

AGENT-BASED POLICY-ENABLED NETWORK MANAGEMENT ARCHITECTURE FOR MOBILE AD HOC NETWORKS

Ritu Chadha, Cho-Yu Jason Chiang, Mike Little, Sunil Samtani

Applied Research, Telcordia Technologies
445 South Street, Morristown, NJ 07960
{chadha, chiang, little, ssamtani}@research.telcordia.com

ABSTRACT*

We describe in this paper a novel design of a network management architecture that incorporates both agent and policy technologies. The conceptual model, main components, and various transports employed in this architecture are discussed. We believe that this network management architecture is highly scalable and capable of adapting network protocols and services on-the-fly with the goal of optimizing the efficiency of operations.

INTRODUCTION

New warfighting paradigms, such as Joint Vision 2010, emphasize mobile, flexible networks that automatically adapt to the warfighter's needs. These paradigms, as expressed in the JTRS ORD, require mobile networking capabilities significantly beyond what is possible with currently fielded technology. JTRS networking protocols must support a variety of services, including automatic neighbor and link quality discovery, automatic network reconfiguration, quality-of-service guarantees, precedence and priority marking, and the automatic routing and relaying of traffic. The development and implementation of mobile networking protocols and agile management capabilities is a technical challenge for the JTRS, FCS and WIN-T efforts. Current military networking technology predominantly supports static, pre-planned legacy networks with only modest capabilities for automated and adaptive (re-)configuration and monitoring of network facilities (e.g. nodes, servers and gateways) and their roles and relationships. In the future, military ad hoc networking calls for flexible composition and on-the-fly operational adjustments to adapt to unpredictable conditions and mission demands in tactical environments.

This paper describes a network management architecture designed under the CECOM DRAMA [2] program. It employs concepts such as policy-enabled [3][4][5][6][7][8] management and agent-based [9][10] technologies to support automatic network

configuration and reconfiguration, fault detection and correction, performance management, and security for Future Combat Systems network environments. It is generic enough to manage a wide range of network elements.

CONCEPTUAL FRAMEWORK

A conceptual framework for an agent-based management system is shown in Figure 1. Such a system is notionally composed of one or more operating environments (e.g. computers) hosting management agents and managed objects. This framework additionally identifies an agent execution environment, which is interposed between the operating environment and the management agents. The agent execution environment provides agent hosting services – inter-agent communications, agent life-cycle management, mobility (if supported), and the like. Managed objects represent any environmental component that may be directly accessed by a management agent to accomplish a management activity.

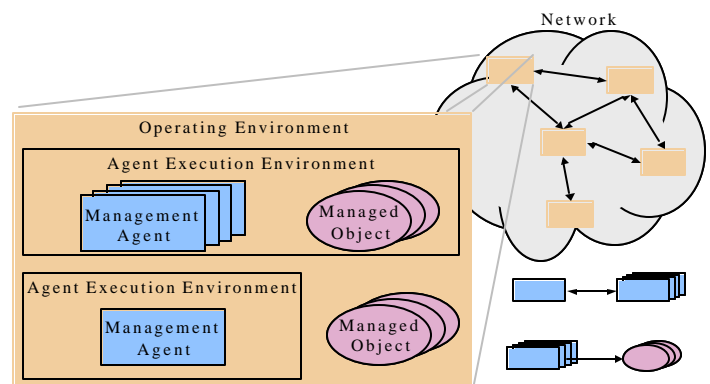


Figure 1 Foundational Conceptual Framework

A conceptual framework for policy-directed systems has a somewhat different composition (see Figure 2). A policy-directed system operates under a set of guidelines (policies) that constrain its behavior relative to its current state and perceived operating environment. Such systems are composed of functional components for (1) sensing environmental conditions, (2) specifying and interpreting policies, (3) distributing and mediating the resultant behavioral constraints, and (4) accommodating the

* This work was supported in part by the U.S. Army Communications and Electronics Command (CECOM) under Contract DAAB07-01-D-G002, 003.

behavioral constraints. There is no requirement that an architecture or its realization follow the functional decomposition illustrated, only that all of the functions be somehow accounted for.

