ARCHITECTURE FOR INDUSTRIAL MONITORING SYSTEMS

Assoc.Prof.Dr.Eng. Sorin Aurel MORARU
MSc.Eng.Drd. Adrian PELCZ
Senior Lecturer Drd.Eng. Liviu PERNIU
Assoc.Prof.Dr.Eng. Adrian CRACIUN

Automatics Department, “Transilvania” University of Brasov, M.Viteazu street, no.5, 500174, Brasov, Romania, phone: +40 0268 418836, smoraru@vision-systems.ro; apelcz@vision-systems.ro; roxanab@unitbv.ro; craciun@vega.unitbv.ro.

Keywords: industrial, monitoring, multi-tier, architecture, communication

Abstract*: This paper presents the architectural model for a multi-layered industrial monitoring system. The main trend for all enterprise applications is guided by transforming the applications into web-accessible, distributed applications. Modern industrial equipment has features that allow them to be connected to computers using serial communications, and so, many old monitoring systems were replaced with computers running specialized software. The problem with these computers is accessibility. The computer is connected to the monitored industrial equipment and usually is located in the proximity of that equipment. This substantially reduces the usability of the information obtained from monitoring to the place where that computer is located.*

By combining the IT trend of building distributed applications based on multi-tier architectures with the modern monitoring software from the industrial systems we have developed a distributed application system for monitoring industrial processes.

1. INTRODUCTION

Many industrial monitoring systems nowadays are still tightly bonded to the place where the industrial process is located.

One of the most common architecture is three-tier. Using this kind of architecture, the applications are distributed on three levels: client, server application and database. This is very suitable for pure software applications such as accounting or management systems.

Modern industrial equipments have features that allow them to be connected to computers using serial communications, and so, many old monitoring systems have been replaced with computers running specialized software. The problem with these computers is accessibility. The computer is connected to the monitored industrial equipment and usually is located in the proximity of that equipment. This substantially reduces the usability of the information obtained from monitoring to the place where that computer is located.

By combining the IT trend of building distributed applications based on multi-tier architectures with the modern monitoring software from the industrial systems we have developed a distributed application system for monitoring industrial processes.

The architecture consists of four levels and it is presented in figure 1. The four levels are:

- hardware;
- data acquisition;
- server;
- client.

