REMOTE ACCESS AND CONTROL FOR AN INDUSTRIAL SYSTEM

Ion TRUICAN
Stefan DAN
Adrian Virgil CRACIUN
Daniel BUCATOS

Automatics Department, "Transilvania" University of Brasov, M.Viteazu Street, no.5, 500174, Brasov, Romania, phone/fax: +40 0268 418836,
itruican@yahoo.com, dan.stefan@unitbv.ro, craciun@vega.unitbv.ro,
daniel_bucatos@yahoo.com

To be able to access and control the Industrial System remotely we are presenting here a model based on Java RMI (Remote Method Invocation). To ensure persistence of data using JDBC (Java Database Connectivity) the data are stored into a database server. The real world entities could be reflected on developed classes using the object oriented programming advantages.

Keywords: Java, RMI, Java Database Connectivity, Object Oriented Programming

1. INTRODUCTION

On our days it is very useful to have the possibility of access and control a system remotely. This future is need it when the presence of personal is not 24 hour required on site.

Every complex system can be divided into small sides, those are fully functioning independently. It is possible to have a complex system which can be access remotely and components those assures the functionality can be analyzed separately.

Using Java RMI (Remote Method Invocation) it is possible to design a system architecture which can be access and controlled remotely. Based on client – server architecture the user can access applications those are developed from the same machine where the server is running, or he can do this from another host located somewhere on the network. The client can invoke methods those are developed and published on the server side. When someone is trying to access the server remotely it must be taken into consideration that because of unknown reason the connection can not be establish because of some network problems for instance. This drawback should be taken into consideration regarding the risks those arise if the network connection is lost. But we can think on the remotely connection as a possibility and this should not be a must to ensure the functionality of the system as a whole.

2. PRESENTATION OF THE CONCEPT

The system will work independently of the remote connection. The client can run on the same machine as the server does or it can be somewhere else.

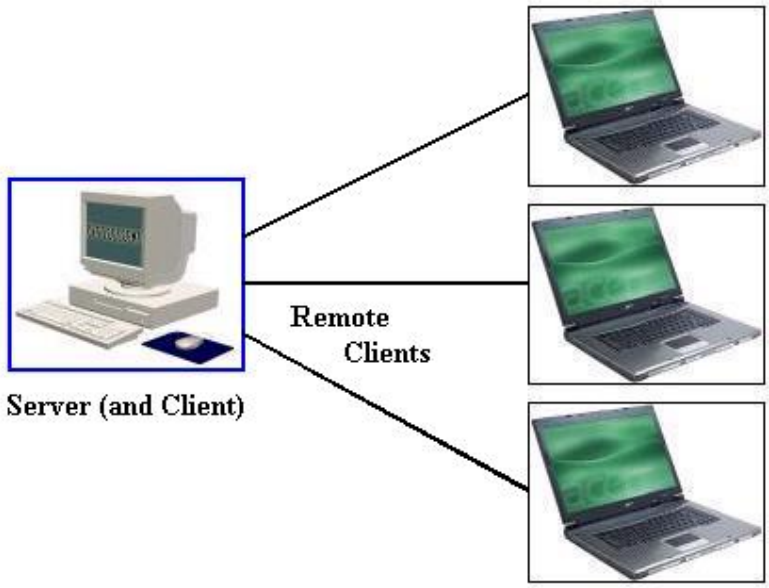

Fig. 1 Client – Server Architecture

To be able to see the history regarding the functionality of the system in most of the cases there should be a database server connected to the application to ensure the persistence of data. In this way we have the possibility to see the system activity for a long period of time; we can study the system functionality from a statistical point of view. When we are choosing the database server we have to take into consideration some criteria for selection:

- Security settings;
- Possibility to backup the database;
- How many concurrent connections are supported;
- Are the database entries easy to manipulate: update, insert, delete;
- Possibility to execute transaction rollback.

We can install the Database server on the same machine where the application server is running, but in this way we can expect some delays in functionality because of resources limitation. If we are taking into consideration the risks those arise when, considering the worst case this machine is damaged, then we have both the application and database servers down and there will be more work to be done to restore the data. It is more easy and safe to install the database server on a separate machine. The connection between database server and our application server can be established using:

- A Local Area Network;
- A Modem;

- Or the Internet.

