DIM
A DISTRIBUTED INFORMATION MANAGEMENT SYSTEM FOR THE DELPHI EXPERIMENT AT CERN

C. Gaspar

and

M. Dönszelmann

CERN, European Organization for Nuclear Research
CH-1211 Geneva 23, Swizerland

Abstract

The Online Control and Monitoring system of the DELPHI Experiment at LEP is distributed over around 20 VAX/VMS Systems and 70 FASTBUS processors. The DIM, Delphi Information Management System allows User Interfaces and Control processes to access information from any of these machines in a transparent way. Using DIM, Delphi User Interfaces can be run anywhere in the world provided that DECNET or TCP-IP are available.

I. Introduction

DELPHI (DEtector with Lepton, Photon and Hadron Identification) [1] is one of the four experiments built for LEP (Large Electron-Positron collider) at CERN . It consists of a central cylindrical section and two end-caps. The overall length and the diameter are over 10 meters and the total weight is 2500 tons.

The electron-positron collisions take place inside the vacuum pipe in the centre of DELPHI and the products of the annihilations fly radially outwards. The products of the annihilations are "tracked" by several layers of detectors and read out via some 200,000 electronic channels. A typical event requires about 1 million bits of information.

II. DELPHI Online System

The Online System of the experiment is divided into four main parts. The Data Acquisition System (DAS) [2] reads event data from the 19 sub-detectors composing DELPHI and writes it onto tape. The Trigger System [3] provides the DAS system with information on whether or not to write the event to tape. The Slow Controls System (SC) [4] controls and monitors slowly moving technical parameters and settings, like temperatures and high voltages, and writes them onto a database. The Lep Communication System controls the flow of data between LEP and DELPHI.

These four parts run distributed over a variety of machines, ranging from 68020 Fastbus processors running OS9, to VAXes of different types running VMS.

The Online System provides many different types of information. Some of the information has a high update frequency (order of seconds) while other are only updated every few hours.

III. DIM

DIM was introduced so that user interfaces, control and monitoring processes could have access to all the information produced in the Online System. DIM was designed in order to meet the following requirements :

- Any process in the Online System should be able to access any information it needs in order to perform its processing or display activities.
- The new system should run within the current Online System, without requiring major changes either to information producers or to consumer processes.
- The information gathered should be consistent over all processes running at a given moment
- Any process, consumer or producer should be able to freely move from one machine to another.
- The system should be very robust, e.g. any process dying should not perturbate the rest of the Online System.

A. Description

Information inside DIM is handled as named services. A service is a set of data with a name.

The name space for services is free (e.g. TRIGGER.RATE or DAS/STATE). The data inside a service can be of any type and size.

DIM uses a Client/Server mechanism and a Name Server.

A Server is a process that has information to publish. It sends the list of services it provides to the Name Server at startup.

A Client is any process that uses a service. When requiring a service the client asks the Name Server which server provides that service and from then on contacts directly the server.

The Name Server knows at any time wich services are available in the system and who provides them.

Figure 1 shows the data flow among Servers, Clients and the Name Server.

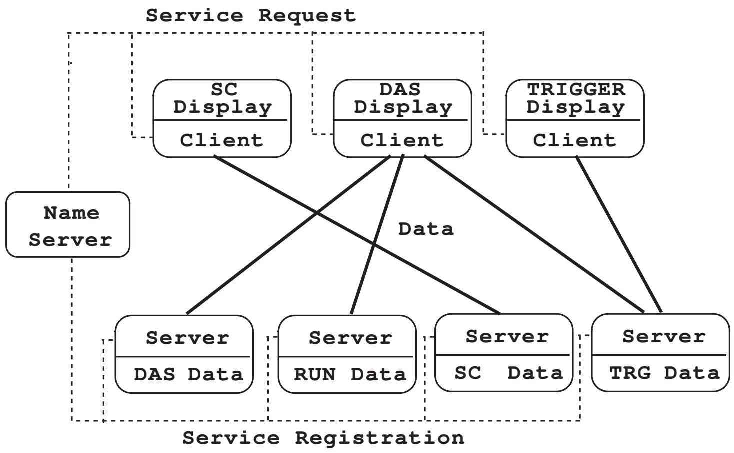

Fig. 1. DIM Data Flow

Four types of services can be requested by clients:

- ONCE-ONLY : The client requests information.
- TIMED : The client requests the information to be updated at regular time intervals.
- MONITORED : The client requests the information to be updated whenever it changes (available depending on whether the server can provide it).
- COMMAND : The client sends a command to the server.

