. Autonomic Network Management

A. An overview of Autonomic Computing

The term *autonomic computing* (AC), introduced by IBM in 2001 [7], refers to computing entities that are capable of performing operations on themselves in order to meet a set of predefined high-level objectives that are mandated by their creators or administrators. By adverting to the human autonomic nervous system which governs vital functionalities without conscious intervention, the vision of AC aims at seeking inspiration from the mechanisms by which biological systems are self-governed. An entity with autonomic computing capabilities is envisaged as one that continuously monitors its own state and adjusts its operations in response to internal or external stimuli. Ultimately, an autonomic entity would be capable of making independent choices to modify its operations in order to meet certain goals.

AC frequently refers to four characteristics distinguishing any AC system, namely, self -configuration, -healing, -optimization and -protection; these properties are collectively referred to as the self-CHOP properties. IBM's vision also defines a general framework in which the system is comprised of an autonomic manager that manages one or more managed elements. The manager is tasked with four main functionalities: *monitoring* the managed entities, *analyzing* their performance, *plan* and *execute* a set of appropriate management actions. These functionalities are collectively referred to as the MAPE loop. Central to this loop is a knowledge-base that maintains the necessary information about the managed entities and their management operations.

B. ANMS Definition

The concept of autonomy is a generic concept that can be applied to different fields (e.g., autonomic space exploration missions or autonomic software applications). When applied to network management the result is an ***autonomic network management system (ANMS)***. More precisely, an NMS is

considered to be autonomic if it performed management operations in a manner that satisfied the self-CHOP characteristics, i.e., self-configurability, self-healing, self-optimization and self-protection according to the following definitions.

An NMS is said to be ***Self-configurable*** if it can detect changes in any of the network components or the surrounding environment that might have or will result in a violation of the management objectives and trigger appropriate configuration or, more precisely, adaptation mechanisms, without disturbing the network performance. For example, detecting management overloads in certain network areas might necessitate the redistribution of management operations among several managers.

An NMS is said to be ***Self-healing*** if it can restore its operations when any type of failures occurs (e.g., a failing monitoring component or a malfunctioning software management entity).

An NMS is ***Self-optimizing*** if it is capable of performing any management task and executing any service in the most efficient manner, such that, given the available resources and environment settings, any change in the current management operations may not result in a better network performance.

Self-protection of a management system relates to its ability to take the necessary measures to protect its operations from any unplanned external influence or threat. For example, the configuration of admission control operations at edge routers can prevent unauthorized traffic from accessing the network.

In general, research efforts towards the realization of ANMSs can be categorized as those that address various functionalities of the ANMS and those that target the development of architectural solutions. In sections IV though VII, we analyze the main requirements and main research approaches to realize the individual ANMS building blocks with reference to the MAPE operations, namely building a network knowledge base, an autonomic monitoring module, an analysis module, a planning module and an execution modules.

IV. Building a Network Knowledge Base

A key factor in achieving autonomic management is the ability to describe an accurate model of the managed system [7], [8]. This process is achieved though building a knowledge base system (KBS) which can range in complexity from a simple database to more advanced expert systems with built-in reasoning mechanisms [9]. The first step in building any KBS involves the specification of the knowledge that must be supplied by the system. A suitable model is then selected for each type of the domain knowledge. Once a model is elicited, the third step aims at building methods for knowledge discovery, acquisition and processing.

While the importance of building knowledge-bases in ANMS is widely acknowledged [8], [10], to the best of the authors' knowledge, this issue has seldom been addressed in the literature and, rather, existing work addresses only isolated related issues. In the following, we provide an analysis of the requirements of a KBS for autonomic network management and summarize related research work in the literature.

A. Knowledge typology in ANMS

A KBS may represent two different forms of knowledge about the managed system [9], typically referred to as the domain knowledge and the control knowledge. The former provides a view or conceptualization of the managed domain while the latter knowledge type represents the ways to manage and control the modeled system, for example, as a set of domain related problems and their corresponding applied solutions. The domain knowledge can be further categorized into a structure knowledge and a behavior knowledge. The structure knowledge refers to information about the sorts of objects of the modeled domain, their properties, the relations among them, in addition to the different factual knowledge about the domain. Behavior knowledge describes patterns of behavior of different components or the overall modeled domain.

From an ANMS perspective, self-awareness is achieved by modeling the different components within the managed network in a network-knowledge-base system (NKBS). This includes defining a set of models describing the users of the network, applications running on top of the underlying infrastructure, the business goals of running the network, the environment encompassing the network, and the network itself. As stated earlier, each of these models may include structure- behavior- and control- knowledge or a combination of these types. For example, the structure knowledge about the network may include the current network topology and a description of the various components capacities and constraints. Behavior knowledge can, for instance, represent a queue operation in case of an overflow (e.g., random drop vs. tail drop) and control knowledge may specify the different actions that can be applied to the network components (e.g., increasing a buffer size or blocking an input port).

A sample of the different types of knowledge within an NKBS is presented in Table I.

B. Approaches to network and system modeling

Developing a model to efficiently reflect the networking environment is not a straightforward task. Current networking systems are highly distributed, complex and heterogeneous; hence, finding a single common representation requires also defining techniques for mapping this model to and from other ones. The developed model must also be robust in capturing the highly dynamic nature of the environment (e.g., traffic loads, active sessions). Current approaches for building an NKBS are mainly limited to representing the network structure knowledge [4], [11] which are commonly referred to as management information models (MIMs) [12] with some efforts to build simple models for control knowledge [13].

The modeled network structure knowledge often includes status information for different network components such as routers, switches and interfaces. Such information typically describes the current network load, latency and different queues and buffers parameters. The management information base (MIB) [4] and the common information model (CIM) [11] developed by the IETF and DMTF standardization organizations, respectively, are the two widely accepted and used structure models. They are used to describe different network components and their performance metrics. For a detailed description and analysis of these models the reader is referred to [12]. The authors in [12] propose a universal

TABLE I
A SAMPLE OF DIFFERENT KNOWLEDGE CLASSES IN ANMS.

Knowledge type	Structure	Behavior	Control
Network	Topology, routers constraints, CPU and memory sizes	Queues operations (e.g., FIFO, tail drop), expected packets treatments by routers	Management actions (e.g., increasing bandwidth assignment, changing packet treatment in a router)
System	servers status/ current configuration	Behavior of services (e.g., web service functionality)	Adding/removing services, adjusting service configuration
Application	Memory requirements, bandwidth usage	Adaptation strategies in response to bandwidth variation	Selection of appropriate adaptation strategies for different bandwidth variation causes
User	Personal and billing information	Mobility and service access patterns	Authority and privacy control

information model that provides a higher abstraction view that can incorporate both the CIM and MIB models. The only standardized model for network control-knowledge is the IETF policy control information model (PCIM) [13]. PCIM provides a formalism for defining and storing policies to control the configurations of individual network components. Nonetheless, PCIM is mainly an information model with no means to support analysis, refinement or reasoning with the defined policies as a first step towards elements autonomy.

Object-oriented models that aim at capturing both the structural and behavior knowledge of networks have been adopted in different projects. Examples include the NESTOR project [14] and the directory enabled networks (DEN) initiative [15]. The model in NESTOR also includes a constraint definition language for asserting relationships and constraints among different network objects. The Directory Enabled Networks-next generation (DEN-ng) [16] is another object-oriented model that describes behavior knowledge using finite state machines (FSM) and patterns. The model takes one step ahead of other approaches that are limited to describing the current state of a managed element where different FSMs of individual elements are populated and combined to produce a FSM that describes the behavior of an overall management solution. However, this approach still relies on the ability to fully formalize the set of all possible states for the modeled elements.

Recently, ontologies have been introduced to represent the semantics of the managed networks [17], [18]. Ontologies are one of the main approaches used in the scope of KBS and artificial intelligence to solve questions related to the semantics of the modeled domain. They describe an abstract model of a domain by defining a set of concepts, their taxonomy, their interrelation and the rules that govern these concepts in a way that can be interpreted by machines. The main difference between ontologies and other information models common to network management, particularly the CIM and MIB models, is that the latter models do not include axioms, semantic relations or constraints on them which facilitate reasoning with them. A network ontology defined using OWL and a reasoning engine based on first-order logic (FOL) calculus is described in [18]. The presented work also investigates the utilization of semantic concepts such as equivalency and containment in order to achieve interpretability between models and commands defined by different vendors.

