Monitoring Environment for Distributed Object-Based Applications *

Grzegorz Aksamit, Andrzej Król, Krzysztof Zieliński
Institute of Computer Science
University of Mining & Metallurgy
Al. Mickiewicza 30, 30-059 Kraków, POLAND
kz@ics.agh.edu.pl

November 2, 1994

Extended Abstract

A very rapid development of distributed applications increases demands for uniform platform for their implementation, debugging, monitoring, and management. A collaboration of major computer manufacturers results in CORBA [4] specification, as a future standard of object based distributed software. Its most important features are language independent model of information processing, structured in a form of objects with interfaces described in IDL notation and common operation invocation mechanism built around ORB functionality, that provides basic transparency services.

Apart from these standardization efforts, exploitation of distributed applications is still a very challenging area, not only during their development phase, but also system management and configuration. This additional aspects are covered by many documents such as [6, 3].

The starting point of any management activity is an information about a current system state, hence a monitoring system is an integral part of any management system [3]. Monitoring of distributed applications is a very complicated task that must be oriented towards the well defined feature of the system behavior. It means that gathering the infomation about a monitored system state should be selectively specified. Hence the monitoring system should be designed having the following in mind:

- A system should be easily expandable and configurable to follow demands of the monitoring process goal,
- Various monitored information generation, compression and filtering mechanisms should be provided,
- A monitor structure should be well fitted in distributed processing paradigm and should obey the same general rules as monitored applications.
- A system implementation should be nonintrusive and effective as much as possible, that means, that overhead of the system should be minimized, and the system should be able to handle any load generated by the monitoring activity.

*This work was sponsored by KBN under grant no.: 8 S503 015 06

In this paper a monitoring framework for distributed applications has been described. This system has been implemented in Orbix [1] that is CORBA complaint software environment and follow the assumptions stated above.

The paper has been structured as follows. In section 2, the system objects have been described by IDL interfaces. A centralized architecture of monitoring system has been chosen. The general system structure consists of the master monitoring module **Master_mon** that collects information generated by the local monitoring modules **Local_mon** distributed around the nodes of the system. Both modules provide client and server functionality so the communication is fully symmetrical. The generic set of **Local_mon** interfaces consists of the following operations:

- **register** – registers **Local_mon** with **Master_mon**,
- **unregister** – unregisters **Local_mon** with **Master_mon**,
- **update** – updates of monitoring data,
- **getnames** – gets monitored parameters names,
- **getvalue** – gets value of a given parameter,
- **kill** – destroys **Local_mon**.

The server **Master_mon** provides two generic interfaces:

- **alarm** – used for the system activation,
- **values** – used to make monitored information accessible from a particular node,

It is quite common, that **Master_mon** object has to cooperate with a greate number of **Local_mon** objects. So this object has to be very efficiently implemented. In section 3, performance evaluation tests of basic Orbix mechanisms have been reported and analyzed. It has created a background for an efficient implementation and structuralization of the monitoring system objects described in section 4. Several optimizations have been proposed to reduce the system overhead and decrease a response time.

The generic version of the monitoring system provides only operations for global load monitoring of nodes (e.g. CPU utilization, average length of CPU queue, etc.). In section 5, an extension of the generic interfaces by operations for per process resource utilization monitoring has been proposed and its implementation has been described. This work has illustrated and proved an extendability of the proposed monitoring framework.

The master monitoring module **Master_mon** cooperates with Visualisation Subsystem (**VS**) implemented as X11/Motif application. It is used for the monitoring framework management and customization of the local monitoring modules **Local_mon** activity. Detailed description of **VS** is presented in section 6. Finally in section 7, the paper is concluded by a few examples of the system application developed by DSRG (Distributed Systems Research Group) at UMM (University of Mining & Metallurgy).

Bibliography

[1] *The Orbix Architecture.* IONA Technologies Ltd., (1993).

[2] *Orbix – A technical Overview.* IONA Technologies Ltd., (1993).

[3] M. Mansouri-Samani, M. Sloman *Monitoring Distributed Systems (A Survey)* Imperial College Research Report No. DOC92/23 Apr. (1993).

[4] *Draft Common Object Request Broker Architecture Revision 1.1.* OMG Report 93-12-29, OMG Inc., (1993).

[5] A. Uszok, A. Król, K. Zieliński *Cluster Management Software for Open Object Oriented Systems* High-Performance Computing and Networking, Munich, Germany, 18-20 April, (1994).

[6] *IDSM/Sysman Management Architecture* report ESPRIT III EP 7026, October, (1993).