Towards the Future Internet
G. Tselentis et al. (Eds.)
IOS Press, 2010
© 2010 The authors and IOS Press. All rights reserved.
doi:10.3233/978-1-60750-539-6-95

An experimental path towards Self-Management for Future Internet Environments

Apostolos Kousaridas [a,1], Gérard Nguengang [b], Julien Boite [b], Vania Conan [b], Vangelis Gazis[a], Tilemachos Raptis[a], Nancy Alonistioti[a]
[a] *Department of Informatics and Telecommunications, University of Athens, Athens, Greece*
{akousar, gazis, traptis, nancy}@di.uoa.gr
[b] *Thales Communications SA*
Paris, France
{Gerard.NGUENGANG, Julien.BOITE, Vania.CONAN}@fr.thalesgroup.com

Abstract. The current challenge for the network systems is the reduction of human intervention in the fundamental management functions and the development of mechanisms that will render the network capable to autonomously configure, optimize, protect and heal itself, handling in parallel the emerging complexity. The in-network cognitive cycle will allow the continuous improvement of management functions of individual network elements and collectively of a whole network system. This paper proposes the software components for the engineering of an innovative self-managed future Internet system that will support visionary research through experimentation.

Keywords. Self-management, cognitive networks, future internet, experimentation

1. Introduction

Communication networks are growing both in term of size and complexity. Existing systems have serious deficiencies in effectively addressing the changing landscape of application demand and traffic patterns. In order to address the increasing complexity, the embodiment of autonomic capabilities in Future Internet networks and management systems emerge as the most appealing solution. In this paper we present the concrete experimental steps that we are investigating in evaluating the application of autonomic management principles to real network systems and equipment. Firstly, we review the major ingredients of self-management, and describe in particular the Monitor-Decide-Execute (MDE) loop concept. We then outline the architecture of the distributed software platform we are developing that implements these principles for the management of Future Internet networks. Finally, we apply the approach to the challenging problem of autonomic wireless network management. We conclude with a discussion of major challenges that lie ahead.

[1] Corresponding Author.

2. Network Systems Self-Management Background

The management systems of future Internet networks are expected to embed autonomic capabilities in order to face the increasing complexity of communication networks that make them difficult to manage [1], [2]. This autonomic enablement implies that networks are able to self-manage their operational features (i.e. self-configure, self-heal, self-optimize, self-protect) [3]. Their behaviour is set from high-level policies that they implement by dynamically adapting network parameters (e.g., routing, protocols, queue sizes, radio frequencies) according to the varying network conditions (e.g., changes in topology, resource availability, traffic demands).

Designing an autonomic system for network management involves several technologies and disciplines and has received significant research effort the last decade ([4], [5], [6], [7]). Moreover, several European research projects are working towards this direction e.g., ANA [8], BIONETS [9], 4WARD [10]. On the implementation side, attempts are made to add more flexibility in network protocol design: Keller et al. propose a framework for self-composing protocol stacks in multiple compartments to replace the existing non-flexible Internet layers [11], and Manzalini provides a platform for autonomic services composition [12]. Focused on the middleware part of the framework, another research trend consists in integrating mobile agents in the network to collect distributed information or perform remote computations [13]. A combination of both mobile and stationary agents to manage mobile ones is proposed in [14].

Cognition development is an important aspect of future Internet self-managed systems, which complements and advances the automation of configuration actions. An in-network cognitive cycle will allow communication systems to improve their inference and reasoning capabilities, by exploiting the feedback from previous events or from historic data that are stored locally. In the literature, there are several simple or more complex multidisciplinary models for cognition development e.g., [15], [16]. However, the majority of them have not being designed considering the restrictions or capabilities that communication networks have. On the other hand, the cognitive models that refer to communication systems do not describe in many cases the available options and how they could be applied in a holistic networking context. Thus, there is the need to present how such ideas could be engineered in a real-world implementation.