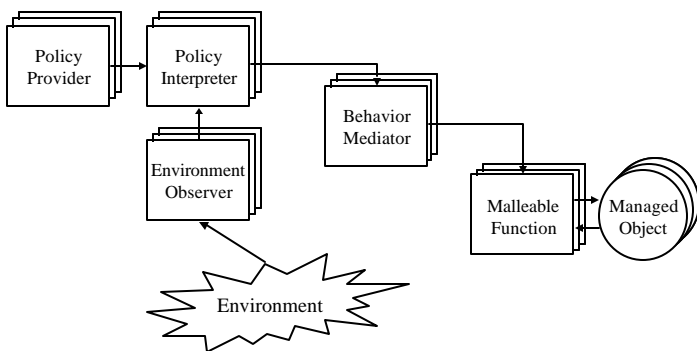


Figure 2 Functional View of Distributed Policy Directed Management System

DESIGN

In this section, we describe a policy-driven, agent-based network management system that is designed to manage a tactical communications environment. The functional architecture is envisioned to provide the following benefits:

- Ability to rapidly provision, configure and maintain (re-provision and re-configure) a heterogeneous agile battlefield communications environment based on the changing mission requirements.
- Network Management bandwidth utilization efficiency through intelligent information collection and dissemination.
- Rapid and autonomous response to network fault and performance problems.
- Ability to tailor management monitoring capabilities and resource demands to node characteristics, resource availability, and networking environment.

A view of the overall design approach is shown in Figure 3. A Policy Manager Agent directs a collection of Management Agents that in turn interact with managed objects. Policies may be stored in a Policy Repository and agents report network conditions and management events. Policies, directives, and events are reported and distributed via protocols and facilities for these purposes.

A Managed Element is a computing environment having one or more managed objects and hosts one or more agents interacting with these managed objects. Managed Objects represent software and hardware components of a managed element. These components can be processes of the operating system (e.g. file system, protocol stacks), services (e.g. web server, database), applications (e.g. situation awareness), hardware sub-assemblies (e.g. GPS),

or the like. In a given context, management processes such as the Management Agent could also be considered as managed objects. Given that the overall networking environment for Future Combat Systems is heterogeneous, the capabilities of the Management Agents are also expected to be heterogeneous.

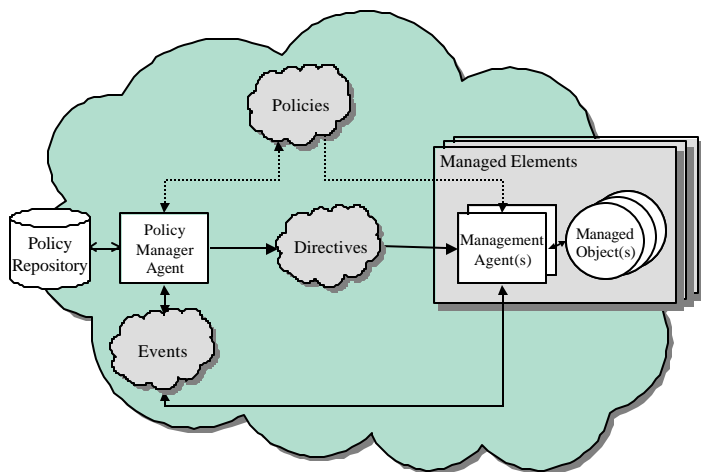


Figure 3 Preliminary Architecture Overview

We envision that depending upon the size of the network environment being managed, there may exist multiple Policy Managers organized in distributed fashion for both scalability and management scope setting purposes. Figure 4 shows one possible management system organization as a hierarchical multi-domain architecture. Policy Managers organized in this manner are capable of interacting in a non-conflicting fashion from the perspective of providing policy distribution and management direction. This architectural assumption allows for management mappings to command structures and provides reasonable scalability (thousands of nodes) while reducing and/or eliminating the need for resource-intensive (bandwidth and computational) coordination and synchronization mechanisms providing duplication removal and conflict resolution. A policy domain essentially represents the scope of influence for a set of policies. The management scope of a policy domain is defined to include all agents and their associated managed objects that incorporate policies directed by a Policy Manager of the policy domain.

A central component of any policy domain is the Policy Manager. In Figure 4, we show a single Global Policy Manager providing policies to two Policy Managers that each manage different policy domains. Each domain-specific Policy Manager in turn may store the received policies within its policy repository and operate on these policies. One or more of the Management Agents in each domain could themselves be Policy Managers, providing additional policy domains. Thus any domain can be hierarchically extended through recursive formulation.

