XVII IMEKO World Congress
Metrology in the 3rd Millennium
June 22–27, 2003, Dubrovnik, Croatia

THE MONITORING AND CONTROL NETWORK SYSTEM OF A ENZYMATIC PROCESSOR VIA THE TCP/IP PROTOCOL UNDER OS LINUX

Miroslav Matýsek, Milan Adámek, Petr Neumann

Department of Automatic Control, Institute of Information Technologies
Tomas Bata University
Zlín, Czech Republic

Abstract – The article deals with a monitoring and controlling system based on the client-server architecture and communicating in frames of TCP/IP communication protocol. That system is applied for monitoring and controlling of real technological processes like fermentation temperature in local dimension area without direct connecting possibilities. The designed structure and its realisation has been proved in a real experiment with chromium tanned leather rests recycling process.

Keywords: Monitoring and controlling system, Linux, TCP/IP protocol.

1. INTRODUCTION

The goal of this work is to create a monitoring system capable to sense, collect and distribute data relevant to a real technological process. Analysing the most efficient structure of the system, the client-server architecture seemed to be the best variant. It fulfils the requirements on the separated sensing, collection, distribution the presentation of data.

The basic division having been made delegated the sensing to the first client, data collection and distribution to the server and presentation to the second client. The necessary freedom is ensured for the environment selection for any of the parts mentioned above.

Another contemplation concerned the choice of proper operating systems for the server realisation. The relevant criteria were:

- simultaneous execution of several programs
- support of communication among processors
- network communication means based on the wide spread TPC/IP protocol
- adequate hardware demands

The underlying criterion was the complex fulfilment with exclusively one operating system. After excluding DOS and the superstructure Windows 3.x, only OS/2, Linux and Windows NT were hot candidates for selection (Tannenbaum, 1992). In fact, the TCP/IP protocol should make it possible to control technological processes in local networks with many other operational systems like Windows 95, Windows 98, Windows NT, Windows 2000, OS/2.

The TCP/IP protocols co-operation has a layered structure. Corresponding particular layers communicate with the partner relevant layers making use of the services offered by the bottom layer of the model. This way of communication is so called “peer-to-peer” communication.

Applications with TCP/IP protocols are based on the *client-server* model. It means that applications exist which provide particular services and the other applications (clients) use them.

The TCP/IP protocol does not offer any mechanism for a process creation on message reception. That is why there should exist a mechanism ensuring a passive waiting for a message delivery. The consequence of that is a certain necessity to start servers offering some services in advance because they can not be started just according to the client requirements. The facts mentioned above implicate the following TCP/IP application structure:

- **Client** - the application initialising the communication, usually with a query
- **Server** - the application awaiting communication which starts a consequent action, usually an answer.

As soon as server accepts the client requirement, there is established a full duplex connection and the processes communicate among them. If the application has been designed for both modes what means that any partner can act as a client or server alternatively, it is called *symmetric* application. If the roles are allocated from the very beginning, it represents the *asymmetric* application, like at the *ftp* service.

The robust operational system Linux has been selected eventually. That decision was influenced by the more than two years of experience with Linux and by the fact that Linux is possible to download from Internet both as a binary file and as a source code. The realisation of both clients is not so dependent on operating systems like server. Generally speaking, there could be designed a platform with access to the BSD sockets network library. In spite of that, Linux has been chosen for the client realisation, too (Stevens, 1993, 1998).

The choice according to comparison between *iterative* and *concurrent* server speaks definitely for a concurrent server because of ability to process more demands simultaneously in spite of the fact that it would be much easier to design an iterative server. The concurrent servers

are widely used for *ftp* and *telnet* services. The established connection between a client and the server represents typically a multiple exchange of queries and answers. The basic algorithm is illustrated in Fig. 1.:

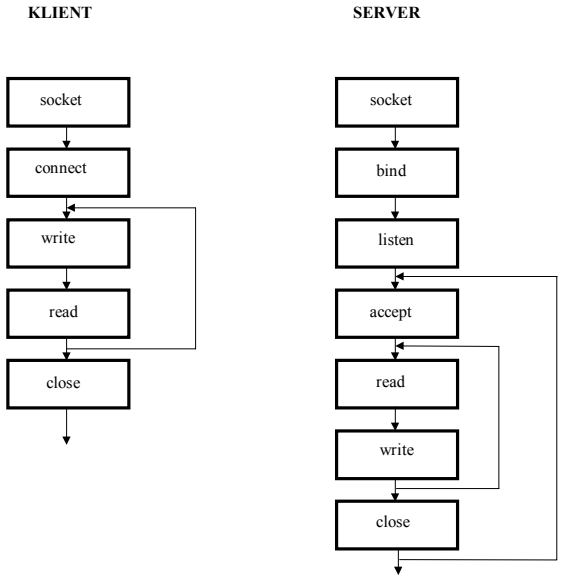

Fig.1. The basic algorithm for the communication between a client and the server.

The concurrency is realised with the function *fork ()* but it is a virtual concurrency performed by sharing space in time division. The server typically creates more processes what means that there is necessary to separate ancestors and descendants. Such algorithm is not the only possibility or a universal solution. The alternative for it could be a realisation of one process with more sockets served by the function *select ()*. A simplified concurrent TCP server is illustrated in Fig.2.:

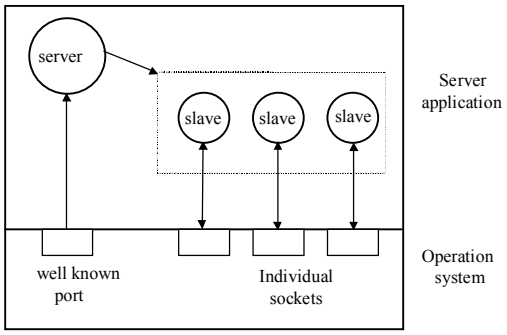

Fig.2. A concurrent TCP server

The expected serving many clients could be realised as a real multiprocessor design for parallel processing or as a time sharing variant.

There are standard means supporting individual processes in the Linux environment. The system function *fork ()* is a very convenient one. That function splits the running program in two almost identical processes. The other relevant functions are *execve ()* which writes over the current process with another code structure and *select ()* which serves a few concurrent I/O activities. It is possible to create for each *client-server* communication a separate process and the operation system takes care for their concurrent demands.

2. MONITORING AND CONTROLLING SYSTEM MODEL

The function of particular modules is as follows (see Fig. 5.):

Data sensing is performed by the first client program titled „recording client“ or maybe better “interface client”. In the real application, such client acts as a driver for the equipment capable to sense the technological process parameters. It sends data to the server and controls the process to the reference value distributed by the presenting client. There are no extra requirements on the environment where this application takes place. We need only the TPC/IP protocol for the connection with the server.

The data collection and distribution are performed with the server program. Its task is divided in three main points:

- Data reception from the recording client.
- Keeping the sensed data copies up-to-date.
- Sending the current data to the presenting client.

Data presentation is executed by the second client program unit having been titled “presenting client”. Its task was formerly to receive data copy from the server and to present them to the user. The modified version task has been accomplished with the reference value setting duty. Those three independent program units are illustrated with the Fig. 5. The dashed line represents the reference setting distribution line. Such structure can ensure the transferability to any network with the TCP/IP protocol. The protocol offers more services and comfort for application development.

3. THE STRUCTURE OF TRANSFERRED DATA

The data transferred between a client and the server is in a format agreed specified by the SLOT structure in the header file. That structure we call simply „slot“ and it carries two groups of data:

- vital data for the client and server program operation
- the data sensed and transported in the network

The first group of data covers the following items:

id - (integer) - an unambiguous slot identification. The PID server fills in the first item what ensures the unique character of it and which serves the recording client,

period - (integer) - the period of data sensing dimensioned in seconds.

According to this item and time elapsed since the last data reception, the presentation client deduces whether the communications session between the server and the recording client takes place.

The second group of data in the slot represents the quantity sensed in a technological process. This group consists of five items. Time of sensing (s-time), name of installation (name), sensed quantity (quantity), unit related to the quantity (unit) and measured value of the quantity (value).

The slot structure is flexible and open for modifications. But it shall contain the obligatory items id and period. The server program is able to distribute data in a new size just after recompilation.

