ADVANCED USE OF WOLRD-WIDE WEB IN THE ONLINE SYSTEM OF DELPHI[a]

M. DöNSZELMANN, D. CARVALHO[b]

CERN European Laboratory for Particle Physics
CH 1211 Geneva 23, Switzerland

L.M. MUNDIM

LAFEX, R. Xavier Sigaud 150, 22290-180 Rio de Janeiro, Brazil

S. DU

LAL, Université de Paris-Sud (Paris XI), Paris (Orsay), France

K. RODDEN

University of Strathclyde, Glasgow G1 1XH, Scotland, U.K.

F. TENNEBØ

Molde College, 6400 Molde, Norway

The World-Wide Web technology is used by the DELPHI experiment at CERN to provide easy access to information of the 'On-line System'. WWW technology on both client and server side is used in five different projects. The World-Wide Web has its advantages concerning the network technology, the practical user interface and its scalability. It however also demands a stateless protocol and format negotiation.

1 Introduction

The use of World-Wide Web[1] can be extended beyond the distribution of static documents. The following sections explain why we interface, where we interface and what the constraints are to interface to the Web. Static documents are normally made available on the Web by an general http server. There is however information that you might like to put on the web for which you first have to build an interface (either on the server side or on the client side). Interfacing to the Web[2] is necessary in the following cases:

• The information is available in a different information system than WWW (Databases).

• The information dynamically changes over time (Monitoring Systems).

---

a. Available via WWW from http://www.cern.ch/~home/duns/papers/chep95_rio/html2/chep95.html

b. Universidade Federal do Rio de Janeiro, Ilha do Fundão, 21995-970 Rio de Janeiro, Brazil

• The information is readable via WWW but in an unsuitable format (Databases).

• The information has to be tightly coupled to an existing application (Help Systems).

• The browser is not capable of understanding the communication protocol or format of the information (Browser Extensions).

The following sections describe both applications on the Server side and on the Client side, with examples of the use of WWW in the field of High Energy Physics Experiments[3].

2 Server Applications

Two common ways of interfacing to servers are currently available. The first one uses the Common Gateway Interface specification[4], which makes it possible to install portable scripts and programs onto a default http server. The second way to interface is to "directly -couple" your program with a standard http daemon. You basically replace two routines (HTRetrieve: which normally retrieves a file, and HTServerInit: which initializes contact with your information system) in the daemon by your own. Directly coupled gateways have the advantage of a once only initialization.

Four examples of server interfaces are shown. The first one relates to DELPHI[5] in general and describes a server to distribute event data using the CGI interface. The second and third distribute on-line status information and are directly coupled. The third example is an attempt for a semi-interactive event display, which also uses CGI.

2.1 Example 1: The DELPHI Event Server (DES)

The specific request of one determined event has been always very important in any experiment for sub-detector studies, code development and graphic visualization of tagged events. In order to fulfill this task an event server (DELEVSRV) was developed on CERNVM (IBM mainframe). However, the workstations proliferation and the CERNVM deactivation imposed the need of a new event server (DES), running via World-Wide Web.

The DES offers to the physicists options to find the tape where the event is, to pick up a copy of it (raw data) and to re-process it by a specific Delphi's reconstruction package (DELANA)[6] version and/or options.

The DES is composed of two CGIs and one HTML document; one of the CGIs if for submitting a request, called from now on Submit part, and the other to query its status, the Query part.

Both CGIs are composed of a URL analyzer, its main part, that evaluates the HTTP request and send it to corresponding modules (see figure 1). Besides the URL analyzer, the Submit CGI is made up of three modules related to the options: Delana, List and Pick and the Query one is composed by the modules Get_Status, Discard_File and Send_File.

The List module communicates with the FATMEN[7] database and send back to the user the tape identification in which the event can be found. The Delana and Pick modules, in a first step, prepare a form[4] and send it back to the browser. In a second step, this form is submitted with the characteristic information to feed the requested module. The same

CGI re-process this form, checks on the DES database to see if this request has not been already submitted and send it to the spool to be processed by a scheduler[8].

