# A LEARNING AND INTENTIONAL LOCAL POLICY DECISION POINT FOR DYNAMIC QOS PROVISIONING

Francine Krief and Dominique Bouthinon
*LIPN Laboratory, UMR CNRS 7030, University of Paris XIII, Avenue Jean-Baptiste Clément 99, 93430 Villetaneuse, France. {krief,db}@lipn.univ-paris13.fr*

**Abstract**: In the policy-based network management, the local policy decision point (LPDP), is used to reach a local decision. This partial decision and the original policy request are next sent to the PDP which renders a final decision. In this paper, we propose to give a real autonomy to the LPDP in term of internal decision and configuration. The LPDP is considered as a learning BDI agent that autonomously adapts the router's behavior to environment changes

**Keywords**: Policy-Based Management, Self-aware Management, Multi-Agent System, BDI architecture, Learning, Quality of Service, DiffServ mechanisms

## 1. INTRODUCTION

Today, service providers must provide the quality required by the users. In the context of fierce competition, this quality is negotiated with the customers. A contract, called SLA (Service Level Agreement), is signed between the service provider and the customer[1]. The SLA specifies the service that must be delivered. This implies differentiated treatments and software infrastructures adapted to implement them.

DiffServ is the model accepted by the network providers to allow them this services differentiation. In this model[2], the traffic is separated in traffic classes which are identified by a value coded in the IP header. DiffServ is well adapted to wide networks because the complex operations (e.g., classification, marking) are realized at the network's entry by the edge routers. The core routers only treat packets according to the class coded in the IP header[3], and an adapted behavior, the PHB (Per Hop Behavior).

The IETF proposed a general framework called PBM (Policy-Based Management)[4] for the control and management of these IP networks. This infrastructure provides a certain level of abstraction and allows the network a flexible behavior according to the various events which can occur during its management by using the policy concept.

We proposed an architecture for the self-aware management of IP networks offering quality of service guarantees by using policy-based management and multi-agent systems[5]. The level of autonomy required is reached by introducing the operational objectives and the parameters to be followed in the infrastructure, as well as by providing respective monitoring and adaptation means. The operator does not need to apply corrections and adaptations himself so much anymore. Thus, the management system is simplified and even more oriented towards the definition of policies and operational parameters.

In this paper, we propose to give a real autonomy to the network components in term of internal decision and configuration by introducing a learning and intentional agent in each network element to autonomously adapt the router's behavior to environment changes. This agent can be seen as a learning and intentional Local Policy Decision Point (LPDP).

First, we present policies, intentional agents and the global architecture proposed for the self-aware management of IP networks offering quality of service guarantees. Then we describe the architecture of a LPDP. Finally we present the future work.

## 2. POLICY APPROACH

The policies can be defined like sets of rules which are applied to the management and control of the access to the network resources[4]. They also allow network administrators or service providers to influence the network element behavior according to certain criteria such as the user's identity or the application type, the traffic required, etc.

In general, the policy rules are in the following form "IF policy_activation_condition THEN policy_action" where the condition describes when policies can be activated.

The IETF introduces the role concept. A role is a type of property that it used to select one or more policies for a set of entities and/or components from among a much larger set of available policies[6].

The policies are centralized in a data base. A Policy Decision Point (PDP) has the responsibility of dispatching the policy rules onto the network elements concerned. The Policy Enforcement Point (PEP), situated in each element, constitutes the application point of the policies.

A policy can be defined at different levels. The highest level corresponds to the business level. Then, this policy must be translated into a network level policy and, then, into a low-level which is understandable by the network element.

The Foundation for Intelligent Physical Agents (FIPA) also defines the policy concept[7]. A policy is a constraint or a set of constraints on the behavior of agents and services.

A policy rule is a conjunction of implications: when a condition holds then an action is permitted, prohibited or whatever.

Policy domains are introduced to efficiently apply policies and simplify policies mechanisms. A policy domain is a set of agents to which a given set of policies apply.

A policy library contains the policies and a distribution mechanism is used to distribute policy rules from originating authorities to mechanisms that have the ability and responsibility of applying policies.

The concept of higher-level policy is introduced to simplify the task of generating specific policy rules for agents and services.

