A Policy Language for the Management of Distributed Agents

Naranker Dulay, Nicodemos Damianou, Emil Lupu, and Morris Sloman

Department of Computing, Imperial College,
180 Queensgate, London SW7 2BZ, United Kingdom
{nd, ncd, e.c.lupu, mss}@doc.ic.ac.uk

Abstract. A key issue in managing distributed agents is the provision of effective policy-based frameworks. To help realise such frameworks we have developed a new policy language that features dynamic fine-grained access controls and event-triggered condition-action rules, with abstractions for grouping objects/agents (domains), and grouping policies (roles, relationships and management structures). In our language policies apply to domains of objects. By changing a policy we change the behaviour of a system. By adding an object or agent to a domain we cause the domain’s policies to be applied to the newly added object. The language is declarative, strongly typed and object-oriented, which makes the language flexible, extensible and adaptable to a wide range of management requirements

1 Introduction

The growth of computer networks and the ever-growing integration of diverse applications and technologies make the task of managing complex distributed agent systems ever more challenging. Distributed systems are increasingly dependent on interaction with other distributed systems, as well as with users and services on the Internet. At the same time distributed agent systems are often required to continue to support legacy applications built with dated tools and technologies.

A promising approach is the use of policy-based frameworks to provide effective management support and for specifying security. Policies are rules that define a choice in the behaviour of a system. Separating policies from the implementation of a system permits policies to be modified in order to dynamically change the strategy for managing the system—and hence modify the behaviour of a system without changing its underlying implementation.

Policy models and implementations are emerging in the research community, what is lacking is a unified approach to expressing policy. There is a need to express and enforce both management and security policies for heterogeneous distributed systems in a single language and to be able to communicate policies between systems administrators and across administrative boundaries. This paper describes PONDER, a declarative language that provides a uniform means for specifying policies that map onto various access control implementation mechanisms such as firewalls, operating systems, databases and Java. Equally importantly, PONDER provides a uniform means for dealing with events that occur in a distributed system such as those related to
faults, auditing, backups, security violations, resource consumption, performance degradation, QoS. PONDER aims to be easy to use and to facilitate improved management interfaces and tools.

The next section outlines our requirements for a distributed systems policy language. Section 3 presents the PONDER language and shows how it can be used to specify policies for distributed systems. Section 4 presents a small example, while section 5 covers related work on policy languages. Section 6 presents our conclusions.

2 Requirements

Many requirements need to be satisfied in order to manage complex distributed systems. This paper does not address all such requirements, but focuses on the requirement of developing a policy-driven language for distributed systems. We take the view that building a framework that supports policy specification and enforcement can significantly enhance the management of any distributed system, but particularly large heterogeneous systems.

We identify the following requirements for policy-driven distributed systems:

• the ability to specify and enforce the policies for a distributed system separately from the implementation of the objects within the system (separation of concerns);
• the ability to change policies and hence change the behaviour of a running system without having to re-implement the objects within the system (evolvability);
• the ability to apply policies to distributed, hierarchical and heterogeneous groups of objects (domains);
• the ability to group policies relating to a single entity (roles);
• the ability to define and group policies between roles (relationships);
• the ability to group roles and relationships (management structures);
• the ability to define what actions an object is permitted or forbidden to perform on available resources, or on other objects (authorisation policies);
• the ability to define the management actions that need to be performed periodically, when triggered by events or simply need to be performed depending on dynamic factors such as the current state of an application (obligation policies);
• the ability to define constraints on a set of policies (meta policies);
• the ability to handle policy conflicts (conflict prevention, detection, resolution);
• the ability to treat policies and domains as objects and subject them to policy control (self management).

The following section gives an overview of the PONDER policy specification language that we have designed to address the policy requirements outlined above. PONDER is declarative, strongly typed and object-oriented (see [5][6] for details on the type system), which makes the language flexible, extensible and adaptable for management of agent communities. PONDER offers a high degree of customisation by supporting parameterisation of any parts of a specification. The language assumes a