From the PC to the Network Computer.

George Shering, CN Division, CERN.

Abstract

The Network Computer is being touted as the alternative/replacement/successor to the PC. Claimed advantages are lower cost and reduced maintenance effort. Technical innovations are a network transferable interpretive programming language (Java), network loadable objects and a simpler stylized user interface. Some of these have been seen before in the "home-built" control systems of the 1970s. This paper describes the advantages of our current PC control systems, traces their development from the 1970s home-built control systems, and speculates on the possible role of the Network Computer in Accelerator Control Systems.

Introduction

In the 1970's the computer control of accelerators was a very hot topic. Accelerators had progressed beyond the capability of manual and hardwired control systems, the computer was there and full of promise, but experience and software for accelerator control had not yet become available. The SPS control system was a high water mark of this period. It brought together concepts such as the touch panel from SLAC, interpretive language ideas from Rutherford, powerful (for the time) minicomputers from Norsk Data, and welded them together into a network based control system which became an example for many other control systems, notably at DESY and KEK.

The problem with the SPS control system was that everything was home designed and custom built. The next generation of developers did not like this (not unreasonably) and an attempt was made to replace the custom designed elements with "commercial solutions". The Norsk Data consoles programmed in Nodal were replaced by workstations programmed in C. The network facilities of Nodal were replaced by Remote Procedure Calls and Equipment Databases. The Norsk Data minicomputers with CAMAC and custom Multiplex I/O were replaced by VME computers. Although each of these steps sounded good on its own, taken together the simplicity of the SPS system was lost and the 1980s' control systems were no better, and were more manpower consuming, than the 1970s' ones, despite enormously increased computer power and memory.

Fortunately in the 1990s the PC came along and gave a solution to many of the problems. Unfortunately the "professional computer" tendency which did not like the custom solutions of the SPS also resisted the PC control system, and as recently as the 1991 International Conference in Tsukuba, proponents of PC control systems were ridiculed for suggesting that a big accelerator could be controlled by PCs, some of which were limited to 640K memory (forgetting that the Norsk Data machines were sub-64K). I hope this first International Conference on PC control systems signifies a change in attitude.

The PC control systems of today have not yet solved all the problems, however, but things are developing fast and the Network Computer is about to appear on stage. The Network Computer, or perhaps more significantly Network Computing and the Control Intranet, will solve our remaining problems and take us back to the streamlined solutions of the 1970s but with full "commercial" components.

Claims of the Network Computer

A main claim for the Network Computer is low cost, namely a Network Computer for under $500. Cost breakdowns given by Oracle indicate that this is perfectly possible. The key feature is no hard disk, peripherals or expansion sockets. Just a screen, keyboard, mouse and speakers, knitted to the network by a single PC board with only a modest amount of memory. In 1984 the first Macintoshes cost 5000 francs for a single PC board and 9" black and white monitor, whereas a TV with the same screen and quantity of electronics cost about 200 francs. If the Mac

Paper presented at the International Workshop on PCs and Particle Accelerator Controls, October 7-9 1996, Desy, Hamburg.

had been sold at 500 instead of 5000 francs at the time then maybe the PC of today would have been a Macintosh! Oracle hope not to repeat that mistake and to keep the cost of the Network Computer down. The only snag is that some full spec PCs are heading down to the $500 level as well! IBM have just announced their "Network Station" at $700 (not counting screen). IBM is aiming squarely at replacing the millions of "dumb terminals" connected to its AS/400 minicomputers, primarily in data entry and customer service environments. The Network Station gives these users a graphical interface, a mouse, access to the Internet and corporate intranets, and e-mail.

Low cost is also an important claim for PC control systems, compared to workstations and VME crates. This aspect has been ignored for too long as computers are traditionally a touchy and highly politicized area. It is disturbing, however, when accelerator labs talk about shortage of money and manpower yet ignore factors of two or more in any area. The claim is that the control system is only a small part of the accelerator so it doesn't matter if too much money is spent. Studies at CERN in the early 1990's showed, however, that total expenditure on computing was a significant fraction of the laboratory's disposable materials budget. Cost studies on the Isolde control system and the Tau-Charm projected facility showed enormous cost savings from the use of PC rather than Workstation+VME. Then there are the claims that PC software can lead to big savings in man-power and that concentration on one kind of computer can bring big savings. It is nice to have every computer imaginable on site, no chance of missing out on the latest gadget! The cost in terms of manpower support is enormous, too great in fact, so what we get is reduced functionality, e.g. mail in ASCII, inter-operability problems, no real expert or support on any one platform.

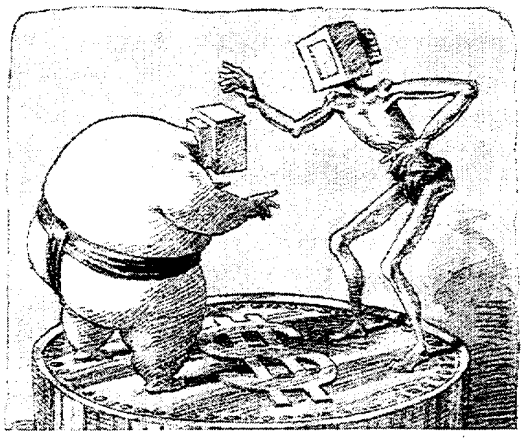
Another claim for the Network Computer is low cost of maintenance. An oft-quoted study found that an office PC can cost $8000 per year to maintain. I have just seen another figure of $13900. This is used by opponents of the PC to counter our "low cost" claim. Most of this is support manpower cost, however, and is clearly very sensitive to the support environment. This figure is easily reached if a support person is needed for every 20 to 30 users. The NICE network at CERN attacks this problem head-on (reference 1), and our NICE (Network Integrated Computing Environment) needs a maximum of a tenth of this. It should be noted, however, that Macintosh users quote the simpler installation of Macintosh software as a big advantage here, maybe a factor of two. However at CERN there is no "NICE" for Macintosh so the support costs are indeed much higher for Macintosh than for PC, as are those for Desktop UNIX. The Network Computer cuts the cost as there is no software to install at all (ROM version), or it is a standard image down-loaded from a server as for the claimed advantage of the X-terminal. It should be noted that IBM's partner for the "Network Station" is none other than NCD, the X-terminal vendor.

