Towards a New Generic Approach for Web Access to Control Data

F.Momal
LHC Division, CERN, 1211 Geneva 23, Switzerland

Abstract

For more than two years, we have offered Web remote access to the data from our supervisory and control systems. All the computing for accessing and presenting the data was previously done on the Web server. During the past two years, Internet/Intranet technologies have greatly evolved. At the same time, more and more industrial control and supervisory systems now offer Internet interfaces.

The structure of our remote access system has conjointly evolved. We aim to install specialized servers that will offer access to different kinds of data. Whenever possible, we use the new Internet components offered by the installed industrial products. In the other cases, developments have been made. On the client side, we use new technologies, such as Java.

1 Towards a new approach

Our group is involved in the control and software supervision of the LHC superconducting magnets tests. For more than two years, we have offered Web remote access to the data from our supervisory and control systems[1]. In a simple and unique way, users can access real-time data, process alarms and archived data. They use the Web access system from home or from the office to follow the tests, to do post-mortem analysis, etc. This service has encountered great success and is now seen as a mandatory part of the supervisory systems.

The first system based on the technologies available at that time intensively used the Web server to access data and to dynamically build the presentation pages (trend curves, graphical synoptics, alarms lists, etc.). The software gateway installed on the server had responsibilities for both acquiring the data from the field and creating the graphical presentation of it. The generic approach of the gateway enabled it to reuse all the different programs available to access the control data.

During the past two years, Internet/Intranet technologies have greatly evolved. The client side (Web browser) is no longer just a displaying tool but it has also acquired the capacity to run many kinds of programs. The Java programming language is now available and the most common desktop software can access Web data. How can these new features help us, and are there any disadvantages?

By bringing the computing power on the client side of the Web, it is now possible to distribute the system. The client is now able to acquire the necessary control data himself, and present it in sophisticated ways.

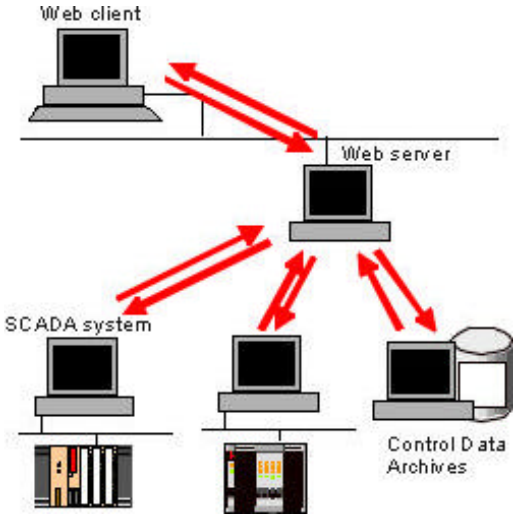

Fig. 1 The old system.

Thus specialized data servers may be installed near the controlled processes. Their goal being limited to transfer the value of some variables, they are lighter and simpler. But to do so can create some problems:

1) Security features are harder to manage due to the fact that more machines are involved.
2) Standardization problems appear for the data transfer between the client and the distributed servers. Depending on the accessed system, different programs may be needed on the client side[1].

Such a system is less prone to errors. Because the data exchanges do not go through a unique gateway, an important bottleneck disappears.

The network load is lowered, because only the value of the control variables is transferred (the only moment when the network traffic is heavy is when the user opens the Web page where the program can be found). In the previous system, complete images were transferred continuously. Obviously, a gain in speed could also be acquired.

The central Web server would no longer be involved in control data acquisition and presentation, but would just contain software and related information. Thus, the load on this server would be greatly reduced and it would result in less waiting time.

The most obvious interest is of course in program interactivity. The user has now the ability to modify the data presentation without interacting with any server. It is also now possible to take actions on the client side such as

[1] Programs refer, on the Web client side, to Java applets or ActiveX components.

sounding an alarm when an error occurs, etc.

From what precedes we see that there are many advantages in the new approach. But the cost for moving from a 100% server approach to a more client oriented one is not low. Programs have to be rewritten. The client machines must be more powerful and they must be able to use the new technologies involved (Java, etc.). One has also to take into account the possibility of having different versions of browsers on the different client machines.

