An Object-Oriented Model for Management of Services in a Distributed System

Geraldina Fernandes*and I. A. Utting

Computing Laboratory, University of Kent, Canterbury, Kent CT2 7NF, UK
Tel: +44 1227 764000 x7754, Fax: +44 1227 762811
email: {gpaf,iau}@ukc.ac.uk

August 23, 1996

Abstract

Distributed computing systems are becoming critical for the working of many enterprises, and are expected to contribute to the financial and operational well-being of the organisations which rely on them. Management of these heterogeneous hardware and software environments, in order to supply and maintain the services required, is one of the most difficult problems facing computer users today. This paper presents an object-oriented approach for the management of services in a distributed system. The main goal of this approach is the distribution of services taking into account the quality of service requirements of the services, and the resources available in the system.
Keywords: management, quality of service, distributed systems, object-oriented

1 Introduction

The importance of network and distributed systems management in supplying and maintaining services required by users has led to a demand for management facilities. For future applications, especially highly interactive applications and those relying on the transfer of multimedia information, it is essential that quality of service (QoS) is guaranteed system-wide, including the distributed system platform, the transport protocol and the network services. Another reason for the interest in automated and integrated management of networks and distributed systems is that some of the systems already installed have grown so large and complex that *ad hoc*, manual management is not coping. The way distributed systems have been constructed, by many vendors working independently, means that management systems from different vendors do not interoperate. Also, there is no integration in the management of different, but related, areas. The fully integrated management systems which will cope with management of large-scale distributed applications and their underlying communication services are still not available.

Existing approaches for management, such as SNMP [15] and the OSI reference model [8], address themselves mainly to network management. Close cooperation between the components of a distributed processing system makes these components much more difficult to manage, when compared with the autonomous components in a network. Different platforms are now available for building distributed applications (*e.g.* APM's ANSAware [1], OMG's CORBA [11] and OSF DCE [12]), however, these platforms do not provide facilities for automatic management of those applications.

*Work supported by JNICT Program *PRAXIS XXI* (Portugal) under grant No. BD/2804/93

This paper presents an object-oriented approach for the management of services in a distributed system. The main goal is to manage the resources available in order to fulfil the QoS requirements of the application services. Another benefit of this approach is to release the application programmer from the job of allocating services to nodes, providing a degree of transparency to the existence of several nodes. The importance of quality of service in distributed systems is discussed in section 2. The proposed management model is described in section 3. A management architecture based on that model is presented in section 4. Section 5 introduces work related to the area. Finally, conclusions and some issues for future work are discussed.

2 Quality of Service in Distributed Systems

Most of the work on quality of service (QoS) has concentrated on the network and communications infrastructure. QoS support in the OSI reference model consists of statically defining parameters intended to be supported at the session and transport layers (*e.g.* throughput, transit delay) [5].

Recently, QoS has also become increasingly important in operating systems because of the interest in their support for multimedia applications. In this research area it is important that resource management strategies are developed to more effectively exploit the limited resources and provide some guarantees of predictability. Resource management strategies are required for all areas of operating systems management, including processor scheduling, communications, device management and memory management. QoS has also become a major issue in distributed systems, due to the emergence of distributed multimedia applications. There has been little work on quality of service support in distributed systems platforms. The role of these platforms is to provide a network- and computer-independent programming environment for the development of distributed applications.

QoS management ensures that the requirements of the users are met. While QoS *dimensions* are abstract characterisations of QoS requirements, QoS management is the concrete realization of required levels of QoS in a real system. QoS management functions include: *specification*; *negotiation, resource allocation and admission control*; *maintenance, monitoring and policing*; and *renegotiation*.

3 Management Model

The role of *management* is to monitor and control the system to be managed, so it fulfils the requirements both of the owners and the users of the system. Management of a distributed system should itself be distributed to reflect the distribution of the system being managed [13]. The management model presented in this paper is a distributed object-oriented model based on the Open Distributed Processing (ODP) Reference Model [9] and the OSI management model [7, 8]. The Reference Model of ODP provides a framework for the standardisation of Open Distributed Processing. It creates an architecture within which support for distribution, interworking and portability can be integrated. The OSI Systems Management provides mechanisms for the monitoring, control and coordination of those resources which allow communication to take place in the OSI environment (OSIE).