The key to the above two advantages is of course software, not hardware. A Network Computer has only one piece of software on board, a Java enabled browser. This can be downloaded from the network at power-on or built-in on ROM. The SPS Control System of the 1970s used the same approach. The console software consisted of a Nodal Interpreter with built in facilities for the man-machine interaction. All other functionality, in particular all the accelerator control functionality, came down over the network in the form of Nodal interpretable code and objects. In the SPS the computer software image was loaded through the network, and for the first ISOLDE control system which was modeled on the SPS one, all the system software was in ROM (EPROM) in the CAMAC based ICC. This was similar to the first personal micro-computers where the system software and BASIC was in ROM. This has the advantage of faster response at power on with no need to load the system image through the network, and so useful network interaction can begin at once.

A third claimed advantage of the Network Computer is ease of use. This springs from two factors: firstly only one interface which is fairly simple and becoming well known. The interface is simpler because everything is a Web page. The primary interaction is clicking on a picture or underlined text. Immediate reactions might be that this is not rich enough in functionality. Even the SPS control system had fancy gadgets

such as a "computer controlled knob". When it came to controlling the Anti-proton Accumulator, despite the fact that it was a "state of the art" machine, Simon van der Meer only wanted a simple touch panel and display, as he considered this to give adequate interaction and all the rest was clutter. And it worked. The touch panel gave only the equivalent of 16 hypertext links and did indeed prove adequate. Web forms are also very simple, and surely adequate for control use. Those of us who work with databases have difficulty accepting that the richness of, for example, Oracle Forms 4.5 is not really necessary for most purposes, but I am sure that, as in the AA, an ounce of application intelligence is worth a ton of interface clutter. The interface will become better known as the WWW continues its explosive advance. Here I feel on solid ground. Bill Gates, non other, has adopted the Web interface and the next version of Windows will have a desktop which is nothing other than a Web page! Another important factor is documentation. This has always been a weak point of control systems. By the time documentation systems had been set up, most of the people involved had moved on to other interests. Now there is only one documentation system, Web pages. From 1997 Microsoft will be providing their documentation in WWW form, despite the fact that their current system is one of the best available. Thus future control systems and future PC interfaces should look just like a Web page with HTML documentation.

The following picture shows how the Financial Times sees the battle between the "thin client" and the "fat client".



Applications Programming.

In the late 1960s the first attempts to harness the potential of computers for accelerator control foundered on the so-called "Software Barrier". In the 1970s people became very aware of this problem and for the SPS the solution was Nodal, an easy to use interpretive language. This was successful and a wide range of accelerator personnel were able to program their knowledge into the control system. In the 1980s a new generation of control personnel appeared who had not lived through the experience of the '60s and were entranced by the power of the workstation. They therefore launched into applications programming in C as the latest in software technology. This resulted in many problems and large teams of programmers had to be employed to produce the code, which was often not what the accelerator specialists would have produced, especially experts such as Nobel Prize winner Simon van der Meer who was also an expert Nodal programmer.

PC control systems came to the rescue in the early '90s with Visual Basic and Excel (reference 2), and once again Machine Physicists and Engineers could create their own applications. These tools are not as easy to use as Nodal was, but the important thing is that they are easy enough. When we started with the ISOLDE control system in 1989 we developed a version on Nodal for the PC which emulated the Touch Panel facilities on the PC screen exactly as used in the AA and in the previous ISOLDE control system. Half way through this development Visual Basic appeared from Microsoft, and Nodal was never seriously used again. Although Visual Basic is more complicated than Nodal it seems to be on the right side of some threshold whereas C and assembly language are on the wrong side. Excel was also popular, even with its then rather arcane "macro language" now replaced by Visual Basic for applications. In the case of Excel it was familiarity through use in non-control applications which was decisive, plus the motivation of learning something which is of general use.

For the Network Computer the programming languages are HTML and Java. These are much more complicated than Nodal. Examples from the Nodal Manual such as

>TYPE BCT(3)
to see the third Beam Current Transformer reading, or
>SET INJPHS=12
to set the RF phase at injection to 12 degrees, were very easy to explain as immediate commands. The simple line numbering technique made it easy to explain how such commands could be strung together to make a program, even to non-programmers, for example

```
1.10 SET A=1
1.30 SET B=2
1.50 TYPE A+B
```

HTML is not quite so obvious, unfortunately. For example:

```
<html><HEAD>
<TITLE>PS Control System Layout</TITLE>
<!-- PS Control Control System Equipment Access, I. Deloose, CERN/PS -->
</HEAD>
<BODY>
<H2><IMG SRC="CERNNormal.gif" ALT="CERN"> European Laboratory for
Particle Physics </H2>
<DIV ID="PSCODemo" STYLE="LAYOUT:FIXED;WIDTH:358pt;HEIGHT:346pt;">
  <OBJECT ID="Label1"
   CLASSID="CLSID:978C9E23-D4B0-11CE-BF2D-00AA003F40D0"
   STYLE="TOP:17pt;LEFT:8pt;WIDTH:338pt;HEIGHT:24pt;ZINDEX:0;">
    <PARAM NAME="VariousPropertyBits" VALUE="276824091">
    <PARAM NAME="Caption" VALUE=" PS Control Control System Equipment Access ">
    <PARAM NAME="Size" VALUE="11924;847">
    <PARAM NAME="SpecialEffect" VALUE="3">
    <PARAM NAME="FontEffects" VALUE="1073741827">
    <PARAM NAME="FontHeight" VALUE="280">
    <PARAM NAME="FontCharSet" VALUE="0">
    <PARAM NAME="FontPitchAndFamily" VALUE="2">
    <PARAM NAME="FontWeight" VALUE="700">
  </OBJECT>
</DIV>
<HR><ADDRESS>
Back to <A HREF="http://hostXX.cern.ch/Welcome.html">PS/CO Home Page</A><BR>
Contact <A HREF="http://xwho.cern.ch/WHO/people/08063">I. Deloose</A> CERN/PS (6-Sep-96)
</ADDRESS>
</BODY>
</html>
```

