Initial Approach Toward Self-configuration and Self-optimization in IP Networks

Elyes Lehtihet[1,2], Hajer Derbel[1], Nazim Agoulmine[1], Yacine Ghamri-Doudane[1], and Sven van der Meer[2]

[1] Laboratoire de Réseaux et Systèmes Multimédia, Institut d'Informatique d'Entreprise and Université d'Evry-Val d'Essonne, Evry, France
{derbel, ghamri}@iie.cnam.fr
Nazim.Agoulmine@iup.univ-evry.fr
[2] Telecommunications Software & Systems Group, Waterford Institute of Technology, Cork Road, Waterford, Ireland
{elehtihet, vdmeer}@tssg.org

Abstract. The growing heterogeneity and scalability of Internet services has complicated, beyond human capabilities, the management of network devices. Therefore, a new paradigm called autonomic networking is being introduced to control, in an efficient and automatic manner, this complex environment. This approach aims to enhance network elements with capabilities that allow them to choose their own behavior for achieving high-level directives. This so called autonomic network element should be able to optimize its configuration, ensure its protection, detect/repair unpredicted conflicts between services requirements and coordinate its behavior with other network elements.

In this paper, we present a research activity that investigates this new concept, and applies it to facilitate the configuration and the optimization of a multi-services IP network. This approach is a first step toward building a self-configured and self-optimized IP network that automatically supports the QoS requirements of heterogeneous applications without any external intervention. Different paradigms have been explored in order to model this behavior and to render network equipment autonomic. A laboratory prototype has been developed to highlight the autonomic behavior of the network to achieve heterogeneous QoS requirements of multimedia and data applications.

1 Introduction

The explosion of Internet technologies, services and applications and their corresponding heterogeneity has exacerbated, beyond human capacity, the complexity of managing the Internet. Traditional management and control techniques are no longer capable of ensuring the efficiency and cost effectiveness of existing and more probably future networks. This problem is already considered as crucial by the research community in almost all computer systems and recently a

J. Dalmau and G. Hasegawa (Eds.): MMNS 2005, LNCS 3754, pp. 371–382, 2005.

new paradigm has emerged as a potential solution. This paradigm, called selfware, is a novel approach to perform network control, as well as management of middle box communication, service creation and composition of network functionalities. It is based on universal and fine-grained multiplexing of numerous policies, rules and events that is done autonomously to facilitates desired behavior of groups of network elements [1]. This approach focuses on the application of analogies from biology and economics to massively distributed computing systems, particularly in the domains of autonomic computing. IBM is considered as the first to introduce this term into the field of computing in 2001. This initiative aims to unify research related to selfware computer systems that are a capable of being self-managed [2]. Self-management encompasses a number of selfware management capabilities such as: Self-configuration, Self-optimizing, Self-healing, Self-protection, Self-awareness, etc.[3].

The concept of selfware is not only applicable to computers but to any system with processing, memory and communication capabilities. It not only affects the design of the system but also the applications, middleware, network equipment, etc. In the networking realm, the objective is to design autonomic networks that are able to manage themselves in an autonomic way and exhibit a global "intelligence" through their interactions. Our objective in this research is to investigate this concept and apply it in order to simplify the control and management of operators' IP networks. The aim is to aid the operator in the complex task of managing their networks by allowing them to control the networks behavior through only high-level goals. The network will automatically adjust its configuration to fulfill these goals without any intervention from the operator. This approach can facilitate cooperation between different administrative domains that share network devices.

This paper is organized as follows: Section 2 presents the limitations of current management approaches that have motivated the investigation of a novel approach. The following section presents the proposed management architecture. Section 4 describes the language we have designed to capture the high-level management goal. The following section details the architecture of the Autonomic Element (AE) as well as the internal functionalities. A prototype of the system as well as a set of conducted tests are presented in section 6. The following section 7 presents general discussions as well as concluding remarks about this work and future directions.

2 A Complexity Beyond the Capacity of Existing Management Systems

Policy Based Management (PBM) is defined as the usage of policy rules to manage the configuration and behavior of one or more entities [4]. In the PBM approach, decisions related to the allocation of network resources and/or security are taken by a central control entity called PDP (Policy Decision Point), which concentrates the entire decision-making activity of the system. However, as the network becomes larger and more heterogeneous and the provided service

varies with different QoS requirement, these approaches become very difficult to specify, design and deploy. In the PBM approach, the number of policy rules, the consistency between the rules and the knowledge expected from the operator to control the entire network render it very complex to achieve. The only known solution to deal with this growing complexity in the realm is the complete decentralization of the decision-making process among the distributed entities.