The aforementioned models are explicit where the utilization of the model is independent from its representation. Other implicit modeling approaches have also been successful in representing a subset of the network model. For example, reinforcement learning (RL), a machine learning approach, has been applied to build look-up tables containing state-to-state transition and corresponding actions pairs in [19]. In [20], concepts of forecasting functions were utilized to gradually construct a dynamic model of the effects of various configuration actions on network components parameters. While implicit learning models are highly suitable for continuously evolving systems, where they replace the periodic manual redesign of the NKBS, one limitation of gradual learning models is their poor performance during the initial phase of learning the model.

Recently, some efforts (e.g., [21]) have been dedicated to model system control using utility functions. These functions are used, for example, to model service-levels as functions of system control parameters, allocated resources and expected service demands. Transfer functions, used within a control loop, are also utilized [22] to represent a server behavior model in response to varying configuration parameters and to model environment noise (e.g., delays and effects of the accuracy of network monitoring measurements).

Figure 1 depicts some of the network modeling approaches classified according to their types.

C. Approaches to user and application modeling

Other than models for networks, knowledge about users and running applications can be modeled within the NKBS. For example, a user's structure model in an NKBS may maintain personal and billing information while a behavior model may contain mobile users' movement patterns and service access frequencies. One way to model user-related information is through the content capabilities/profile preferences (CC/PP) framework [23] or through the use of ontologies (e.g., [24]).

An application structure model describes a set of the expected input and output variables such as the bandwidth utilization rate, acceptable loss range in data rates and tolerable data loss patterns (e.g., distributed or bursty), whereas a behavior model for the application can contain feasible adaptation strategies in the face of variations of bandwidth availability. While there have been extensive research efforts in developing application models (e.g., [25]), they have been seldom used as part of automating network management tasks.

Usage of user and application models to facilitate mobility related network management tasks has been presented in [26].

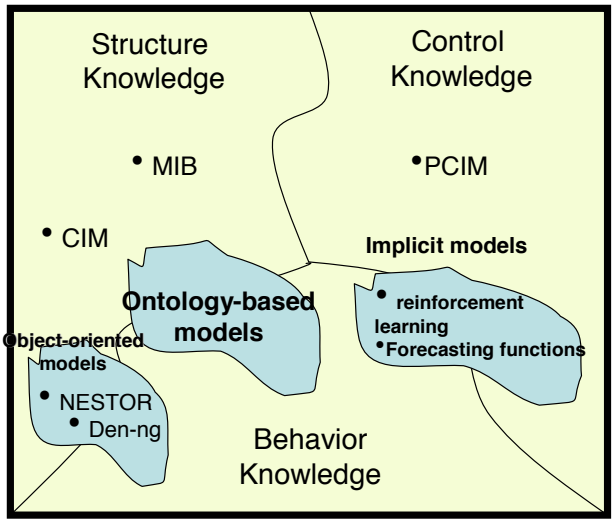

Fig. 1. Classification of network modeling approaches

In [27], Yan et al. developed a model for customers' behavior in telecommunication networks and utilized the model to reduce customer dissatisfaction and, in turn, the churn rate. The effects of the maximum number of users on the performance of a set of resources is modeled using auto-regressive functions in [22]. A biologically inspired approach for representing the structure and behavior models of network applications is described in [28] where a network application is implemented as a group of distributed, autonomic objects called cyber-entities (CEs) that are analogous in their functionalities to a bee colony consisting of multiple bees.

D. Other generic Approaches

General purpose models for autonomic systems can also be applied to networks. A general purpose knowledge engine is built in [29] which describes an autonomic entity as a set of behavioral rules and programmed behaviors. Rules are used, for example, to control the functioning of the entity and the interactions with other autonomic entities. The pre-programmed behaviors include algorithms to describe patterns of system actions in response to internal and external events.

An ontology-based reference model consisting of three layers representing resources, events and rules is described in [17]. The resource layer can be regarded as a domain knowledge layer which maintains information about the managed system components and resources. The event layer maintains the system behavior knowledge where a system learns that a certain change in a resource status is considered a significant event that must be reported to the control layer and can further apply correlation rules to further represent more complex forms of knowledge through the correlation of elementary events. Finally, action rules at the third layer defines the control knowledge where rules are used to control the behavior of the managed resources.

E. Data acquisition

Once a model has been elicited, the next step in building an NKBS is to identify various information sources and to collect the needed data. Two main approaches can be applied to realize this step, pre-embedding knowledge about the sources or discovering their existence. Discovery can be as simple as that of sending ping messages to IP addresses in a certain domain in order to build a map of the living network nodes from a pre-existing map of all connected nodes. In current implementations of NMSs no explicit knowledge base is built; rather, a manager entity queries the network devices, or agents, to periodically assess the status of the network components using a predefined communication protocol such as SNMP.

An interesting approach for communicating network related events through the utilization of content based networking (CBN) principles has been recently investigated by Keeney et al. [30]. The authors add semantics to the matching, filtering and distribution processes in a publish-subscribe system that routes network related events triggered by SNMP agents using a CBN. While the approach significantly enhances the performance of the management system, it is still limited to communicating the current status of network components using MIB/CIM models.

V. Autonomic Monitoring

Network monitoring is concerned with the collection of the necessary measurements to determine the underlying network health, the achieved service quality as well as the possibility of network faults or attacks. In general, traditional monitoring tools rely mainly on the systematic collection of measurements of predetermined parameters collected by network devices (e.g., using counters, logs or SNMP MIBs). In contrast, autonomic monitoring approaches strive to continuously adjust their operations in order to maintain an equilibrium between the needed accurate view of the network status and the processing overhead. In this sense, autonomy implies the ability to decide, at runtime, which network components to be monitored, where to install monitors, how to adjust monitoring parameters, when to execute monitoring functionalities and for how long. In the following, we provide a classification methodology for autonomic monitoring methods and apply it to analyze various approaches in the literature.

A. Classification of autonomic monitoring approaches

1) Active versus passive monitoring: In both traditional and autonomic approaches, monitoring can be achieved in either a passive or an active manner. Passive monitoring is limited to receiving measurement data from a set of sensors residing in the network. On the other hand, in active monitoring, network measurements are obtained by sending additional testing messages. Active monitoring introduces additional traffic overhead to the network while passive monitoring is limited to periods of time when there is traffic on the network between the hosts of interest. Hybrid schemes can combine both monitoring methods to achieve a more accurate view of the network while maintaining lower overheads.

2) Distributed versus centralized monitoring: Autonomic monitoring functionalities may be initiated and controlled by centralized autonomic mangers that collect and consolidate data obtained from different locations. Another approach to autonomic monitoring is to distribute this functionality over a set of individual autonomic nodes in the system which interact together directly to achieve the task of monitoring the entire network. These nodes can, for example, probe neighboring nodes to learn and track network behavior on their own.

3) Monitoring granularity: Monitoring granularity can be measured with respect to several aspects. The measurement unit granularity determines the level of detail in capturing the status of the managed network. For example, measurements can be performed at the byte, packet, flow or aggregated-traffic levels. Rate granularity defines the sampling rate which can range from few seconds to several hours. Finally, spatial granularity refers to the scope of the measured parameter which can reflect end-to-end properties (e.g., round tripe time) or a link property (e.g., link delay).

4) Monitoring timing: Most monitoring entities preform their functionalities on fixed time intervals. On the contrary, event-based and on demand monitoring [31] trigger monitoring functionalities only when necessary. Obviously, these approaches provide a better management of the consumed network resources on the account of the maintained up-to-date status.

5) Monitoring programmability: An autonomic monitoring approach is expected to dynamically modify its operation as needed, hence, requiring a degree of self-programmability. The simplest way to achieve this property is through the dynamic adjustment of a set of parameters. For example, it can dynamically adapt the rate granularity of a monitor component according to current network loads. Advanced approaches can dynamically modify entire monitoring functionalities such as switching between passive and active modes and installing new monitors.

B. Approaches to autonomic network monitoring

In this section, we limit our review to monitoring approaches that exhibit some degree of self-programmability. These approaches represent a first step towards the realization of a fully autonomic monitoring functionality.

An earlier approach to monitoring autonomy is introduced in Ganglia [32] where a node monitors its local resources and sends multicast packets containing monitoring data whenever significant updates occur. Using a set of programmable interfaces, applications are allowed to request from different nodes to perform different monitoring functionalities using application-specific metrics. Ganglia follows a passive distributed monitoring approach where monitoring programmability is left to the applications.