The above HTML will generate the page as shown below:

European Laboratory for Particle Physics

PS Control Control System Equipment Access

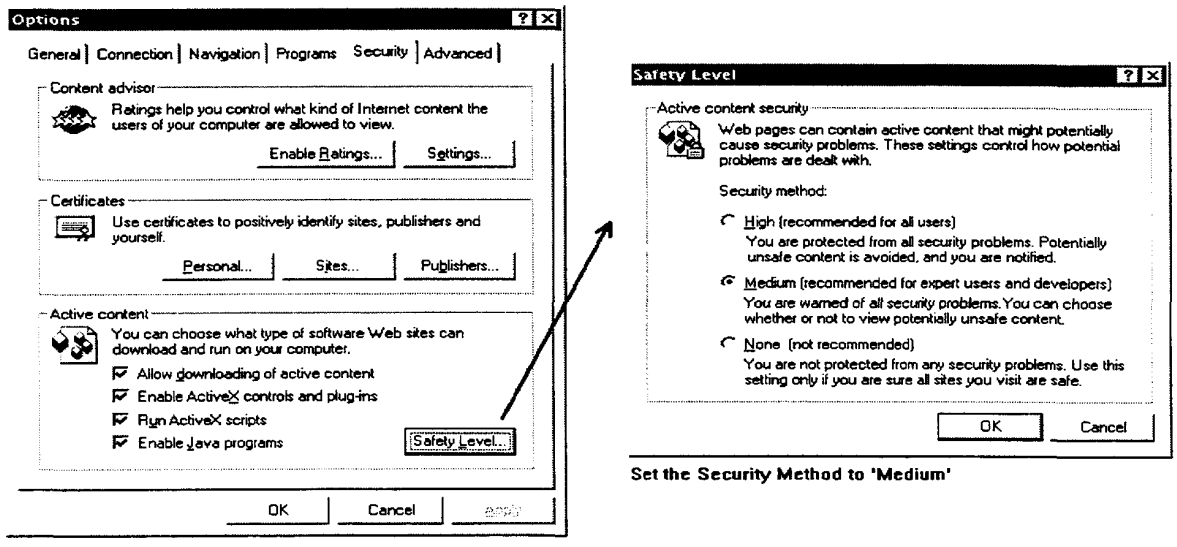
Back to PS/CO Home Page
Contact I. Deloose CERN/PS (6-Sep-96)

How can we be sure that the machine physicists and engineers will "catch on" to HTML? Accelerator personnel are now much more computer literate than they were 20 years ago. Nodal was significant for its very low "learning threshold". Nowadays people expect to cross a reasonable threshold and even Visual Basic takes a certain amount of effort to learn. But is HTML too arcane? Two considerations give me hope. The first is motivation. Everybody and his little brother want to have their own "home page" as the Web is visually rewarding. It is a technology that people want to learn, like Excel macros were, and a lot easier too. For people who are responsible for a part of the accelerator it is motivating to be able to create a specific home page. The second reason for hope is that automated aids are becoming available. The new versions of the Browsers have page editing and creation facilities built-in, for example the Netscape Page Wizard in Netscape Gold 3.0, and Microsoft's Internet Assistant for Word. Such products could well be customized for control applications. Even if the HTML has to be edited by hand to include the control aspects, the fact that the basic page layout with pictures etc. can already be created easily with commercial software packages is an immense advance. The final nail in Nodal's coffin was our inability to keep up with user expectations concerning such support facilities, on-line help etc. That is why it is so important to latch-on to a successful standard such as the Web.

A Web Control Example

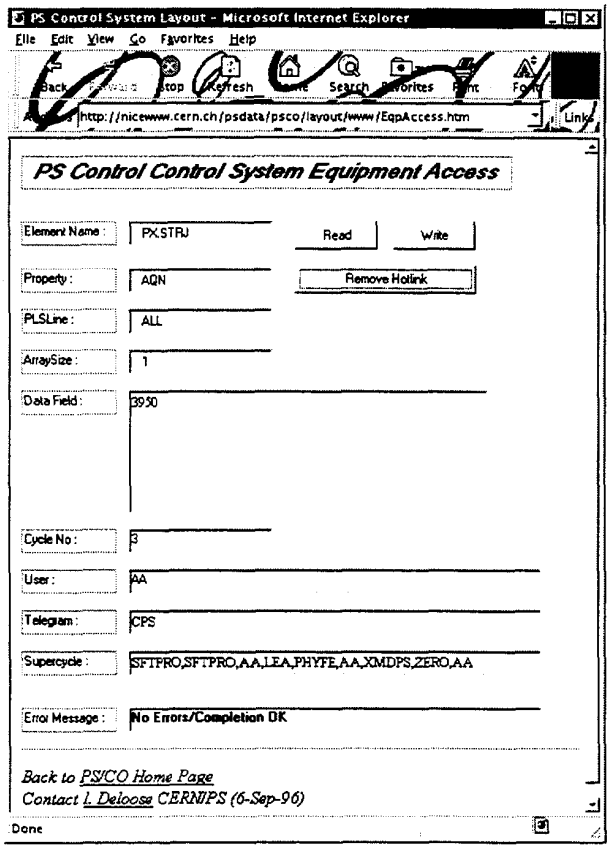
This example was written by Ivan Deloose in the PS Controls group (reference 3). What follows should only be considered as a demonstration and is not (yet) an operational tool, though using the WWW as diagnostic environment is currently under investigation. With the release of the latest Microsoft Internet Explorer version 3, OLE objects and Visual Basic scripts can be inserted into HMTL documents. The object has to be first entered in the Windows 95 registry and its components must be accessible by the Internet Explorer.

