CONTROLLING A LARGE PHYSICS EXPERIMENT
A COMMUNICATION ISSUE

C. Gaspar

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

J. J. Schwarz

INSA L3I. B502 IF
20 av A. Einstein
69621 Villeurbanne Cedex, France

Abstract In the past a physics experiment could be set up in a small laboratory by a single genius. These days, physics experiments involve international collaborations of hundreds of scientists, and the laboratory of the "crazy scientist" has been replaced by kilometers of accelarator tunnels, enormous underground caverns, tons of detector apparatus, hundreds of kilometers of cables, thousands of electronic boards and dozens of computers. The control of such an experiment is a complex system involving constraints of reliability, efficiency, reconfigurability and maintainability. This paper defends that an efficient communication system is an important issue when building such a control system.

Keywords Large-scale systems, computer communication, control systems, distributed control, load regulation, reliability, robustness

1.INTRODUCTION

DELPHI (DEtector with Lepton, Photon and Hadron Identification) (Aarnio at al, 1991) is one of the four experiments built for the 27 kilometer long LEP - Large Electron-Positron - collider at CERN, the European Organization for Particle Physics.

DELPHI 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.

The DELPHI detector is composed of 20 sub-detectors, as described in Fig.1, which were built by different teams in the laboratories of the DELPHI collaboration (around 500 scientists from 42 laboratories in 20 different countries all over the world). The main aim of the experiment is the verification of the theory known as the "Standard Model".

The DELPHI experiment started collecting data in 1989 and it has to be up and running for 8 months/year (24h a day) until around the year 2000. During its live time the experiment is constantly being modified, to allow for different physics studies. New sub-detectors can be introduced and old ones can be upgraded or replaced.

The control system of the experiment has to ensure that the experiment works efficiently and reliably during the running periods and it has to allow for an easy reconfiguration of any part of the experiment according to the physicists wishes.

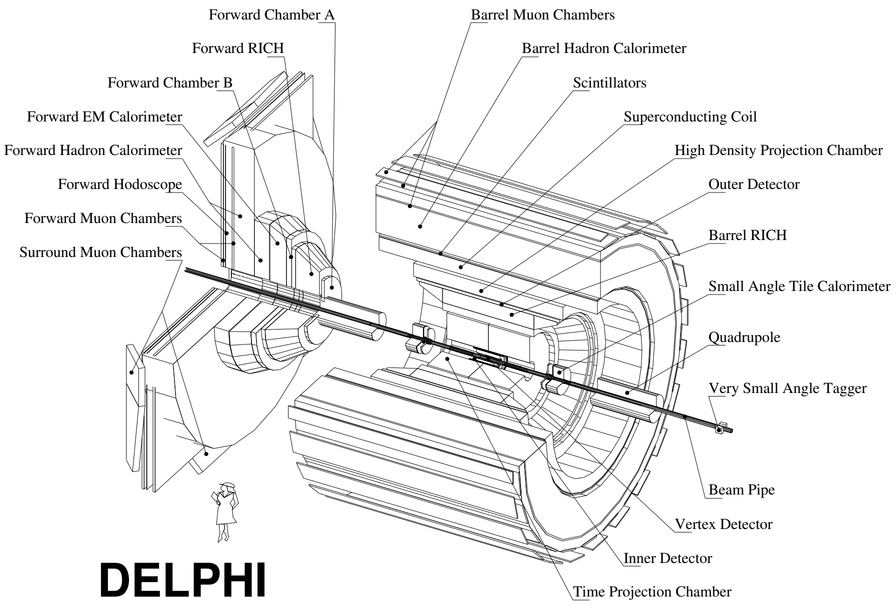

Fig. 1. The DELPHI Detector

2.DELPHI ONLINE SYSTEM

The online system of a physics experiment is composed of many different parts. Its main tasks are:

- The Data Acquisition System (DAS) (Charpentier et al., 1991) reads event data from the 20 sub-detectors composing DELPHI and writes it onto tape. In order to provide a high degree of independence to the individual sub-detectors the DAS system has been split into 20 autonomous partitions. These partitions are normally combined to form a full detector but they can also work in stand-alone mode for test and calibration purposes.