TIMED and MONITORED services are only requested once by the client (normally at startup), the service will then be updated automatically by the server.

When using MONITORED services the server will update the information of all clients whenever it changes, thus making shure the data is coherent over all the clients of a certain service.

COMMANDs and ONCE-ONLY services can be requested whenever needed.

The COMMAND service is a special type of service, it's the only one for which no data is returned to the client.

As a communication mechanism the DIM aproach can be compared for instance to the RPC (Remote Procedure Call) approach.

RPCs are, in a way, contained in DIM, by using the ONCE-ONLY service. But DIM provides also asynchrounous communications with the MONITORED and TIMED services, this way, since the client requests the information only once and the information gets updated automatically, the overhead in time and network traffic is much less than when using RPC.

B. *Integration*

The DIM System was designed bearing in mind that it had to be integrated in a running system, so it was made as easy to use as possible.

Library routines exist for server and client use. They give all the DIM functionality to an existing process just by inserting one or two lines of code. This is possible because the system is completely asynchronous, both on the server and on the client side.

When a server declares its services at startup time, the library call will set up everything so that the services are registered with the Name Server, client requests are handled and data is sent to the clients whenever needed.

When a service is requested, on the client side, a buffer address and/or the address of a user routine is required so that when new data arrives the buffer is updated and/or the user routine is called. The library takes care of all the communications with the Name Server and with the server.

The largest users of DIM are :

- SMI [5], the State Manager Interface concepts allows the description of the experiment in terms of objects in well defined states. Commands can be sent to these objects causing them to perform actions and to change state. SMI is responsible for the control of the Data Acquisition and the Slow Controls of the 19 sub-detectors of DELPHI. The full SMI system is composed of more than 500 objects. Dim has been integrated into SMI, it provides the communication support for passing states and commands among the SMI objects and between the user interfaces and SMI.
- DUI [6], The MOTIF DELPHI User Interface uses DIM to get all the quantities to be displayed and to send commands to the various parts of the system (via SMI). The DIM mechanism works very well within MOTIF, since the user routine supplied when doing a client request can be used to update the screen whenever new data arrives.

C. *Robustness*

The DIM System has become the most important means of communication between processes on the Online System, so a very special care has been taken on the recovery from error situations.

The establishment of a communication channel between a server and a client is independent of the order in which the processes are started. The Name Server keeps track