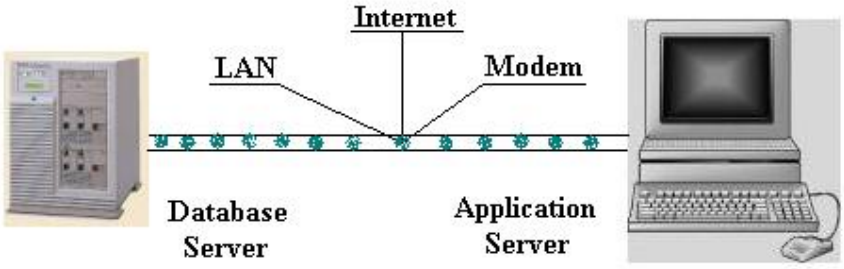

Fig. 2 Database Server - Application Server connection

Using Java RMI with this architecture the database server should provide a JDBC (Java DataBase Connectivity) driver, which can be used to connect to database. Developing the application using Java language, we can assure some advantages those are inherited. Java was designed to be platform independent and in this way our server and clients can run on any computer regardless on the operating system, which can be Windows or Linux as well. Using the Thread mechanism, every client can be served when is trying to access the application regardless of any other client connections. Java is an object-oriented language and we can structure our code in such a manner that we can reflect the real world objects into developed classes.

To restrict the access to the application we have two possibilities: hardware and software. When a client is trying to access the application he has to provide a password to be able to continue. Because we have there a remote connection and the password will be send it through the network we have to encrypt it to be sure that an unauthorized person is not able to use it even that he has intercept it. To check the password to see if this is matching we can specify the right answer only once in our code. This could be the simplest way, but is not so efficient. After the source code is compiled, we are getting the application classes and if someone at one moment wants to change the password, then we have to modify the code and re-compile the whole application again, and this is not so nice. There exist some other possibilities to store the password and this does not imply to re-compile the code if we want to do some changes. We can store the password in a specific table into database, assigning a common password for all users or specific ones for every connection. Another possibility is to put the user and passwords into property or encrypted files and read it from there afterwards.

The hardware solution of restricting the access to the application is to deny the connection from unknown IP addresses. We can put the application behind a firewall. Moreover there can be created a VPN virtual private network between server and clients and in this way we can ensure that only authorized user has access. We have to take a look now to the last part from our architecture, the process, which must be accessed and controlled. Depending on the complexity of the system, we have to

communicate with only one device or if it is necessary the communication must be implemented for many devices. As a general overview we can look at the process as a black box (sometimes this is delivered in this way to the client because of proprietary devices), there can be a main device, which is used for communication.

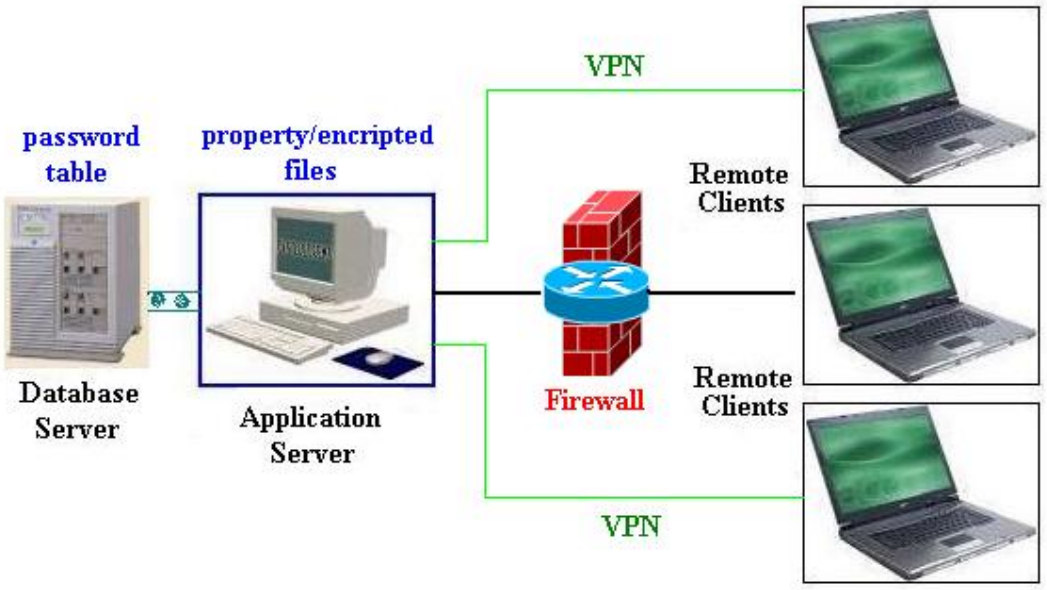

Fig. 3 How to restrict the connection

To establish the connection between the server machine and process device we can use:

- Serial port;
- Parallel port;
- Ethernet interface;
- Infrared port (IrDA);
- Bluetooth.