Other approaches rely on a set of predefined policies to achieve monitoring programmability. For example a set of policies are used to control the creation and configuration of monitoring channels that allow the dynamic configurations of various monitoring parameters and monitoring granularity is presented in QMON [33]. A more advanced approach is described by Han et al. [34] in which policies are used to control monitoring functionalities based on the network status and that of the running applications. For example, policies are activated at run-time to adjust the sampling rate of certain parameters according the network overload. In a similar manner, another policy selects an appropriate monitoring algorithm based on the type of the running application (e.g., with or without timeliness requirements). Although in this approach monitoring functionalities are changed at run time, the creation of the policies that control this change is static. A distributed peer-to-peer monitoring paradigm has been introduced in the DYSWIS architecture [35]. In DYWIS, each node follows a set of predefined rules to decide whether to trigger a query to its peers to further investigate an incurred problem according to its own observations. Here, programmability is pre-coded as a set of distributed monitoring rules embedded in the nodes.

Adaptive sampling and dynamic placement of monitors are also extensively addressed in the literature. For example, in [36], using a centralized monitoring control methodology, the authors dynamically calculate the appropriate sampling rate based on the severity of the network problems and the availability of resources. Similarly, Cantieni et al. [37] reformulate the monitors placement problem as an optimization problem and solve it using Lagrange multipliers. The A-Gap system [38] applies stochastic modeling to find the optimal state updates versus the state accuracy. A self-stabilizing spanning tree which maintains different monitoring filters is created using the solution to the problem to provide different views of the monitored system. A main limitation of modeling monitoring programmability as optimization problems is the implicit static representation of the problem constraints and objectives.

Sterrit et al. [39] present an event-based distributed monitoring mechanism for speeding up the propagation of network data through the use of heartbeats with dynamic rates. The approach is inspired by animal reflex reactions where heartbeats transmitted to managers by network components contain health indicator summaries that can assist in fault isolation and recovery. One limitation of this approach is the resulting excess of messages which are responsible for changing the monitoring granularity during the increase in the transmission rate of heartbeats. These messages may further contribute to decreasing the performance of the network. An interesting approach for centralized monitoring self-programmability is presented in NetQuest [40]. In this project, the problem of selecting appropriate monitoring functionalities is modeled as a design problem and Bayesian experimental design concepts are applied to determine various parameters of the solution such as the selection of a set of active measurements that maximize the amount of information obtained about the network.

Distributed peer-to-peer based approaches for network monitoring have also been investigated. For example, the AutoMon project [41] uses a distributed overlay of distributed network agent, acting as peers and hosted on various nodes, to perform distributed tests as well as distributed monitoring for fault and performance management. AutoMon applies both passive and active monitoring techniques and provides users with interfaces to define monitoring requirements.

VI. Autonomic Analysis

The second functionality in an ANMS is the analysis of the monitoring information. In traditional management systems, analysis is mainly limited to interpreting the collected data into a state description of the managed network and is performed alongside the monitoring functionality. This step is referred to as network diagnosis and is carried out with the help of pre-existing rules or heuristics. In ANMSs, autonomic diagnosis differs from traditional diagnosis systems in its ability to self-learn (e.g., newer states such as the detection of new types of anomalies or security attacks or means to enhance the analysis mechanism) without the reliance on pre-embedded rules. Furthermore, analysis also includes a second functionality, namely, network anticipation; this functionality relates to the utilization of the system knowledge to further anticipate the future state of the network. State anticipation is necessary to achieve the self-CHOP functionalities as it is needed to avoid either reaching non-optimal or failing states (i.e., failing to achieve self-optimization and self-healing).

A. Approaches to autonomic diagnosis

Network diagnosis, and in particular the problems of determining fault root-causes, network anomalies and security threats, has been the focus of extensive research efforts. In some of these efforts, self-learning has been achieved using various techniques such as the use of expert systems, probability models and Bayesian networks [42]. For a detailed comparison of these techniques the reader is referred to [42]

Recently, diagnosis of other aspects of the managed network has also been addressed in the literature. For example, an interesting approach is presented by Zhang et al. [43] to diagnose causes of violation of service-level objectives (e.g., service response time and request throughput). Pattern recognition and probability modeling techniques were used to identify which low-level system properties are related to such violations. Hariri et al. [44] investigated the development of abnormality metrics that can be used to analyze and diagnose the behavior of different network flows. The developed metrics use indications of flows characteristics such as packet rate and successful sessions rate to classify traffic into normal, probable normal, probable abnormal, and abnormal.

B. Classification of autonomic anticipation

In contrast to network diagnosis, the problem of network anticipation[1] represents a recent research area. Traditional approaches for management are reactive in nature and are mostly limited to applying a feedback loop in which they rely on current system diagnosis while making management decisions. A main limitation with approaches following this trend is their costly performance and, in some cases, their inability to avoid performance degradation and system failures. ANMSs are expected to be anticipatory [45], i.e., management decisions are made based on information pertaining to present as well as future internal state of the system (e.g., the queue length within a few minutes) and that of the external environment (e.g., the user's future location). Anticipation approaches can be classified according to their utilized knowledge, anticipation timing and the applied anticipation mechanism.

1) Anticipation knowledge: Anticipation approaches can be classified, based on the type of input knowledge, into state-based and history-based approaches. The former can be further classified into two categories: formula-based and state-transition based. In a state-to-state transition model, the knowledge of current system state is sufficient for prediction [16]. On the other hand, a mathematical model in formula-based approaches is used to derive the future state (e.g., TCP throughput [46]).

History-based prediction techniques rely on the availability of a performance trend or history to predict future states [47]. A major issue with history-based anticipation is an implicit assumption that state characteristics observed in the past are likely to prevail in the future.

2) Anticipation timing: Anticipation approaches can also be classified as either off-line or on-line. Off-line approaches are mostly suitable for less dynamic systems that are expected to exhibit periodicity over time. In such cases, using a simulation model or injecting synthesized traffic into the system off-line is sufficient to derive a model of expected future states. Online anticipation refers to continuously anticipating the system behavior using up-to-date knowledge about its current status. Although online anticipation approaches provide more accurate future states for dynamic systems (e.g., anticipating router workload), they usually compete for the running system resources.

3) Anticipation mechanisms: Anticipation approaches can use different mechanisms to achieve their tasks. Examples of such mechanisms are: probabilistic anticipation, statistical anticipation, anticipation through machine learning, and simulation-based anticipation. A comparison of the effectiveness of a number of traditional prediction techniques such as time-series algorithms, rule-based techniques and Bayesian network models in predicting network failures can be found in [48].

C. Approaches to network state anticipation

Anticipatory management solutions have been centered around a limited subset of network related properties, namely, patterns of network traffic [49], host loads [50], and network anomalies [51].

Several general purpose autonomic anticipation approaches have also been presented recently. These approaches provide a generic solution that can be customized to predict the behavior of any single component of the managed system. Examples of these approaches are the resource prediction system (RPS) toolkit [47], IBM's Clockwork [52] and the model-driven hierarchy [53].

The RPS toolkit provides a formula-based solution for online prediction of a network or resource related scalar valued variable that can be periodically sampled using time series and wavelet analysis. Clockwork [52] is also a general purpose online prediction method introduced by Russell et al. It applies feed-forward control loops to construct a predictive autonomic system using auto-regression. The method was applied to two

[1]The terms anticipatory, proactive, predictive, are commonly used to refer to the same concepts of knowledge about a future state

network specific prediction functionalities: forecasting cyclic variations in system loads and tracking the response time of the system in cases of different loads. The model-driven online anticipation approach, presented in [53], presents different system components as separate models. Each model represents a management functionality (e.g., configuration) as a set of possibly encountered problems and their candidate solutions. A machine learning mechanism is then used to reinforce or eliminate the stored solutions based on the system feedback. The utilization of probabilistic models in order to predict service-level agreements (SLA) violations, at run time, has been introduced in [54]. A Bayesian network corresponding to different network performance variables (e.g., CPU utilization) is constructed using past network measurements, and consequently used to predict any possible service violations in the future.

D. Approaches to user and application anticipation

Anticipating the performance of a running application and its resources demand is critical for an ANMS to make appropriate adaptation and configuration decisions that maximize the satisfaction of current management goals. This problem has been the focus of various research efforts. For example, the demand behavior of applications mining histories has been investigated in [55] while predicting the response time of transaction-oriented applications was analyzed in [56]. Predicting the user's behavior for accessing services was also proposed in [57]. In a previous work, the authors studied the significance of predicting users' mobility in various management tasks, and applied concepts of Dempster-Shafer reasoning to both the user and the environment models in order to predict the mobile user's traveled trajectory [26].