## 3. INTENTIONAL AGENTS

An agent is a temporal persistent computational system, able to act autonomously to meet its objectives or goals when it is situated in some environment. In order to be perceived as intelligent, a software agent must exhibit a particular kind of behavior, identified by Michael Wooldridge and Nick Jennings[8] as flexible autonomous behavior and characterized by:

- reactivity: intelligent agents must be able to perceive their environment and respond at time to changes on it though its actions;
- pro-activeness: intelligent agents exhibit goal oriented behavior by taking the initiative to satisfy its design objectives;
- social ability : intelligent agents must be able to interact with other agents or humans in order to satisfy their objectives.

The study of intelligent agents has received a great deal of attention in recent years. This paper explores a particular type of intelligent agent, a BDI (Belief-Desire-Intention) agent. BDI agents have been widely used in relatively complex and dynamically changing environments[9]. They are based on the following core data structures: beliefs, desires, intentions, and plans[10]. These data structures represent respectively, information gathered from the environment, a set of tasks or goals contextual to the environment, a set of sub-goals that the agent is currently committed, and specification of how sub-goals may be achieved via primitive actions. The BDI architecture comes with the specification of how these four entities interact, and provides

a powerful basis for modeling, specifying, implementing, and verifying agent-based systems.

## 4. GLOBAL ARCHITECTURE

We proposed an architecture for the self-aware management of IP networks by using policy-based management and multi-agent systems[5]. Using this architecture, the QoS management within the framework of the DiffServ model is dynamic. It includes three levels corresponding to the three mediation components recommended by the architecture of the IST CADENUS project[11]. Moreover, monitoring functions are introduced to allow each level to adapt its behavior to the environment which it is controlling.

Each level implement their own tools of monitoring and have a meta-control level which allows it to adapt its behavior to the dynamicity of the environment it is managing (see figure 1).

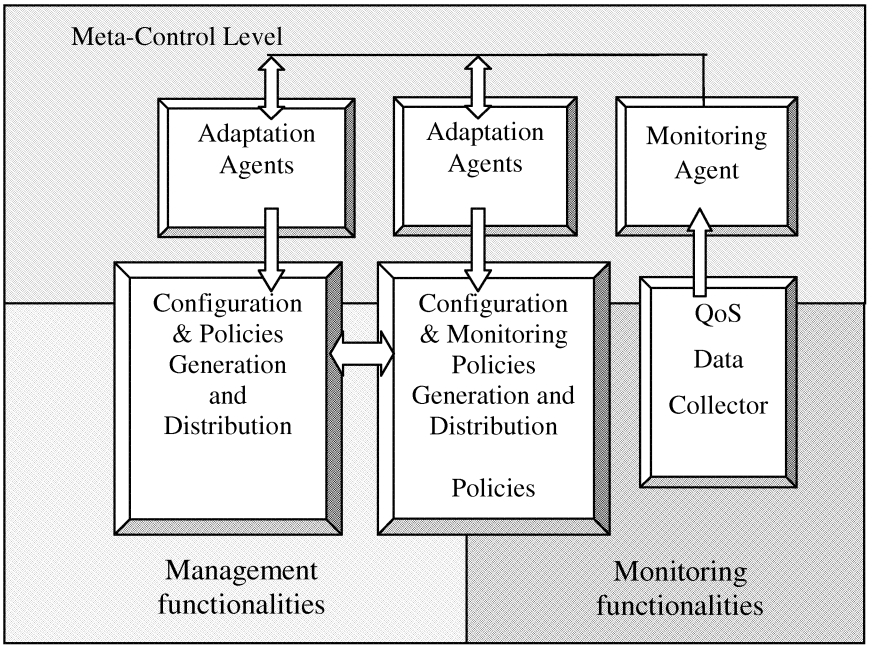



*Figure 1.* A Service/Resource Mediator

The meta-control level contains two categories of agents:

- The monitoring agent. It controls the coherence of network/network element behavior with the policies which were applied. It makes the

decision to inform the others agents or the higher level before a SLA violation;

- The adaptation agent. It modifies the Mediator/Network Element behavior in order to improve its operation and to optimize the service configuration.

## 5. THE LOCAL PDP

In the architecture proposed, PDPs (i.e. Resource Mediators) send network-level policies which are not directly executable by the network elements in order to give them more autonomy. The Provisioning Agent, situated in each network element, receives the decisions and the policy rules from the PDP (see figure 2). Then, it must translate these policy rules into policy rules/commands understandable by the PEP. Therefore, it can be seen as a local PDP.

