

# A Policy Manager for Enterprise Service Management\*

Mani Subramanian, Minaxi Gupta  
*College of Computing*  
*Georgia Institute of Technology, USA*  
manis,minaxi@cc.gatech.edu

## 1 Introduction and Motivation

A distributed computing environment (DCE) is a collection of clients and servers where servers offer services to clients requesting for service. The DCE network could be wired, wireless, or a combination of both. An enterprise network management system that is in operation currently addresses the management of network, system resources, and applications.

As DCEs grow in size, number of devices and the type of services they require from servers increase as well. For example a large number of cable modems (CMs) share bandwidth in an broadband HFC (Hybrid Fiber Coaxial) network. Under peak load conditions, decisions need to be made in real-time as to how to do fair allocation of bandwidth amongst the various CMs. There would in general be several choices and a real-time **Policy Manager** is needed to make the right choice.

Another example is the use of mobile devices such as palm pilots as clients. During stock market crisis, users may want to access information and execute transactions in a hurry from any geographic location. A **Policy Manager** that keeps track of the location of the palm pilots and servers could optimize the traffic load for performance.

The above examples illustrate the necessity to augment the enterprise management system with a policy-based management system that monitors and manages a distributed computing environment as whole. A paradigm shift is needed for enterprise management systems from managing the network, system resources, and applications, to managing services as well that the clients request from servers. As the number of things that the enterprise management system has to manage increase, conventional management techniques that require polling and human intervention for decision making would be unable to perform management in real-time. Hence, another paradigm shift is needed to move from *polling-based* architecture to *event-based* architecture that uses *push* technology.

A *policy-based, event-driven* architecture for management of client-server based DCEs is described. The system functions at the application level and comprises **network clients**, *intelligent Policy Agents*, embedded in *network clients*, *active Policy Manager*, and **network servers**. Policy Agents monitor specified management parameters at respective network clients and inform the active Policy Manager if need arises. The Policy Manager evaluates the information received from Policy Agent, makes a policy-based decision, and directs the network server for appropriate action. This is accomplished transparently without requiring any manual intervention. In addition, because the architecture is event-based, the overhead to perform the management functionality is significantly reduced. Policy Manager makes a policy-based decision using a policy engine, which we will describe later. The architecture is not limited to any particular infra-structure.

## 2 Architecture

Main components of this architecture are **network clients**, **Policy Agents** embedded in network clients, **network servers**, and **Policy Manager** as shown in figure 1. Network clients request services from network servers using *client-server communication path* shown in figure 1 by dotted lines. Intelligent Policy Agents in each network client monitor these services and generate *policy objects* if either service quality degrades

---

\*Supported by grant from Aprisma Management Technologies (formerly Cabletron)

or there is some disruption in the service. These objects are *pushed* over to Policy Manager using *policy path* shown in figure 1 by solid lines. The Policy Manager maintains information about all network clients and network servers in its *information base*. It also has a *policy engine* which is a plug-and-play part of the architecture. Policy engine contains guidelines for deciding policies given the situation and can be of various types. It can be *rule-based*, *case-based*, *model-based*, *probability based* or any other kind. Policy Manager uses policy engine and information base to *actively* decide on the policy to remedy the situation. Enforcing the policy can be done in an *automated* way by sending *policy enforcement objects* as *events* to the network server, directing it to take appropriate corrective action. These objects are sent over to network servers using *policy path* shown in solid lines in figure 1. Policy Manager can also communicate with the Policy Agent by pushing events using the same policy path shown in figure in solid lines.