VII. Autonomic Planning and Execution

The third and fourth functionalities of an autonomic manager are the planning and execution of system solutions. Planning in network operations refers to the task of configuring network topologies, operations and services and is carried out off-line by network administrators. On the other hand, network configuration and performance management functionalities in the FCAPS model bear some resemblance to the planning and execution functions of autonomic managers. As described in Section II, in traditional NMSs, these functionalities are either performed manually or are partially automated (e.g., using policies or mobile agents). Recently, network adaptation has been used to reflect the continuous fine tuning of network components in order to meet a set of management goals, which reflects in a more accurate manner the objectives of the planning and execution functionalities.

In general, adaptation of an entity refers to changes that make such entity more fit to its environment [58]. ***System adaptation*** refers to the enforcement of one or more strategies, algorithms, rules or configurations in order to change one or more aspects of the system with the objective of achieving a set of higher-level goals. ***Autonomic adaptation*** refers to the ability of the system to perform adaptation operations using its internal knowledge to decide why, when, where and how adaptation is performed, without any external intervention in the decision making process.

In the following, we derive a classification methodology for autonomic adaptation guided by work in other domains (e.g., adaptive software [58]).

A. Adaptation dimensions

1) Adaptation knowledge: In order to perform adaptation, adaptation policies or algorithms must rely on input knowledge to decide why, when, how and where to perform the adaptation operations. This knowledge can be a simple set of measurements or a model of the entire system.

2) Adaptation strategy: Adaptation mechanisms can either be planned or spontaneous. The goals of planned adaptations are related to enhancing the performance of the system, for example, by offering a new service. Spontaneous adaptations aim at finding immediate solutions to current or anticipated problems.

3) Adaptation purposefulness: Refers to the goal of the adaptation process that may modify the behavior of a one or more network components or that of the entire network.

4) Degree of adaptation autonomy: Refers to the degree of human involvement in the adaptation process. Adaptation may be limited to offering alternative adaptation decisions to the administrator who is then responsible for selecting and enforcing the necessary actions. While the other extreme represents systems that develop and apply the most appropriate decisions without any human support.

5) Adaptation stimuli: Refers to the conditions under which the system undergoes adaptation. While planned adaptation may not necessary require a stimulus to trigger adaptation, spontaneous adaptation may be triggered in response to external or internal events.

6) Adaptation rate: Refers to the frequency of applying the adaptation strategies to the managed system. While a higher rate of minor adaptation operations (e.g., parameter tuning) may result in a better system performance, a frequent application of exhaustive adaptation decisions my consume the system resources and may even result in an unstable system.

Lazy adaptation refers to reactive techniques that are triggered only in response to already occurring events, while opportunistic systems seek appropriate time windows to enhance the performance of the managed entities. Finally, continuous adaptation may be scheduled on equally spaced time intervals achieving a constant adaptation rate.

7) Adaptation temporal scope: Adaptation strategies may also be classified according to the temporal scope of their effects. A short-term adaptation strategy is a one-time applied technique to fix an anticipated or a sensed problem. For example, a policy may be applied once to adapt the behavior of a router (e.g., clearing a buffer contents). Long-term strategies are applied to achieve a desired long-term goal. A typical example of a long-term adaptation strategy is the addition of a new resource or service. A short-term adaptation solution achieves an instantaneous goal while a long-term strategy cumulatively achieves a goal over a longer period.

TABLE II
THE BASE FOR ADAPTATION CLASSIFICATION WITHIN AN ANMS.

Classification Dimension	types and description
Knowledge	complete/partial self-aware environment-aware
Purposefulness	local/global goals
Strategy	spontaneous/planned
Degree of autonomy	human-in-the-loop/ autonomic
Stimuli	None/ internal /external
Rate	opportunistic/lazy/continuous
Temporal scope	short-term/long-term
Spatial scope	local-effects/global-effects
Open vs. closed	static strategies/learning
Vulnerability	smaller/wider coping range

8) Adaptation spatial scope: Refers to the spread of the adaptation effects. Adaptation may be limited to affecting the local internal state (i.e., the structure, control or behavior properties) of a single component, or may affect the behavior of the entire network.

9) Open- versus closed- adaptation: An ANMS is closed-adaptive [58] if the same adaptation strategy is applied repetitively in the same context regardless of whether the strategy was successful or not. A closed-adaptive system maintains a predefined number of adaptation strategies (e.g., predefined set of configuration policies for a router). Open adaptive schemes continuously evolve and enhance their applied adaptation strategies.

10) Vulnerability of adaptation: Refers to the ability to adapt in response to different levels of variations ranging from a small change from the expected normal system conditions (e.g., a small change in server workload) up to extreme cases of change (e.g., simultaneous failure of a number of components in a network due to a malicious attack). The term *coping range* refers to the range of acceptable variations within which the system can adapt its behavior and outside which it may collapse.

A summary of the bases for classification of adaptation approaches within an ANMS is presented in Table II.

B. Autonomic adaptation approaches

Current network adaptation approaches mainly utilize configuration policies or rules. Nonetheless, static policy configurations built a-priori into network devices lack the flexibility and may not be sufficient to handle different changes in these underlying environments. Various research trends (e.g., [59]) have highlighted the notion of policy adaptation and the central role that it can play in achieving autonomic adaptation in policy-enabled networks. In [60], Granville et al. proposed an architecture to support standard policy replacement strategies. Closed-adaptation is performed through meta-policies that control management policies. In [61], open-adaptation is achieved via a genetic algorithm which exports policies that improve the system performance while de-activating policies that degrade its performance. A major limitation of policy-based autonomic adaptation is that the hard-wired control makes no use of any experiences gained from previous adaptation actions. If the system internal or external context changes in a way that is not suitable for the pre-programmed adaptation, external control is required to adjust the existing policies. In general, utilized knowledge in policy-based adaptation approaches is limited the hard-coded conditions using state variables. These approaches follow a planned adaptation strategy with a very limited coping range where the long-term behavior of the managed components is controlled by the a-priori defined policies.

Within the Unity project [62], utility functions are used to arbitrate resource allocation among different application environments where a resource usage is expressed by individual application environments through utility functions. This approach was extended in [63] by the introduction of utility policies that define the desired performance goals in terms of utility functions. Autonomic adaptation is then achieved by selecting the best system configurations as a result of solving an optimization problem. The optimization problem aims at maximizing the resources utilization given these functions. In a similar manner, forecasting functions were used [26] to adapt the network configurations. In this approach, the most appropriate policy actions for the current context of the network are selected from a predefined pool of all possible actions to be applied at the network-levels based on their forecasted behavior. Optimization is also used to compose and update the end-to-end communication paths [64] with the highest service quality. The optimization engine takes, as an input, a set of route requirements in terms of service quality, and a weighted graph representing the current capabilities and constraints of each participating network domain and finds the optimal paths by solving a multi-constrained optimal path problem. One problem in viewing adaptation of ANMSs as an optimization problem or a utility function is that, in general, most mathematical programming approaches deal with static, non-changing, and well defined problems and lack the ability to continuously adapt the *problem formulation* to represent the changing environment. Furthermore, knowledge in such approaches is limited to either current configurations or time series of one or more variables. These approaches are limited in their coping range where sudden changes in any network component can cause the failure of the applied adaptation strategy. They are also limited to targeting short-term adaptation goals.

Recently, a goal-based approach to policy refinement has been introduced in [65] where low-level actions are selected to satisfy a high-level goal using inference and concepts of the event-calculus theory. This approach takes advantage of availability of previous experience gained from already applied policies and their behavior to make decisions concerning the creation of future policies. Another approach has been presented in [66] which attaches a description of the system behavior, in terms of resource utilization, such as network bandwidth and response time, to each specified rule and uses behavior implications to derive new adaptations.

An open-adaptation approach is introduced by Dowling et al. [67] where concepts of collaborative reinforcement learning are used to allow a set of agents to collaboratively adapt their system behavior at run-time. Reinforcement learning is achieved through simple positive and negative feedback among

agents and is used to learn optimal routing policies. An agent's experience with a non-optimal routing (e.g., congested route) results in exchanging negative feedback messages and vice versa, a good connection is reflected by an increase in the reward messages. One limitation of reinforcement learning approaches is the need to maintain a precise model of all exhaustive system states.