The Query part also works in two steps: first it gets into the Get_Status module, which communicates with the DES database and generates a new HTML document containing the status of all active requests sent by the user. If there is no finished request, this document contains just a message telling the job's position in the queue. On the contrary, if there are finished jobs, this document will contain another form (using GET method), that allows the user to fetch the output and/or the log file to the local machine (done by the Send_File module) or discard it (Discard_File module). The Send_File module communicates with the DES database after sending the file to update the request status for future garbage collection. The Discard_File module will perform its action by deleting the files related to the present request, updating the DES database and calling the Get_Status module in order to send an updated HTML document to the user. The process then repeats until the last file is sent.

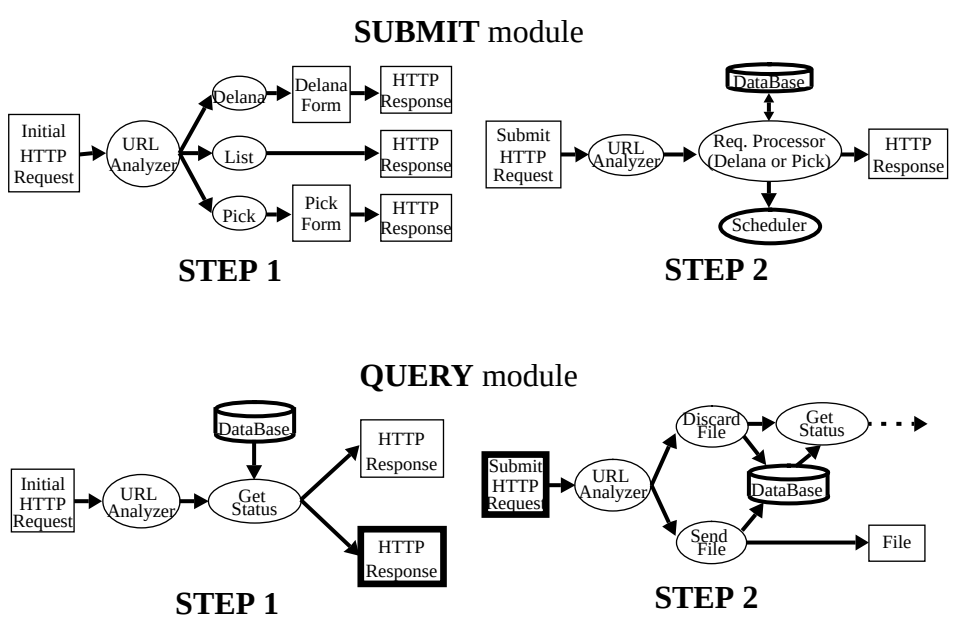

Figure 1: Dataflow of the DELPHI Event Server (DES)

2.2 Example 2: HIPE WWW Gateway

The next two examples relate to the Data Acquisition[9] and Slow Controls[10] of the On-line System. In the On-line System computers run monitoring processes to check the flow and control of the data acquisition system, the slow control values and settings and the performance of the triggering system. Each of these processes set a state as the result of their inspections made. These states are monitored and controlled by state-management pro-

cesses. The DELPHI on-line system relies heavily on network communications to interact between the state-managers, the monitoring processes and the DELPHI User Interface[11].

One way of communicating is via memory. HIPE (Human Interface for the Elementary Process)[12] uses shared memory to read values from slow controls (high voltage) channels. It is menu driven and displays channels, groups of channels and groups of groups in textual menus on a vt200 terminal. The HIPE WWW gateway[13] was designed to give people network access to the slow controls channels of DELPHI via the Web. It uses the core part of HIPE, but instead of producing textual menus it generates HTML. Figure 2 shows some of the Outer Detector high voltage channel readings.