  The main constraints of the DAS system are in the areas of efficiency and real-time behaviour; the aim of the system is to record with zero losses the data produced in all LEP collisions (one every $11\mu s$). Another important constraint is adaptability; the DAS system has to cope with the introduction of new detectors, from their test phase (often in a different environment) up to their complete integration in the experiment.

- The Trigger System (Fuster et al., 1992) provides the DAS system with the information on whether or not an event is interesting and should be written to tape. The final trigger decision is a combination of the partial decisions of the sub-detectors.

  The main constraints of the Trigger system are also related to timing requirements imposed by the LEP crossing rate and to the flexibility necessary for easy modification of the combinatory logics between sub-detectors according to LEP's behaviour

- The Slow Controls System (SC) (Adye et al., 1992) controls and monitors slowly moving technical parameters and settings, like temperatures and high voltages of each sub-detector, and writes them onto a database.

  The SC constraints are mainly related to safety and security (of people and of the detectors that could be irreversibly damaged) since it controls dangerous parameters, like high voltages, sensitive gas pressures and temperatures.

- The Lep Communication System (Dönszelman and Gaspar, 1994) controls the exchange of data between the LEP control system and DELPHI.

  The main constraint is the availability of the system. If wrong information is distributed either to the experiment or to LEP the SC system could react erroneously and consequently damage the detector.

- The Quality Checking System (QC) provides automatic and human interfaced tools for checking the quality of the data being written on tape.

  The main constraint of the QC system is related to processing power; all the data produced by the detector has to be processed and analysed within a reasonable delay (1/2 hour) so that faulty parts of the detector can be repaired.

Information about the physical characteristics (timing properties) involved in data logging and triggering in such high-energy physics experiments were described by Zalewski (1993).

The complexity of controlling such a system comes from the fact that although the different parts of the system have different requirements and constraints, they have to work together for the common goal of providing "good" data for physics analysis.

In previous experiments the control of the different areas was always designed separately by different experts, using different methodologies and tools resulting in a set of dedicated control systems.

DELPHI noticed that due to the diversity of the different systems and of their interfaces the interaction between the different domains was practically impossible, the operation of the experiment was complicated and the evolution and maintenance of the system was in serious danger. DELPHI went through a re-investigation of the complete needs of the system and decided to take a common approach to the full "experiment control" system. The result was the design of a system that can be used for the control and monitoring of all parts of the experiment, and consequently for obtaining a system that is easier to operate, because it is homogeneous, and easier to maintain and upgrade.

The full online readout and control system is composed of around 40 VAX machines of different types running VMS, 70 Fastbus (M68020) processors running OS9, 80 G64 (M6809) processors and thousands of other items of electronic equipment (digitizers, timing units, etc.).

The online control system is characterized by a highly distributed architecture; like most current computer control systems, it consists of workstations interconnected by a local area network. Each workstation (through a Graphical User Interface - GUI) controls and monitors a part of the system, either a sub-detector (Det) or a central task, like DAS or SC, as shown in the diagram of Fig. 2.

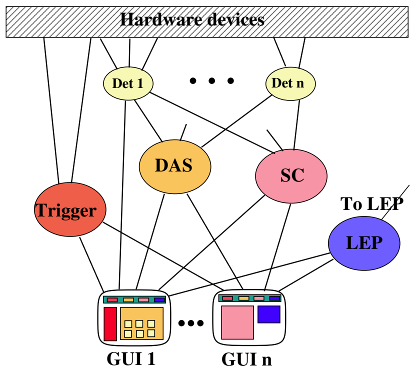

Fig. 2. The Online System

The need for the multiple tasks composing the DELPHI control application to run in different machines give rise to the problem of communicating easily, effectively and reliably among processes and processors.