Because we are working into industrial environment we can use the serial port for communication and there exist predefined classes those could be used for an RS232 connection. When we are writing the code, we don't have to concentrate the attention on the lower level of communication, because this is already implemented, we just have to pay attention for the send it and received data.

The protocol used for communication is mainly implemented on the process device side and we just have to adapt our code to this. The data could be collected at predefined time interval from that device. Information is taken from the process and stored into database and then send it to the clients if they need it, in this way the client is never interacting directly with the process only via server.

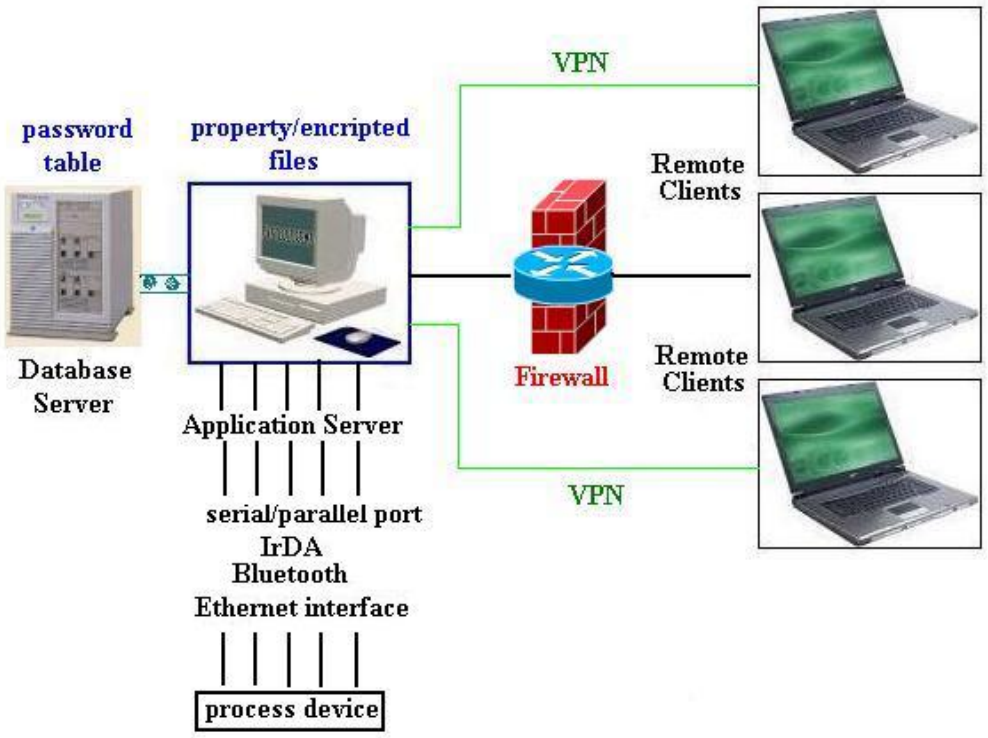

Fig. 4 System architecture

3. CONCLUSIONS

With this architecture we can split a complex system into small pieces those are fully functioning and at the same time there exist the possibility to add some new implementation for future requirements. The server side is the main entry here and it represents the central part. The database server ensures the persistence of data and JDBC is used for communication. Using RMI the clients are able to communicate with server using the Internet.

4. REFERENCES

[1] Fielding R.T., Taylor R.N., *Principled Design of the Modern Web Architecture*, ACM Transactions on Internet Technology, Vol. 2, No. 2, pp. 115-150, 2002

[2] Barnett B., Kirtland M., Ganapathy M., *An Architecture for Distributed Applications on the Internet: Overview of the Microsoft .NET Framework*, Microsoft Corporation, 2003

[3] Hill J.C., Knight J.C., *ANDREA: Scalable Command-and-Control of Distributed Applications*, Department of Computer Science, University of Virgina, 2002

[4] Gilmore S., Palomino M., *Monitoring and Controlling Distributed Applications with Relocatable Objects,* Proceedings of the 2n IEEE/ACM International Symposium on Cluster Computing and Grid, 2002

[5] Yeung K.C., Kelly P., Optimizing Java RMI Programs by Communications Restructuring, Department of Computing, 2001

[6] http://java.sun.com