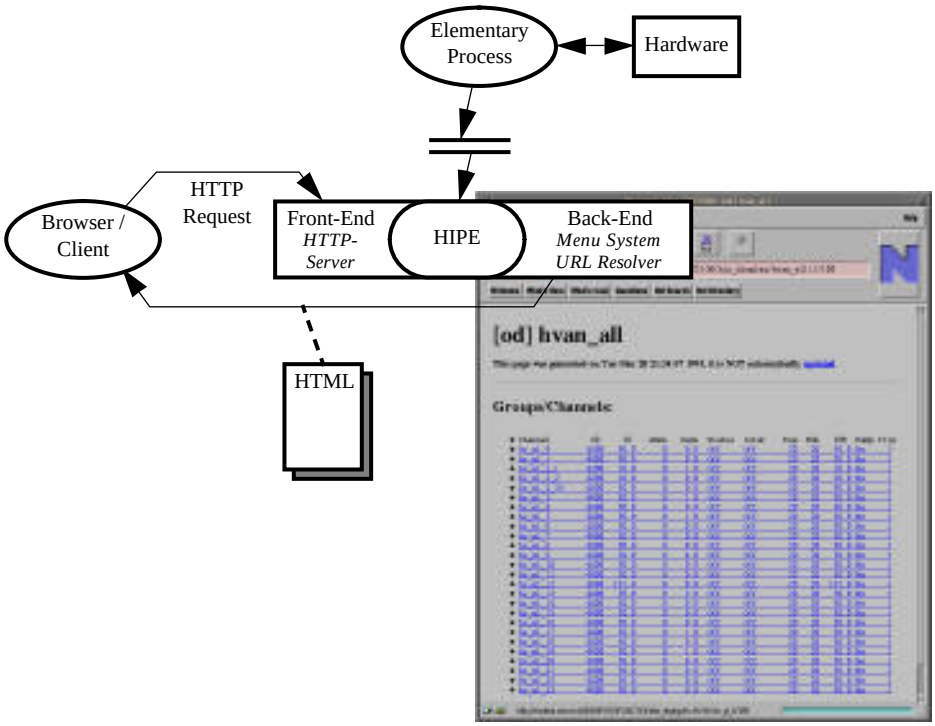

Figure 2: Dataflow of the HIPE WWW Gateway and its appearance on the Web

Figure 2 also shows the dataflow for the HIPE WWW gateway. The Elementary Process reads the hardware and stores the information in shared memory. It is an autonomous process which is always running in the background. The HIPE WWW gateway, when started, associates with the shared memory and waits for a request from the browser. When this request comes in, a URL analyzer looks what the user wants. Requests look like:

• http://machinename.cern.ch/HIP/OD/function/hic_display/hv_channels/1/100

where OD equals Outer Detector, hic_display is the function to execute, hv_channels is the group of channels to show and 1 and 100 are the parameters to provide to hic_display.

The URL analyzer now calls the hic_display function which forces HIPE to read the channels from shared memory and compose a display.

The functions which used to provide a display were replaced by function that compose HTML, in the form of unnumbered lists instead of menus. The output, which is thus created, is a passive HTML file. HIPE WWW is aware of what functions to execute when you select an item. The URL resolver basically re-composes URLs from these functions and inserts them into the output. The end result is an HTML document showing current slow controls channels, where each channel points to a function to explore that channel even further. The whole setup turns out to be faster than the old HIPE vt200 interface.

2.3 Example 3: DIM WWW Gateway

The DELPHI On-line System uses the DIM[14] system (Distributed Information Management) for its internal network communications. This system uses a publish-and-subscribe mechanism in a client-server model. Monitoring processes publish information by name onto the network. Other applications, like user interfaces, subscribe to this information, and once subscribed are kept up-to-date by the monitoring jobs. The DIM system uses a central nameserver to tell clients where certain information (services) may be found. The system can be used for internal DELPHI running only, because both the central nameserver and the continuous connections do not scale.

To ensure that people at their home institutes have access to the DIM services a DIM WWW gateway[13] was created. The gateway works by doing on-the-fly conversions of so called DIM files. These files are basically HTML files, but contain extra DIM tags. When retrieved these tags are replaced by information (values) from the DIM system.

Figure 3 shows the On-line System Status of DELPHI. Information like trigger rates, state of the data acquisition and slow controls is shown in a table marked up in HTML3.