The possibility of using a dedicated distributed operating system, like Amoeba (Tanenbaum et al., 1991), which would have allowed a better use of the available computing power and facilitated the coding of the application, was not considered due to the need for the use of commercial software, the reluctance of users to abandon their preferred operating system and also the constraints imposed by maintenance and support.

A communication system providing a transparent network extension to the chosen operating system could nevertheless allow the same goals to be achieved.

After a careful investigation of the products available in the market, it was discovered that either:

- they did not fullfill all the requirements needed (see next section), or
- they were not capable of handling such a large system, or
- they did not provide an easy interface to the user code, or
- they were self contained, i.e., they provided their own main loop and took over all operating system resources, not allowing the user to write or use his own software freely.

The final decision was therefore taken to design a new communication system - DIM (Distributed Information Management System) (Gaspar and

Dönszelman, 1993).

3.DESIGN REQUIREMENTS

The control system of the experiment is composed of more than 500 processes, distributed over around 40 workstations. The basic control consists of sending commands and getting back status reports in a tree-like structure, from the main operators down to the different hardware modules. Another very important part is monitoring: the monitoring of the experiment involves large amounts of data produced by the various machines dealing with specific parts of the system. In order to process, check and display these quantities a considerable amount of communication among stations has to take place.

The main purpose of the DIM System is to make data available where and when it is necessary, both for control and for monitoring purposes, independently of where it is produced.

In order to accomplish its mission DIM was designed according to the following requirements:

- Efficient Communication Mechanism

  DELPHI has some requirements for what concerns the communication mechanism. In the Online system most of the processes should be able to react to asynchronous changes of conditions and the communication package should allow for this. Another important requirement is one-to-many communications, as in the case of change of conditions in most cases more than one process has to be notified. For DELPHI's purposes an efficient communication mechanism is one that respects these requirements and is, of course, fast.

- Uniformity

  The DIM system should be capable of handling all exchanges within the online system, All processes involved with control, monitoring, processing or display should use the same communication system. A homogeneous system is much easier to program and to maintain.

- Run-time Transparency

  An important goal for a distributed communication system is transparency. No matter where a process runs, it should be able to communicate with any other process in the system, using a single mechanism that is independent of where the processes are located.

  The DIM system should allow processes to move freely from one machine to another. All communications should be automatically reestablished. This feature would allow for load balancing between machines, using either manual balancing for a static evaluation or dynamic balancing using automatic methods like the "Bidding Algorithm" (Ferguson, et al., 1988).

- User-coding Transparency

  Distributed applications are often very difficult to program. When coding a distributed application the user should not be concerned with machine boundaries; the communication system should provide a location-transparent interface.

  The programming interface should hide from the user all communication issues and reduce to the minimum the necessity for additional user code.

- Reliability and Robustness

  In an environment with many processes, processors and networks, it often happens that a process, a processor or a network link breaks down. The loss of one of these items should not perturb the rest of the application. DIM should provide for automatic recovery from crash situations or the migration of processes.

- Wide-area Transparency

  DELPHI is an international collaboration; any necessary information should be available to the outside world, using the same system.

By fullfilling such requirements, a communication system can greatly improve the development and performance of the complete system. It provides a decoupling layer between software modules, that makes coding, maintenance and upgrading of the system easier and improves efficiency and reliability at run-time.

4.COMMUNICATION MECHANISM CONSIDERATIONS

When designing a distributed control system, the choice of the communication mechanism to be used is an important issue.

Distributed applications are often based on remote procedure calls (RPC) (Birrell and Nelson, 1984). In the RPC mechanism the client sends a message containing the name of a routine to be executed and its parameters to a server, the server executes the routine and sends a message back containing the result. This implies that the communication is point-to-point and synchronous, since the client always waits until the routine finishes execution, as shown in the diagram of Fig. 3.