Before starting the application, the security method must be set to 'Medium'. This can be done by opening the 'Options' dialogue from the 'View' menu in Explorer, selecting the 'Security' tab and clicking the 'Safety Level' button.



Set the Security Method to 'Medium'

The example URL http://nicewww.cern.ch/psdata/psco/layout/www/EqpAccess.htm creates the page as follows:



To use the page one should fill in the parameters Element Name, Property, PLSLine, and click for instance the 'Hotlink' button. The *DataField*, *CycleNo* and *User* labels will be refreshed in synchronism with the PS timing system (PLS).

A selection from the HTML for the above example is shown below.

```
<html><HEAD>
<TITLE>PS Control System Layout</TITLE>
<!-- PS Control Control System Equipment Access, I. Deloose, CERN/PS
-->
</HEAD>
<BODY>
<SCRIPT LANGUAGE="VBScript">
<!--
Sub WriteButton_Click()
RpcOle.ElemName = ElemName.Text
RpcOle.Property = Property.Text
RpcOle.PLSLine = PLSLine.Text
RpcOle.ArraySize = ArraySize.Text
RpcOle.Action = 2
ErrorMessage.Caption = " " + RpcOle.ErrorMessage
ReadTelegram
end sub
</SCRIPT>
```

```
<OBJECT ID="ReadButton"
CLASSID="CLSID:D7053240-CE69-11CD-A777-00DD01143C57"
STYLE="TOP:66pt;LEFT:215pt;WIDTH:58pt;HEIGHT:25pt;.............
   HSPACE=10
   ALIGN=TOP>
        <PARAM NAME="Caption" VALUE="Read">
        <PARAM NAME="Size" VALUE="2000;682">
        <PARAM NAME="FontCharSet" VALUE="0">
        <PARAM NAME="FontPitchAndFamily" VALUE="2">
        <PARAM NAME="ParagraphAlign" VALUE="3">
        <PARAM NAME="FontWeight" VALUE="0">
    </OBJECT>
<BR><BR><BR>
<OBJECT ID="RpcOle"
CLASSID="CLSID:C4FAADB0-F779-11CF-A0AE-00AA00AC621F"
STYLE="TOP:173pt;LEFT:91pt;WIDTH:248pt;HEIGHT:83pt;...........
   HSPACE=10>
        <PARAM NAME="_Version" VALUE="65536">
        <PARAM NAME="_ExtentX" VALUE="8749">
        <PARAM NAME="_ExtentY" VALUE="3000">
        <PARAM NAME="_StockProps" VALUE="221">
        <PARAM NAME="BackColor" VALUE="16777215">
        <PARAM NAME="Appearance" VALUE="1">
    </OBJECT>
<HR><ADDRESS>
Back to <A HREF="http://hostXX.cern.ch/Welcome.html">PS/CO Home
Page</A><BR>
Contact <A HREF="http://xwho.cern.ch/WHO/people/08063">I. Deloose</A>
CERN/PS (6-Sep-96)
</ADDRESS>
</BODY>
</html>
```

The above HTML illustrates three important things. Firstly the <SCRIPT...> tag which enables VBscript to be included in the HTML document. Secondly the inclusion of normal Windows objects such as buttons. Thirdly the key object, "RpcOle" which is the OCX loaded to do the work.

Java and the Interpretive approach

The above approach is very powerful and Microsoft is promoting it strongly as "ActiveX". There are some problems, however, including the need to register the OCX control, the security problem of loading a binary into your computer, and the size of the binary.

An interpretive approach to network computing was used in the SPS control system to surmount these and other problems, and the result was the Nodal system for the SPS (reference 4). The interpretive approach has now been re-discovered, and the hottest item in computing today is "Java and the Web".

The interpretive approach has advantages for the user and for the system. For the user the advantages are an easy interactive approach to program development and a simpler language. For the system the advantages are the security offered by the interpretation, sometimes called running in a "virtual machine", and the network transferability of the code.

The requirements for an interpretive language do not exclude compilation. In fact "Just In Time" compilers for Java are already available whereas it took us from the SPS to the AA, five or six years, to get the "Nodiler" for Nodal into operation. The important thing is that an applet running over the network must be self contained, must need no parts pre-loaded into the target computer and must leave no parts behind.

The network transferability of code can be illustrated by the following example from the Nodal Manual.

```
1.1 WAIT-CYCLE 6
1.2 FOR I=1,6; EXECUTE (I) 2
1.3 DIMENSION A(108)
1.4 FOR I=1,6; WAIT (I); FOR J=1,18; SET
A(18*(I 1)+J)=B(J)
1.5 EXECUTE(DISP) 3 A
1.6 GOTO 1.1

2.1 DIM B(18)
2.1 FOR I=1,18; SET B(I)=HPOS(I)
2.3 REMIT B

3.1 SET ODEV=15; CLEAR; HISTO(A)
```

Line 1.2 shows how group 2, an interpretive applet is sent to each of computers 1 to 6, the so-called General Purpose computers at each of the SPS's six access points. In the remote computer the code is interpreted and the work done. Line 2.3 shows how the data is sent back. B is an object, in this case an 18 element array, but it is self-contained. It consists of a set of bytes containing its size, type, number of elements and then the data, so that it can be interpreted in the client computer on arrival. This example also shows parallel execution of the remote code, also a feature of Java.