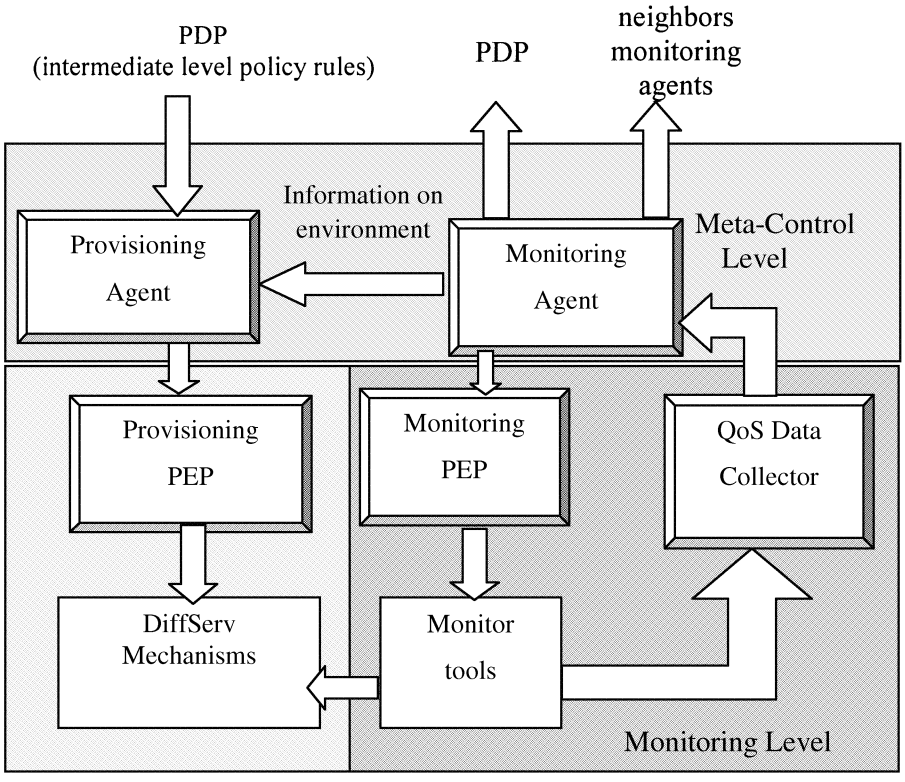



*Figure 2.* a network element

## 5.1 bdi architecture

The LPDP has a rational agent's architecture. Its rationality is turned towards the execution of a set of plans to maintain a certain QoS. Depending on the network state and the policy rules sent by the PDP, it pushes new configuration rules to the PEP. Using this architecture, the reallocation and management of network resources is based on current network state and applications QoS requirements.

The LPDP architecture is based on BDI (Beliefs Desires Intention) model and the system implemented by A. Guerra Hernandez[12]. This system is composed of four key data structures : beliefs, desires, intentions and a plan library. In addition, a queue is used to store temporarily the events perceived by the LPDP (see figure 3). Theses structures are presented in the following:

- **Beliefs** can be viewed as the informative component of the system and environment state. This component can be implemented as a set of logical expressions[12, 13]. For example, the fact that the AF queuing size reached a threshold of 70% can be represented by the statement *position (AF_queuing, 70).*
  This information comes from the monitoring agent that filters the information received from the QoS Data Collector and translates them into logical expressions. It also receives information from the neighbors monitoring agents about their state.
  Beliefs are updated by the monitoring agent and the execution of intentions.
- **Desires** are identified in our architecture as goals. **Goals** are descriptions of desired tasks or behaviors. They are provided by the PDP in the form of policy rules. An example of such policy rule is given in the following using the Ponder language[14]:

  *Inst oblig* *EFConfigurationPolicy {*
  *Subject* *DiffServManager;*
  *Target* *r = /DomainA/Routers/CoreRouters;*
  *On* *EFConfigRequest(DS,max_input_rate,min_output_rate);*
  *Do* *applyEFPHB(DS,max_input_rate,min_output_rate);}*

  EFPHB specifies the relative observable traffic characteristics (e.g., delay, loss)[2]. This policy rule is not directly executable by the node because it does not specify the particular algorithms or the mechanisms used to implement the PHB. A node may have a set of parameters that can be used to control how the packets are scheduled onto an output interface (e.g., N separate queues with settable priorities, queue lengths, round-robin weights, drop algorithm, drop preference weights and

threshold, etc): for example weighted round-robin (WRR) queue servicing or drop-preference queue management[3].