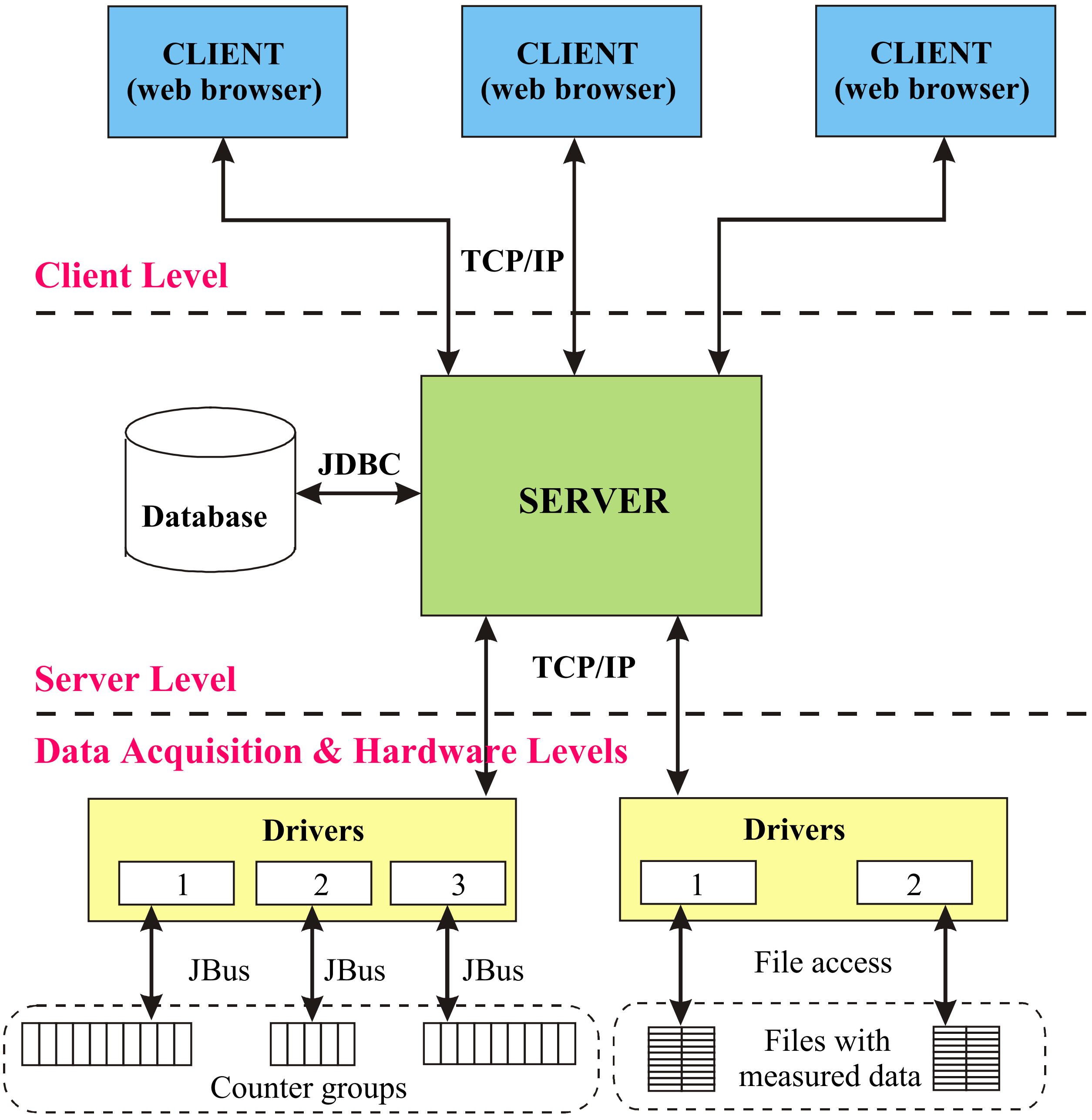


Fig. 1. Architecture for industrial monitoring systems

2. LEVELS DESCRIPTION

2.1. The Hardware Level

This level it is represented by the monitored equipment, the serial connections, the computer and all the auxiliary hardware needed to make the software able to communicate with the equipment.

From the monitoring system's point of view, it does not matter what kind of hardware is being used. The system is based on the fact that the hardware level is able to transform and transport the raw measured data into a computer. This can be done in various ways (from the computer point of view, the connection to the industrial equipment can be made by a serial or parallel communication port, by a specialized data acquisition board, or another type of communication equipment).

For the system that we have already implemented, the hardware level was represented by sets of industrial electronic counters for 6.4 KV power cells. The counters are linked together into a serial RS-485 network. Because the distance from the counters to the computer was quite big (500 m) and the electro-magnetic interferences on the cable path were strong, we have decided to transport the data through a fiber-optics (FO) cable from the counters to the PC. For this we used two RS-485 / FO converters to extend the RS-485 network using FO. The FO end located near the computer was transformed back into copper RS-485 and afterwards into RS-232, using a RS-485/RS-232 converter. The RS-232 cable was plugged into the 9-pin serial port of the computer that does the data acquisition.

So, in the example above, the hardware level is composed by: the industrial data acquisition equipment (the counters), the RS-485/FO converters, the copper and FO cables and the RS-485/RS-232 converter.

2.2. The Data Acquisition Level

As everything that belongs to hardware in the industrial sense is found on the hardware level, the three remaining levels are 100% software.

So, on the data acquisition level we find a software program. This program, as all the four levels has the role of transforming and transporting information. It is basically a translator between the hardware level and the server level. It has three main roles, which practically describes the business logic:

- Acquires the data from the hardware level, using the communication protocol provided by the hardware (serial communication, driver for DAQ board, etc);
- Processes the raw data received from the hardware, and packs it into telegrams for sending it to the server level.
- Sends the processed information to the server level, using a network connection (local area network (LAN) or an Internet connection).

In the system we have implemented, the hardware equipment is connected to the computer using the 9-pin serial port. The communication protocol used by the counters is JBus, a protocol based on ModBus. The software program developed to serve as Data Acquisition level is written in Visual C++ and it reads the data from the serial port, processes it and it sends it to the server, using a TCP/IP connection over the LAN or the Internet.