Spontaneous adaptation strategies can be achieved through the use of biologically inspired solutions. Swarm intelligence concepts are applied to automate the control of network management problems such as load balancing [68] and route construction and maintenance [69]. These approaches exhibit high resemblance to the well established multi-agent systems [1], and can even be regarded as an extension of such systems in which agents are much simpler in their functionalities. The main advantage to such approaches is their extended temporal and spatial scopes and the flexibility in modifying their purposefulness and strategies. However, they rely on the use of random generators and incur higher computational costs [70].

Autonomic adaptation approaches using evolutionary computing aim at moving or expanding the coping range of the adaptation operations by learning new or modified adaptations. For example, in [71], proxy servers or routers are modeled as bacteria and service requests as food, where each bacterium has an amount of genetic material that codes for the rule set by which it lives. The system then evolves its adaptation through gene migration and random mutation within each server.

VIII. Autonomic Network Management Architectures

In addition to research focusing on the individual functionalities of ANMSs, complete architectural solutions that can tie together these functionalities are investigated in the literature. In this section, we first derive the main criteria to compare various autonomic network management architectures and apply them to compare main contributions in this area.

A. Classification of autonomic architectures

Figure 2 depicts six axes representing the various properties of an autonomic system, namely its degree of activity, degree of adaptability, degree of intelligence, degree of awareness, memory strength, and degree of autonomy.

1) Degree of activity: An autonomic system can satisfy the self-CHOP properties through a reactive or a proactive behavior. A reactive autonomic system attempts to identify significant events or problems that resulted in impairing its performance and then finds an appropriate action or solution after the problem has been already detected. On the other hand, proactive autonomicity [72] promotes the use of preventive measures to maintain the system performance based on the analysis of the current state, anticipated events and the predicted system reaction to them. It is also opportunistic in taking actions to optimize the overall performance (e.g., moving data files near users as their movements are anticipated). As described earlier, 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 (e.g., space missions [73] and network failures due to anomalies [51]).

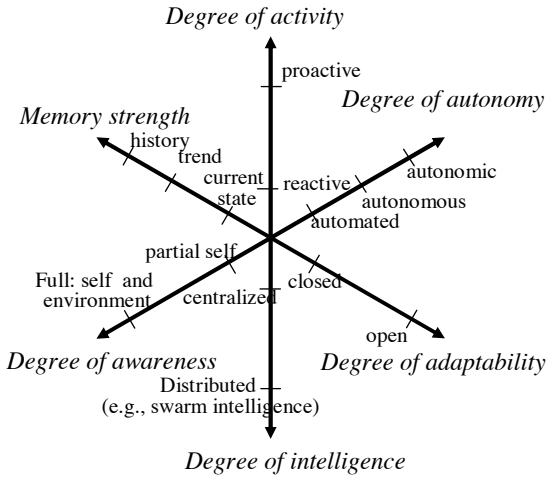

Fig. 2. Classification dimensions of autonomic systems.

2) Degree of adaptability: Similar to all software architectures [58], an autonomic system can be closed-adaptive, open-adaptive or somewhere in between these two extremes. As discussed earlier, closed adaptation refers to the inability to learn new behaviors, while open-adaptability allows the autonomic system to evolve its functionalities either through components interactions [28] or through gained experience [12], [19].

3) Degree of intelligence: In dynamic environments, as the case for network management, an autonomic system must exhibit some intelligence in performing each of the MAPE functions. In this context, intelligence refers to the ability to learn (e.g., from past experiences) in order to enhance its operations. One solution is to limit intelligence to a centralized autonomic manager [63] that controls all other components in the system. The other extreme of distributed intelligence across the managed system may be achieved though the use of swarms or other biologically inspired solutions where intelligence emerges from the collective behavior of (un)intelligent small entities (e.g., [73]).

4) Degree of awareness: A certain degree of self- and environment- awareness is necessary in performing any autonomic functionality. As stated earlier, self-awareness is realized through an efficient KBS. A KBS may maintain a small set of operational parameters and may be extended to the ability of precisely describing the overall system state and behavior.

5) Memory strength: This criterion refers to the ability of the system to remember past behaviors, or problems and their solutions and utilize them in its management decisions. In some autonomic systems, the knowledge of the current system state is sufficient to perform the self-CHOP functionalities (e.g., management of data servers [63]), whereas, in highly dynamic environments, knowledge of behavior trends and histories of past actions can dramatically enhance the system performance (e.g., resource allocation in mobile environments [20]). This knowledge is typically stored as a behavior or control structure in the NKBS.

6) Degree of autonomy and autonomicity: An automated system is one that uses a closed-control loop; the heart of this loop is a manager where the performance of various components in the system is monitored and compared to some pre-specified or desired value. The manager may trigger one or more pre-specified actions if necessary. A human operator specifies, in advance, the desired system performance and the solution of every possible management problem which the system should apply once the problem is encountered.

Autonomy and autonomicity move humans from the control to supervision [72]. Autonomous computing [73] introduces some degree of independence in sensing and reacting to stimuli in the environment. For example, the decisions of when and why to execute (or not) a given function (e.g., system backup) is based on internal system calculations rather than explicit commands or execution programs.

An Autonomic system performs operations on itself to guarantee continuity and efficiency of its expected functionality. Autonomicity mandates self-CHOP properties while hiding details of operations from operators. In contrast to autonomous systems, autonomicity mandates the ability to learn and integrate new types of knowledge and new functionalities. Since achieving the self-properties requires the system to act independently, it is clear that autonomicity implies autonomy [73].

As a typical AC system, one can measure the efficiency of an ANMS using the classification presented in Figure 2. The higher the value of one property on an axis the more advanced it is. For example, a favorable ANMS is highly proactive rather than reactive. Similarly, an open-adaptive ANMS with a high-degree of distributed intelligence is expected to perform better in highly dynamic environments (e.g., wireless and moving networks) where new learnt behaviors are needed to face changes in the environment. Likewise, the more the system is self- and environment- aware the more accurate the management decisions are.

B. Architectural approaches to ANMSs

In this section, we discuss some of the major research projects targeting the development of autonomic network management architectures. Due to space constraints, we limit our discussion to five prominent research projects.

One of the earliest research efforts towards the realization of autonomic management is Unity [74]. Unity is a multi-agent architecture in which agents rely on utility functions to achieve self-optimization, and hence, closed-adaptation of system resources. Monitoring operations are performed via *sentinel elements*, while application resource demands prediction are calculated by *autonomic elements* within the application environments. Closed-adaptation of managed resources is realized through a *resource arbiter element* which computes a globally optimal allocation of servers across the applications based on their forecasted demands. Hard-coded policies are placed into replicated repositories to specify high-level system objectives.

The IST funded autonomic network architecture project, DASADA [75], includes a reference architecture that was developed by a consortium of researchers. The architecture provides the means for automating the management of legacy systems. It is comprised of four types of components, namely, sensors, gauges, controllers, and effectors. Sensors and effectors functionalities are similar to the monitoring and execution functionalities within the MAPE loop. On the other hand, gauges can be regarded as special types of correlation engines that can, based on data supplied from sensors, generate alarms or detect events that necessitate system adaptation which is carried out by controllers. Adaptation is performed in DASDA through the use of mobile codes.

Autonomia [76] is also a multi-agent based system, developed at the University of Arizona, that aims at achieving the self-CHOP properties through the automated deployment and migration of agents. Autonomia is built on top of legacy networks using two software modules: *the component management interface* and the *component runtime manger*. The former provides interfaces to define system behavior and interaction rules, while the latter manages the component operations following the MAPE loop.

The ACCORD project [77] defines an autonomic architecture as a set of autonomic elements. An autonomic element is, in turn, defined as a *self-contained modular software unit of composition with specified interfaces and explicit context dependencies*. Context, internal and external rules are used to guide the elements adaptive behavior and their composition.

The Autonomic Network Architecture (ANA) project [78] targets the development of a new ANMS that solves the non-scalability problem of network functionalities within the traditional Internet and the OSI models. The concept of functional compartments is introduced to encapsulate one or more network functionality rather than using the traditional layering approach.

Table III provides a detailed comparison of the aforementioned projects with respect to the developed classification criteria.

IX. Summary and analysis of future Research Directions

Having discussed the main contributions towards building ANMSs in the literature, this section is dedicated to analyzing the main challenges facing further research advancements.