The policy rules are stored in a policy repository and analyzed by a policy conflict detection and resolution module[15]. To be understandable by the intentional LPDP, each policy rule is then translated into a logic expression such as

*Achieve (efphb_ds, ds, max_input_rate, min_output_rate)*

expressing the desire of the LPDP to associate a certain QoS corresponding to PHB type with a certain DSCP value. The LPDP interacts with its environment through its database and through the basic actions that it performs when its intentions are carried out.

The perceptions of the LPDP are mapped to events stored in a **queue**. Events can be the acquisition or removal of a belief, e.g., the reception of a message coming from the monitoring agent or the acquisition of a new goal coming from the distant PDP.

- **Intentions** are the plans that the LPDP has been chosen for execution.
- **Plan Library** is the set of predefined plans. Plans describe how certain sequences of actions may be performed to achieve given goals or react to particular situations. Each plan has an identifier, a trigger or an invocation condition, which specifies in what situations the plan is useful and applicable, a context or precondition, and a body, which describes the steps of the procedure. An example of plan is given where a Weighted Round Robin scheduling algorithm is used to satisfy the goal presented above.

*Plan-id: p012*

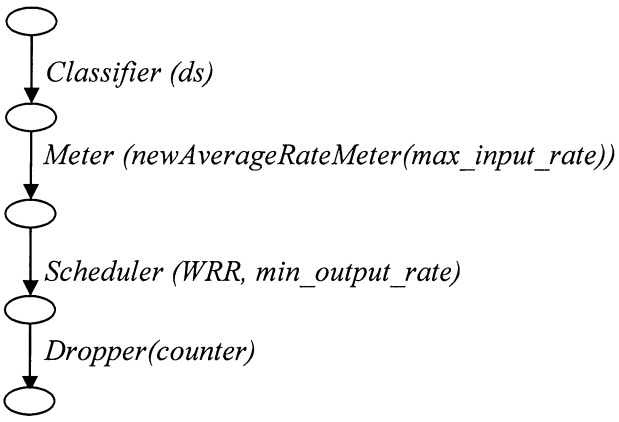
*Trigger:*

*achieve(efphb_ds, ds, max_input_rate, min_output_rate)*

*Context:*

*max_input_rate <= min_output_rate*

*Body:*



Plan body can be represented as a tree which nodes are considered as states and branches are actions or sub-goals of the LPDP. The executable plans are ordered by utility before selecting the first one. Therefore, a specific queuing or scheduling algorithm can be privileged for example.

**Actions** are of two kinds, internal and external ones. External actions are low-level policy rules directly executable by the PEP. They affect the environment where the LPDP is situated. Internal actions affect only the beliefs of the LPDP. Once a plan instance is executed, the LPDP executes a sequence of internal actions, i.e. add and delete beliefs. These internal actions are predefined for each plan in the plan library.

An **interpreter** manipulates these components by selecting appropriate plans based on the system's beliefs and goals, placing those selected on the intention structure, and executing them. It is responsible for the behavior of the LPDP. The interpreter verifies that the terms associated with an action are grounded before executing an external action. Different algorithms are proposed in literature to execute intentions. Some of them are well adapted to dynamic environment[16].