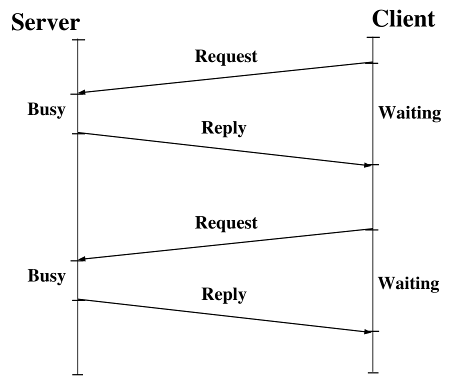

Fig. 3. RPC mechanism

For an application like DELPHI (but probably also for others such as industrial control or real-time applications) the RPC mechanism is very heavy and ill-suited. In most of these applications important tasks are the monitoring of parameters (sensors) and reacting to predefined or exceptional conditions; furthermore these tasks are normally repetitive and have to be executed indefinitely.

The solution that semmed the best in this case is for clients to declare an interest in a service provided by a server only once (at startup), and then to get updates at regular time intervals or when the conditions change, i.e., an asynchronous and one-to-many (group communications) protocol (Kaashoek and Tanenbaum, 1991), as depicted in Fig. 4.

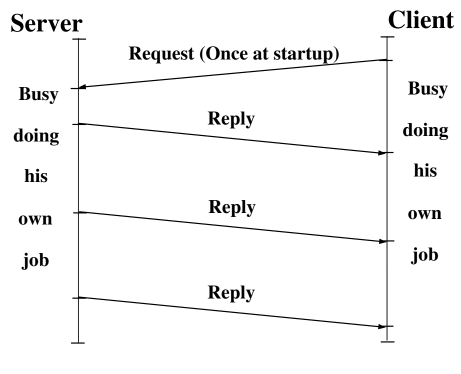

Fig. 4. DIM's mechanism

This mechanism - interrupt-driven, as opposed to RPC's polling approach - involves twice less messages sent over the network, i.e. it is faster and saves in network bandwidth. It has also the advantages of allowing parallelism (since the client does not have to wait for the server to reply and so can be busy with other tasks) and of allowing multiple clients to receive updates in parallel.

This approach, together with the possibility of sending commands to servers (more RPC-like), are the main features of the DIM communication mechanism.

5.DESIGN PHILOSOPHY

DIM, like most communication systems, is based on the client/server paradigm.

The basic concept in the DIM approach is the concept of "service". Servers provide services to clients. A service is normally a set of data (of any type or size) and it is recognized by a name - "named services". The name space for services is free.

Four commonly used types of services have been identified :

- 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 (availability depends on whether the server can provide it).
- COMMAND: The client sends a command to the server.

The TIMED and MONITORED services are only requested once by the client (normally at start-up); the service will then be updated automatically by the server.

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

The updating mechanism can be of two types, implemented either by executing a client callback routine or by updating a client buffer with the new set of data, or both. In fact this last type works as if the clients maintain a copy of the server's data in cache, the cache coherence being assured by the server.

In order to allow for the required transparency (i.e, a client does not need to know where a server is running) as well as to allow for easy recovery from crashes and migration of servers, a name server was introduced.

Servers "publish" their services by registering them with the name server (normally once, at startup).

Clients "subscribe" to services by asking the name server which server provides the service and then contacting the server directly, providing the type of service and the type of update as parameters.

The name server keeps an up-to-date directory of all the servers and services available in the system.