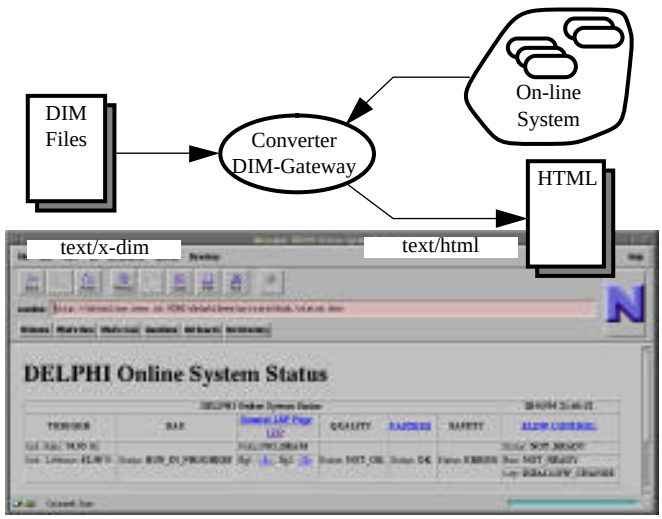

Figure 3: Dataflow of the DIM WWW Gateway and its appearance on the Web

Figure 3 also shows the dataflow for this gateway. When the user retrieves a DIM file the gateway searches for a converter from text/x-dim (the MIME[15] type associated with DIM files) to text/html (the MIME type associated with HTML files). The converter in its turn looks for <DIM SRC=”service”> tags. If found, it subscribes itself to the service and replaces the tag with the values it receives. It then releases the service again. All DIM tags are thus replaced by values and strings, resulting in a valid HTML document. This document may then be displayed by the user. If the user reloads the file, the same conversion takes place again, updating all values with more current ones.

The gateway provides only a one level interface, which certifies the statelessness. Conversions of DIM files tend to be slow, since the gateway needs to subscribe and release services all the time. By caching the actual service (the information is kept up-to-date by DIM) the performance is greatly improved.

2.4 Example 4: EDWIN Gateway

High Energy Physics experiments are probably easiest explained by visualizing collisions. Specialized programs, like event displays or viewers, must be used in order to see these events from different angles and with different zoom factors. These programs are normally quite complex and difficult to use. To make event viewing available to the general public, especially for educational purposes, DELPHI undertook the effort to provide their event display via the Web.

EDWIN (Event Display WWW INterface) was designed to provide a generic way of interfacing event displays to the Web. The user uses HTML forms to request a certain event and view (angle, zoom factor). The server provides a view in the form of a bitmap. The user may now iterate by resubmitting the form with different values, until he is satisfied with the result. This setup works but is very slow. Figure 4 shows an example of an event viewed via the Web.

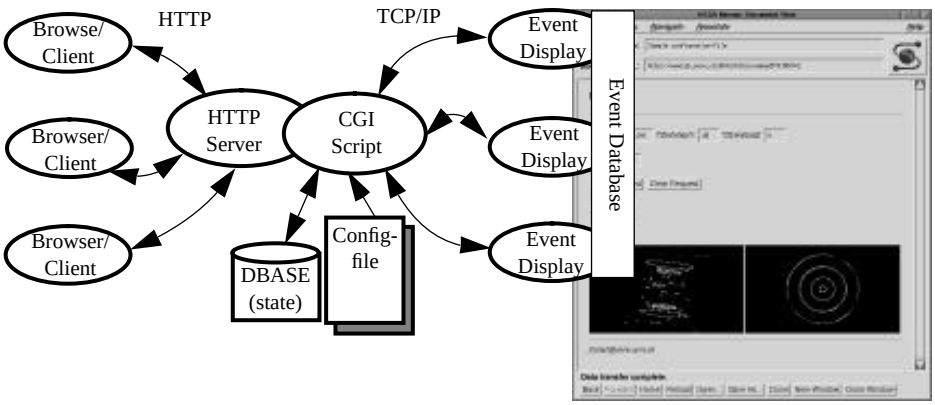

Figure 4: Dataflow of EDWIN and its appearance on the Web

Figure 4 also shows the dataflow of EDWIN. When the user requests the first view of an event, the server allocates an event display to him and sends back an ID number in the next form. It stores this ID and the association to the event display in a state database. When the user requests the next view, the server looks in the database for the ID and thus knows to which event display the request has to go. Ordinary event displays are used, which redirect their (XWindows) output to a virtual X server. The EDWIN package, which is linked into the event display, grabs the window, converts it into GIF and sends it to the client. Internal communications between the http server and the event displays is done with the XTCP package[16], which provides a TCP/IP communication channel to an X client.