3. Cognition Development

In the Autonomic Network vision, each network device (e.g., router, gateway, access point), is potentially considered as an autonomic element. The latter is capable of monitoring its network-related state and modifying it based on conditions that administrators have specified. This cognitive cycle consists in constantly monitoring this network state, making decisions and executing reconfiguration actions (see Figure 1). The cognitive cycle is implemented in software, and the autonomic element executing this function is further referred to as the cognitive network manager (CNM). Thanks to its inference capability, a CNM can perform rapid detection of device anomalies or network services disruption, diagnose the root cause, compute possible corrective actions to finally select and enforce the one that best fits the operator's goals and objectives. The monitoring process is carried out by the CNM's sensors, while effectors perform the execution of reconfigurations.

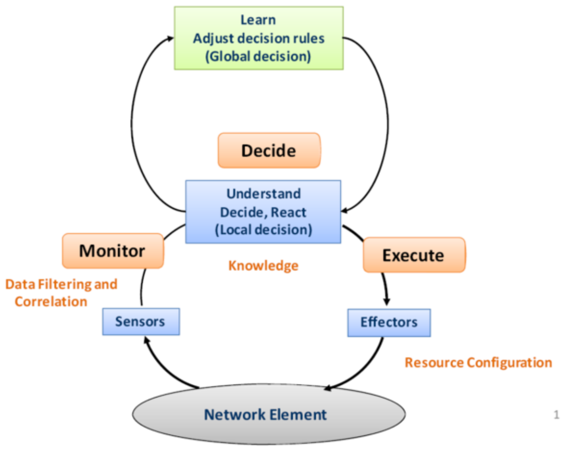

Figure 1. Cognitive Network Managers implement the cognitive cycle

Network devices are physically distributed and the configuration of particular element often interplays with other nodes' configuration. As a consequence, the representation of the network context that each CNM uses for making decisions cannot be limited to its local point of view. The adoption of a centralized approach in order to provide this broader picture is not considered effective, since scalability and fault tolerance issues incur. Thus, cooperation among neighboring autonomic elements is required so as to exchange information that is necessary for the decision making process.

Broadcasting messages throughout the entire network is not an efficient option, since the autonomic management system should limit the communication to a defined range. Considering that knowledge about an entire network is not necessary in order to make fast local decisions either, information exchange is limited to compartments (subsets of neighbouring network elements). As a consequence, each CNM makes local decisions based on a situated knowledge (i.e. information gathered and exchanged between neighbouring entities in the defined compartment view). Figure 2 represents the compartment view of two CNMs, namely A and B. In this example, the compartment is limited to a one hop neighbourhood, but this range can dynamically be extended if the capturing of a broader view of the network state is necessary. However, there is a trade-off between extending this local view and limiting the communication cost for the management of the compartment: the larger the compartment, the more the amount of exchanged information increases.

In the case that information broader than the knowledge confined in the compartment view of a CNM is required for making some decisions (e.g., regarding end-to-end connectivity), the Domain Cognitive Network Manager (DCNM) is used. The DCNM, as it is depicted in Figure 2, is a more sophisticated CNM that can deal with higher-level knowledge, collected by a set of subsuming CNMs in the specified domain that it manages. Domain CNMs having this richer knowledge possibly complemented with interactions with other DCNMs, can proceed with global decision making, thus solving problems that standard CNMs cannot address locally, adjust policies to fit the desired system behaviour and learn from previous experiences to improve the impact of future decisions (upper part of Figure 1).

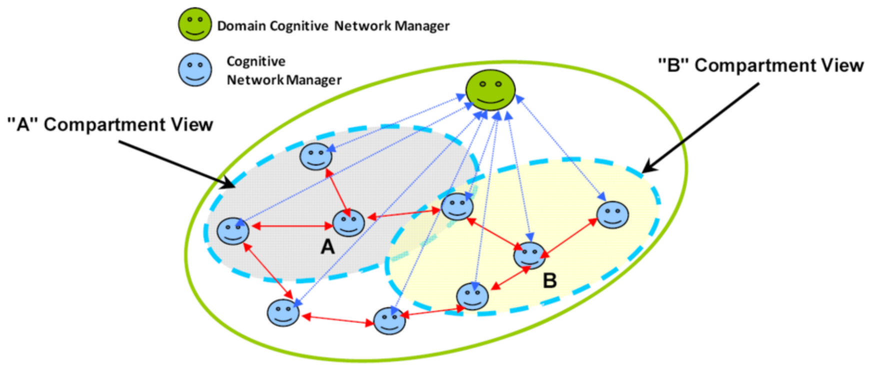

Figure 2. One-hop compartment views

4. Engineering the Cognitive cycle for Network Management

4.1. Software requirements

Implementing the architecture described in section 3, where distributed cognitive managers are embedded in network elements, implies meeting several software requirements. First of all, the network management framework needs to be fully distributed in order to scale up to large or constrained networks and to be fault tolerant. Moreover, it should provide features that are necessary to implement the cognitive cycle. This includes functionalities allowing a CNM to:

- perceive the network state and act on it
- communicate with neighbouring entities
- make fast decisions in order to deal with applications or services that are not tolerant to delays.

This last point implies that a CNM manipulates knowledge (representation of the network context, operator policies). The distribution of entities means that knowledge is distributed. As a consequence, the knowledge representation must be uniform in order for entities to exchange understandable information.

Besides the requirements that are needed in order to implement the cognitive cycle, a key issue for the viability of the proposed in-network management solution is the integration in network devices. The various characteristics of such equipment imply specific software requirements that should also be considered. In fact, devices in today’s networks are widely heterogeneous and often resource-constrained. In order to fit these characteristics, the software solution must be easily adaptable. Modularity is necessary not only to plug the platform easily on any type of devices but also to load only required parts of it on resource-constrained devices.

4.2. Framework description

The framework we have implemented in the Self-NET project [17] meets the requirements stated above. Developed in Java, its main characteristics are full distribution of cognitive entities and components modularity. A java-based solution has

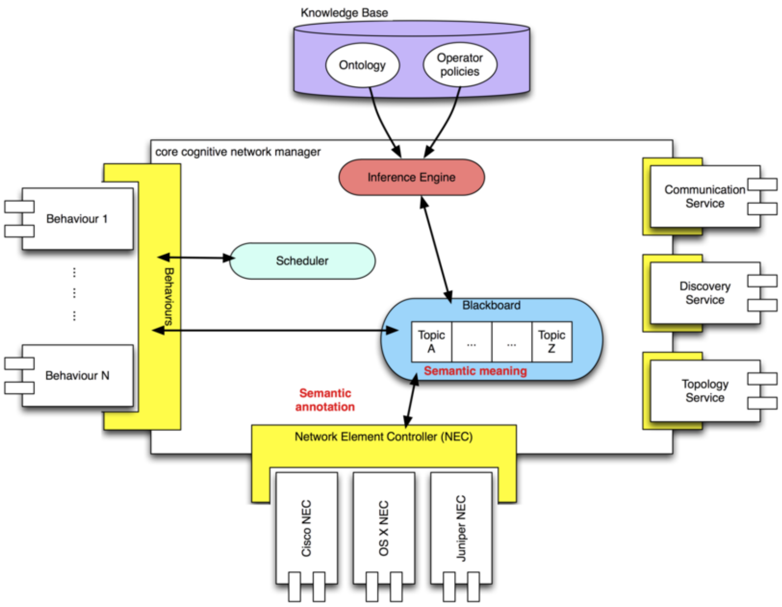

Figure 3. Software modules of the Cognitive Network Manager

been selected for the experimentation, since it offers platform-independence and embedded distributed computing features. However, alternative programming environments (e.g., scripting languages) could be selected, especially in the case of resource-constrained devices (e.g., wireless mesh network router).

Figure 3 depicts the different components of the platform, which main benefit resides in the simplicity of adding, deleting or changing one of its components (e.g., network element controllers need to be adapted to the type of device the CNM is embedded in). This modularity has been brought by developing the framework in a component-oriented way, on top of OSGi (Open Services Gateway initiative) [18]. Each component of the cognitive network manager is implemented as a module, called a bundle in the OSGi terminology.

Some of these modules undertake to capture the network context or to enforce the decided (re-)configurations:

- *Topology service:* based on a regular exchange of hello messages, constructs, updates and exchanges adjacency tables that reflect the physical network topology.
- *Discovery service:* maintains information concerning neighbouring cognitive network managers (e.g., IP addresses, network interfaces used to reach them).
- *Network Element Controller*: gathers network equipment status and publishes these different pieces of information (interfaces, routing…) in the blackboard. It also provides means to enforce reconfigurations in the network device.