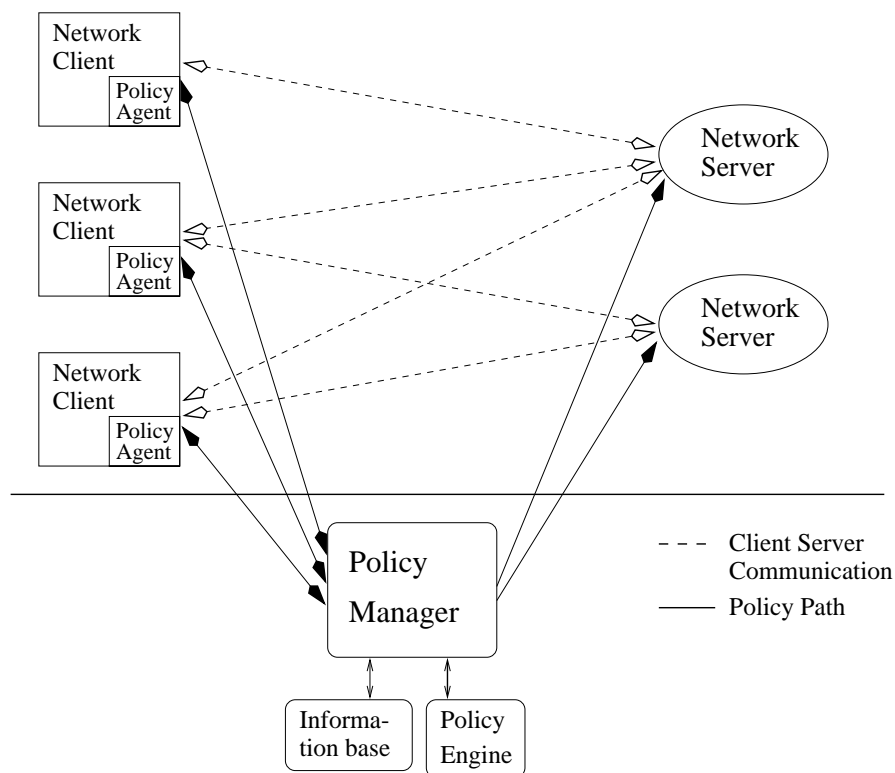


Figure 1: Components of architecture

The novelty of the architecture can be summarized as;

- Since *events* are pushed only when need arises, there is no *polling* involved in this architecture for the purposes of performing *service management*.
- The policy engine implementation is flexible. As long as it answers Policy Manager's queries about policies, it can be rule-based, case-based, model-based, probability-based, or any other kind.
- The architecture is not limited to any particular infra-structure. The implementation can be done in object-oriented way using technology like CORBA (Common Object Request Broker Architecture), or in a scalar way. It can be web-based if desired, and the Policy Agents can be implemented as Java applets.

### 3 Applications

The architecture is generic enough to be applied for service management of a wide variety of networks. Although many examples are possible, we would restrict ourselves to just two of them. The first example

is QoS (Quality of Service) management in broadband access networks like HFC (Hybrid Fiber-Coaxial) networks. This is an example of a wired network. The second example is to manage services needed by mobile users using devices like palm tops, palm pilots, laptops, pagers, cellular phones etc., while they are on the move, using wired or wireless (mostly latter) connection.

**Broadband Access Networks:** Main components of broadband networks like HFC networks are CMTS (Cable Modem Terminating System), CMs (Cable Modem), and CPEs (Customer Premise Equipment). The CPEs are connected to CMs and CMs transmit and receive data from outside world through CMTS. Both upstream and downstream channels are shared by multiple CMs [1] and availability of resources like bandwidth, processing time, and queuing buffers in CMTSs etc. to each CPE automatically becomes dependent on the usage of these resources by other CPEs and it is necessary to ensure QoS in such a system.

By embedding Policy Agents in CMs, filling in Policy Manager's information base with appropriate information about CMs and CMTS, and by plugging in a policy engine that can answer queries related to QoS management in such a network, the architecture can be mapped elegantly to automatically manage QoS in an HFC network, without any polling. We have a prototype implementation of this architecture for simulated HFC networks [2].

**Mobile Users using Wired or Wireless Connection:** Mobility is a way of life and people want to have access to information while they are on the move using pervasive devices like palm tops, palm pilots, laptops, pagers, and cellular phones etc. Most of these devices use wireless connection, but a few have wired connection as well. Depending on the capability of the device, users ask for services from network servers. Such services exist presently in a limited way and are likely to grow in future. For example, using Palm VII wireless service subscription, users can get stock and weather updates as services [3]. Managing such services would be very crucial. Also, since these devices are mobile, locating a server for fast real-time response for a service request by the device can also be done by Policy Manager.

As in case of HFC networks described above, a Policy Agent can be embedded in each pervasive device. Once Policy Controller's information base is filled with appropriate information about pervasive devices and servers and an appropriate policy engine is plugged in, the architecture can be used to manage services offered by servers for pervasive devices.

## 4 Summary

An *automated policy-based* service management in enterprise networks using *push technology* is described. An *active* Policy Manager manages the services provided to the network clients by the network servers using *intelligent, embedded* Policy Agents. The Policy Manager uses a *plug-in policy engine* and an *information base* containing information on network servers and network clients to make and enforce *policy decisions*. The architecture is independent of the transport infra-structure.

## 5 Acknowledgement

The authors acknowledge Cabletron with special thanks to Dr. Lundy Lewis for supporting this research activity.

## References

- [1] DOCSIS MIB Specifications. <http://www.cablemodem.com>.
- [2] QoS Management System Using Active Policy Server. Minaxi Gupta and Mani Subramanian. Submitted to NOMS 99 for consideration.
- [3] Palm web site. <http://www.palm.com>.