In any case if one wants to adopt the forthcoming Internet solutions offered by the SCADA industry, one has to move towards a more client side approach.

2 The Internet solutions offered by the SCADA industry

Logically wishing to integrate Intranet/Internet technolo-gy, SCADA systems vendors are starting to offer extensions compliant with the standards involved. But each has its own approach and the result is a rather anarchic offer. We have noticed the following approaches:

1) The first and simplest one is to transfer "screen images" across the network. A Web server, closely tied to the supervisory software converts on demand the graphical output of it into bitmap images and sends them back to the Web client (ex: Labview[tm]). This solution can be very convenient and is very easy to set up. But it is a closed solution. One can visualize the data but can't do anything else with it. It is also demanding in terms of computing resources and can dangerously overload the machine on which the server and supervisory software is installed.
2) At the other end, one of the products we saw offers total control of the SCADA system via ActiveX (FactoryLink[tm] from USDATA[2]). It ports the operator console in a Web browser. This can also be very convenient in some cases. It can be seen as the Windows equivalent to an Xterminal running the SCADA system in the Unix environment. But this option doesn't really respond to our needs. We don't want to offer a very limited number of people full control of the system (a license is in that case needed for each client) but rather a more synthesized view to a large group of people. It is also a closed solution and doesn't offer programmatic interface.
3) Some vendors intend to propose "light clients" using ActiveX. These clients offer the users limited functions. For example, an ActiveX object could be dedicated to viewing the alarms or the data stored in the archives of the SCADA system. This is effectively a lighter solution but, like all ActiveX solutions, it is dedicated to a unique platform.
4) Rarely, some products offer an open Java based interface which allows simple integration.
5) Of course the "old solution" of a classic TCP/IP server using the sockets can be easily integrated as long as the protocol is known. Java applets can access the server.

Some of these approaches are more open than others. The first three can't be integrated in a global system. They lack a programmatic interface. But, on the other hand, it is an off the shelf solution which can be used immediately. The diversity is certainly due to the immaturity of these technologies. We can hope that one day a standard will be adopted by a majority of vendors (for example CORBA to communicate between distributed objects)

3 An attempt at a generic and distributed system

Our system must address the process experts as well as physicists, managers or system experts. Thus we need to access, in a similar way, data coming from a broad range of systems such as supervisory systems, archived files, databases, etc. The formatting of the data is also diverse to reflect the different needs: it can be trend curves, operators' console like graphical views of the process, tables to be inserted in Excel, etc. To cope with this diversity, we must have a generic approach.

3.1 The structure of our system

Our system is composed of a central server and a set of decentralized data servers. The central Web server has three main functions:

1) It stores the HTML pages and the Java programs.
2) It acts as a location server: it keeps information on the locations of the data servers.
3) It stores information concerning data visualization.

So, for example, for a particular process it keeps on one side the name of the data server which handles the real-time data related to that process, and on the other side the description of the way the data has to be graphically represented.

Local data servers are attached to a process. We have defined three types of data servers (see fig.2): real-time data servers, alarms data servers and archives data servers. These servers collect the data which is to be published on the Web. They manage all security features and control the load on the control machines themselves. The three servers may of course coexist physically on the same machine.

3.2 Standardizing access to the data servers

Whenever possible, we try to standardize the access to the different data servers. When integrating closed solutions offered from the SCADA vendors, this is not possible and we only integrate them in the HTML pages. In other cases, standardization may be achieved on the server or on the client side. For the archived data servers, because we use our own software, this is done on the server side. By doing so, we allow unique access to this server from any software, linked to the Web or not.

For the other servers, we minimize as much as possible the number of protocols used. We currently accept two protocols:

1) A simple protocol based on httpd. It is not the most

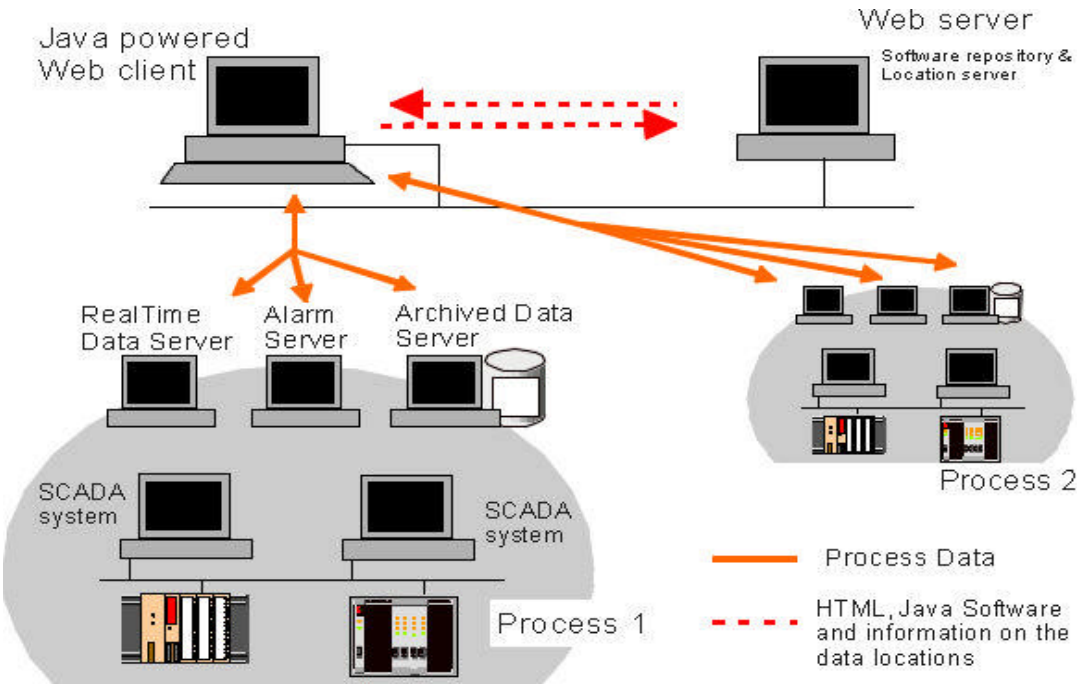

Fig. 2 The structure of the our new system.

efficient but it is very easy to implement with any kind of supervisory software. Real-time values are returned as a simple list of names and values.

2) A socket based protocol defined by the vendor of a product we have purchased (MD2S[tm]). This program, which runs on the data servers, gathers data in a real-time database. It offers local and remote programmatic interfaces (Java classes, C++ library, Excel interface, etc.).

On the client side, a Java interface related to data access has been defined. It hides the diversity beneath.

3.3 An applet to visualize control data

On top of the client side Java interface, an applet to visualize control data has been developed. The user can interactively define the data he is interested in and how he wishes to view it. He may then save the defined configurations on the central server to retrieve them later from any computer. This applet is also able to interpret the graphical language used by the previous system to animate supervisory displays (see an example on fig.3). This language tells to the graphical engine of the applet how to go about viewing the data. With a simple text editor, any user may easily describe the way he wants to view the data. Putting this description on the server allows everyone to immediately use it. The applet can load the description from the server and animate it after connecting to the proper data server. Because the access protocol to this data server has been hidden beneath a Java interface, the same description may be applied to any kind of data server.

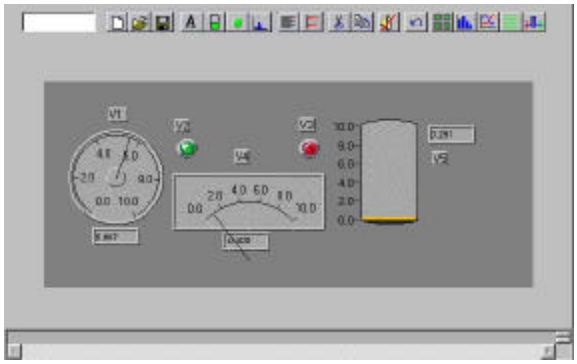

Fig. 3 An applet to visualize control data.

3 Conclusion

The diversity of the solutions offered by the SCADA industry is too wide today to allow easy installation of a generic system. Important developments have still to be made. But standards are on their way and we can hope that their adoption by many will once again simplify our work like the Web did.

[1] F.Momal, C. Pinto-Pereira, "Using World-Wide-Web for Control Systems", ICALEPCS'95, http://wwwlhc.cern.ch/ICALEPCS95/icalep95.htm

[2] http://www.usdata.com/