A Monitoring Network System and its Practical Demonstration on a Model Technological Process

MILAN ADAMEK, MIROSLAV MATYSEK, VACLAV MACH,
PETR NEUMANN, MARTIN POSPISILIK
Faculty of Applied Informatics
Tomas Bata University in Zlin
Nad Stranemi 4511, 76005, Zlin
CZECH REPUBLIC
adamek@utb.cz, matysek@fai.utb.cz
http://fai.utb.cz/en

Abstract: - In order to deepen students´ practical knowledge and understanding of the "Computer Networks" subject, a Monitoring Network System was developed that is able to read, collect and distribute data describing real technological processes. The Monitoring System design is based on Client/Server architecture. The model is represented by the real chemical reactor, which was created for the study of the enzymatic hydrolysis of potentially hazardous chrome-tanned wastes generated during leather manufacturing. Final system should be implemented at the university to the graduates and the system should be modular with possible communication support. The setup of the monitoring system comes with all needed explanation and procedures.

Key-Words: - Monitoring network system, Client/Server, TCP/IP protocol, Chemical reactor, Hydrolysis.

1 Introduction

Students educated in the field of Informatics at tertiary education institutions absolve, in various forms, the "Computer Network(s)" course during their Bachelor's or Master's degree studies. The basic content of these courses is to introduce students to the problems and issues of computer network administration from a user's point-of-view. Usually, the courses more-or-less copy the following structure: The Topology and Architecture of Computer Networks, Communication Protocols and – of course, the problems of Computer Network Security. Nevertheless, computer networks represent a broad and dynamically-developing area, which is difficult to encompass in the limited time dedicated to this subject such that students acquire practical skills through practical tasks and applications.

IT education at the Faculty of Applied Informatics is founded on a worthy tradition and historical achievements in the field of Automation and Control of Industrial Processes. This specific orientation is also reflected in the long-term demand of employers for skilled graduates in this field. For these reasons, and with the express purpose of extending students` practical knowledge of computer networks, a Monitoring Network System was developed within the context of a student's year seminar project - which is able to read, collect and distribute data describing a technological process.

A client – server architecture was chosen for the system design, which meets the requirements for the separation of those parts which execute data reading, collecting, distribution and presentation operations [1]. The above-mentioned system architecture works on the assumption that the reading of technological data is executed by the first client, data collection and distribution is realized by the server, and data presentation is performed by the second client. This system structure enables us to achieve the necessary freedom when selecting the environment in which any part of the system can be realized.

The selection of the operating system was a very important step in the development of the monitoring system. The realization of any of the clients is not as strongly dependent on the type of operating system as is the realization of the server. Generally, it can be stated that a client application can be realized on any platform for which the BSD Sockets Network Library is available. The selection process of the operating system for the server application was determined by the following parameters:

- Support of Inter-Process Communication (IPC),
- Simultaneous Multiple Program Processing,
- Network Communication Resources based on TCP/IP protocols,
- Reasonable hardware demands.

As a result, preference was given to the Linux Operating System (OS) for the realization of the server as well as the individual clients. This operating system showed the best match with the criteria above [2, 3].

2 THE ARCHITECTURE OF THE MONITORING SYSTEM

In network applications, the client - server model is a standard [4]. In this case, the term server denotes a process that expects a contact from the client - in the sense that the server can perform an operation for the client. The client in the said model also represents a certain process. The behavior of the server and client in this monitoring system can be described as follows:

- The server process is started on a PC. It initializes itself, then goes into sleep mode and waits for a contact from the client.
- The client process is started on the same or another PC. In the event that the client and server are running on the same PC (under one OS), they communicate between themselves by means of the OS.
- If the client and server run on different PCs, communication between them is through a computer network. The client process sends its request to the server and waits for the result of the processing to be sent back by the server.
- After completing the request of the client, the server goes into a sleep mode and waits for further requests from the client.

Client

The role of the client in the proposed monitoring system is to read the technological data and depict their visualization. A Standard Client works with a known port, which in this case is also called the so-called "reserved port". The reserved ports are installed in order to establish connection with the server. To establish a connection, the client must know the number of the port on which the service is offered by the server (e.g. ftp). Other programs of a client type operate as non-standard clients.

Therefore, after establishing a connection between the client and server, it is not necessary to use the reserved port (and even, this is not desirable due to the freeing-up of the port for further communications) and the communications are performed on randomly-selected and mutually conflict-free ports from the range of dynamic and private ports available. The reserved port is, after establishing the connection, released immediately for eventual communication with another client.