Based on the surveyed literature, it is the authors' opinion that the autonomic network management concept does not constitute an explicit new alternative for communication management but is rather a new concept which presents a unification of recent advancements and trends in various areas of network research. Based on this premise, ANMSs can only be realized by addressing two issues. The first issue relates to identifying and solving the various challenges in each of these research areas. The second issue addresses the development of architectures that can host these solutions and coordinate their interactions. With respect to the second issue, generic autonomic computing architectures fail to give insights on the needed building blocks specific to ANMSs. On the other hand, current network-specific autonomic architectures are not usually developed with the purpose of hosting a set of external functionalities.

To this end, we propose a reference framework for ANMSs that serves two goals. The first goal is to provide a holistic

TABLE III
A COMPARISON OF DIFFERENT ANMS ARCHITECTURES.

Dimension	Unity	DASADA	Autonomia	ACCORD	ANA
Activity	Proactive optimization actions based on forecasted service demands	Reactive responses to guages and probes	Reactive responses to events detected through the component management interface	An elements within ACCORD acts reactively according to its sensors	Dependent on the hosted mechanism within a network compartment (external to the architecture)
Adaptability	closed-adaptability through the use of static policies	Partial open-adaptability through the use of Worklets (mobile code for software reconfigurability)	closed-adaptability though the use of static policies and utility functions	closed-adaptability though the use of static policies but global intelligence is attained through element composition	closed-adaptability though the dynamic composition of compartments and the compartment stored policies
Intelligence	Policy-based and utility-based management via centralized resource arbitrator	Limited distributed intelligence in controllers that instantiates and coordinates processes	Limited central intelligence in the component runtime manager	Limited distributed intelligence in elements that encapsulate rules, constraints and mechanism for self-management	Dependent on the hosted mechanism within a network compartment (external to the architecture)
Awareness	Partial self-awareness of resources and partial environment awareness (application resource demands)	A structural and behavior knowledge models	Problem and solution knowledge represented in behavior and interaction rules	elements maintain self-knowledge about own functionalities as well as behavior and interaction Rules	Self-awareness is distributed in compartments monitors and exchanged through the use of content networks
Memory	Trends of resource usage and demands modeled as utility functions and control knowledge as policies	Short memory of probes and guages	no memory of previous behavior but a behavior model is pre-built	no memory of previous behavior but behavior rules are pre-built	no memory of previous behavior but behavior rules are stored in compartments
Autonomy	Autonomous (utility functions must be built externally)	Autonomous management using mobile codes	Automated management (policies must be defined externally)	Autonomous management is achieved via the dynamic composition of autonomic elements	The level of autonomy is dependent on the developed functional

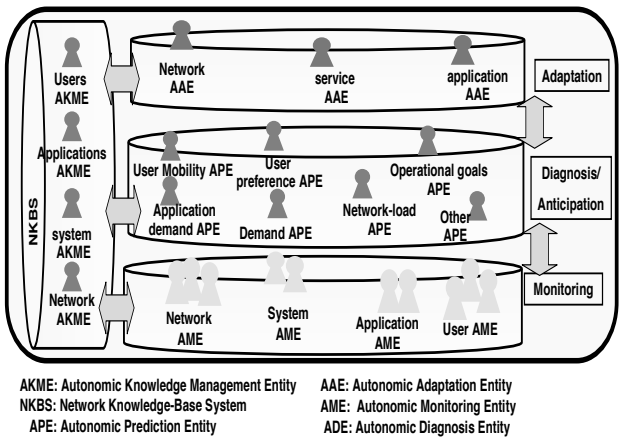

Fig. 3. A general reference framework for ANMSs

view of the diversified research efforts in the literature and, in turn, it can be used to identify future research directions towards the realization of ANMSs. Secondly, it provides the guidelines for the building blocks of ANMS that are needed for the development of future ANMS architectures.

Figure 3 depicts a schematic representation for the reference framework proposed to realize the concepts of AC in network management systems. The framework may be viewed as a realization of the MAPE operations but tailored, as discussed in earlier sections, to the specific network management functionalities.

Similar to generic AC systems, the framework maintains a NKBS that stores a model of the environment of the managed domain, and is accessible to all other components of the system. The NKBS will host various autonomic entities that represent knowledge about users, devices, running applications and the network itself.

We have shown that a major problem of building NKBSs is the development of an expressive network model that can be adopted by different network hardware and software vendors. Current network administrators are more accustomed to the utilization of plain and simple data models and are reluctant to learn newer models. On the other hand, introducing more advanced network models relies heavily on their cooperation in transferring their experience into a structured behavior and control models.

The second component of the ANMS corresponds to the monitoring component of the MAPE operations. In ANMSs, entities within this ***autonomic monitoring layer*** will be tasked with observing the behavior of the network, system, running applications, and serviced users. This step typically involves monitoring a large number of performance metrics with a typically high volume. A major problem with current monitoring approaches is their limited applicability to network-level fault management using a limited set of network performance measurement variables and the lack of techniques for efficient monitoring programmability. A major research challenge in this area is to investigate the use of different types of available knowledge (e.g., context of users and applications) and to

broaden the scope of diagnosis to the detection of even the slight change in the underlying network.

The ***autonomic diagnosis/anticipation layer*** is mainly responsible for analyzing the current system state and predicting future events. It may also be responsible for projecting predictions that might affect the smoothness of network service delivery to the adaptation layer. Most of the current approaches for management are limited to responding to effects rather than anticipating and proactively managing the actual events that caused such effects. The move from reactive to proactive management is still faced with many challenges such as developing algorithms that can identify such events and deal with the variations in certainties in the anticipated data.

The entities within the ***autonomic adaptation layer***, in Figure 3, continuously adapt the behavior of the various components of the managed network to optimize the network performance. Autonomic adaptation has been the focus of intensive research efforts lately, and current literature has investigated the utilization of different theories coming from different disciplines towards the realization of this functionality. This includes the utilization of concepts of the control theory, evolutionary computing and artificial intelligence. Yet, more advances in this area will further contribute to adding more autonomy to the management systems.

A final interesting point that has not been investigated in the literature is the issue of communication and coordination among various autonomic entities in future management systems.

In addition to the aforementioned challenges, other limiting factors to advancing research in this area include the readiness of users of current traditional management systems to accept a transitional period and the difficulty in modifying traditional management thinking with respect to, for example, the long-standing communication protocol stack.

X. Conclusions

While the vision of autonomic computing has attracted considerable attention from research communities in both industry and academia, realizing autonomicity in current network management systems remains a challenging task. This article aspired to help advancing current research efforts in this area by surveying current approaches and putting them into perspective deriving a more holistic view. A coherent classification methodology for approaches to achieve each of the management functionalities is presented and utilized to classify existing research efforts. A reference framework for future management systems was also proposed and applied to identify future research directions and open issues.

A final note is that the authors by no means claim that the survey within this article is exhaustive. Autonomic communication is an emerging field of research that is attracting lots of attention of a large number of researchers and describing all relevant approaches in this area can never be achieved within a single article. Rather, the authors mainly aimed at providing a valid taxonomy that can be used to put current and emerging approaches into perspective.

References

[1] A. Karmouch, "Mobile software agents for telecommunications", *IEEE Commun. Mag.*, vol. 36, n. 7, 1998.

[2] R. Boutaba and A. Polyrakis, "Projecting advanced enterprise network and service management to active networks", *IEEE Network*, vol. 16, n. 1, pp. 28 – 33, Jan.-Feb. 2002.

[3] S. Calo and M. Sloman, "Guest Editorial: Policy-Based Management of Networks and Services", *J. Network and Systems Management*, vol. 11, n. 3, pp. 249–252, 2003.

[4] J. Galvin and K. McCloghrie, "Administrative Model for version 2 of the Simple Network Management Protocol (SNMPv2), RFC 1445, Trusted Information Systems, Hughes, LAN Systems", Apr. 1993.

[5] G. Pavlou, P. Flegkas, S. Gouveris and A. Liotta, "On management technologies and the potential of Web services", *IEEE Commun. Mag.*, vol. 42, n. 7, pp. 58–66, 2004.

[6] "Meta Object Facility (MOF) Specification, Version 1.4, Object Management Group, April 2002, http://www.omg.org/docs/formal/02-04-03.pdf".

[7] J. Kephart and D. Chess, "The Vision of Autonomic Computing", *IEEE Computer*, vol. 36, n. 1, pp. 41–50, Jan. 2003.

[8] J. Strassner, "Knowledge management issues for autonomic systems", in *Database and Expert Systems Applications, 2005. Proc. Sixteenth International Workshop on*, pp. 398– 402, 22-26 Aug. 2005.

[9] Mark A. Musen, *Automated generation of model-based knowledge acquisition tools*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1989.