The corresponding modification has to be done also in the presentation client program. Those modifications shall be implemented in the part ensuring the displaying of received data. There is to be stressed that the data size modification does not call for the modification of functions performing the client - server connection and data transport.

4. THE NETWORK ENCUMBRANCE

The implementation of network applications should always be accompanied with two vital questions:

- How will the application to be implemented encumber the general network operation?
- What performance can we expect from our application in case of encumbered network?

The first question can be positively answered providing the analysts and the programmers have created a system with efficient data transport not loading the network excessively. The answer in our case is positive because the application of the monitoring system makes use of small infrequent packets for the data exchange between clients and server. That is why no extra load is added to the network loaded with applications for transferring and sharing files (FTP or NFS). For the evaluation of the second question, we need to resolve how important the just in time reception of monitored process data is our case. During the technological process monitoring we need mostly data for further calculation or for verification of process control efficiency so that time is not so critical like at control itself.

5. REAL APPLICATION

As the research and design work continues, the recording client has been extended with a monitoring and controlling module for the enzymatic processor process temperature. The interface between enzymatic processor and network is realised with the 12-bit multipurpose converter card PCL-818L and a control box with actor elements and Pt100 converts designed at our department. The following picture illustrates the enzymatic process diagram (Fig. 3.):

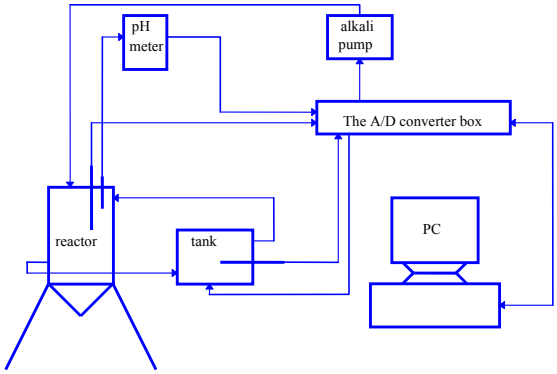

Fig.3. The block diagram of the enzymatic processor

The simplified schematic of the enzymatic reactor is in the Fig.4.

The recording client has been accomplished with a modified three-level controller with penalisation and a digital PID self-tuning controller based on the Ziegler-Nichols criterion with a fly identification of system parameters by least mean square optimisation method. The penalisation constant is fixed in the program and in case of its change, it is necessary to compile the interface client again.

The local network monitoring and control system has been debugged in the C-language in the OS Linux Slackware 1.2.13 and RedHat 7.1 environment and verified on the laboratory enzymatic processor installed in the Institute of Information Technologies (IIT), Thomas Bata University Zlin, Czech Republic. The interface client was established on *and3.iit.utb.cz* (PC at the enzymatic processor), server and presentation clients were established on *cas3.iit.utb.cz* (PC in the IIT server room). Computers were interconnected via Ethernet 10BASE2 - 10 Mb/s local network.

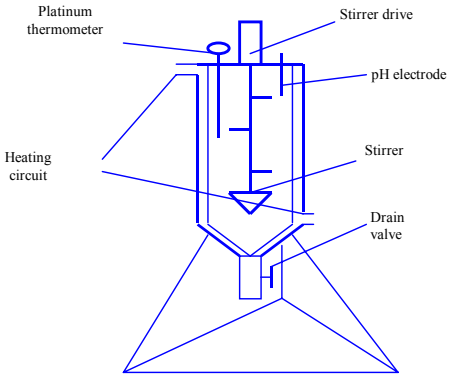

Fig.4. The enzymatic reactor

6. CONCLUSION

The designed systems are quite modest in the sense of communication sources. The communication based on the TCP protocol guarantees a sufficient integrity of the transferred data. The system architecture makes possible the dynamic connecting and disconnecting of both the monitoring places and the programs for monitored processes data presentation.

The modularity of the system has prepared the good precondition for implementing the client application in the environment of any operating system equipped with the support of communication based on the TCP/IP protocol.