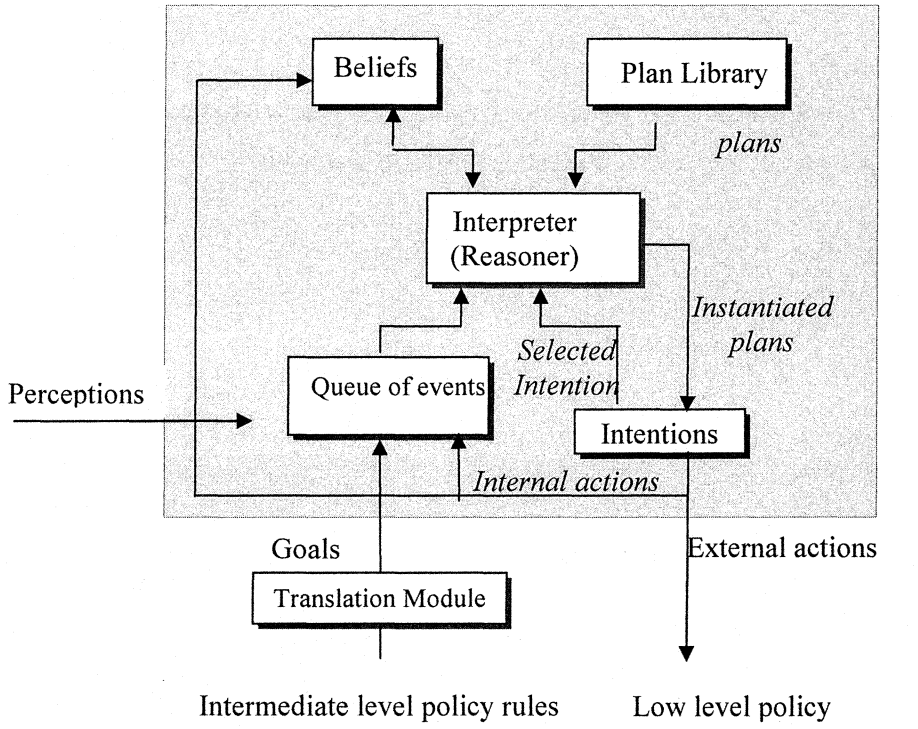



*Figure 3.* The intentional LPDP architecture

## 5.2 Learning

The success or failure in the execution of the LPDP's plans depends on many factors connected with the environment such as the input traffic type or the network load. All theses factors are difficult to well identify. Moreover, the implantation of PHBs, which is the basis for DiffServ operation, involves a hard task of choosing among a set of buffer management and scheduling techniques. This is a crucial issue to an effective QoS management. Adaptation by learning are the most suitable policy configuration strategies[14]. Therefore, the BDI architecture presented is extended with the introduction of a learning module. This module allows the LPDP to use its passed experience (i.e. plans instances that have succeed or failed in a particular context and environment) to refine the context of its plans. This context represents the reasons a LPDP has to act in a particular way. By learning the context of plans execution, it selects plans, and consequently policies which are the most suitable depending on the environment.

### 5.2.1 Learning and intentional LPDP architecture

The learning BDI architecture we propose is an extension of the intentional LPDP architecture based on supervised concept-learning (see figure 4).

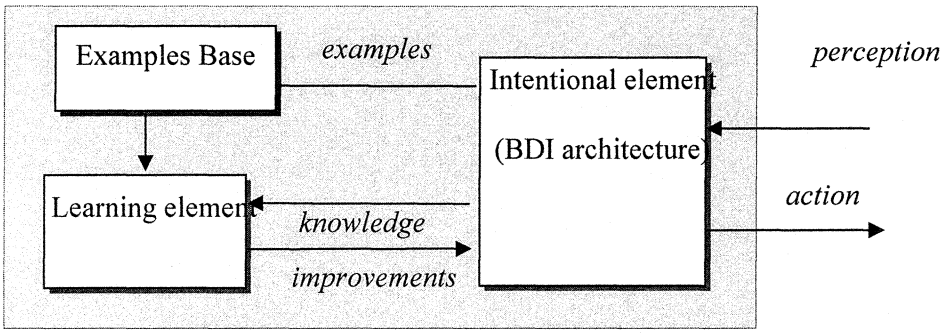



*Figure 4.* The learning and intentional LPDP architecture

The **examples base** contains labeled examples provided by the intentional element, i.e. the BDI architecture. Each example refers a plan and a set of beliefs that represents a given state of the environment, where the plan has succeeded or failed.

The **learning element** is responsible for the learning task. It acts from examples base and background knowledge provided by the intentional element. In result the learning element proposes improvements, precisely refined contexts of the plans, to the intentional element.

### 5.2.2 Learning Process

The building of the examples base precedes the learning phase. The intentional element uses the evaluation of its actions from the environment to fill the examples base with labeled examples. Each labeled example characterizes a success or a failure in the execution of a plan in a given state of the environment (i.e. a given set of beliefs). A positive example *(plan-id, e, success)* is built each time the intentional element selects and applies the plan *plan-id* in an environment *e* represented by a set of beliefs. A negative example *(plan-id, e', failure)* is built when the intentional element detects that a previously applied plan *plan-id* is no longer available in the current environment *e'*.