We consider that the operational parameters (Routing services, QoS services, Connectivity Services, ...) offered by individual network devices must be modeled in accordance with an **agreed-upon data model**. The definition of a common data model enables the network administrator to map business rules to the network by refining abstract entities into concrete objects/devices [5]; and the network designer to engineer the network such that it can provide different functions for different requirements which is equivalent to the refinement of high-level goals into operations, supported by the concrete objects/devices, that when performed will achieve the high-level goal [5].

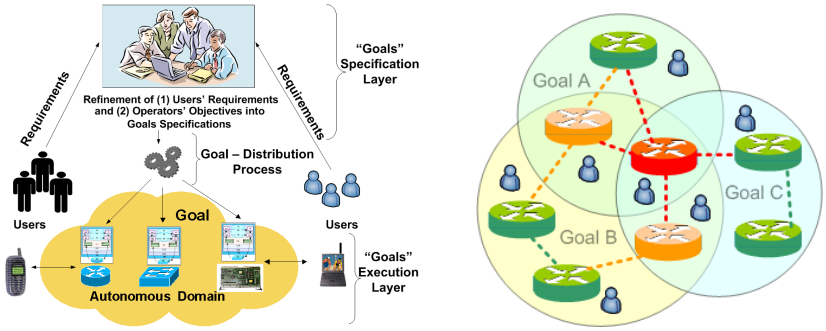
In our approach we aim to achieve two main objectives; (1) to avoid the centralization of the decision making process and (2) facilitate the specification and the enforcement of operators objectives. We argue that every element in the network should be autonomic i.e. having the capability to take its own decision and to supervise several management objectives. The enforcement of this autonomy is achieved by (1) the specification of high-level goals (we have used an approach based on Finite State Machine theory, where each state represents a target behavior of the autonomic elements in a particular context), and (2) the specification of an interaction schema between AEs to coordinate their behavior and achieve their goals.

3 Autonomic Management Based on Goals

The management of today's communication systems needs more autonomy and decentralization. With this in mind, we propose a new management approach called "Goal-based management". The aim of this approach is to design a network capable of to organizing itself in such way that the aggregate behavior of each autonomic element satisfies the high-level operational goals defined by the administrator.

We define a goal as a semantic association between system resources. By subscribing a Goal, an autonomic element becomes a part of an entire autonomous domain (i.e. regrouping of elements with a single objective). All the elements associated with a single Goal are viewed as a single autonomic entity though an element can be associated to multiple Goals. Consequently, the behavior of an autonomic element can be driven by one or multiple Goals. The ability of the autonomic element to deal with multiple Goal definitions (multiple business requirements) is handled by the goal language. The setting of the managed element allows him to detect unpredicted conflicts between different objectives.

Systems resources (autonomic elements) inherit their properties and relationships from a general information model, which we have defined to model both



a. Goal-based network architecture

b. Multiple overlapped autonomous domains sharing common resources

Fig. 1. Goal-based Management

AE behavior and communication mechanisms. The knowledge associated with these relationships is essential for almost all autonomic management functions. However, it is not sufficient to only establish the relationships between resources but it is also important to capture the semantics of these relations. In fact, without this explicit meaning the resolution of a problem is not possible [6]. As depicted in fig.1, our framework is organized into two layers:

- *"Goals"* specification layer: its aim is to define the high-level objectives (Goals). This layer introduces a number of "goal specifications" in the network that allows the specification of Goals in terms of explicit behaviors expected from the target autonomic elements in various contexts. . In this first approach, the goals are specified as a state machine representing different possible network element behaviors as well as transition conditions between these states. These Goals drive the network equipment's behavior to exhibit self-management capabilities. This entity is called "Goal Server".
- *"Goals"* execution layer: the lower layer contains a set of autonomic network elements (AE) that behave in an autonomic manner while trying to achieve the high-level goals. The autonomic equipment is self-managed and capable of adapting it's behavior according to the context. The context of an AE corresponds to all the information about its environment i.e. its local state, remote AE states, active services, etc. AEs are able to interact in order to exchange knowledge and update their context. In real networks, these elements can be routers, switches, gateways, software,...