ACKNOWLEDGMENTS

The work has been supported by the Ministry of Education of the Czech Republic under grants CEZ J22/98:260000014, MSM 281100001 and by the Grant Agency of the Czech Republic under grant No. 102/02/0204. This support is very gratefully acknowledged.

REFERENCES

[1] A.S. Tanenbaum, "Modern Operating Systems", *Science Book,* Prentice-Hall, 1992.
[2] A.S. Tanenbaum, A.S. Woodhull, "Operating systems: design and implementation", *Science Book,* Prentice-Hall,.1998.
[3] W.R Stevens, "Advanced Programming in the UNIX Environment", *Science Book,* Addison-Wesley, 1993.
[4] W.R Stevens, "TI - UNIX network programming", *Science Book,* Prentice-Hall, 1998.
[5] G. Silberschatz, "Operating systems concepts", *Science Book,* John Wiley & sons, Inc., 2002.

Authors: Ing. Miroslav Matýsek, Ph.D., Mgr. Milan Adámek, Ph.D., Ing. Petr Neumann, Ph.D., Thomas Bata University, Faculty of Technology Zlín, Department of Automatic Control, Mostni 5139, 760 01 Zlín, Czech Republic.
+420 57 6033203 +420 57 6033333
matysek@ft.utb.cz
adamek@ft.utb.cz
neumann@ft.utb.cz

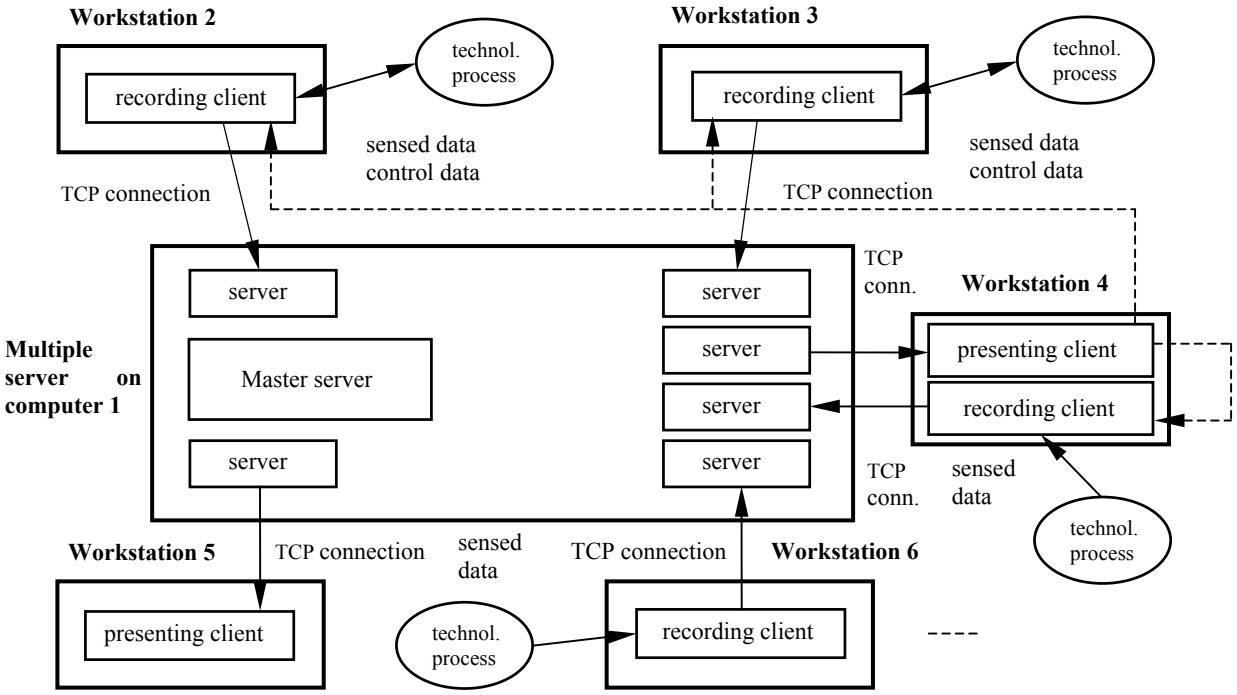

Fig. 5. The designed model of the monitoring and controlling system