A XML based Policy-Driven Management Information Service

R. Natarajan, A.P. Mathur
Software Engineering Research Center
1398 Department of Computer Sciences
Purdue University, West Lafayette, IN 47907
USA
{nrk, apm}@cs.purdue.edu

P. McKee
BT, Adastral Park
Martlesham Heath
Ipswich, IP5 3RE
UK
paul.mckee@bt.com

Abstract

This paper presents the design and architecture of a prototype implementation of a XML based policy-driven management information server. It also describes the usage of such a server in building a flexible, extensible architecture for managing heterogeneous distributed systems.

1. Introduction

Due to the rapid growth in the number and flexibility of services in current networks and integration across organizational boundaries, present day and evolving distributed systems tend to be highly heterogeneous and dynamic. A management solution to such systems must possess the following attributes:

- Ability to devolve authority to the lowest possible level, to handle the sheer scale of Internet based distributed systems.
- A vendor, implementation and operating system neutral information model for resource management to facilitate interchange of management information. Such an information model should be capable adequately representing any entity of the system and is a must given the heterogeneity and dynamic nature of emerging distributed systems.
- Ability to be tailored at run-time to meet new requirements, either in terms of providing additional functionality or adapting to environmental changes. This ability is required to handle the dynamic nature of evolving network systems.
- Ability to interact and cooperate with other management solutions, in order to deal with inter-organizational integration and interaction.

We describe a prototype implementation of a management information server that can be used as a building block in creating a management system that exhibits the above properties.

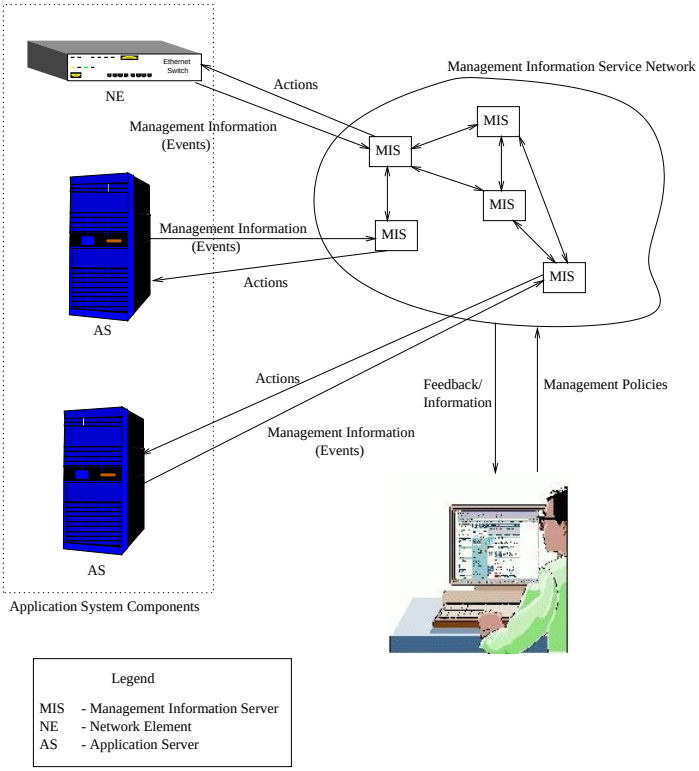

Figure 1: Architecture of a Management System built around the Management Information Server

2. Architecture of a Management Information Server

In our context, management information consists of events and management policies. Events are generated by application system components such as software objects, servers, smart appliances, network elements and the like, and typically provide operational data about the application system. Policies are generated by management systems, and are an expression of directives from the management that drive the application system towards meeting management specified goals. At the information server level, policies would constitute a set of action-event pairs that express the management goal. These lower-level policies would be hierarchically derived from a set of higher-level policies that describe the management goal in progressively more abstract terms.

Our management information server is a management system component that receives events (management information) from application system components and acts on them based on specified policies. Thus, the management information server

can be viewed as a generic extension of the basic unit of a publish and subscribe service. The type of actions that a Management Information Server can perform on the information it receives can range from simple subscription based forwarding to sophisticated aggregation, filtering and logging of information. Apart from this, custom extensions can be provided to take specific actions based on recognized information. Figure 1 shows the architecture of our management system and where the proposed management information server fits in such an architecture.

