Management and QoS in Distributed Systems

N. Cottin, O.Baala, J. Gaber and M. Wack
Département Génie Informatique
Université de Technologie de Belfort-Montbéliard
France

Abstract *Recent emerging of advanced networking technologies (IPv6, ATM, FastEthernet) together with the development of Internet technologies and distributed computing such as World Wide Web, Java and CORBA have the goal of offering heterogeneous services to users across various geographical and network boundaries. Consequently, many innovations in developing and adopting such emerged technologies are needed to guarantee the Quality of Service (QoS) in real-time distributed applications. An environment called Distributed Objects Management Protocol (DOMP) is defined to supervise and manage real-time distributed applications in order to improve their QoS.*

Keywords: QoS, Real-time applications, SNMP, Data Mining, CORBA, Java.

1 Introduction

Customers nowadays not only ask for applications which functionally respond to their needs but also which provide a high Quality of Service level during transactions. The evolution of advanced communications [1] and computing technologies [2] comes up with new challenges. The first challenge is the management of large networks while not imposing an excessive overhead on network resources to enable flexible, scalable and robust operations. The second challenge is to effectively exploit the emerging communications and computing technologies to improve the quality of services in real-time distributed applications. To measure the Quality of Service, the Distributed Objects Management Protocol (DOMP) collects relevant information during the execution of the distributed application. Collected information concern the distributed application itself and the underlying network. These measures are exploited to maintain or improve the QoS.

Anatomy of a distributed application

Distributed computing can be seen as breaking down an application into components that can be distributed on a network of computers, yet still work together to do cooperative tasks. Higher-level services are needed to achieve a good performance . Here are a few of the more common motivations for supervising distributed applications :

- Large problems are broken into smaller pieces spread over the network. The nodes of the network cooperate to accomplish the application executions. Each node initially has a local view of the network. To enlarge their view of the network, the nodes need a distributed protocol to accomplish this network supervision service.
- Greedy tasks would be better assigned to the powerful processing units of the network. Within a distributed environment, we want to avoid the situation where a task is a slave of the slowest, most heavily loaded unit in the system.
- For systems that need fault-tolerance, a machine crash or a communication failure must not affect the correct functioning of the system. Redundant processing

on multiple networked computers can be used : when a machine goes down, the job can still carry on.

In this paper, we outline our effort on building an observation tool that collects remote data on objects spread over the network in order to improve the Quality of Service. We define the Quality of Service at two levels. The reliability at the application level where the application must guarantee a correct behavior with respect to its specification. The robustness at the network level where the network must guarantee the communications even in case of failure occurrences.

The rest of the paper is organized as follows : in the first section, we describe the distributed object management protocol architecture. In the second section, we give a detailed overview of the relevant information processed by the distributed object management protocol. In the third section, we describe the internal functioning of the protocol and present some implementation features. We finally conclude by pointing out future work directions.

2 The Distributed Objects Management Protocol (DOMP)

The most widespread technology for developing distributed applications is Common Object Request Broker Architecture (CORBA) defined by the OMG [3]. CORBA is an emerging standard of distributed object technology providing the interconnection network between distributed objects. It enables multiple clients and servers (commonly designed as objects) to communicate locally or remotely, through a LAN or the Internet.

An agent-based architecture for dynamic resource management, called Hector [5], under development at Mississippi State University, is designed using MPI to provide the infrastructure to control parallel programs during their execution and monitor their performance by running in a distributed and centralized manner. Our research is based on the Simple Network Management Protocol (SNMP) [4] extended to objects management. The Distributed Objects Management Protocol DOMP integrates objects management and network supervision.