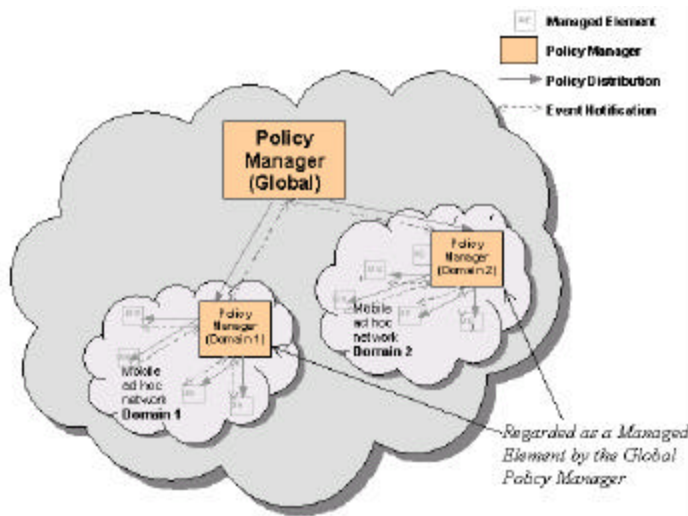


Figure 4 Multi-Domain System Architecture View

With respect to event distribution, a Policy Manager and its Management Agents can pass events to each other bi-directionally, as shown in Figure 3. These events could trigger management actions based on the policies an agent is implementing. These actions themselves may trigger the generation of more events to be consumed by the entities interested in receiving those events.

POLICY MANAGER

A Policy Manager (PM) is a logical entity that oversees policy implementation. Each Policy Manager has its own management scope. For example, there may be a global Policy Manager that oversees the implementation of the policies having global influence while several domain Policy Managers may be in charge of the implementation of those policies only applicable to the domain in which they reside. A Policy Manager performs two major functions: distribution of policies and implementation of policies. With respect to the distribution of policies there are two channels by which a Policy Manager may receive policies. Policies could be entered from an associated Policy Manager GUI by human policy administrators, or delivered to it from a Policy Manager elsewhere (e.g. one or more levels up in the hierarchy). Either the Policy Manager or the GUI itself validates the policies it receives and saves them to the local policy repository.

In order to implement policies, a Policy Manager listens for events of interest, where an event is of interest if it is relevant to an active Policy Manager policy. Once such an event arrives, the Policy Manager examines its active policy set to determine which policy rule(s) would need to be enforced as a result of this event. If the conditions associated with such rules evaluate to true, the actions associated with the rules are executed.

A Policy Manager is viewed as a collection of functions, all of which can be realized as a single Policy Manager Agent or created as distinct agents operating together to perform the role of the Policy Manager. Figure 5 illustrates the functional organization of the Policy Manager.

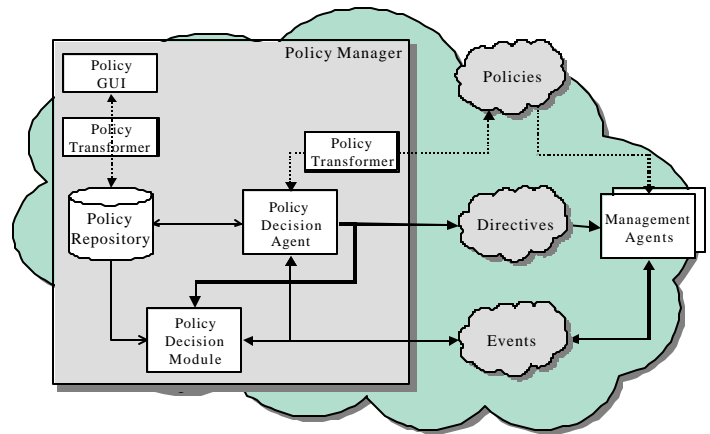


Figure 5 Policy Manager Component Architecture

Policy Decision Agent

The Policy Decision Agent is responsible for implementing policies. Once valid policies are installed, the Policy Decision Agent accesses the policy repository to retrieve policies as needed. It evaluates the prerequisite conditions for enforcing the policies. If these conditions are true, the Policy Decision Agent launches the actions associated with the policies. To enhance extensibility and to promote modular design, the Policy Decision Agent may coordinate with one or more Policy Decision Modules. These adjunct modules can handle policy decision processing for specific functional requirements.

Policy Repository

The Policy Repository (PR) provides a persistent policy store for policies known by the policy manager. These policies may or may not be currently "active" with regard to the policy decision process. For Policy Managers with distributed components, the PR also provides secure access.