Why develop a new language? Of all the myriad languages that have been devised to program computers relatively few survive, and popular wisdom holds that we need a new programming language like a hole in the head! The same was said in 1972 when we started with Nodal, although there were many fewer languages then. Nodal was developed from Focal, a DEC language for the PDP 8 (now we are on a history lesson!). It was necessary to ADD so many features that we gave it a new name which reflected its network capability. In the case of Java the opposite was the case! It started with C++ with its good things such as objects, but they had to SUBTRACT the things which could not be interpreted (and which made it bug-prone), such as pointers. Interpretive languages tend to be simpler as the interpreter must be resident and ready to handle any language construct, whereas compiled languages can be more complicated as no code will be generated if a language construct is not used. The result is that Java is simple and compact, yet powerful and object oriented. In my opinion, as a neophyte in the matter, it well justifies the hype it is experiencing at present.

A language can fail, despite its beauty, if there is not enough effort put into its support environment. This was the final end of Nodal and the PS developed language P+ at CERN. Even Pascal and Modula 2, also invented in Switzerland though not at CERN, are suffering from this problem. There are alternative approaches to bringing life to Web pages, for example Netscape's plug-in technology, Microsoft's ActiveX which is based on downloading OCX's, and Oracle's network loadable objects for database access. None of these use the interpretive approach, however, with the possible exception of Oracle's PL/SQL, nor do they enjoy the simplicity and power of the Java object oriented language. I would say that Java will make it. The biggest assurance for me with Java is that Microsoft have produced Visual Java ++ with all the goodies of Visual Basic and Visual C++. Microsoft's Internet Explorer 3.0 is fully Java enabled, and all the signs are that Microsoft will absorb it rather than fight it.

It is interesting to note that Java, like Nodal, started life as an equipment control language and system. It was originally intended, by its development team at Sun Microsystems, to allow a large, distributed and heterogeneous network of consumer electronic devices to talk to each other and coordinate activity. Ideally every device in the network could communicate with every other, for example the television could turn the kettle on when the favorite program is near to the end. There was no intention to release the language for general use. When the Internet "exploded" with the World Wide Web the Java team realized their product was ideally suited to the new environment as it would allow executable content to be distributed over the Internet. Java has become so successful that Sun hope to move it back into its original on-line use. Sun is predicting that their software sales of on-line control type Java based software will contribute more to their turnover than server sales by the end of the century. If this

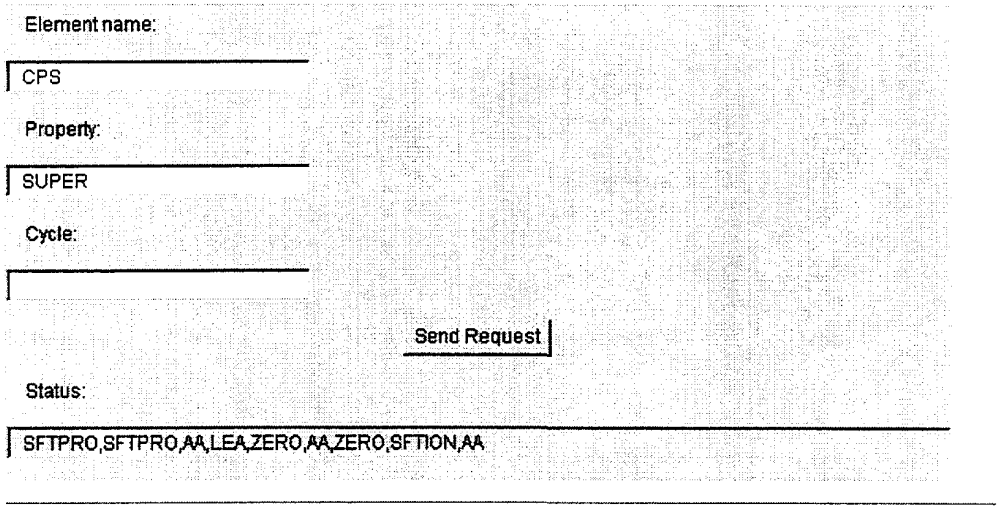
happens then maybe in accelerator controls we will see Network Computing spreading even below the control system into the equipment itself.

An Web Control Example using Java

This application was written by Eric Roux, again in the PS, as a simple example for this conference. It is being extended as a possible operational tool and it is hoped that the extended version will be demonstrated live. This example provides similar functionality to the ActiveX example shown earlier, but is implemented in Java. Let us start with the HTML which is shown below.

```
<HTML>
<HEAD>
<TITLE>PS Control</TITLE>
</HEAD>
<BODY>
<HR>
<APPLET CODE=PS_Control.class WIDTH=600 HEIGHT=300>
<param name=Host value="hostXX.cern.ch"></APPLET>
<HR>
</BODY>
</HTML>
```

Here the HTML is simpler, the page can be beautified later. The main thing to note is the applet defined by the <APPLRT...> tag. This is loaded and interpreted by the browser to give the output as shown below.