One of the most important ideas in OSI Systems Management is the use of object-oriented principles to define management information and interfaces. The OSI management model views the devices in the network that are subject to management as *managed objects*. A managed object is an abstraction that represents the properties of data processing and data communications resources, for the purposes of management. The resource exists independently of management, but its internal functioning is only visible to management through the managed object . Apart from network elements, managed objects can also be considered as representations of other entities, such as distributed services and applications.

Managers monitor the activities of managed objects, make management decisions based on that information and perform control actions on the managed objects. The access to a managed object is allowed through the object interface. A managed object, besides its normal service interfaces, can provide more than one *management interface*, giving different management views of the resource represented by the managed object. There are three types of interactions between managers and managed objects: control actions, requests for information, and notifications.

Within the ODP environment there are multiple coexisting management views and boundaries of responsibility. To reflect these different views, *domains* provide a means of specifying boundaries of management responsibility and authority. It is necessary to permit different domains, with members in common, to coexist in order to reflect the different required views of management. All the managed objects in a domain are controlled under a common management policy.

4 Management Architecture

In this section we present an architecture to support management of distributed systems, addressing in particular the issue of distributing the workload submitted to a distributed system by its users. Distributed scheduling is used in order to locate a new application service on an appropriate node, taking into account the current state of the system and the quality of service (QoS) requirements of the service.

A centrally-located allocator – *Distributed System Manager (*DSM*)* – is responsible for taking decisions in order to determine to which node in the system each service will be allocated. To determine if a node is suitable to instantiate a service, the DSM has to compare the QoS requirements of the service with the resources provided by the node. Information about the resources available on each node is gathered. This information includes load indexes giving, for instance, the amount of processing provided by the node, memory available and communication bandwidth. Information concerning specific hardware devices can also be collected. The DSM makes placement decisions based on its last known state of the system. This information, stored by the DSM, is updated by the monitoring information it receives from node managers.

Having the DSM as a central unit, to which monitoring information is sent by all node managers in the system, can create a bottleneck in the DSM. This overhead can be reduced by sending monitoring information to the DSM on demand. Demand-driven information transfer also brings other benefits. Not all the node managers have to be polled, thus reducing communication overhead. When the DSM is first started, it polls all nodes under its control and uses the information received to initialise the monitoring information it stores concerning the resources available on each node. After initialisation, the DSM issues requests for monitoring information only when it is trying to find a suitable location for a service. Considering the service requirements, the DSM selects a set of nodes that can satisfy those requirements, polls them to check if they are still able to provide the same resources, and selects the one that best satisfies the service requirements. If the optimum QoS level cannot be granted, the DSM has to notify the client that requested the service and eventually negotiate new QoS settings. A purely centralised solution is not very reliable, since the failure of the central entity could cause failure of the entire management system. A solution to this problem may be based on the approach used in MICROS for processor allocation [16]. Instead of having one single entity responsible for global management, as in a centralised approach, there could be a *federation* of managers, each responsible for a group of nodes.

All newly created services are instantiated by the DSM upon request by the *trader*. The trader is an object that provides a service which accepts and stores service offers from potential providers (servers) and hands out this information on request to potential clients. The DSM selects a suitable location for the service requested and asks the local node manager to instantiate that service.

The is a *Node Manager* on each node, which reports monitoring information to a DSM and is responsible for managing the objects within that node. It is capable of performing management operations on managed objects on behalf of a DSM and of emitting management notifications on behalf of a managed object to inform the DSM about the occurrence of an event. The managed objects within each node represent the servers that provide the services allocated to that node and the resources provided by the node.

In order to be managed, all services must be started dynamically via the DSM. A description of each service (*alias*) is dynamically installed in a DSM . After aliases have been installed, the DSM posts proxies for the services with the trader and instantiates them dynamically whenever a client tries to import them. When a client asks the trader for a service, the reference in the trader for that service is a proxy exported by the DSM. The actual activation/deactivation of a service in a node is the responsibility of the *factory* local to that node. The procedure for activating a service is illustrated in figure 1.