4 A Language to Capture Goals Description

In order to facilitate interoperability between management entities in the context of autonomous management, it is necessary to define a shared understanding of both domain information and the problem statement. Unless each autonomic entity in a system can share information with every other autonomic entity and

so contribute to an overall system awareness, the vision of autonomic communications will not really be reached. Thus, we need a common data model to represent all resources in a uniform manner [6].

To achieve this objective, we need a language to express the specification of the Goal. Instead of developing a specific language, we have chosen to extend an existing one. This extension of the language is done at the metamodel level i.e. where the language itself is defined. We have added the necessary primitives to the language to model a Goal. Our reference information model is the Common Information Model (CIM) from the DMTF [7]. In the following, we will describe how we have extended this model to fulfill our requirements. As shown in the fig.2, the Goal concept is a specialization of a CIM Class. Like the CIM Notification concept depicted in the CIM Meta Schema, the managed elements have to subscribe, via an association, to a particular Goal. This subscription is called the Goal-distribution process and is detailed in section 5. The instances of the Goal conform to the CIM-XML mapping for the CIM Classes as provided by the DMTF [8], see fig.3. The Goal structure aggregates new extensions to the CIM Schema. Every extension is a specialization of a CIM Named Element and conforms to the CIM-XML representation of the properties, methods and parameters. We have added specific Qualifiers to every new structure of the CIM Meta Schema in order to capture the additional semantics to represent a Goal. The Goal concept is in fact a grouping of elements that permit to achieve a set of management objectives in specific contexts. We have structured the Goal as an aggregation of "Goal-Behaviors" and a "Goal-Setting". The Goal-Setting determines the behavior of the Goal according to a particular context and describes the AE's Role, Location and Identifier. It is also a composition of management data locating the other peer AEs sharing the same Goal (Element-Setting) and describing the management context of the user's application services

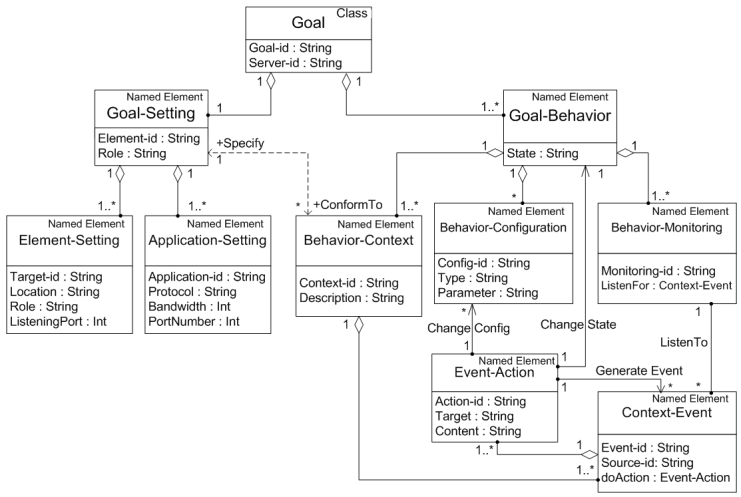


Fig. 2. Specification of the Goal Language

(Application-Setting). The Application-Setting aims to represent the user-level and the business-level requirements. This Application-Setting is specific to our case study; a more generic representation of the management context should be proposed in future work.

Every Goal-Behavior aggregates a set of Configuration, Monitoring, and Context specifications. The State is the unique identifier for the behavior of a Goal. The Behavior-Configuration of an AE corresponds to the execution of platform specific code correlated with the expected behavior. The Behavior-Monitoring implements the listeners for the exact type of Context-Events that need to be considered by the element in a particular state. Context-Event defines the occurrence of (1) low-level events collected by the sensors and provided by the platform-specific listeners, and (2) high-level events (messages) coming from goal servers and/or other AEs. The Behavior-Context part conforms to the Goal-Setting and identifies the collected events (Context-Event), and the equivalent actions (Event-Action), to achieve the Settings of the Goal in a particular context (behavior's state). A Context-Event is triggered from the monitoring module when sets of conditions are realized. These conditions are implemented in the system level of an AE by a set of activated sensors. When a Context-Event is verified, it triggers the execution of a set of Event-Actions according to the new context. An Event-Action specifies the execution of some platform-specific code and/or the emission of messages to the Goal Servers and other AEs. Here again, the internal implementation of the AE, which is specific to an execution environment, must match the description of the Event-Action. These actions can be reflexive i.e. change behavior state, or reactive i.e. change the Behavior-Configuration of the element and/or trigger changes in the behavior of other AEs. Changes in the AE's state trigger a specific re-configuration of the element, to apply the Context related to this new state.