A small section of the Java code is shown below.

```
/*#########################################################################

      FILE          PS_Control.JAVA
      PROGRAM       PS_Control.CLASS
      VERSION       1.0
      AUTHOR        Eric ROUX & Ivan DELOOSE
      DATE          24/09/96
      PURPOSE       PS passerelle access from an applet.

#########################################################################*/

//============================
// imports
//============================
import java.applet.*;
import java.awt.*;
import java.io.*;


// PS_Control class declaration
public class PS_Control extends Applet
{
      private     TextArea          StatusTextArea;
      private     Button            ReadButton;
      private     Choice            TelegramChoice, PropertyChoice,
PLSLineChoice;

      // Initializes the applet.
      public void init()
      {
            // Set the applet background color & default font.
      setBackground(Color.lightGray);
      setFont(new Font("Helvetica", Font.PLAIN, 12));
```

The Java code is processed into a byte stream which can be interpreted. Although Java is more complicated than earlier interpretive languages such as Nodal and Basic, it has the advantages of being modern, simple compared to C++, and fully object oriented. On the interpretive side it has the advantage that the resultant byte stream is small. One program was 14Kbytes for the Java code compared with 447Kbytes for the Visual Basic compiled version (not quite identical versions, I must say, but similar). Of course an interpreter for Java exists for all computers which have a Java enabled browser; that means all computers.

The Server Side

The anchor point of any control system is of course the Front End Computer as we used to call it, now perhaps best called the Equipment Server in the new Network Computing terminology. This is where most of the money goes, at least the initial up-front investment, both in hardware and software. The equipment interface electronics does seem to last well, however, and the life-cycle cost is therefore lower. The most common problem is lack of follow-up knowledge and documentation.

An important is fact that the server hardware and software has to be ready at an early point in the accelerator commissioning cycle. Powerful readily available and well known hardware, together with an easy, rapid, and well known development and testing system is therefore called for - or so one might think.

What, however, do most control systems use? VME crates, with specialized complex processors, arcane operating systems and crude C or C++ development environments! All in the

name of better performance in a field where performance is increasing by a factor of two every 18 months and the accelerator may have a life of over 18 years!

For Isolde we used simple PCs with ISA bus interface, DOS, and Nodal plus Microsoft QuickC development environment. This worked and the rest is history. Nowadays we can have server PCs with four Pentium Pro processors and PCI interface. CERN has - adopted/accepted - PCI as a bus standard for LHC experiments despite its association with the common PC! More seriously there is no way small VME manufacturers can rival the research and development efforts of Intel and others in the PC field. The PC now has huge demands on its performance due to the advent of multimedia and 3D modelling, so performance is likely to be led by the PC for the foreseeable future. Such "commodity computing" is recognized as essential for HEP experiments.

More important, however, is the software and the software development environment. The PC is evolving rapidly in this area, especially with the "Visual" products from Microsoft, and again there is no way specialist operating system companies can match the resources of Microsoft to produce powerful and easy to use development environments.

All this applies to any type of PC control system, but the next section will show how it particularly applies to a "Network Computing" control system.

The Network Computing Server

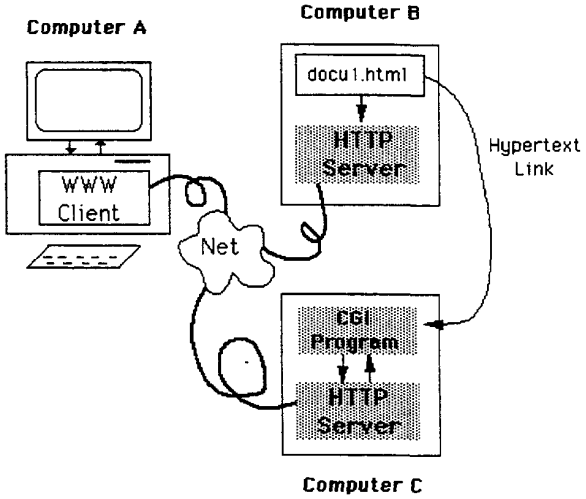
Most of today's control systems are based on the Remote Procedure Call (RPC) interface to the Front End Computer. The Isolde control system was so based and, as I will explain later, this was its (only?) weak point. We wrote our own RPC based on Novell IPX. Some effort has gone into standards for RPC but the usage level is small and convergence level low compared with the World Wide Web.

In the "Network Computing" control system the Front End Computer would be a WWW server and the protocol would be HTTP. The original World Wide Web servers were simply page servers, yielding up files in HTML, GIF and other formats on request, at the click of a hypertext link. A simple hypertext link in HTML has the form, for example:

<A HREF="http://consult.cern.ch/xwho">Directories</A>

where the word "Directories" is underlined and when clicked causes the browser to load from the Uniform Resource Locator enclosed in the quotes, i.e. http://consult.cern.ch/xwho

The first attempt at interactive work uses an interface called Common Gateway Interface (CGI). The diagram below shows an hypertext document on Computer B with a link to a file on Computer C that holds the CGI program that will be executed if a user activates the link. This link is a "normal" http: link, but the file is stored in such a way that the HTTP server on Computer C can tell that the file contains a program that is to be run, rather than a document that is to be sent to the client as usual.



When the program runs, it prepares an HTML document on the fly, and sends that document to the client, which displays the document as it would any other HTML document. Such programs are sometimes called HTTP scripts or "Common Gateway Interface" (CGI) scripts. Note that CGI scripts may be written in scripting languages (like Perl, TCL, etc.) or in any other programming language (like C++, Pascal, Basic, Java).

As implemented in HTTP 1.0 the above protocol might be inadequate for control use. The objections would be the same as those made

against the PC, too slow, not designed for the purpose, no multi-tasking, etc. Two reactions are possible: drop it and go back to workstations and VME, or put some effort into the critical areas, like we did on Isolde, to overcome the disadvantages and so reap all the advantages. For the WWW, however, we have been pre-empted and I don't think we have to do anything. Just read the following quotes from the WWW consortium:

FastCGI is a new, open extension to CGI that provides high performance for all Internet applications without any of the limitations of existing Web server APIs.

"A large proportion of the Web has always been data from all kinds of systems made visible on the Web through gateway servers," said Tim Berners-Lee, Director of the World Wide Web Consortium. "The first simple method of connecting custom gateways into existing servers was known as CGI.