Policy Decision Module

A Policy Decision Module (PDM) is a separable component that assists the policy decision-making process. One or more Policy Decision Modules could provide modular and/or function-specific decision services for a Policy Decision Agent. The Policy Decision Agent can send directives to a particular Policy Decision Module to perform various computations. Depending on the results of the PDM decision processing, directives could be issued directly and/or relevant events could be generated and reported.

Policy Manager GUI

The Policy Manager GUI (PM-GUI) enables human policy administrators to create and manage stored policies. Policy creation can be a tedious and exacting process that may need to consider operational impacts, existing policies, mission requirements, and knowledge of events and managed objects. The PM-GUI can provide automated support to ease this task. It also can provide administrators the ability to modify existing policies, remove policies, or to activate or disable specific known policies.

Policy Transformer

The Policy Transformer component translates policies from one policy representation form to another. For example, policies abstracted as high-level mission goals obtained from the Policy Manager GUI may need to be transformed to an appropriate representation that is compliant with the form adopted by the policy repository and policy decision agent. Alternatively, policies distributed to management agents may need to be transformed into a format appropriate for that management agent. The Policy Transformer can be generalized in terms of the transformation capability – it transforms the policies obtained from a policy producer (e.g. Policy Manager GUI) to an appropriate representation that is conformant to the format adopted by the policy consumer.

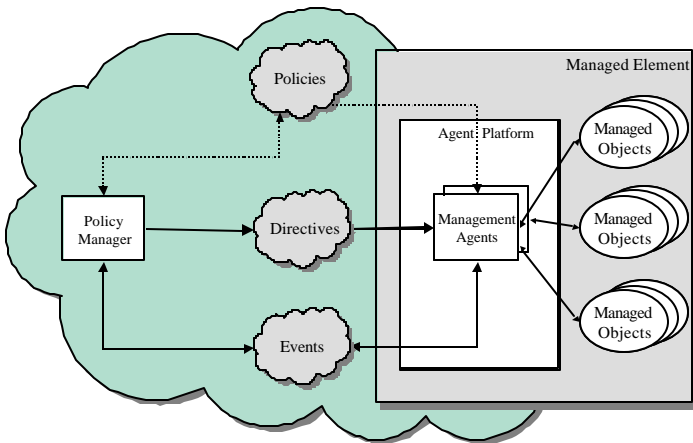


Figure 6 Managed Element Component Architecture

MANAGED ELEMENTS

A Managed Element is notionally a network element that has one or more Managed Objects and is capable of hosting Management Agents (see Figure 6). Management Agents are responsible for all management activities not performed by the Policy Manager. This includes activities such as interacting with Managed Objects, coordinating element-specific policies, aggregating and distributing network event information, and performing element configuration. A Managed Element is capable of hosting

Management Agents by employing an Agent Platform component.

Agent Platform

The Agent Platform minimally provides communications facilities and lifecycle services to the Management Agent. It may also provide convenient common infrastructure services for Management Agents as well as services facilitating interaction with managed objects. An agent platform on the managed element providing an agent execution environment that is independent of the underlying operating system and hardware allows the capabilities provided by the management agents to be easily translated across heterogeneous platforms. Mobile agent technologies allow code modules to be transported on-the-fly from one node to the other, carry necessary state information, and resume their execution on the destination node. The major advantages of using such a mobile agent platform are:

- *Upgrading existing software agents:* Facilitates addition of new policies and updates of existing policies that require customized code modules for the associated actions.
- *Adding new functionality without preloading software:* Allows multiple programs to share limited memory resources in embedded systems. Code modules can be shipped on demand to a managed element when the specific functionality is needed and can be removed immediately after they have completed their mission.

Management Agents

Management agents perform a wide variety of management functions, not all of which are directly related to interacting with managed objects. Below are some examples of different types of management agents that may exist:

Intelligent Monitoring Agents: Such agents perform context sensitive aggregation and compression operations on the network event information derived from monitoring managed objects and/or produced by other management agents. Whether or not aggregation and/or compression are performed depends upon the currently active monitoring policies. These policies would define the event(s) of interest, the type of compression and/or aggregation to perform, and the circumstances (time of day, event value, event quantity, etc) that must exist. For example, multiple occurrences of the same event within a short period of time can be condensed into a single event under a policy that dictates that duplicates are to be discarded when closely spaced. An alternative policy could direct a similar single event to be reported, which is to include the number of times the event actually occurred and when it first occurred.