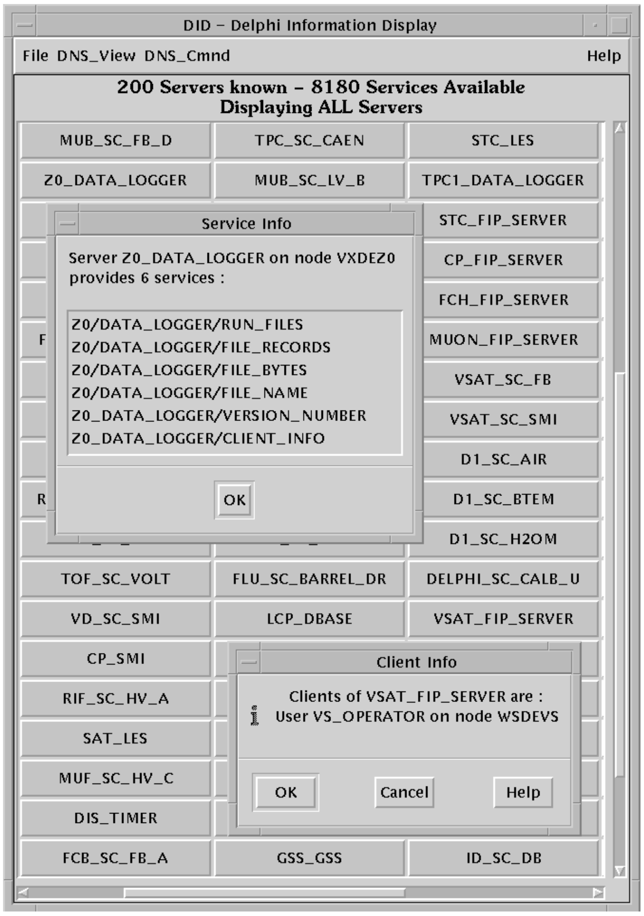

Fig. 2. DIM - Display Tool

of the clients waiting for "non-available" services and contacts them as soon as the servers start up. More generally when any server or client dies its partners will reconnect as soon as it comes back up.

When the Name Server starts all the servers will (re)register their services. And the clients will (re)request the services they need.

The Communication between servers and clients once established is independent of the presence of the Name Server.

In order to make sure that the servers are in a good state, each server sends watchdog messages to the Name Server, this way the Name Server can disconnect from a server or kill it if its behaviour is anomalous.

D. DIM Tools

The behaviour of the system can get very complex when the number of servers and clients becomes high. To be able to check and monitor the state of the system a Display tool has been developed, the DID - DELPHI Information Display.

DID allows the visualisition of the servers present in the system in three different ways :

- View all Servers
- View Servers by node
- View Servers by Service Name (wildcards allowed)

For each Server the list of Services available or the list of Clients connected can be displayed, as show on the example of Fig. 2.

E. Implementation

DIM is, for the moment, only available under the VMS Operating System, but it is foreseen to extend it to OS9 so that the FASTBUS embedded processes can publish information directly.

DIM uses as network support DECNET or TCP/IP, The protocol used is chosen by servers depending on what is available on the machine, the clients will (if possible) use the same protocol.

The Libraries available for Server and Client usage were written in "C" and they provide a "C" and a FORTRAN callable interface.

The DID - DIM display tool was developed using the OSF/MOTIF standard and VUIT as an interface building facility.

IV. Conclusions

The DIM approach provides a good communication machanism over different processes and different machines, we could call it an "Asynchrounous RPC" mechanism.

The DIM system has been installed progressively during '92 and it proved to be fast and reliable. Since it is very easy to integrate its usage is still growing. More and more users decide to use it to access any available services.

DIM is now responsible for most of the communication aspects within the DELPHI Online System, In DELPHI it makes available 9300 services, provided by 250 servers.

Using DIM, User Interfaces or other processes can access all DELPHI Online information from any machine inside or outside the CERN site.

DIM is now also being used by the NA50 experiment at CERN.

The Display tool is found to be very usefull not only for monitoring the state of the system but also to find out which services are available at a given time.

V. Acknowledgments

We would like to thank many participants of the DELPHI collaboration who have been very patient during the commissioning of the system.

In particular we are gratefull to Ph. Gavillet, Ph. Charpentier, B. Franek and M. Jonker for the many discussions during the design of the system.

VI. References

[1] DELPHI Collaboration, Aarnio, P. et al.,"The DELPHI Detector at LEP" in **Nuclear Instruments and Methods in Physics Research A303** (1991) pp.233–276

[2] Adye, T. et al., "Architecture and Performance of the DELPHI Data Acquisition and Control System", in **Proceedings of the International Conference on Computing in High Energy Physics '91** (Tsukuba, Japan, March 1991)

[3] Fuster, J.A. et al., "Architecture and Performance of the DELPHI Trigger system", in **Proceedings of the IEEE 1992 Nuclear Science Symposium** (Orlando, Florida, October 1992)

[4] Adye, T. et al., "The Slow Controls of the DELPHI Experiment at LEP", in **Proceedings of the International Conference on Computing in High Energy Physics '92** (Annecy, France, September 1992)

[5] Barlow, J. et al., "Run Control in MODEL: The State Manager", in **IEEE trans. nucl. sci. 36** (1989) pp.1549–1553

[6] Dönszelmann, M. and Gaspar, C. "A Configurable MOTIF Interface for the Delphi Experiment at LEP" in **Proceedings of the Second Anual International MOTIF Users Meeting** (Washington DC, December 1992) pp.156–162