The *Blackboard* is a key component of the framework. It provides writing and reading facilities in a shared space, where information are semantically organized in different topics. This component can be seen as the short-term memory of the system, where recently gathered information can be published or retrieved.

A Cognitive Network Manager needs to communicate with neighbouring entities. This function is assured by the *Communication Service* that provides communication functions with high-level semantics (communication acts, content identification).

During the execution of the cognitive cycle, some specific tasks might be necessary (e.g., sending information periodically or computing some specific information needed for possible reconfigurations). *Behaviours* are in charge of these specific tasks and can be configured to be executed only once or periodically. The execution of a behaviour is triggered by the *Scheduler*.

Finally, the reasoning capabilities of a CNM rely on the *Inference Engine* module, which can be seen as the brain of the system. The Jess inference engine has been integrated in our framework for such purpose [19]. A-priori knowledge is loaded into it. This a-priori knowledge is composed of operator policies and an ontology [20], which is the appropriate tool providing a uniform way to represent the "world" and capture a domain technical know-how. During the execution of the cognitive cycle, collected information are confronted to the policies defined by administrators to autonomously detect anomalies and face them. The inference engine makes high-level decisions that may need additional information for reconfiguring the network equipment. Such specific additional information are computed by behaviours and published in the blackboard.

Each CNM and Domain CNM, runs this set of modules. The difference between them is mainly the level and scope of knowledge they manipulate. In fact, Domain CNMs manipulate richer knowledge, resulting from CNMs inferred data or coming from exchanges with other DCNMs. Moreover, computations and decisions a DCNM makes are different from those a CNMs processes. Being at a higher level, the response time of their decisions can be longer whereas CNMs role is to react fast.

5. Experimentation on Capacity Optimization of Future Internet Wireless Networks

One of the key target areas where Autonomic management can bring significant benefits is wireless networks. In such a wireless infrastructure, network (self-) management is particularly challenging because of the volatile and unpredictable nature of the wireless medium and the mobility patterns of terminal devices. Due to the strong demand for efficient wireless resources utilization, wireless network planning and management is a sophisticated task, which requires expert knowledge, especially in a dense urban environment. Network nodes (e.g., access points) that have several configuration capabilities and observe their local operational status should coordinate to improve overall performance and the complex problem of efficient wireless resource management that arises. A centralized approach is not effective due to scalability and complexity issues. Thus, a more localized and distributed architecture is necessary for the orchestration of the various heterogeneous or homogenous access points (APs).

5.1. Radio Resource Allocation Problem in Wireless Networks

The CNM model is applied to the coverage and capacity optimization of future Internet wireless network in order to illustrate how collective interaction of CNMs can automate and solve a network management problem about resource allocation. In order to present and test the key functionalities of the CNM model, we have selected a specific network management problem, from the family of coverage and capacity

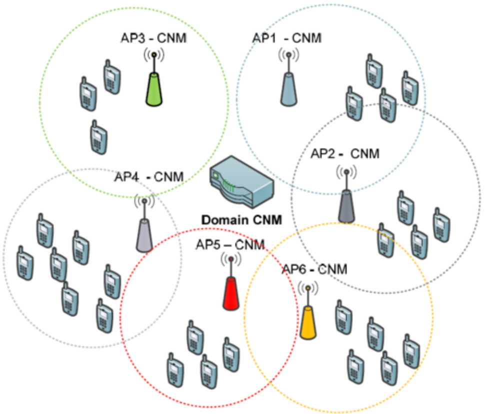

Figure 4. Sample Wireless Topology and CNMs

optimization of Future Internet wireless network.

Particularly for IEEE 802.11a/b/g, efficient radio resource allocation in a dense urban environment can be quite a challenge, due to the finite number of interference-free channels available. Channel allocation in IEEE 802.11a/b/g systems may result in conflict where more than one adjacent (in terms of radio coverage) access points use the same or different channels but with a substantial nonetheless spectrum overlap, thus causing a substantial drop in performance. Channel assignment is a determinant factor of performance in WLAN installations that requires intelligence and coordination to achieve maximum potential.