The interface is made stateless by sending back and forth ID numbers between the server and client. The gateway however does not scale because the number of event displays that can be started is limited. There is also no way to know when an ID number can be thrown away or reused. The performance turned out to be very cumbersome. Conversions and manipulation of graphics plus the transfer across the network make things very slow. Future applications of this type should probably transfer data once, after which the user may interact with the browser/viewer.

3 Client Applications

Adding applications to the client side is possible by installing viewers or using a protocol called Common Client Interface[a] (CCI)[17,18]. Viewer extend the browsers capability of recognizing document formats. It uses a one way connection (e.g. the browser instruct the viewer what to display). CCI enables a two way connection. Since the connection lasts a state-full protocol may be used here.

CCI communication may be used in two directions, see figure 5. The browser can instruct an application (viewer) what to show, but the application may also tell the browser what to do. The latter was used by DELPHI, as also shown in figure 5, to implement a context sensitive help system. The figure shows a histogram presenter (much like PAW)[19] which starts Mosaic and retrieves specific help pages, when asked for.

The communication may be truly interactive. One could think of an application where the browser trains a user about a certain program. While the user is going through the tutorial on the Web, the program follows and demonstrates how things will really look. If at a certain moment the user decides to try something in the program, the browser may follow what he does and point out what to do next.

In future clients may probably be dynamically extended with some functionality. In this case the user will not only transfer the information, but also the interface to this information. EDWIN could be set up like that. One would transfer all information of an event, including the code to view an event from different angles. Currently HotJava[20] from Sun MicroSystems[b] is the only browser which supports this type of extensions. They use a generic programming language called Java[21].

---

a. Although CGI is supported by many server, CCI is only supported by XMosaic.

b. HotJava is currently only available for Solaris, Windows NT and Windows 95. A port for Macintosh 7.5 is underway.

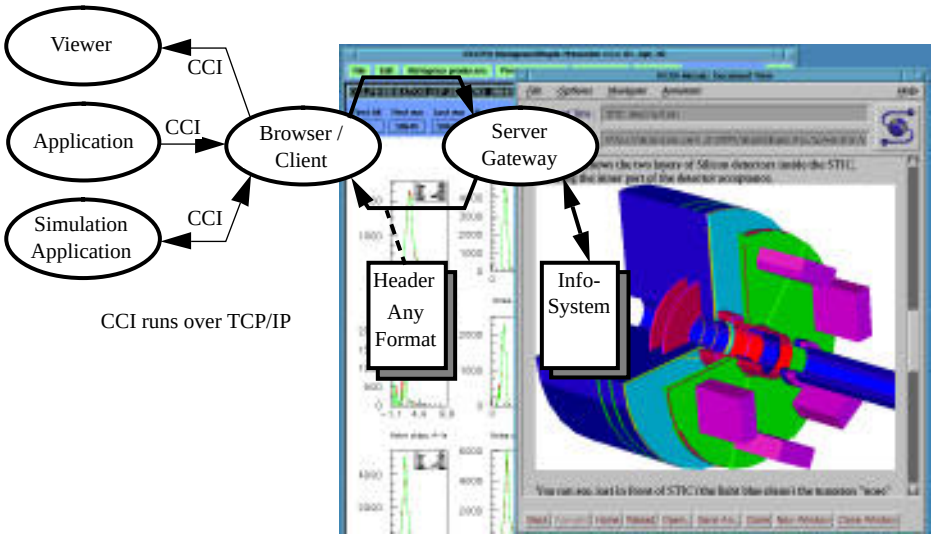

Figure 5: Connected applications and the DELPHI help on histogram system

4 Conclusions

The World-Wide Web can be used for more than just distributing static documents. Access to synthesized and dynamic information is currently possible by creating interfaces to the Web on the server side. A standard (CGI) has been written to do so. Interfaces which are created this way have to conform to the http protocol. Its stateless character has proven to be sometimes hard, but not impossible, to cope with. The performance of some server side interfaces is also an issue.