Reading of technological data is ensured by the *Recording Client*, which is realized as a controller of the device reading the relevant technological process quantity - which is subsequently sent to the server. It is even possible to run multiple recording clients under one operating system (PC). In such a case, each client ensures the reading and transfer of different technological quantities. No specific requirements are made on the environment in which this part is realized, apart from the ability to use TCP/IP protocols for the connection with the program server.

The *Presentation Client* then receives the current copies of the technological process quantities data from the server and ensures their presentation. It is possible to have more presentation clients running within a network under one operating system (PC).

Server

The collection and distribution of data is performed by the program server. Its activities can be divided into three main points:

- Receiving data from the recording clients
- Maintaining the current copies of the data read
- Sending the current data to the presentation clients

Two kinds of servers are used in the proposed system, depending on the time needed to perform the required processes. Their activities can be characterized as follows:

- In the event the server is able to handle the clients` requests in a short time interval, the server processes these requests itself. This type of process in the model is called an Iterative Server. A typical example of an iterative server is a service providing date and time data.
- If the client's request processing time is unlimited, the server processes the request in a competitive way. This type of server is called a Competitive Server. The competitive server - parental process (i.e. Master Server) creates its copy – its descendant. This descendant then processes the client`s request further; thus the original (parental) server process may go back into sleep mode and await the next client request. Of course, this type of server requires an operating system that allows the simultaneous running of multiple processes. Competitive servers are widely used for example in ftp and Telnet services. The