In the huge mass consumer market segment targeted by modern WLAN technologies, the typical consumer does not possess the technical expertise required to fully comprehend and efficiently solve resource allocation problems e.g., frequency planning. Moreover, existing standards for popular wireless access technologies do not provide the capacity to interact with peers on the basis of local and exchanged information for purposes of improving coverage and capacity. Therefore, the introduction of an autonomic mechanism (i.e. CNM) that undertakes the discovery of conflicting frequency plans and initiates reactive measures to adapt frequency assignments, collaboratively among the concerned network elements is necessary.

5.2. Cognitive Network Managers for dynamic optimization of Frequency Channel

In this sub-section it is presented how the CNM framework is used in order to handle and automate the problem that is described in section 5.1. Moreover, the role of each component of the CNM is outlined.

We consider a dense wireless network environment consisting of several APs (e.g., IEEE 802.11), each embedding the CNM, while a Domain CNM is assigned for the underlying APs (Figure 4). The CNM that is placed per AP undertakes a) to make deductions about its operational status b) to proactively prepare solutions to face possible problems and c) to react fast when a problem occurs by enforcing the anticipated reconfiguration actions, thus seizing the need for human intervention.

In the context of the frequency allocation problem, the knowledge that the CNMs need in order to reason and make decisions refers to the operational status of an AP that is deduced through various metrics (e.g., Signal to Interference plus Noise Ratio, packet error rate, number of associated mobile devices). In order for a CNM to understand its current operational state, the metrics that it gathers from the AP are

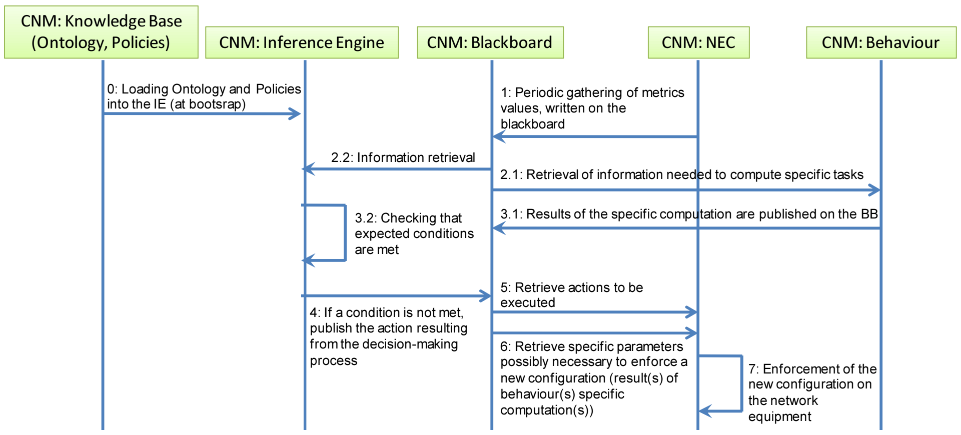
CNM: Knowledge Base (Ontology, Policies)
CNM: Inference Engine
CNM: Blackboard
CNM: NEC
CNM: Behaviour
0: Loading Ontology and Policies into the IE (at bootsrap)
1: Periodic gathering of metrics values, written on the blackboard
2.2: Information retrieval
2.1: Retrieval of information needed to compute specific tasks
3.2: Checking that expected conditions are met
3.1: Results of the specific computation are published on the BB
5: Retrieve actions to be executed
4: If a condition is not met, publish the action resulting from the decision-making process
6: Retrieve specific parameters possibly necessary to enforce a new configuration (result(s) of behaviour(s) specific computation(s))
7: Enforcement of the new configuration on the network equipment

Figure 5. Flow Chart for internal interaction of CNM modules

mapped to the concepts that are described in the ontology, including the concept of "InterferenceLevel". Moreover, the decision making process relies on a set of policies defined by the operator. For instance in our use case, two rules can be defined:

- A first one in order to estimate the level of interference (the value of *threshold* can be configured):

  ```
  If ( SINR ) >= threshold Then InterferenceLevel is HIGH
  ```

- A second one in order to actually make the decision of changing the channel frequency to face a detected *HIGH* level of interference:

  ```
  If ( InterferenceLevel is HIGH ) Then ChangeFrequency
  ```