2.3. The Server Level

This level is the most important. The server is an application that centralizes and unifies all the data acquired by the drivers from the different locations. The technology we used to develop our server was Java, version 1.4. For the database connection we used JDBC (Java Data-Base Connectivity). The database server used in this project is MySQL, but it can be any other database server. The server has the following main roles:

- Collects the data from the drivers and retains it into a buffer;
- Generates the medium, minimum and maximum values for each measure for retaining them in the database (separately for the current and previous day, for the current month and for the past months);
- Maintains the connections with the client applications to which it sends the data, depending on their requests (instant or history values);
- Has connection management mechanisms for the driver and client programs;
- Analyzes the data received and interprets the timings in order to detect errors in the monitoring system's functionality.

The database attached to the server retains information regarding the monitoring system, such as:

- User names and passwords for the persons who are allowed to use the monitoring system (the access to the system is secured);
- Equipment names (in our case, the counter description);
- Group names and belonging equipments (the equipments can be grouped in one or more groups);
- The minimum, average and maximum values for retaining the time evolution for each measure of any monitored equipment with different sampling periods (20 seconds for values in the current or previous day, 10 minutes for values from the current month and 2 hours for values from the past months).

From the network point of view, the server is a program that listens to a TCP/IP port and waits for connections. The connections can come from driver of from client programs. The server opens a new connection and a new thread for each connection. A connection is maintained active until the server or the connected applications closes it.

The server has a robust connection management system, which allows it to run stable under any circumstance, independent of the connected applications behavior.

The server has an advanced information and data management system, optimizing the data processing and storing by sampling with different periods depending on time. By using a large input buffer, the system can manage any configuration of information request from clients and temporary store in all the received data.