[10] Mazeiar Salehie and Ladan Tahvildari, "Autonomic Computing: Emerging Trends and Open Problems", in *Proc. IEEE ICSE Workshop on Design and Evolution of Autonomic Application Software (DEAS), St. Louis, Missouri, USA*, pp. 82–88, May 2005.

[11] Distributed Management Task Force, *Common Information Model (CIM) Specification*, Jun 1999.

[12] J.-P. Martin-Flatin, D. Srivastava and A. Westerinen, "Iterative multi-tier management information modeling", *IEEE Commun. Mag.*, vol. 41, n. 12, pp. 92 – 99, Dec. 2003.

[13] B. Moore, E. Ellesson, J. Strassner and A. Westerinen, "Policy Core Information Model, Version 1 Specification. RFC 3060, IBM, LongBoard Inc, Cisco Systems, January 2001".

[14] Y. Yemini, A.V. Konstantinou and D. Florissi, "NESTOR: an architecture for network self-management and organization", *IEEE J. Select. Areas Commun.*, vol. 18, n. 5, pp. 758–766, 2000.

[15] S. Judd and J. Strassner, "Directory-enabled networks: Information model and base schema", 1998.

[16] J. Strassner, "DEN-ng: achieving business-driven network management", in *Network Operations and Management Symposium, 2002. NOMS*, pp. 753–766, 2002.

[17] L. Stojanovic, J. Schneider, A. Maedche, S. Libischer, R. Studer, Th. Lumpp, A. Abecker, G. Breiter and J. Dinger, "The role of ontologies in autonomic computing systems", *IBM Syst. J.*, vol. 43, n. 3, pp. 598–616, 2004.

[18] A.K.Y. Wong, P. Ray, N. Parameswaran and J. Strassner, "Ontology mapping for the interoperability problem in network management", *IEEE J. Select. Areas Commun.*, vol. 23, n. 10, pp. 2058–2068, Oct. 2005.

[19] G. Tesauro, "Reinforcement Learning in Autonomic Computing: A Manifesto and Case Studies", *IEEE Internet Comput.*, vol. 11, n. 1, pp. 22–30, Jan.-Feb. 2007.

[20] N. Samaan and A. Karmouch, "An Automated Policy Based Management Framework for Differentiated Communication Systems", *IEEE J. Select. Areas Commun.*, vol. 23, n. 12, pp. 2236–2247, December 2005.

[21] William E. Walsh, Gerald Tesauro, Jeffrey O. Kephart and Rajarshi Das, "Utility Functions in Autonomic Systems", in *International Conference on Autonomic Computing (ICAC-04)*, 2004.

[22] Diao Y., J. Hellerstein, S. Parekh, R. Griffith, G.E. Kaiser and D. Phung, "control theory foundation for self-managing computing systems", *IEEE J. Select. Areas Commun.*, vol. 23, n. 12, pp. 2213 – 2222, Dec. 2005.

[23] Mikael Nilsson, "Composite Capabilities/Preference Profiles: Terminology and Abbreviations", in *W3C Working Draft*, Jul. 2000.

[24] L. Razmerita, A. Angehrn and A. Maedche, "Ontology based user modeling for Knowledge Management Systems", in *Proc. the User Modeling Conference, Pittsburgh, USA, Springer-Verlag*, pp. 213–217, 2003.

[25] D. Liu and F. Huebner, "Application Profiling of IP Traffic", in *27th Annual IEEE International Conference on Local Computer Networks (LCN02)*, pp. 220–229, Nov.06–08 2002.

[26] N. Samaan and A. Karmouch, "A Mobility Prediction Scheme based on Dempster-Shafer Theory of Evidence", *IEEE Transactions on Mobile Computing*, vol. 4, n. 5, pp. 537 –551, Sept.-Oct. 2005.

[27] Lian Yan, R.H. Wolniewicz and R. Dodier, "Predicting customer behavior in telecommunications", *Intelligent Systems, IEEE*, vol. 19, n. 2, pp. 50– 58, Mar-Apr 2004.

[28] J. Suzuki and T. Suda, "A middleware platform for a biologically inspired network architecture supporting autonomous and adaptive applications", *IEEE J. Select. Areas Commun.*, vol. 23, n. 2, pp. 249 – 260, Feb. 2005.

[29] S. Hariri, B. Khargharia, H. Chen, J. Yang, Y. Zhang, M. Parashar and H. Liu, "The Autonomic Computing Paradigm", *Cluster Computing: The Journal of Networks, Software Tools, and Applications" , Springer Science and Business Media B.V. (Kluwer Academic Publishers)*, vol. 9, n. 1, pp. 5 17, 2006.

[30] J. Keeney, D. Lewis and D. O'Sullivan, "Ontological Semantics for Distributing Contextual Knowledge in Highly Distributed Autonomic Systems", *J. Network and System Management, Special Issue on Autonomic Pervasive and Context-aware Systems*, vol. 15, n. 1, Mar. 2007.

[31] R. Chaparadza, "Introducing on-Demand MIBs to Traffic Engineering", in *14th IEEE International Conference on Networks,ICON06*, pp. 1–6, Sept. 2006.

[32] M. Massie, B. Chun and D. Culler, "The Ganglia Distributed Monitoring System: Design, Implementation, and Experience", *Parallel Computing*, vol. 30, n. 7, pp. 817840, 2004.

[33] S. Agarwala, Y. Chen and K. Milojicic, D. Schwan, "QMON: QoS- and Utility-Aware Monitoring in Enterprise Systems", in *IEEE International Conference on Autonomic Computing*, pp. 124–133, 2006.

[34] Q. Han, S. Gutierrez-Nolasco and N. Venkatasubramanian, "Reflective middleware for integrating network monitoring with adaptive object messaging", *IEEE Network*, vol. 18, n. 1, pp. 56–65, 2004.

[35] K. Miao, H. Schulzrinne, V. Singh and Q. Deng, "Distributed Self Fault-Diagnosis for SIP Multimedia Applications", *Real-Time Mobile Multimedia Services*, vol. 4787, pp. 187–190, 2007.

[36] E.A. Hernandez, M.C. Chidester and A.D. George, "Adaptive Sampling for Network Management", *J. Network and Systems Management*, vol. 9, n. 4, pp. 409–434, 2001.

[37] G. Cantieni, G. Iannaccone, C. Barakat, C. Diot and P. Thiran, "Reformulating the Monitor Placement Problem: Optimal Network-Wide Sampling", in *40th Annual Conference on Information Sciences and Systems, 2006*, pp. 1725–1731, 22-24 March 2006.

[38] Gonzalez Prieto and R.Stadler, "A-GAP: An Adaptive Protocol for Continuous Network Monitoring with Accuracy Objectives", *IEEE Trans. Network Service Management*, to appear.

[39] R. Sterritt, D.W. Bustard, D. Gunning and P. Henning, "Autonomic communications and the reflex unified fault management architecture", *Advanced Engineering Informatics*, vol. 19, n. 3, pp. 189–198, 2005.

[40] Han Hee Song, Lili Qiu and Yin Zhang, "NetQuest: a flexible framework for large-scale network measurement", in *SIGMETRICS/Performance*, pp. 121–132, 2006.

[41] A. Binzenhofer, K. Tutschku, B. auf dem Graben, M. Fiedler and P. Carlsson, "A P2P-based framework for distributed network management", in *Proc. Wireless Systems and Network Architectures in Next Generation Internet*, pp. 198–210. Springer, 2006.

[42] L. Steinder and —A. Sethi, "A survey of fault localization techniques in computer networks", *Science of Computer Programming*, vol. 53, n. 2, pp. 165–194, Nov. 2004.

[43] S. Zhang, I. Cohen, M. Goldszmidt, J. Symons and A. Fox, "Ensembles of Models for Automated Diagnosis of System Performance Problems", *Dependable Systems and Networks, 2005. DSN 2005. Proceedings. International Conference on*, pp. 644–653, 2005.

[44] S. Hariri, G. Qu, R. Modukuri, H. Chen and M. Yousif, "Quality-of-protection (QoP)-an online monitoring and self-protection mechanism", *IEEE J. Select. Areas Commun.*, vol. 23, n. 10, pp. 1983–1993, 2005.

[45] D. L. Tennenhouse, "Proactive Computing", *Communications of the ACM*, vol. 43, n. 5, pp. 4350, May 2000.

[46] Qi He, Constantinos Dovrolis and Mostafa Ammar, "Prediction of TCP throughput: formula-based and history-based methods", *SIGMETRICS Perform. Eval. Rev.*, vol. 33, n. 1, pp. 388–389, 2005.