The initial step of a CNM after its initiation includes the loading of the ontology and the rules that are defined above into the inference engine that is responsible for reasoning. The interactions of the CNM modules are briefly depicted in Figure 5, and the role of the essential CNM components instantiated for this particular scenario are presented as follows:

- **Step 1:** at regular intervals of time, the **Network Element Controller** (NEC) provides updates of specific metrics and performance parameters of the access point (e.g., Signal to Interference plus Noise Ratio (SINR), packet error rate, number of associated devices, operational channel) and publishes these raw data into the appropriate topic of the blackboard (e.g., SINR values are published in a topic named "Interference").
- **Steps 2 and 3:** this step is composed of multiple tasks executed in parallel, where different components realize the following processes, respectively:
  - o **Inference Engine** (2.2 and 3.2 in Figure 5): retrieves information stored in the different topics of the blackboard to check if expected conditions are met. In our example, measurements gathered by the NEC and stored in the blackboard are retrieved and compared to the defined rules in order to determine whether the interference level is acceptable (*LOW* or *MEDIUM* levels) or not (*HIGH*). In the case where interference level is considered *HIGH*, the second rule (presented above) is triggered thus producing the order of executing the action *ChangeFrequency*.

- o **Behaviours** (2.1 and 3.1 in Figure 5): realize specific tasks regarding to the problem that a CNM addresses. For instance, in the described use case, a behaviour uses the **Communication Service** to exchange information with neighbouring CNMs (e.g., currently used channel). All the inform**ation that is receive**d from neighb**o**urs is maintained in the **Discovery Service**.
  Another behaviour is in charge of executing the specific task of selecting the most appropriate frequency channel to use. To achieve this, needed information is retrieved from the corresponding topics in the blackboard (local metrics) or from the Discovery Service (frequency channels used by neighbours and number of devices connected to them). The most appropriate frequency channel is then selected by minimizing the objective function $\sum_{i=a}^{b} w_i * Overlap(Ch_i, Ch_j)$, where:
  - Ch_i is the channel of the APs that are sensed by the AP j,
  - w_i is the number of users that are connected per neighbouring AP,
  - $Overlap(Ch_i, Ch_j)$ provides the level of channels i and j theoretical interference overlap. An example of this function is provided in [21].

  The result of this channel selection is published in a dedicated topic of the blackboard, named "optimalFrequencyChannel".
- **Step 4:** in the case that the **Inference Engine** triggers a rule that orders a reconfiguration action, this last is published in a dedicated topic of the blackboard, named "actions". For instance, the "*changeFrequency*" resulting from the second rule, in this use case, is stored in the topic "actions" of the blackboard in order for the Network Element Controller to deal with the corresponding reconfiguration in the following steps.
- **Steps 5, 6 and 7:** these steps actually involve the **Network Element Controller** in dealing with the reconfiguration order. The NEC is triggered when an action is ordered in the respective topic of the blackboard (entitled "action") and should change the network equipment accordingly. In the specific example, the action includes the change of the existing channel of the AP with the most appropriate value, computed by the corresponding behaviour and stored in the topic "optimalFrequencyChannel" of the blackboard. Finally, the NEC retrieves this optimal frequency channel and enforces the new configuration.

As a result, the Cognitive Network Manager locally executing the cognitive control loop, as it is discussed in this section, provides an efficient way to optimize frequency channel allocation in a dense wireless network environment.

Furthermore, the domain CNM supports the behaviours of local CNMs as well as some additional behaviours that emerge due to the greater spatial situation awareness and configuration capabilities that is collectively provided by the underlying CNMs. In the specific network management case the domain CNM can include the following behaviours: a) check the coverage levels of the network area, b) activate an Access Point, c) deactivate an Access Point. Similarly, the domain CNM uses the communication means that its device offers in order to provide the Topology service, the Discovery service, and the Communications service.

The CNM model described above could be applied to various other network management problems, for coverage and capacity optimization, such as load balancing, handover parameter optimization, QoS parameter optimization or even for fault management tasks e.g., cell outage detection.