One sample scenario of usage for this management information server would be in managing a diverse set of web servers capable of externalizing events in some form. In such a case, we can convert the externalized events into XML fragments which will form management information input to our management information servers. The manager can now specify policies that take different actions, either on the application system's components or otherwise, based on events. As an example, a policy to divert traffic or alert administrators in the presence of high loads can be enforced on a web server that is capable of providing traffic events per unit time.

The policies and management information in our system are specified in XML. We use a generic template for all incoming communications to the server. This consists of a standard template for a message that is divided into a header and body. The header serves as a wrapper that contains essential information such as message type, origin, destination, authorization information (if needed), timestamp etc.

Using the header, the server identifies the message and passes it on to appropriate subsystems for further processing. The body of a message also has a generic template that is determined by the contents of the message header. For instance, an event message would have a body that contains event type, event priority, timestamp, event text and other such event related elements.

Similarly a policy message would have a body that contains policy user (the intended subsystem that should use this policy), policy data (the actual policy contents), activation timestamp, active interval and other policy related elements. Each of these elements themselves can be XML fragments that can be further interpreted. For instance, the policy data can be a XML fragment that specifies the event template and the actions to be taken when events matching the templates are detected. Due to the self-descriptive nature of XML documents, we believe that this model will lead to easy extension of the generic templates to match any particular representation of events and policies chosen by the user.

Figure 2 shows the internal architecture of a management information server. The operation of each functional block (sub-system) in the architecture can be independently controlled through management policies. The server prototype has been built in Java, and components of a single information server themselves are capable of being distributed over a network or run on a single machine.

The router component provides the flexibility and extensibility of the system, by identifying the generic templates and classifying messages based on this, passing them on to the relevant subsystem. The policy handler and policy registrar, handle and store policies respectively. The event handler uses policy registrar services to determine actions to be taken on receiving events from the application system

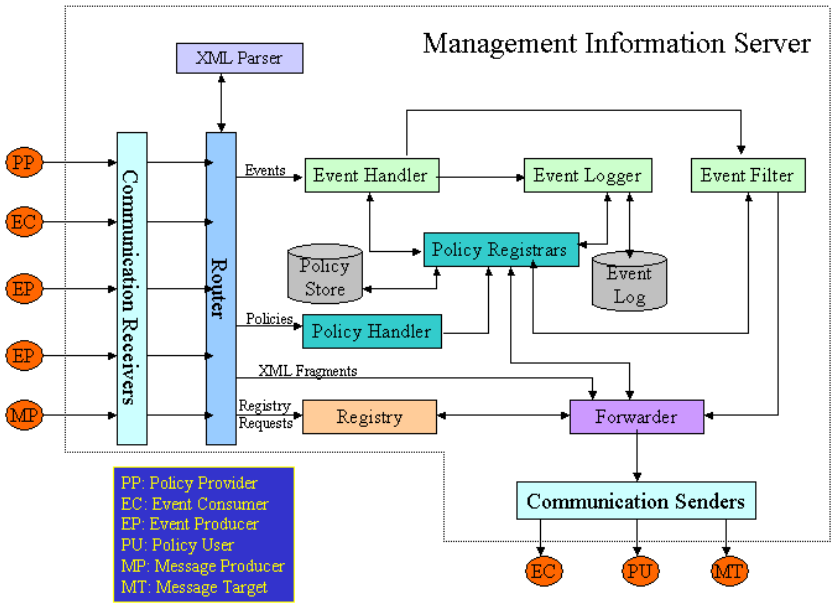

Figure 2: Architecture of the Management Information Server

components. The event logger is used to log events into a database for later analysis and audits. The event filter is the means of delegating intelligence and authority to the management information server. It can aid in the management decision-making process by performing sophisticated prioritisation, filtering, aggregation, averaging, threshold detection and other operations on incoming events. The forwarder and registry service combine to form an efficient addressing unit to perform conventional subscription based forwarding of events.

3. Conclusions

By implementing management policy at all levels, our proposed management information server based architecture differs from conventional publish-subscribe semantics based systems. With the management information server, we have developed a building block for a extensible, flexible management solution capable of handling heterogeneity. Future work involves experimental evaluation of the system in handling heterogeneity apart from efficiency, overhead and scalability of the system. The benefits and limitations of XML in expressing policies and events in such a scenario also need to be analyzed further.