[47] P. Dinda, "Design, Implementation, and Performance of an Extensible Toolkit for Resource Prediction in Distributed Systems", *IEEE Trans. Parallel Distrib. Syst.*, vol. 17, n. 2, pp. 160–173, 2006.

[48] R. Sahoo, A. Oliner, I. Rish, M. Gupta, J. Moreira, S. Ma, R. Vilalta and A. Sivasubramaniam, "Critical event prediction for proactive management in large-scale computer clusters", *Proc. the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 426–435, 2003.

[49] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford and F. True, "Deriving traffic demands for operational IP networks: methodology and experience", *IEEE/ACM Trans. Networking*, vol. 9, n. 3, pp. 265 – 279, June 2001.

[50] P.A. Dinda and D.R. O'Hallaron, "Host load prediction using linear models", *Cluster Computing*, vol. 3, n. 4, pp. 265–280, 2000.

[51] L. P. Gaspary, R. N. Sanchez, D. W. Antunes and E. Meneghetti, "A SNMP-based platform for distributed stateful intrusion detection in enterprise networks", *IEEE J. Select. Areas Commun.*, vol. 23, n. 10, pp. 1973 – 1982, Oct, 2005.

[52] L.W. Russell, S.P. Morgan and E.G. Chron, "Clockwork: A new movement in autonomic systems", *IBM Systems Journal*, vol. 42, n. 1, pp. 77–84, 2003.

[53] Y.S. Dai, T. Marshall and X. Guan, "Autonomic and dependable computing: moving towards a model-driven approach", *J. Computer Science*, vol. 2, n. 6, pp. 496–504, 2006.

[54] Songyun Duan and Shivnath Babu, "Proactive identification of performance problems", in *SIGMOD '06: Proc. the 2006 ACM SIGMOD international conference on Management of data*, pp. 766–768, New York, NY, USA, 2006, ACM Press.

[55] A. Andrzejak and M. Ceyran, "Characterizing and Predicting Resource Demand by Periodicity Mining", *J. Network and Systems Management*, vol. 13, n. 2, pp. 175–196, 2005.

[56] S. Kirtane and J. Martin, "Application performance prediction in autonomic systems", *Proc. the 44th annual southeast regional conference*, pp. 566–572, 2006.

[57] M. Deshpande and G. Karypis, "Selective Markov models for predicting Web page accesses", *ACM Transactions on Internet Technology (TOIT)*, vol. 4, n. 2, pp. 163–184, 2004.

[58] Peyman Oreizy, Michael M. Gorlick, Richard N. Taylor, Dennis Heimbigner, Gregory Johnson, Nenad Medvidovic, Alex Quilici, David S. Rosenblum and Alexander L. Wolf, "An Architecture-Based Approach to Self-Adaptive Software", *IEEE Intelligent Systems*, vol. 14, n. 3, pp. 54–62, 1999.

[59] L. Lymberopoulos, E. Lupu and M. Sloman, "An Adaptive Policy Based Management Framework for Differentiated Services Networks", in *IEEE 3rd International Workshop on Policies for Distributed Systems and Networks (POLICY'02), Monterey, California*, pp. 147–158, Jun. 2002.

[60] Z. Granville, L., A. Faraco de S Coelho, G., M. Almeida and L. Tarouco, "An Architecture for Automated Replacement of QoS Policies", in *The 7th IEEE Symposium on Computers and Communications (ISCC02), Italy*, Jul. 2002.

[61] I. Marshall and C. Roadknight, "Provision of Quality of Service for Active Services", *Computer Networks*, vol. 36, n. 1, pp. 75–85, Jun. 2001.

[62] D.M. Chess, A. Segal, I. Whalley and S.R. White, "Unity: experiences with a prototype autonomic computing system", in *Autonomic Computing, 2004. Proceedings. International Conference on*, pp. 140– 147, 17-18 May 2004.

[63] J. Kephart and R. Das, "Achieving Self-Management via Utility Functions", *IEEE Internet Comput.*, vol. 11, n. 1, pp. 40 – 48, Jan.-Feb. 2007.

[64] J. Xiao and R. Boutaba, "QoS-aware service composition and adaptation in autonomic communication", *IEEE J. Select. Areas Commun.*, vol. 23, n. 12, pp. 2344–2360, 2005.

[65] A. Bandara, E. Lupu, J. Moffet and A. Russo, "A Goal-based Approach to Policy Refinement", in *5th IEEE Wrkshp on Policies for Distributed Systems and Networks (Policy 2004), New York, USA*, Jun. 2004.

[66] S. Uttamchandaniand, C. Talcott and D. Pease, "Eos: An Approach of Using Behavior Implications for Policy-based Self-management", in *14th IFIP/IEEE Intl Wrkshp on Distributed Systems: Operations and Management, DSOM 2003, Heidelberg, Germany, Oct. 20-22*, pp. 16–27, 2003.

[67] J. Dowling, E. Curran, R. Cunningham, and V. Cahill, "Using feedback in collaborative reinforcement learning to adapt and optimise decentralized distributed systems", *IEEE Trans. Systems, Man, Cybern. A, Special Issue on Engineering Self-Organized Distributed Systems*, vol. 35, n. 3, pp. 360–372, 2005.

[68] F. Chiang, R. Braun and J. Hughes, "A Biologically Inspired Multi-Agent Framework for Autonomic Service Management", *J. Pervasive Computing and Communications*, vol. 2, n. 3, pp. 261275, 2006.

[69] K.M. Sim and W.H. Sun, "Ant colony optimization for routing and load-balancing: survey and new directions", *IEEE Trans. Systems, Man, Cybern. A*, vol. 33, n. 5, pp. 560–572, 2003.

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

[71] Ian W. Marshall and Chris M. Roadknight, "Provision of quality of service for active services.", *Computer Networks*, vol. 36, n. 1, pp. 75–85, 2001.

[72] R. Want, T. Pering and D. Tennenhouse, "Comparing autonomic and proactive computing", *IBM Systems Journal*, vol. 42, n. 1, 2003.

[73] W. Truszkowski, M. Hinchey, J. Rash and C. Rouff, "Autonomous and autonomic systems: a paradigm for future space exploration missions", *IEEE Trans. Systems, Man, Cybern. C*, vol. 36, n. 3, pp. 279–291, 2006.

[74] G. Tesauro, D. Chess, W. Walsh, R. Das, A. Segal, I. Whalley, J. Kephart and S. White, "A Multi-Agent Systems Approach to Autonomic Computing", in *AAMAS '04: Proc. the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 464–471. IEEE Computer Society, 2004.

[75] G. Kaiser, J. Parekh, P. Gross and G. Valetto, "Retrofitting Autonomic Capabilities onto Legacy Systems", *Journal of Cluster Computing*, 2005.

[76] X. Dong, S. Hariri, L. Xue, H. Chen, M. Zhang, S. Pavuluri and S. Rao, "Autonomia: an autonomic computing environment", *Performance, Computing, and Communications Conference, 2003. Conference Proc. the 2003 IEEE International*, pp. 61–68, 2003.

[77] Hua Liu and M. Parashar, "Accord: a programming framework for autonomic applications", *IEEE Trans. Systems, Man, Cybern. C*, vol. 36, n. 3, pp. 341–352, May 2006.

[78] "The ANA blueprint, ANA Project - Deliverable D1.4/5/6v1 ANA Blueprint First Version, FP6-IST-27489", Technical report, ANA: Autonomic Network Architecture, A European Union funded project in Situated and Autonomic Communications, 2007.

Nancy Samaan {nsamaan@site.uottawa.ca}received the Ph.D. degree in computer science from the University of Ottawa, Canada, in 2007 and her BSc and MSc degrees from the department of computer science, Alexandria University, Egypt, in 1998 and 2001, respectively. She is currently an assistant professor with the School of Information Technology and Engineering at the University of Ottawa, Canada. Her current research interests include wireless communications, autonomic computing, mobile computing, network diagnosis, quality of Service management and policy-based management.

Ahmed Karmouch {karmouch@site.uottawa.ca} is a Professor of Electrical and Computer Engineering and computer Science at the School of Information Technology and Engineering, University of Ottawa, Canada. He is involved in several projects with industry and government laboratories in Canada and Europe. His current research interests are in mobile computing, autonomic networking, context aware communications, and ambient networks. He has served in several program committees, organized several conferences and workshops, edited several books and served as Guest Editor for IEEE Communications Magazine, Computer Communications, and others.