Policy Management Agents: Examples of agents that can be constructed specific to policy management are policy enforcement, policy audit and policy synchronizing agents. A Policy Enforcement Agent is responsible for implementing and executing policies local to the managed element. Policy Auditing Agents could be implemented as mobile agents and dispatched to a managed element to verify if the policies stored on the Managed Elements are in concordance with domain or global policies. If a policy inconsistency is detected, a Policy Synchronizing agent can be dispatched to the Managed Element to correct the situation.

Incarnation Manager Agents: A management agent needs to interface with non-agent based software components to fulfill monitoring, configuration and reporting functions. An Incarnation Manager Agent manages an “agent incarnation instance” for non agent-based software components. The incarnation instance works like an “adaptor” to bridge the agent-enabled platform with the corresponding non agent-based software components. This allows us to present a consistent view of the agent-based management framework.

Each non agent-based software component would have a corresponding incarnation instance running in the agent execution environment. The incarnation instances function as agent wrappers, which enable us to obtain a consistent view of the agent-based management architecture even when there exist non agent-based software components on the Managed Elements. In addition, the Management Interface Wrapper is introduced for the software components that have not implemented the required management interface.

There are several advantages to this approach, which are summarized below:

- It shields the non agent-based software components from being exposed to the local policy-triggered directives that affect these components. It also reduces necessary code changes to the components to a minimum and simplifies the task of integrating any non agent-based software components.
- It shields the Policy Enforcement Agent from being aware of the existence of the non agent-based software components, which (1) promotes a cleaner design, (2) allows the co-existence of agent and non agent-based software components on the same Managed Elements, and (3) presents a consistent view to the external components that need to communicate with the Managed Element for any management purpose.
- It facilitates coding of non agent-based modules. The consistent agent view can be achieved with assistance from the Incarnation Manager Agent.

SCENARIOS

Two scenarios are presented to help illustrate the functionality of our management system. These capabilities are useful in most tactical ad hoc networking environments.

Scenario 1 illustrates automated reconfiguration of network resources in response to a network fault by re-negotiating QoS assurances on behalf of affected flows with a Bandwidth Broker. The goal of this scenario is to salvage as many flows as possible in the event of a network fault based on the high level QoS policies for a particular user. The Bandwidth Broker has a snapshot of the network topology and the current route between a source and a destination. This route information, current utilization on this network path and the flow QoS requirements are input parameters into an admission control algorithm to decide whether to admit or reject a flow. As faults occur on the network path (for example, an interface on a network element goes down), the monitoring agents running on the network element detect this fault and report it via YAP [1] to the network management agent. The agents running on the network element would ascertain that this event is indeed a fault (more than an alarm caused by an interface status toggle).

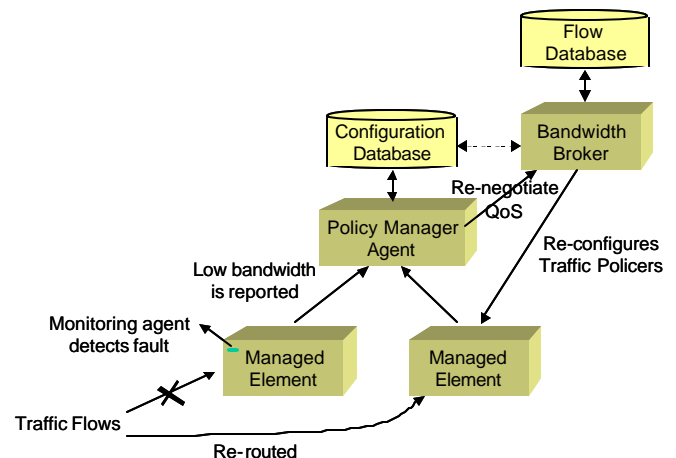


Figure 7 Example Management Scenario – 1

The Policy Manager agent finds the flows that are affected by this fault. It checks and interprets the policies stored for a particular user (for example, if the flow has high priority traffic, then re-negotiate flow QoS with the Bandwidth Broker). If the policy requires maintaining QoS assurances for a particular flow, the management agent communicates with the Bandwidth Broker to re-admit the flow on the re-routed path at a certain QoS level. If the Bandwidth Broker re-admits the flow, it will install the appropriate policers on the network element. Also, it will remove policers that are no longer appropriate by performing cleanup

operations and will re-claim unused bandwidth on stable links downstream that could now be used for other flows. This scenario a) maintains QoS assurances in the network for relevant users, b) improves network bandwidth utilization by updating state information in the Bandwidth Broker and c) increases the probability of flows being admitted in the network.