The Behavior-Context module specification conforms to the Goal-Setting and it identifies the collected events, and the equivalent actions for realizing the settings of the Goal in a particular context (behavior's state).

The Goal representation as a Finite State Machine seems to be implicit in our case. The Goal aggregates multiple behaviors. Each behavior represents a State and is modeled by a set of configurations and policies to enforce as well as a set of sensors to activate. The transition between the states is defined by the Context according to the Setting. In simple terms, autonomic elements tailor their behavior to user- and business-requirements. The complete XML representation of the goal is presented in the fig.3. The AE is aware of the network context via the Goal-Setting. Conflicts are detected when the Context-Events, from different Goal instances, reported by the Behavior-Monitoring module do not conform to the Application-Setting of all the Goals. In this case the AE collects the Behavior-Context identifiers and sends them to the Goal Servers. The problem can be temporarily solved by assigning priorities to the goals depending on predefined policies between autonomic domains.

The Goal-language supports the communication mechanism between AEs and the Goal Servers. Every element exchanges information (knowledge) in a

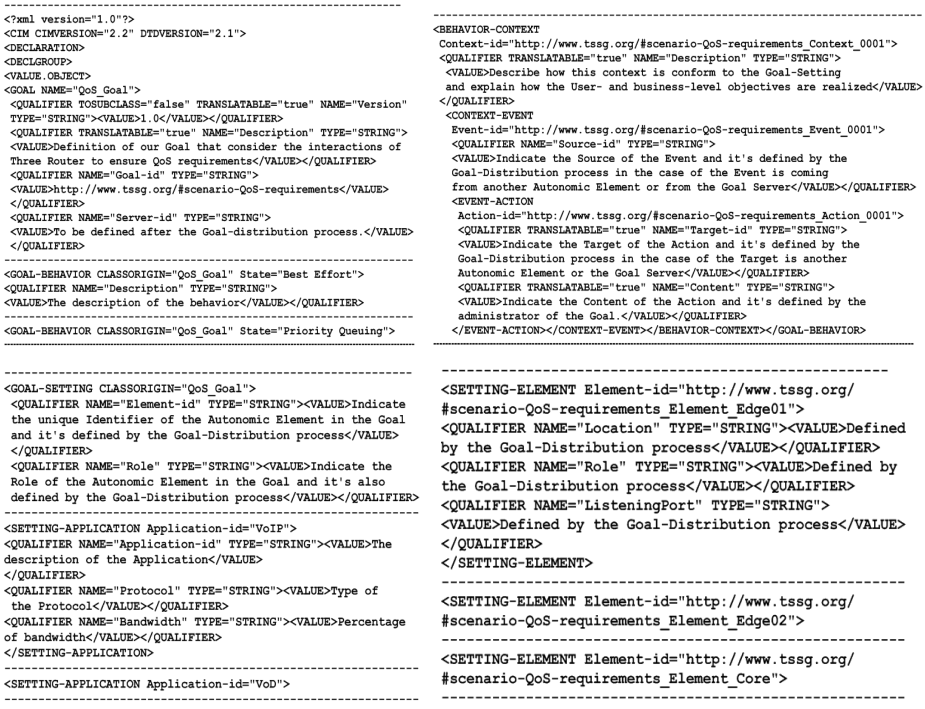


Fig. 3. Goal Representation in CIM-XML format

common specification. The Element reasons with this knowledge by applying the Goal configurations, monitoring the system events, detecting/reporting configuration conflicts and adapting its behavior (changing state).