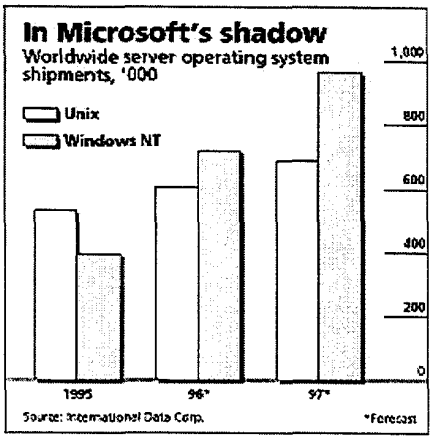
"CGI, whilst a common interface used by many servers, is intrinsically slow. The W3 Consortium is looking forward to the development of new open standards for this interface, and FastCGI is one promising proposal in the spirit of open standards."

Web server APIs increase application performance compared to CGI, but are vendor-specific, complex, language-dependent, difficult to maintain, prone to security risks, and inherently unstable. FastCGI delivers performance increases that are as great as -- and often greater than -- APIs without these compromises. "CGI, which was developed here at NCSA, has become an important de facto standard on the Web," said Elizabeth Frank, httpd Technical Manager at NCSA. "But it also significantly impacts the performance of heavily accessed sites. One response has been the proliferation of Web server APIs. Unfortunately, applications produced to a specific server API are rarely usuable with a different server. Applications using FastCGI will not have this problem. This is why we are happy to announce support for FastCGI in our next server release."

Even the current version of CGI has adequate capability to implement the Nodal system used for the SPS, and also of course its Java equivalent.

At CERN we are starting to use the Oracle Web Server approach to database access. This is, of course, vendor-specific and does not use FastCGI (yet). It is already showing a lot of promise. Microsoft are also announcing new developments in this field. There is little conceptual difference between a database and a control system equipment server, in the SPS we even called the equipment interface "data modules". Fast speed of response is required in both cases.

The number of Web servers on the Internet is increasing dramatically. Soon every organization, club, maybe even every shop, will need its own server. Originally the Web software was developed at CERN on a Norsk Data computer! With increasing popularity, of course, the focus moved to UNIX and most of the current implementations and documentation are UNIX based. With the explosion in demand this is changing and Windows NT is rapidly becoming the server software of choice, see the following graph thanks to the Economist:



One can be fairly sure that the latest, best and fastest Web software will be available for NT before UNIX, especially the more obscure versions of UNIX. The attraction is, of course, the lower cost of the hardware, especially the big disks and fast network cards required for commercial servers. For control systems already using PC Front Ends all we have to do is to upgrade to NT.

The Intranet and security

A frequent worry in control system design is security. An unscrupulous control system designer can scare his management into accepting some obscure software system by claiming it is more secure, especially as none of these "bad hackers" will ever have heard of it!.

Security is indeed a problem on the World Wide Web (WWW). Although it was only invented in 1989 at CERN, the Web has already become the primary information interchange mechanism on the Internet. The computing industry worldwide is working on solutions to provide WWW (Network Computer) shopping and even banking. There is a W3C consortium web page on security, http://www.w3.org/pub/WWW/Security/.

For control systems there are two approaches to security. The first is the "Intranet" and "firewall" approach. The Local Area Network (LAN) used at CERN is currently Novell Netware, so the Isolde Control system was designed to use this, the fundamental tenet of PC control systems being to use industry standard solutions where possible.

There are strong indications that the industry will move from proprietary LAN software (Netware, NT, Netbios, etc.) to the Intranet. An Intranet is just a local area Internet segment, usually using existing LAN hardware (Ethernet, ATM, etc.) but using standard Internet and in particular WWW software. This local segment is then isolated from, or rather connected to, the global Internet via a "firewall". The firewall serves two functions. One is to isolate the local segment from outside traffic, so guaranteeing the fast response needed for control systems. The second is to implement a security layer. Firewall systems are available commercially, though apart from fictitious problems as in the film, there are practical problems of speed and ease of use.

The second approach to security is authorization. In the SPS control system a user had to log in with a password and was allocated a 16 bit section/group identifier and a 16 bit capability word. Each bit in the capability word allowed particular actions to be done (properties to be accessed). This could also be checked against the individual or his section. For Isolde we used the Netware log in facilities. Credit card shopping on the Internet, banking, and Internet money will need much more secure authorization, and this is indeed being developed (see WWW page mentioned above). This could be adapted to control system use, maybe even in conjunction with a firewall. The firewall could use a heavy authorization process to allow outsiders in, but internally a lighter and faster solution could be used. It should be noted that the main utility of the SPS protection system was to stop unintentional errors due for example to mis-typing a Nodal command, rather than to stop malicious hackers.

The Global Scaling Philosophy

This chapter contains the main message I want to get across, and is perhaps the most revolutionary proposal in the paper. When we were building the ISOLDE control system I regularly asked the question "would it scale if we were building a control system for the whole world". In the end this objective was not met, mainly because of the database required for the Remote Procedure Call software. The old SPS control system would have scaled to the whole world, and so will the Network Computing based control system proposed here.

The Internet and Intranet are based on a whole new philosophy. They represent a move away from targeted information distribution towards active "information seeking". So it is with the Network Computer based control system. The control system does not exist in any one place. It exists only as the amorphous sum of all the applications programs. This is a difficult concept to understand, and also difficult for control system owners to accept as it makes it hard for any one person or group to claim ownership.