Scenario 2 illustrates on how proactive node management agents can be used to provide oversight without involving a global management agent. The capabilities include, for example, saying:

“This node can be a DNS server with priority 0; SIP server with priority 1; etc.”

A low priority may reflect that a node is a mobile node or it is running on battery power. Current state of the art allows these capabilities to be pre-defined. However, in a more dynamic environment, where nodes are constantly on the move, capabilities need to be re-configured due to changes in the environment. Changing roles can be based on battery power, signal strength, hardware, mobility pattern, etc. Capabilities that are important in a certain environment may not be relevant in others (e.g. battery power may be important at night but not during the daytime for solar-powered nodes); such constraints can be captured as local policies on a node, and executed by the configuration agents running at these nodes.

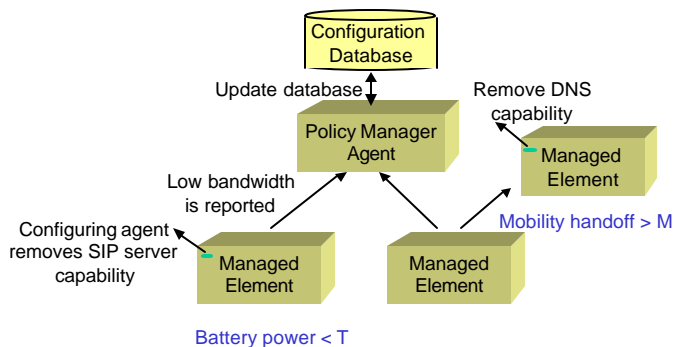


Figure 8 Example Management Scenario – 2

It may be most efficient and effective to distribute the relevant management agent policy rules to local agents, each having its scope of operations. This distributed decision-making scenario allows for lower bandwidth overhead and faster decision-making. The scenario also illustrates dynamic and automatic alteration of node capabilities based on network events. For example, in Figure 8, we demonstrate that when the battery power of a laptop falls below a certain threshold, the configuration agent reduces the capability set on this node to exclude the SIP server capability. Similarly, if a node is highly mobile as determined by the number of mobility handoffs, the configuration agent will find it unfit to serve as a DNS

server. When these changes are made locally, the management agent need only be informed of the result. Such an approach can reduce management reaction time and the load on the network since data can be used locally to re-configure the network appropriately.

CONCLUSION[†]

We have presented a novel design of a network management architecture for managing mobile ad hoc networks in this paper. By incorporating both agent and policy technologies, this architecture enables an efficient, agile, and extensible approach to network management desirable for network-centric warfighting environments.

REFERENCES

- [1] A. McAuley, C. J. Chiang, D. Chee, and L. Wong, “Generic Protocol for Network Management Data Collection”, Milcom 2003.
- [2] R. Chadha, C. J. Chiang, M. Little and S. Samtani, “Preliminary Design of an Agent-based Management System for Ad Hoc Networking Environments”, Drama Deliverable Task 5 for CECOM, December, 2002.
- [3] R. Chadha, G. Lapiotis, S. Wright, “Policy-Based Networking”, IEEE Network special issue, March/April 2002, Vol. 16 No. 2, guest editors.
- [4] J. Saperia, “IETF Wrangles over Policy Definitions”, Network Computing, January 2002, p. 36.
- [5] IETF Policy Framework Working Group, <http://www.ietf.org/html.charters/policy-charter.html>.
- [6] Randeep Bhatia, Jorge Lobo, Madhur Kohli, “Policy Evaluation for Network Management”, Proceedings of INFOCOM’00, pp. 1107-1116, 2000.
- [7] R. Chadha et al., “Policy Framework MPLS Information Model for QoS and TE”, <http://search.ietf.org/internet-drafts/draft-chadha-policy-mpls-te-01.txt>, December 2000.
- [8] J. Moffett, M. Sloman, “Policy Hierarchies for Distributed Systems Management”, IEEE JSAC, Vol 11, No. 9, Dec. 1993.
- [9] C. Stergiou, G. Arys, “Policy Based Agent Management using Conversation Patterns”, AGENTS’01, May 28-June 1, 2001, Montréal, Québec, Canada.
- [10] C. A. Iglesias, M. Garijo, and J. C. Gonzalez. “A survey of agent-oriented methodologies”, In Intelligent Agent V, Proc. of ATAL-98, LNAI 1555, Springer, 1999, pp. 317-330.

[†] The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of CECOM or the U.S. Government.