DOMP architecture

This protocol defines the relevant information to collect from the objects and the network. The aim is to report information, perform statistics and send commands to the nodes. Distributed objects created by an application are running on different nodes (e.g., computers) of the network. They call each other to handle a request and perform a service. The protocol is in charge of gathering and displaying information from the objects, the nodes and the network. DOMP performs the following tasks:

- Gathering loading information. For example, in order to be informed which machines are the most available to run jobs.
- Processing collected information. The Optimizer Engine (OE), based on a Data Mining system, uses the collected information to make optimization decisions at run-time. For example, migration notification can be sent to some objects held by a machine becoming busy or to objects that communicate at high rate.
- Displaying performance measurements.

3 Relevant information description

In this section, we define the relevant information to be collected from objects. The most important is the frequency of objects invocations. An Object Invocation History (OIH) is associated with each object to keep track of the invocation on that object. Based on the OIH knowledge, we determine whether objects are communicating at high rate. In this case,

migration indication is sent to the involved objects.

DOMP supervises the local resources consumed by each object. An Object List Resources (OLR) is associated with each object. This OLR contains the local machine resources of the object such as the CPU-time and memory needed to process the object . So we can determine among the objects those who are the most CPU-time and/or memory consumers.

A list, called Object Thrown Exception (OTE), is associated with each object. OTE contains the exceptions thrown by the object. The OTE length determines the object level of reliability: the longer is the OTE list, the less reliable is the object.

Using these criteria, we can classify all the objects in order to make decisions that improve the Quality of Service. We also define the relevant information regarding the network. The node availability (NAV) is a parameter that indicates if a machine is down. In this case, all the objects on this machine are no more reachable. To know the objects in concern, we dispose of a table that contains the objects localization. Another important parameter is the Node Transfer Rates (NTR) on each node of the network involved in objects invocations. In addition, on each node, the following information are considered such as the CPU power, the load percentages (CPL), the percentage of global memory usage (GMU) and its failures frequency (statistics per minute) or NFF.

Implementation Features

An implementation of DOMP called OMENS (Object Manager, Environment and Network Supervisor) is being developed using Java and CORBA. The DOMP runtime environment, under development, is designed in the following manner. Declared objects to be observed have an interface that is in charge of sending messages to the DOMP manager according to the relevant information described in the previous section. For example, the objects invocation counts or the CPU time being spent. In our implementation of the DOMP protocol, called OMENS, the information is gathered from CORBA-based objects only. The network area is composed of a finite number of known machines.

An agent (i.e., a DOMP daemon) is running on each computer. The OMENS agent is in charge of collecting the local objects information. The OMENS server stores into a repository the objects information provided by the OMENS agents. A component of the OMENS server, the Optimizer Engine (OE), uses the repository to perform statistics and figures out optimizing decisions regarding to the QoS criteria. The OMENS client can also display the statistics sent by the OMENS server.

4 Conclusion

The ultimate goal of the DOMP is to guarantee the QoS of real-time applications when executed on a distributed network. DOMP provides a dynamic runtime environment that enables applications to be adapted to the varying network resources and optimize their execution. Our prototype is under development. We are currently working on the formal statement of the DOMP protocol.

References

[1] James E. Goldman. *Applied data communications.* John Wiley & sons Inc, 1995.

[2] Jim Farley. *Java distributed computing.* O'Reilly ed. 1998.

[3] R.Orfali, D.Harkey and J.Edwards. *The essential Distributed objects guide.* John Wiley & sons Inc. 1996.

[4] D. Zeltserman. *A practical guide to SNMPv3 and network management.* Prentice Hall series in computer networking and distributed systems, 1999.

[5] S. H. Russ, K. Reece, J. Robinson, B. Meyers, R. Rajan, L. Rajagopalan and C. H. Tan. *Hector:An Agent-Based Architecture*

for Dynamic Resource Management. IEEE Concurrency, p47-55, April-June 1999.

[6] H. Ma and S. T. Tan. *Dynamic Mobile Agent Based Distributed Network Management Using Internet Technology and CORBA.* IEEE International Conference on Systems, Man, and Cybernetcis, Tokyo, Japan,1999.