Figure 5 shows a small example of the use of the DIM system within the DELPHI Online System

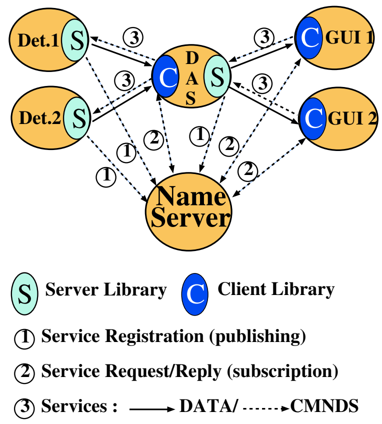

Fig. 5. DIM example

Whenever one of the processes (a server or even the name server) in the system crashes or dies all processes connected to it will be notified and will reconnect as soon as it comes back to life. This feature not only allows for easy recovery, it also allows for the easy migration of a server from one machine to another (by stopping it in the first machine and starting it in the second one), and so for the possibility of balancing the machine load of the different workstations.

6.IMPLEMENTATION ISSUES

- Transparency/Ease-of-use

  The user-coding transparency required by users is achived by hiding all communications, both server-client, and with the name server, inside library routines. These routines implement all the server and client functionality and allow any process to become a server or a client by doing a subroutine call.

  Once a server has "published" its services or a client has "subscribed" to the services it needs, the handling of client requests or server updates can be done (if desired) without any notification or intervention of the user process.

- Monitoring and Debugging

  The behaviour of complex distributed applications can be very difficult to understand without the help of a dedicated tool.

  The DIM System provides a tool - DID, the Distributed Information Display - that allows the visualization of the processes involved in the application as shown in Fig. 6.

  DID provides information on the servers and services available in the system at a given moment and on the clients using them.

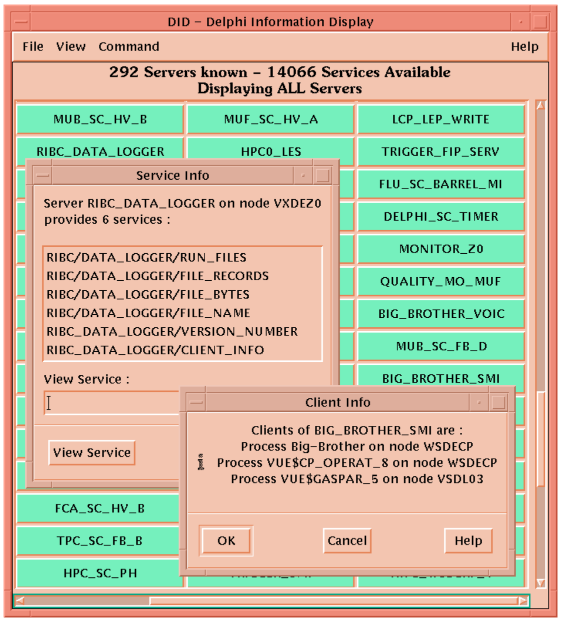

Fig. 6. DID - Display Tool

- World-Wide Access

  An application built using DIM can be distributed across the world provided that either TCP/IP or DECNET is available.

  The information available as DIM services can also be accessed by WWW (World Wide Web) through a WWW-gateway. The WWW page can be composed with the help of a dedicated editor.

7.CURRENT DEVELOPMENTS

Although the DIM system is currently in use in the DELPHI experiment, the project is not finished. Its installation on different platforms and over other network protocols would be of great use to both DELPHI and other potential users.

The DIM system is available for the moment under VMS and UNIX operating systems but not in mixed environments. It uses as network support TCP/IP and/or DECNET. The extension to a mixed platform environment supporting UNIX, OS9 and VMS over TCP/IP is being studied with the associated problems of different data representations over different machines. The main change is the need to describe the structure of the service data (that was previously irrelevant) in order to be able to convert to the local machine format.

The data formats and the network protocol to be used on each connection will have to be negotiated between the server, the client and the name server.

8.SYSTEM ENGENEERING

The engineering of the complete DELPHI Online System was a very large project. DIM is only a layer in the system, but this layer provides services to all the (500) processes composing the online system as if it was an extension to the operating system providing high-level networking capabilities.