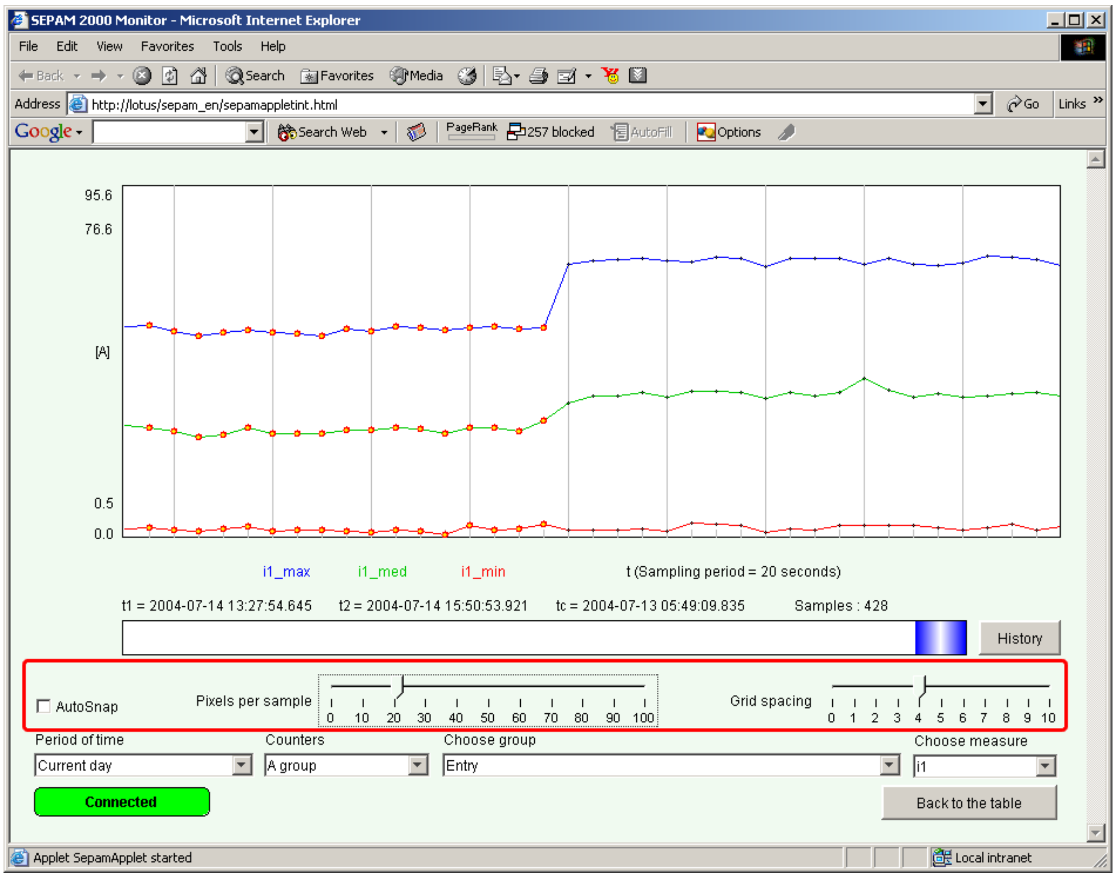
2.4. The Client Level

On this level is the client application, which is accessible using a web-browser. Like this, any user with the right privileges can enter the monitoring system from anywhere using the local network or an Internet connection and can have a global view of the monitored network of equipments.

For the client application we recommend the Java Applet technology. This allows the client application to be embedded into a simple HTML web page, making it accessible from anywhere. The Java technology also provides very good portability, enabling the client application to run from a large variety of operating systems and browsers.

These two features (accessibility and portability) are the two most important features for the users. Using Java Applets, the client application can be just as any other kind of local application, from the complexity point of view. When it is first accessed, the applet code is downloaded on the client machine and it is run. From the second time forward, the code is being run from the browser cache, this making it accessible very fast. We can look at the Java applet like at an application, which installs automatically from the LAN or the Internet and runs into a web browser.

Below, in figure 2, we present a screenshot with the client application developed for our project:



Address: http://lotus/sepam_en/sepamappletint.html

t1 = 2004-07-14 13:27:54.645   t2 = 2004-07-14 15:50:53.921   tc = 2004-07-13 05:49:09.835   Samples : 428

Period of time: Current day
Counters: A group
Choose group: Entry
Choose measure: i1

Applet SepamApplet started

Fig. 2. Screenshot with a client application developed using Java Applet technology

The client application connects to the server like the Driver, using a TCP/IP connection. This makes it accessible in the Internet or Intranet.

The user can choose that kind of information he wants to see. In the image in fig. 2 are four combo-boxes from which the user can choose if he wants to see instant values or history values (and for these last, which period of time), what kind of grouping for the equipment (all, a group, a single equipment), the specific group or equipment and the measure (currents, voltages, consumed energies, frequency, etc.).

3. CONCLUSIONS

This architecture, although is very complex from the implementation's point of view, brings huge advantages in production. From these advantages, we point out a few important ones:

- Flexibility - the system's levels can be geographically placed anywhere;
- Stability - this system is stable even under problematic circumstances such as a driver failure (the system will signal the problem and work correct for all the other components);
- Accessibility - using the Java technology we harvest it's benefits in terms of portability (cross-platform, cross-browser) and accessibility (by using TCP/IP connections it is widely accessible);
- Security - the access to the monitoring system is secured using user accounts and passwords;
- Speed - it may seem that this system is slow, but it is just the opposite. Due to the optimizations we made, the system can process hundreds of measured values per second, coming in asynchronously from several drivers, and being watched simultaneously by several clients;
- Data storage - we have developed a powerful database storage mechanism, with variable sampling periods, which will allow the time variations for all the measures of any equipment to be stored for years without overloading the database;
- Low cost - using free and open-source technologies such as Java and MySQL, the running costs are very low, as there are no third-party software components to buy.

4. REFERENCES

[1] Flanagan D., *Java in a Nutshell,* O’Reilly, 1998
[2] Taylor A., *JDBC – Database Programming On The Internet*, Informix Press, 1997.
[3] *Sepam 2000 – Metering and protection Functions*, Merlin-Gerin.
[4] Geary D.M., *Java – Mastering the JFC*, The Sun Microsystems Press, 1999
[5] Horstmann C., Cornell G., *Core Java Fundamentals (vol I and II)*, Sun Microsystems Press, 2001
[6] Sun Microsystems, *Java 1.4.1. Electronic documentation*
[7] Microsoft, *Visual C++ .net MSDN*