The learning task starts at the end of an outputs sequence of the intentional element, or ideally when the intentional element is idle and the examples base contains a sufficient number of examples. Tilde (Top-Down Induction of Logical Decision Trees) is used as learning method[12]. Tilde represents learnt concepts as decisions trees which suit the disjunctive form of the plans contexts.

At the end of the learning process the learning element gives the intentional element modified contexts of applied plans. These new learnt contexts match the environments in which these plans must be applied.

Cyclic incremental learning sessions can be activated by emptying the examples base and restarting the process described above each time the learning element produces its results. A non incremental batch learning session can also be carried out from all collected examples when the network is inactive.

### 5.2.3 Distributed learning

The learning presented in this paper is centralized. However, a distributed learning could be envisaged. When a LPDP learns something all the LPDPs having the same role could be beneficiary. All LPDPs, situated in the core network for example, have the same internal structure including goals, background knowledge and possibly competence. They also have the same procedure to select their actions. The only difference among them is their experience, i.e., their perception since they are situated differently in the environment. Thus the performance of a LPDPs group (i.e. core router LPDPs) could be improved by direct interactions, as exchanging the learnt contexts of their plans, among neighbors LPDPs having the same role.

## 6. CONCLUSION AND FUTURE WORK

We have proposed an architecture for the Self-aware management of IP networks by using policy-based management and multi-agent systems. In this architecture, the role of the administrator is limited to the guidance of these processes in their laying down operational objectives and parameters.

In this paper we propose to give a real autonomy to the LPDP in term of internal decision and configuration. The LPDP is considered as a learning BDI agent that autonomously adapts the router's behavior to environment changes.

This approach presents many advantages : The performance of the LPDP is improved because the introduction of BDI concept allows it to adapt its behavior in an autonomous manner. It chooses the policy rules according to the policy rules sent by the distant PDP. Therefore, the network element becomes relatively autonomous. Policies are distributed to all the routers that are concerned and every one of them according to its context will seek the good parameters to configure its part of the service.

By introducing learning methods, the LPDP is able to use its past actions to improve its future actions. It learns more accurately the environment in which a plan must be applied.

Our future work concerns the distributed learning and the simulation of the LPDP by adapting A. Guerra Hernandez's system[12].

## REFERENCES

1. E. Marilly, O. Martinot, S. Betgé-Brezetz, "Service Level Agreement Management: Business Requirements and Research Issues", DNAC'01, December 2001
2. S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss,"An Architecture for Differentiated Services", RFC 2475, December 1998
3. K. Nichols, S. Blake, F. Baker, D. Black, "Definition of the Differentiated Services Field in the IPv4 and IPv6 Headers", RFC 2474, December 1998.
4. R. Yavatkar, D. Pendarakis, R. Guerin, "A Framework for Policy Based Admission Control", RFC 2753, January 2000.
5. F. Krief, "Self-aware management of IP networks with QoS guarantees", International Journal of Network Management, Vol.14, Issue 5, September-October 2004
6. B. Moore and al. : "Policy Core Information Model (PCIM)"- IETF- RFC 3060.
7. FIPA, "Fipa Policies and Domains specification", Document number PC00089D, August 2001.
8. M. Wooldridge and N.R. Jennings. "Intelligent agents: Theory and practice. The Knowledge Engineering Review, 10(2): 115-152, 1995.
9. C.Olivia, C.-F. Chang, C. F. Enguix, A. K. Ghose, "Case-Based BDI Agents: an Effective Approach for Intelligent Search on the World Wide Web", AAAI Symposium on Intelligent Agents, Stanford, CA., USA, 1999.
10. Rao A., Georgeff M., "BDI Agents:From Theory to Practice", Tech. Note 56, 1995
11. CADENUS Project. QoS Control in SLA Networks. IST-1999-11017, March 2001.

12 Alejandro Guerra Hernandez, "Learning in intentional BDI Multi-Agent Systems", PHD thesis, University of Paris 13, December 2003
13 M. Ljungberg, A. Lucas, "The OASIS Air Traffic Management System", PRICAI'92, Seoul, Koea, 1992
14 J.C. Strassner, " Policy-based network management: Solutions for the next generation", Morgan Kaufmann, 2003
15 L. Lymberopoulos, E. Lupu and M. Sloman, "An Adaptative Policy Based Management Framework for Differentiated Services Networks", Policy 2002
16 M. Wooldridge. "Reasoning about rational agents". MIT Press, Cambridge, MA., USA, 2000.