Let us look again at the SPS control system. Each General Purpose or Front End Computer had its own connected hardware and its own set of "data modules" to drive them. The only connection to the outside world was through Nodal programs. Some of these programs set initial values, others set operational conditions, others took measurements and gave feedback. Any of these computers could have been part of

an enormous set controlling all the hardware of all the accelerators in the world, and even everything else in the world! It would all be connected to the same network, e.g. the Internet, and all controlled in the same way, of course with different suites of Nodal programs for different applications. The integrating function for any one accelerator or application is only its own group of Nodal programs, plus the security and permission facilities to ensure correct use.

At first this led to confusion as the number of programs grew and grew. Then organization was put in place and the programs were arranged in a "tree" structure. The typical "node" program consisted of the lines:

```
2.1 FOR I=1,17; $SET LEGEND(I-1)=LEGS(I)
2.2  SET Z=BUTTON; NEXT(PROGS(Z+1)
```

and the two string arrays LEGS and PROGS which contained touch button legends and the program names to be called. This tree structure allowed service programs to be written which scanned the tree and gave documentation of all the programs that made up the control system. Of course the documentation was always "chasing after" reality, and it was always a great debate as to whether this approach was not more realistic that the more obvious "carefully pre-planned and pre-documented" approach.

HTML with its hypertext links implements a kind of branching "tree-leaf" structure essentially similar to the SPS "tree" above. "Web Crawlers" to search and track the links are already in evidence. In the beginning we were always asked why there was not a Web directory like a telephone directory where you could "find everything". Now, after two years, we are not asked that question anymore and it is being accepted that "post facto" services like Digital's Alta Vista are the way to go. Similarly we hope that control system designers will accept the SPS approach and abandon the chimera that all accelerator details and application programs can be carefully (and ever so heavily) planned in advance.

The Role of the Network Computer

Will the Network Computer as such have a place in our control systems, displacing the PC? The answer is yes and no! If Microsoft's ambition to embrace the Web is realized it may be difficult to distinguish between a Network Computer and a PC. If all documentation is HTML and the PC desktop is a Web page, then PC control systems will clearly follow.

The big unknown is whether the Java applet approach to spreadsheets, word processors or application scripting systems will ever rival or replace the standalone Excel, Word or Visual Basic elements of today's PC. Not in the immediate future, that's certain.

Nevertheless a Network Computing control system could come quickly. The "real time computing" community are finding X11 too heavy for browsing applications (reference 5), and there are proposals to use Web serving instead. It is even easier to add Web serving to an NT equipment server. So simple applications could be handled by a Browser and more demanding applications could use the existing RPC techniques until the Web techniques such as Fast CGI become adequately developed.

This could bring convergence with the UNIX workstation control systems. Most applications could be handled by the Browser on either UNIX or the PC, whereas C on the workstation or Excel and Visual Basic on the PC would handle the "legacy" applications in "traditional" VME or PC control environments.

As many of the "simpler" applications could quickly be converted to Network Computing applications, then a simple Network Computer could handle all the local control, public terminal, type applications, and a Browser could suffice for office access to the accelerator.

Will it ever happen?

Control system design for accelerators seems to have changed from a pioneering subject to being as conservative as controls for the chemical industry (some would say even more conservative). This is understandable from the confidence point of view, but not when there is a shortage of money and manpower.

The pioneering developments such as Nodal for the SPS were copied and further developed at

the PS, Desy and KEK to good effect. They never made it across the Atlantic, however. When PC control systems were first proposed there was enormous resistance at CERN. It took a lot of courage to decide that Isolde, even though a rather small project at the time and very short of money, should be used as a test-bed for a PC control system, against the established wisdom of the Controls fraternity. Subsequently this was adopted by Desy and so we are all here today! Nevertheless the PS has gone back and invested in over a hundred AIX workstations rather than PCs, and is continuing with VME front ends. In LEP, where PC front end computers were installed "in extremis" to get LEP going, they are being replaced by VME systems! So not everyone is convinced about PC control systems!

What then is the chance of not a PC or NC based Network Computing control system seeing the light of day. Perhaps in time for the next PCaPAC conference someone will have the courage to try.

Conclusion

Networking Computing, with or without a specialized Network Computer could well be the next logical step in PC control systems. The essence of the PC control system is to harness the huge investment in hardware and software by firms such as Intel, Microsoft and Oracle for use in control systems, rather than use home brew or small production vendors. It looks increasingly probable that the main thrust of PC development will be towards network computing and the Intranet, so the "true" PC control system should follow.

Hardware will not change much. Four processor Pentium Pros with PCI bus interface already make the front end look almost overpowered! At the client level we may just see the specialized Network Computer replace the PC for auxiliary terminal type applications in maintenance and local operation.

Software will change the most. IP will remain but HTTP may replace TCP and RPC. JAVA may replace C++, and HTML may replace Visual Basic as the normal scripting language.

Above all convergence between the control system world and the world of everyday computing will increase, embracing not only the proficient PC user as with today's PC control systems, but also the whole Internet proficient world of tomorrow which will comprise wives, children, and who knows, accelerator physicists.

Acknowledgments

I would like to thank Ivan Deloose and Eric Roux of the PS at CERN for the examples of Network Controls, the Financial Times and Economist for the relevant images, and my colleagues in the Database section at CERN for the background in Oracle and Network Computing.

References

[1] A. Pace, "The CERN PC Network", this conference.

[2] R. Billinge, A. Bret, I. Deloose, A. Pace and G. Shering, "A PC Based Control System for the CERN ISOLDE Separators", ICALEPCS '91, Tsukuba, Japan, November 1991

[3] I. Deloose, "Integrating the New Generation of ISOLDE Controls into a Multi Platform Environment", this conference.

[4] M. C. Crowley-Milling and G. Shering, "The Nodal System for the SPS", CERN 78-07

[5] J. Blake, C. Eck, M. Merkel and L. Pregernig, "Real-Time Operating Systems at CERN", CERN/CN/96/11, VITA Europe Congress, 8-9 October 1996.