The "heart" of the online system was developed by a small set of people (around 5 to 10) but the complete system includes work done by several members of the collaboration (mainly physicists wishing to set up the control and monitoring of their detector-specific parts). The development of software by external users of the system has largely benefited from a well-defined interface. Users could freely develop their software, using their preferred languages, tools or methodologies, but all exchanges with the outside were regulated by the DIM protocol.

9.CONCLUSIONS

DELPHI is one of the largest physics experiments in the world; its online control system is composed of many different components, distributed over many machines. In order to allow for efficient communication among machines and processes, a communication system - DIM - was developed.

The design of this system involved some of the main features of the engineering of complex computer systems:

- Apart from the functional aims (mainly high speed data acquisition and control), important non-functional objectives were the availability, flexibility, fault-tolerance and safety of the final system.
- The multi-disciplinary cooperation (physicists, electronics engineers and software engineers) was a key issue in the design of the system.
- Due to the need for evolution, parts of the system have to be re-engineered periodically, bearing in mind an overall view of the system.

DIM has greatly simplified the coding and maintenance of the DELPHI online software, by providing a network-transparent inter-process communication mechanism. The distribution of and access to up-to-date information of all parts of the system takes place with the minimum addition of user code.

DIM's asynchronous communication mechanism allows for task parallelism and multiple destination updates. Its characteristics of efficiency and reliability have considerably improved the performance and robustness of the complete online system. The number of crashes was reduced from once a week (taking about two or three hours to recover) to none during the last year.

Access to the DELPHI information is possible from all over the world, either directly through DIM or via WWW through a WWW-DIM gateway.

DIM is responsible for most of the communications inside the DELPHI Online System; in this environment it makes available around 15000 services provided by 300 servers. DIM is now also being used by other experiments at CERN.

The extension of the DIM system to other platforms would be of great use to DELPHI (and other users), and is being studied.

10.REFERENCES

DELPHI Collaboration, Aarnio, P. et al. (1991). The DELPHI Detector at LEP. In: *Nuclear Instruments and Methods in Physics Research A303*, pp. 233-276.

Charpentier, Ph. et al. (1991). 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.

Fuster, J.A. et al. (1992). Architecture and Performance of the DELPHI Trigger system. In: *Proceedings of the IEEE 1992 Nuclear Science Symposium.* Orlando, Florida.

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

Zalewski, J. (1993). Real Time Data Acquisition in High Energy Physics Experiments. In: *Proceedings of the IEEE Real Time Applications Workshop '93*, pp. 112-115.

Dönszelmann, M. and Gaspar, C. (1994). The DELPHI distributed information system for exchanging LEP machine related information. In: *Nuclear Instruments and Methods in Physics Research A352*, pp. 280-282.

Ferguson, D., Yemini, Y., and Nkolaou C. (1988) Microeconomic Algorithms for Load Balancing in Distributed Computer Systems. In: *Proceedings of the 8th IEEE International Conference on Distributed Computing Systems*, pp. 491-499.

Gaspar, C. and Dönszelmann, M. (1993). DIM - A Distributed Information Management System for the DELPHI experiment at CERN. In: *Proceedings of the IEEE Eight Conference REAL TIME '93 on Computer Applications in Nuclear, Particle and Plasma Physics.* Vancouver, Canada.

Tanenbaum, A. S., Kaashoek, M.F., Renesse, R. van and Bal, H. (1991). The Amoeba distributed operating system - A Status Report. In: *Computer Communications.* Vol 14, pp 324-335.

Birrell, A. D. and Nelson, B. J. (1984). Implementing Remote Procedure Call. In: *ACM Trans. Comp. Syst.* Vol. 2, No. 1, pp. 39-59.

Kaashoek, M. F. and Tanenbaum, A. S. (1991). Group communication in the Amoeba distributed operating system. In: *Proceedings of the 11th International Conference on Distributed Computing Systems.* Arlington.