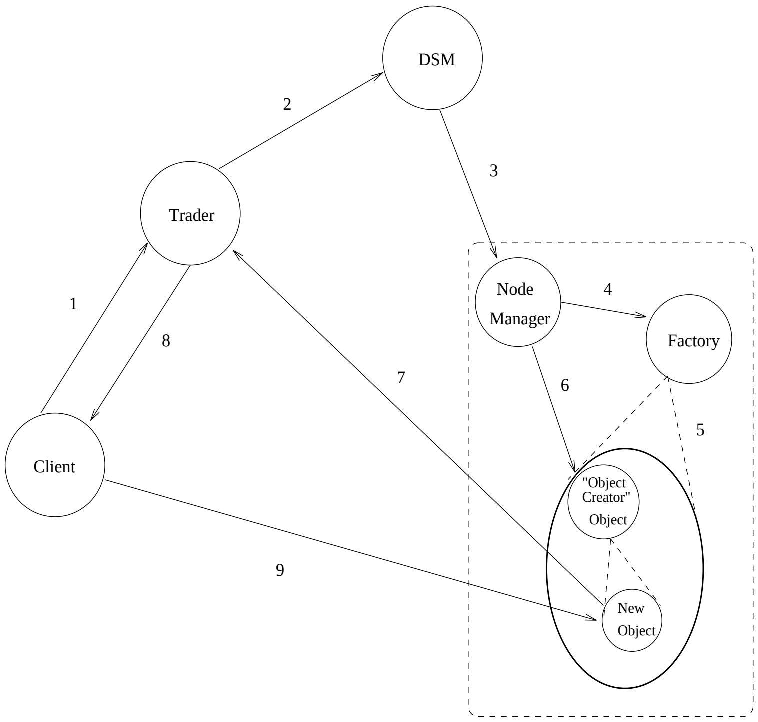

Figure 1: Activation of a service.

When a client needs a service, it tries to import a reference to that service from the trader (1). The trader recognises the offer as a proxy and forwards the import to the DSM (2). The DSM selects a suitable node on which to instantiate the service and asks the corresponding node manager to instantiate it (3). The node manager asks the factory to create a server capsule (every object is instantiated within a capsule, it is a collection of objects) (4). The factory instantiates the capsule and returns its interface reference to the node manager (5). The node manager calls an operation on the capsule interface to instantiate the object that provides the service required (6). Once the service has been instantiated, it exports itself to the trader (7). The trader returns the reference received from the service to the client (8). The client can now interact with the service (9).

When a node is running out of resources, it may issue an alarm to inform the DSM. The DSM may then decide to *passivate* (the server is terminated and a *snapshot* is stored so the server can be later re-activated) or *migrate* one of the services that have been activated in that node. Another situation where passivation could be considered is when a high-priority service activation is requested and a suitable node cannot be found. It may be necessary to passivate a lower-priority service in order to provide the resources required by the high-priority service.

5 Related Work

There has been a lot of research on the conceptual basis and description of a management architecture for the design of a distributed management system [6, 14]. The emphasis, in this architecture, is on the use of domains to group managed objects and partition the management structure to cope with very large scale inter-organisational distributed systems.

A number of organisations, namely X/Open, IEEE POSIX and NMF, are trying to define frameworks to integrate management. The majority of work is still focused on the definition of objectives and requirements [4]. However, an Open Distributed Management Architecture (ODMA) [10] is already being developed by ISO as a specific interpretation of the ODP Reference Model, to provide an architecture reference model for the management of distributed resources, systems and applications.

A systems architectural approach to QoS provision in distributed systems has been proposed recently by a number of research teams. Research on an architectural approach to integrated QoS support for multimedia applications has been undertaken at the University of Lancaster [2]. They have developed a QoS architecture which offers a framework to specify and implement the required performance properties of multimedia applications over high-performance ATM-based networks.