5 Autonomic Element Architecture

The role of an AE is to satisfy the goals specified by the administrator through the Goal Server during its initialization, to interact with its peers in order to propagate knowledge, and provide a global "intelligence" for achieving the desired goal in a cooperative manner. The AEs can assign priorities to the enforced goals and reason with them i.e. loading behavior modules, configuring state and apply settings conforming to the contexts of the behavior. Therefore, the business-level and the user-level objectives are managed in a completely decentralized manner and conflicts can be solved more easily. When an AE faces a conflict, it takes it own decision depending on its knowledge and sends an event containing a Goal and Behavior-Context Identifiers, responsible of the conflict, to the Goal Servers. This event will help the administrator to understand the behavior of its network in order to enhance the specification of his goals and solve the unpredicted configurations conflicts.

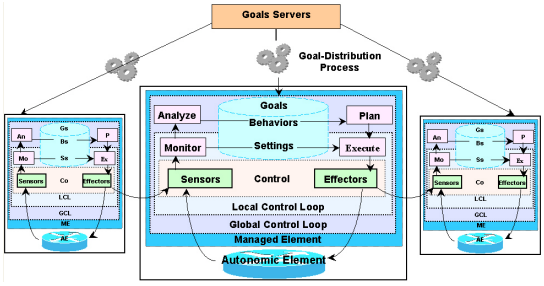


Fig. 4. Autonomic Element Architecture

As shown in fig.4, the reasoning capabilities of an AE are distributed between four functional modules that compose its internal architecture: monitoring; analyzing; planning and execution. Hence, the AE exhibits communication capabilities that allow it to interact with its environment. At initialization, the AE subscribes to autonomous domains via the Goal-distribution process. This process defines how an AE is associated to an autonomous domain and corresponding goal. A Goal can be enforced in many AEs (1-N) and an AE can be associated to several Goals (M-1). The administrator defines the set of elements that are required to achieve a Goal. This is accomplished using our defined Goal-language. Once a goal is specified, the Goal-distribution process is performed in two phases. In the first phase, every AE subscribes to a Goal via the Goal Server (GS). The role of the GS is to complete the Setting of the Goal by referencing every entity participating in its realization (autonomic domain elements). The identification process assigns a unique ID and a role to every AE (this can also be configured directly in the AE during its local configuration). The interactions between AEs are determined by their role, thus every AE is aware of its role in the autonomic domain. Once the Goal-Setting is completed, the AEs download the goal specification and use it to drive their behavior.

The internal architecture of our Autonomic Element is presented in fig.4. This architecture is aligned with the one presented in [2] and is composed of a number of functional modules that enable the expected autonomic behavior. The **Control Module** allows an AE to interact with other AEs as well as with its internal and external environment. It introduces two entities called sensors and effectors. Sensors provide mechanisms to collect events from the environment while the effectors allow the configuration of its managed resources. In our work, the Control module affects mainly the configuration of the underlying traffic engineering mechanisms of the IP router; in our test-bed, we have used the Linux Traffic Control TC tool [9]. The **Monitoring Module** provides different mechanisms to collect, aggregate, filter and manage information collected by sensors. Whereas, the **Analyze Module** performs the diagnosis of the monitoring results and detects any disruptions in the network or system resources. This information is then transformed into events. The **Planning Module** defines the set of elementary actions to perform accordingly to these events. These actions can be atomic Behavior-Configuration (e.g. QoS class modification, QoS class

creation/removing,...) or Event-Action installation (e.g. configuration actions, messages, change behavior). The **Execution Module** provides the mechanisms that control the execution of the specified set of actions, as defined in the planning module. It mainly translates the Behavior-Context into calls in the Control module.

Once the goals are specified, the workflow interaction between the different modules of the AE allows the router to behave in an autonomic manner without any human intervention. The behavior defines two levels of control over the instrumented managed resources and the AE as a whole. The local control loop of the AE (change Behavior-Configuration) allows **reactive behavior**, to situation changes in the AE, to be enforced. Another general loop, called global control loop permits to achieve a **reflexive behavior** in the AE (behavior changes) according to more important changes in the context.