6. Conclusions

The automation of the existing network management systems is very limited as well as their ability to collectively address complex management problems. The degree of cognition is very small, since machine learning techniques or reasoning tools are not extensively used. CNM attempts to provide the software architecture for a realistic and implementable self-managed network system. The inherent distribution of cognitive agents as well as components modularity, and the adoption of an open standard (i.e. OSGi) assure the applicability of CNM in the Future Internet.

Acknowledgement

This work is supported by the European Commission Seventh Framework Programme ICT-2008-224344 through the Self-NET Project (https://www.ict-selfnet.eu).

References

[1] Strassner, J. K., “Autonomic Systems and Networks: Theory and Practice”, 10th IEEE/IFIP Network Operations and Management Symposium,(NOMS) , pp. 588-588, 2006.
[2] Ganek, A. G., “The dawning of the autonomic computing era”, IBM Systems Journal, 2003.
[3] Horn, P., "Autonomic computing : IBM's perspective on the state of Information Technology", 2001.
[4] M. A. Siqueira, F. Luciano V. Rafael Pasquini, M. Magalhães, “An Architecture for Autonomic Management of Ambient Networks”, Autonomic Networking, pp. 255-267, 2006.
[5] C. Foley, S. Balasubramaniam, E. Power, M. Ponce de Leon, et al. “A Framework for In-Network Management in Heterogeneous Future Communication Networks”, MACE, 2008.
[6] Jennings, B., Van der Meer, S., Balasubramaniam, , et al. .“Towards Autonomic Management of Communications Networks”, IEEE Communications Magazine 45(10), pp. 112–121, 2007.
[7] R. Chaparadza, S. Papavassiliou, T. Kastrinogiannis, M. Vigoureux, et al. Creating a viable Evolution Path towards Self-Managing Future Internet via a Standardizable Reference Model for Autonomic Network Engineering. FIA Prague 2009 Conference, published in the FI Book produced by FIA, 2009.
[8] The ANA Project, [Online]. Available: http://www.ana-project.org [Accessed: Jan. 20, 2010]
[9] The BIONETS Project [Online]. Available: http://www.bionets.eu [Accessed: Jan. 20, 2010]
[10] The 4WARD Project [Online]. Available: http://www.4ward-project.eu [Accessed: Jan. 20, 2010]
[11] A. Keller , T. Hossmann , M. May, G. Bouabene, C. Jelger, C. Tschudin, “A System Architecture for Evolving Protocol Stacks”, Computer Communications and Networks. (ICCCN), pp. 1-7, 2008.
[12] Manzalini A., Zambonelli F., “Towards Autonomic and Situation-Aware Communication Services: the CASCADAS Vision”. Proceedings of the IEEE Workshop on Distributed Intelligent Systems: Collective Intelligence and Its Applications (DIS’06) , pp. 383-388, 2006.
[13] Lefebvre, J. Chamberland, S. Pierre, S. “A Network Management Framework Using Mobile Agents”, Journal of Computer Science , 2005.
[14] P. Ray, N. Parameswaran, L. Lewis, G. Jakobson, “Distributed Autonomic Management: An Approach and Experiment towards Managing Service-Centric Networks”, Internat. Conf. on Service Systems and Service Management, pp. 1-6, 2008.
[15] Dave Cliff, “Biologically-Inspired Computing Approaches To Cognitive Systems: a partial tour of the literature”, Technical Report, http://www.hpl.hp.com/techreports/2003/HPL-2003-11.html.
[16] D. Vernon, G. Metta, G. Sandini, “A Survey of Artificial Cognitive Systems: Implications for the Autonomous Development of Mental Capabilities in Computational Agents”, IEEE Transactions on Evolutionary Computation, Special Issue on Autonomous Mental Development, 2007.
[17] Self-NET Project, [Online]. Available: https://www.ict-selfnet.eu. [Accessed: Jan. 20, 2010]
[18] OSGi Alliance, [Online]. Available: http://www.osgi.org. [Accessed: Jan. 20, 2010]
[19] Jess: The rule Engine for the Java Platform, [Online]. Available: http//www.jessrules.com
[20] D. Fensel, “Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce”, Springer-Verlag, 2001.
[21] Arunesh Mishra, Suman Banerjee, William A. Arbaugh: Weighted coloring based channel assignment for WLANs. Mobile Computing and Communications Review 9(3): 19-31, 2005.