Due to the fact that client side interfacing and client extensions are still in development, not many applications have been created in that area. Context-Help systems are possible by using the upcoming CCI standard. Extensible browsers (HotJava) will present a whole new view on the Web. The provider will in future not only publish information but also publish an interface or interpreter for that information. It will provide better interactivity for web users.

References

1. T.J.Berners-Lee, R.Cailliau, J.F.Groff, and B.Pollermann, *"World-Wide Web: The Information Universe"*, Electronic Networking: Research, Applications and Policy, 2(1), pp. 52-58 (1992).
2. M.Dönszelmann, *"Interfacing to the Web"*, to be published in the proceedings of the CERN School of Computing, 20 Aug - 2 Sept, Arles, France (1995).
3. M.Dönszelmann, *"World-Wide Web and High Energy Physics Experiments, A Status Report"*, Mod. Phys. C - Physics and Computers 5(5), pp. 755-908 (1994).

4. NCSA Team, *"CGI, Common Gateway Interface (specification)"*, http://hoohoo.ncsa.uiuc.edu/cgi/overview.html, (1994).
5. P.Aarnio et al., *"The Delphi Detector at LEP"*, DELPHI Collaboration, Nucl. Instr. and Methods in Physics Research A 303, pp. 233-276 (1991).
6. Delphi Collaboration, *"Delphi Data Analysis Program (DELANA) - User's Guide"*, Internal Delphi Note, 89-44 PROG 137, CERN, Geneva, Switzerland (1989).
7. CERN, *"FATMEN"*, CN/AS Division, CERN Program Library Long Write-up Q123, Geneva, Switzerland (1993).
8. A.S.Tannenbaum, *"Operating Systems: Design and Implementation"*, Prentice-Hall International Editions (1987).
9. T.Adye et al., *"Architecture and Performance of the DELPHI Data Acquisition and Control System"*, Proceedings of the International Conference on Computing in High Energy Physics 91, pp. 619-626, Tsukuba, Japan (1991).
10. T.Adye et al., *"The Design and Operation of the Slow Controls for the DELPHI Experiment at LEP"*, Nucl. Instr. and Methods in Physics Research A 349, pp. 160-182 (1994).
11. M.Dönszelmann, C.Gaspar and J.A.Valls, *"A Configurable Motif Interface for the DELPHI experiment at LEP"*, Proceedings of the International Motif User Conference 92, pp. 156-162, Washington D.C., U.S.A. (1992).
12. M.Dönszelmann, *"DELPHI HIPE system user's manual v2.30"*, Internal DELPHI Note, 92-26 DAS 124 Rev, CERN, Geneva, Switzerland (1992).
13. M.Dönszelmann and K.Rodden, *"Gateways for World-Wide Web in the 'Online' Data Acquisition System of the DELPHI Experiment at CERN"*, Advance Proceedings of the Second International WWW Conference 94 "Mosaic and the Web", pp. 985-992, Chicago, U.S.A, October 17-20 (1994).
14. C.Gaspar and M.Dönszelmann, *"DIM - A Distributed Information Management System for the DELPHI Experiment at CERN"*, Proceedings of the 8th Conference on Real-Time Applications in Nuclear, Particle and Plasma Physics, Vancouver, Canada, June 8-11 (1993).
15. MIME, "Multipurpose Internet Mail Extensions", RFC1341 (1992).
16. M.Dönszelmann, *"XTCP, Communication Channel Interface for a Generic Server or X Client"*, Internal DELPHI Note, 95-17 DAS 161, CERN, Geneva, Switzerland (1995).
17. NCSA Team, *"CCI, Common Client Interface (specification)"*, http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/CCI/ccispec.html, (1994).
18. M.Dönszelmann, *"RMI, Remote Mosaic Interface (for VMS)"*, Internal DELPHI Note, 95-19 DAS 162, CERN, Geneva, Switzerland (1995).
19. V.Chorowicz, *"The real-time Data Monitoring in DELPHI"*, Internal DELPHI Note, 95-31 DAS 163, CERN, Geneva, Switzerland (1995).
20. SUN Microsystems, *"The HotJava Browser: A White Paper"*, http://java.sun/com/documentation.html, (1994).
21. J.Gosling and H.McGilton, *"The Java Language Environment: A White Paper"*, http://java.sun.com/documentation.html, (1995).