6 Experimentation

We have implemented a proof-of-concept prototype of an autonomous network that exhibits self-configuring and self-optimizing behaviors in fulfilling high-level goals. The aim of our prototype is mainly to demonstrate the ability of the network to control its own behavior, without human intervention, while all the time meeting the QoS requirements of heterogeneous user applications. The supported applications are FTP, Voice over IP and MPEG Video streaming. Our deployed test-bed, as shown in the fig.5, is composed of three routers (Edge Router ER1, Core Router and Edge Router ER2), a Goal Server, an Application Server and two client terminals supporting different types of applications (FTP, VoIP, Video Streaming). In this example, we have used a simple application identification technique based on a combination of layer 3 and 4 information (Port numbers and IP Address).

The high-level goals are defined by an authorized authority using the Goal Server (GS). The goals are specified to ask autonomic routers to adapt automatically their behavior and exhibit a self-configuration and self-optimizing properties according to the applications that are running in the network and the network capacity. In our scenario, the goal specification defines three behaviors for the AEs: BE (Best Effort), PQ (Priority Queuing) and DiffServ (Differentiated Service). The goal is enforced in the autonomic router using the GS.

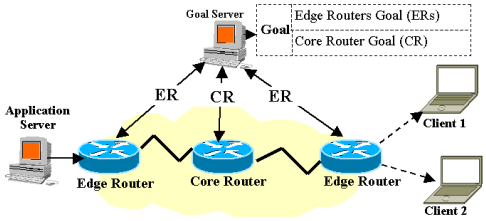


Fig. 5. Test-bed Architecture

This goal is interpreted by the autonomic router, which in turn enforces it locally i.e. execute corresponding low-level configurations, monitoring actions and enforce the corresponding context rules. These rules define the local configuration parameters of the router. More particularly, it defines the configuration of each router interface in term of scheduler, queue management, and buffering according to the context i.e. running applications streams, traffic load, etc.

At the starting of the experimentation, every router interacts with the GS to download their associated goal. The goal specification contains (1) the behavior specification of the autonomic router according to its role, (2) the applications identification and QoS requirement specification. In our experimentation, we have only two roles: Edge Router role and Core Router role. During time, an autonomic router interacts with a peer autonomic router to exchange context information, which allows him to have a global view of the network behavior and reacts immediately when any change occurs (new application launch, per class QoS degradation, etc.). The objective of this experimentation is to highlight the capability of an autonomic router to evaluate a situation and react accordingly to try to fulfill its assigned Goal. Figure 6 shows the evolution of the configuration of Classes of Service (CS) as well as the distribution of bandwidth between these classes. This evolution of the CS configuration corresponds also to a self-adaptation of the autonomic router behavior. Once the initialization phase is complete, all the routers initialize their behaviors to BE. This default behavior is motivated by the existence of only one type of application stream in the network (same priority); therefore only one class of service CS is needed (BE) to support this application. All the available bandwidth is allocated to the BE class. During time, ER1 detects the launch of a new application (VoIP application) through its sensors and using its knowledge base identifies its Settings i.e. the targeted QoS. Based on these properties, it determines the most accurate actions to adopt in order to maintain the QoS objectives (reflexive behavior). This situation is depicted in fig.6 at t=60 sec. At this instant, ER1 informs the peer Core router about this new situation (cooperative behavior) so that they cooperatively find a solution and take the most accurate actions. In this case, the cooperative decision is to adopt the "PQ" behavior, which allows to support the QoS for two classes

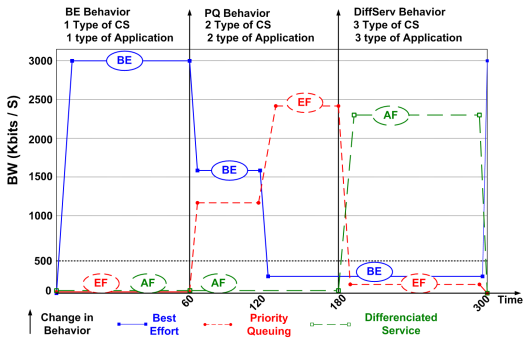


Fig. 6. Experimental Results

of application, one with low delay requirement and the other with best effort requirement. The bandwidth distribution between the two classes is controlled by a shaper, which ensures that the low priority class is not starved by the high priority class. Thus, the routers synchronously change their behaviors from BE to PQ.