6 Conclusions and Future Work

The main goal of the work reported in this paper is to provide an architecture to support the management of services in a distributed systems. The architecture presented includes a scheduling strategy, making the system itself responsible for service allocation decisions. Those decisions are made considering the resources available and the QoS requirements specified by the services.

Large distributed systems will inevitably be partitioned into multiple domains with different responsibilities and management purposes . Management domains is an issue for further development. A DSM will be responsible for the management of the nodes which are members of one domain. One domain could have more than one DSM, for instance a *committee* of DSMs, to protect against faults.

The management architecture presented could be used to support the inclusion of management in existing platforms for developing distributed applications, such as ANSAware and CORBA.

The model described will be specified using ZEST [3, 17], an extension of the formal specification language Z, developed by PROST Objects to facilitate a clear, unambiguous specification of managed objects. From this model tests will be generated in order to validate the implementation.

References

[1] APM. ANSAware 4.1: Application Programming in ANSAware. RM.102.02, Poseidon House, Castle Park, Cambridge, CB3 0RD, UK, February 1993.

[2] Campbell, A. and Coulson, G. and Hutchison, D. A Quality of Service Architecture. Technical report mpg-94-08, Department of Computing, Lancaster University, 1994.

[3] E. Cusack and G. H. B. Rafsanjani. ZEST. In S. Stepney, R. Barden, and D. Cooper, editors, *Object Orientation in Z*, Workshops in Computing, pages 113–126. Springer-Verlag, 1992.

[4] J. Hungate and G. Fernandes. Distributed Systems: Survey of Open Management Approaches. Technical Report NISTIR 5735, NIST - National Institute of Standards and Technology, Distributed Systems Engineering, Computer Systems Laboratory, Technology Administration, U.S. Department of Commerce, Gaithersburg, MD 20899, U.S.A., September 1995.

[5] D. Hutchison, G. Coulson, A. Campbell, and G. S. Blair. Quality of Service Management in Distributed Systems. In M. Sloman, editor, *Network an Distributed Systems Management*. Addison Wesley, 1994.

[6] IDSM and Sysman. IDSM/SysMan Management Architecture. Technical report, October 1993.

[7] ISO. ISO/IEC 7498-4 | CCITT REC. X.700 Information Processing Systems - Open Systems Interconnection - Basic Reference Model - Part4: Management Framework, 1989. OSI Mgt.

[8] ISO. ISO/IEC 10040 Information Technology - Open Systems Interconnection - Systems Management Overview, 1992. OSI Mgt model.

[9] ISO. ISO/IEC DIS 10746-3 | ITU-T Rec. X.903 Part 3 Open Distributed Processing - Reference Model: Architecture. ISO/IEC JTC1/SC21/WG7, 1995. ODP-RM.

[10] ISO. ISO/IEC JTC1/SC21 N 9947: Open Distributed Management Architecture, Working Draft 3, November 1995. ODMA.

[11] Object Management Group. The Common Object Request Broker: Architecture and Specification. Tc document number 91.12.1, revision 1.1, OMG, December 1991. OMG CORBA.

[12] Open Software Foundation. *OSF DCE Application Development Guide*. Open Software Foundation, 11 Cambridge Center, Cambridge, MA, 1994.

[13] M. Sloman, J. Magee, K. Twidle, and J. Kramer. An Architecture for Managing Distributed Systems. Technical report, Imperial College, September 1993.

[14] M. Sloman, J. Magee, K. Twidle, and J. Kramer. An Architecture for Managing Distributed Systems. In *Proceddings of the Fourth IEEE Workshop on Future Trends of Distributed Computing Systems*, pages 40–46. IEEE Computer Science Press, September 1993.

[15] W. Stallings. *SNMP, SNMPv2 and CMIP - The Practical Guide to Network-Management Standards*. Addison Wesley, 1993.

[16] A. S. Tanenbaum. *Modern Operating Systems*. Prentice Hall, 1992.

[17] C. Wezeman and A. J. Judge. Z for Managed Objects. In J. P. Bowen and J. A. Hall, editors, *Eight Annual Z User Workshop*, pages 108–119, Cambridge, 1994. Springer-Verlag.