ER1 continues to monitor any new application traffic using its sensors while the Core router controls the aggregated QoS for each class of service. In the case where it detects a high loss rate for the higher priority service class, it sends a notification (PQ) to the ER1 and a reconfiguration automatically occurs to redistribute the bandwidth between the two classes more efficiently. In the same figure, we can see that at t=126 sec ER1 has detected the launch of a second VoD application and immediately triggers a notification to inform the core router. The later then reconfigures itself automatically by changing the bandwidth distribution between the higher and the lower classes. At t=180 sec, a new application with the highest priority is detected. Three types of application are now running at the same time in the network. In order to maintain the QoS objective of each application, three different classes of service are necessary. The autonomic router's behavior then changes automatically from "PQ" to "Diff-Serv" and three classes of services are defined: Expedited Forwarding for VoIP; Assured Forwarding for VoD and Default Class for Best Effort traffic.

For our prototype to have a global view of the network, a number of monitoring sensors have been installed in the autonomic routers that collect information about their behavior and their context. This information is collected by a monitoring application and presents it in a useful manner to the administrator. The monitoring application presents also a topology map of the autonomic network as well as tables, statistics and graphs related to the autonomic routers' behavior, existing service classes, bandwidth occupation per class, and loss rate per service class.

The objective of this experimentation was to highlight the adaptive behavior of the routers based on the context. The tests have shown that the network effectively has achieved the enforced goal through the local behavior adaptation of AEs and their exchange of context information. Nevertheless, it is important to note here that the objective was not to highlight the benefit of having three types of scheduler in the network but rather the benefit of autonomous and coordinated behavior-adaptation, causing automatic router re-configuration based on the network context.

7 Conclusions and Perspectives

In this paper we have introduced an initial approach for introducing autonomous capabilities into IP routers. The idea behind this work is to show that it is possible to model a goal in terms of a state machine that specifies the expected behaviors from target autonomic elements in various situations. Conforming to the goal, network elements take stand-alone decisions based on their local information collected from cooperative autonomic peers. We have extended the

CIM in order to introduce the new concepts necessary to model a goal and we have specified a global architecture based on a Goal Server (GS) and Autonomic Elements (AE). We have implemented this concept in a small-scale test bed that allowed us to validate some aspects of the model and highlight the AEs autonomous behavior. The obtained results are very promising and have shown that some aspects of autonomic networks are realizable and simplify the tedious work of IP network configuration and optimization. However, this work should be seen as a first step towards the achievement of a truly autonomic network.

In the future we aim to use our model for a large scale IP network where the interactions and the behavior coordination between autonomic routers will be more complex. We will consider the case of a unique domain or multiple overlapped autonomic domains fulfilling different goals. The modeling of autonomic behavior and the introduction of cognitive and cooperative capabilities based on a generic representation of the management context are certainly the most important issues that we will address in future work.

References

1. Smirnov M., Popescu-Zeletin R.: Autonomic Communication, Communication Paradigms for 2020, Future and Emerging Technologies (FET), 22/07/2003, Brussels.
2. Kephart J. O., Chess D. : Vision of Autonomic Computing, Computer Magazine, IEEE, 2003.
3. Kephart J. O., Walsh W. E. : An Artificial Intelligence Perspective on Autonomic Computing Policies, in Proceedings of the 4th international Workshop on Policies for Distributed Systems and Networks, June 07 - 09, 2004, New York.
4. Strassner J. : Realizing on demand networking, in Proceedings of the 6th IFIP/IEEE International Conference on Management of Multimedia Networks and Services, 7th-10th September 2003, Queen's University of Belfast, Northern Ireland.
5. Bandara A. K., Lupu E. C., Moffett J. D., Russo A. : A Goal-based Approach to Policy Refinement, IEEE 5th International Workshop on Policies for Distributed Systems and Networks, POLICY 2004, June 7-9, 2004,
6. Stojanovic L., Schneider J., Maedche A., Libischer S., Studer R., Lumpp Th., Abecker A., Breiter G., Dinger J. : The role of ontologies in autonomic computing, Published in IBM Systems Journal, Volume 43, Issue 3, 2004.
7. DMTF : Common Information Model (CIM) Specification, V2.2, DSP0004, June 14, 1999.
8. DMTF : Representation of CIM in XML, v2.2, DSP0201, December 09, 2004.
9. Brown M. A. : Guide to IP Layer Network Administration with Linux, Traffic-Control-HOWTO, http://linux-ip.net/articles/Traffic-Control-HOWTO/.