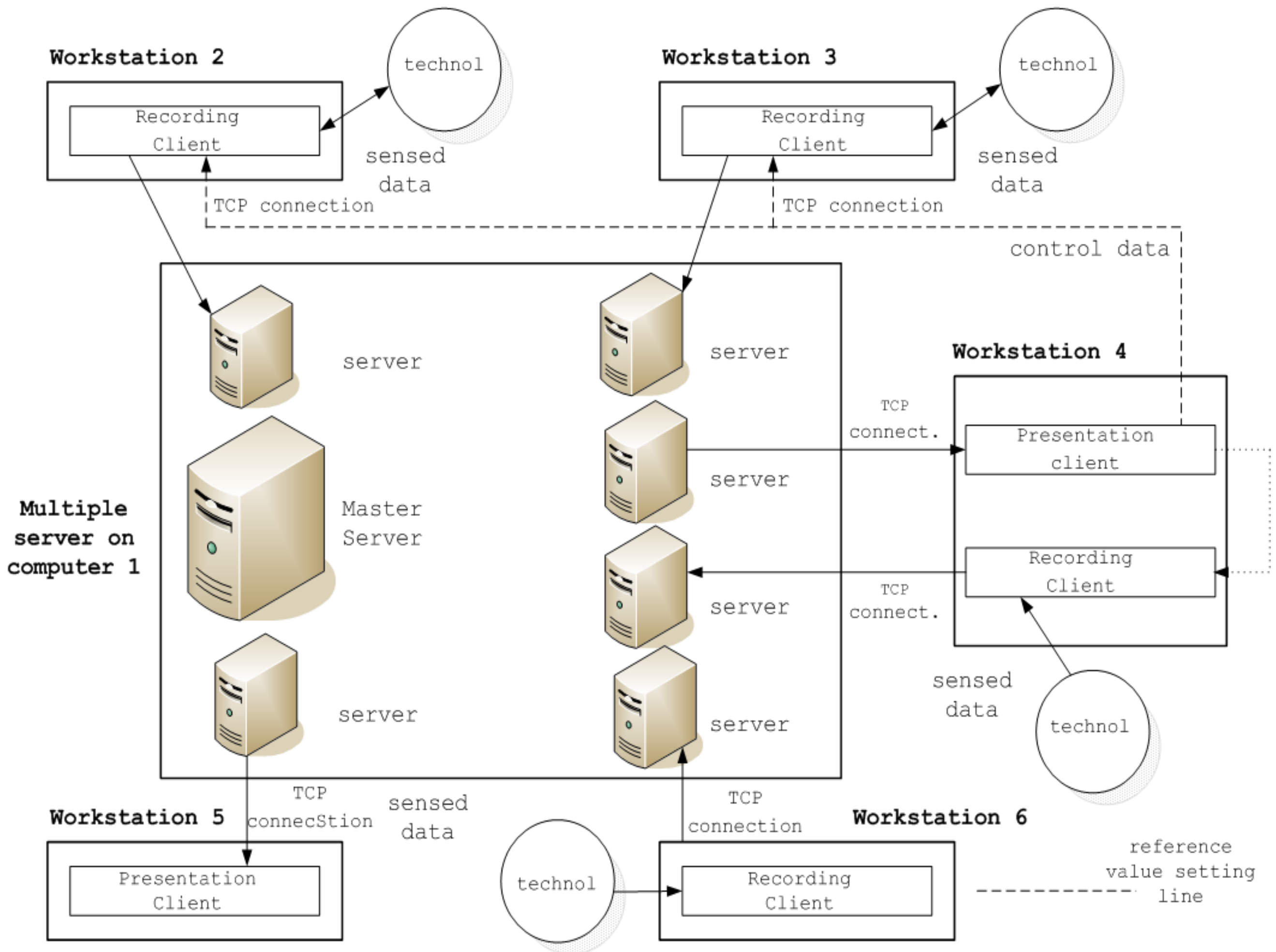

Fig 1 The designed model of the monitoring and controlling system.

established connection between client and server usually contains multiple requests – in other words, multiple exchanges of queries and answers take place between the client and the server.

3 SOFTWARE

The Client Program

Communication between a client and server by means of TCP protocols is carried out according to the following algorithms:

- Finding the server's IP address from the specified name or address inputted in dot notation Creation of a clipboard for communication - socket() function
- Connection to the server - connect() function
- Data exchanges with the server - read() or write() functions
- Terminating communications with the server

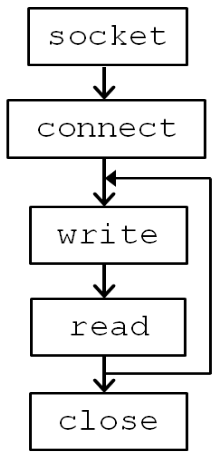

Fig 2 The basic algorithm for communications between a client and server

The Server Program

Processing client requirements is performed by the server in a competitive way. In most cases, this is an apparent competition - achieved by sharing the process in time.

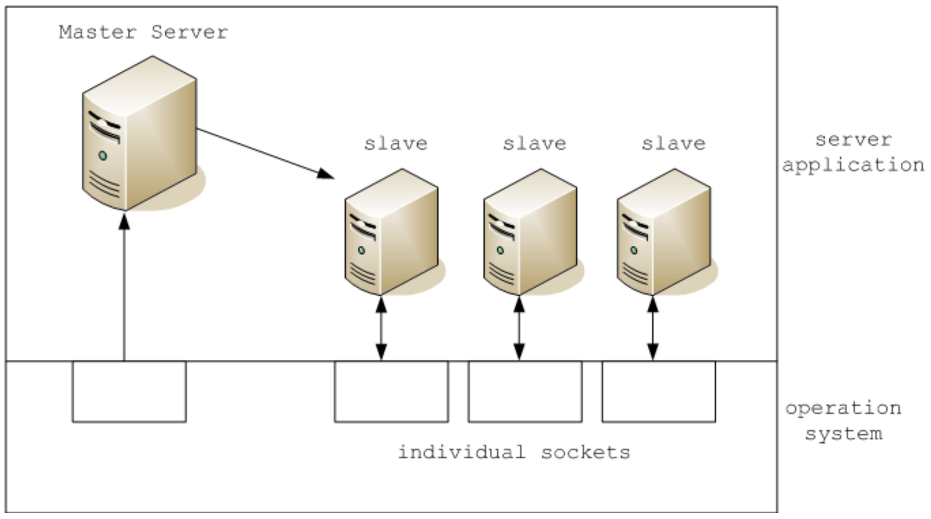

Fig 3 A competitive TCP server.

The basic algorithm of the Server program is as follows:

- Immediately after starting, the Master server program creates clipboards for communications associated with the reserved ports - socket() and bind() functions. The ports for connections with clients are defined in the server configuration file.
- After their creation, the clipboards are set into a passive state, in which they wait for connections from clients - listen() and select() functions. Using the select() function, the kernel can put the master server processes into a standby mode. It waits in this mode until an event occurs, then it activates itself. In the proposed model, the kernel only notifies the master server process in the case when connection initialization data (i.e. data for reading) appear on any of the clipboards.
- In the course of an occurrence of the "connection request" event, the master server receives the request on the corresponding port - connect() function. In the case of a recording client, the master server controls the number of the connected recording clients. If the new client exceeds the maximum number of the recording clients, the connection is closed and the master server returns again to the sleep mode. If there is enough room for the connection of a new recording client, a call to the kernel fork() function is executed, resulting in the creation of a descendant to serve the operator's requirements. If a presentation client is connected - the controlling of the maximum number of connected recording clients is left out; otherwise, the connection request process remains the same. After that, the master server closes the unnecessary mailbox that was created by the prior calling of the connect() function.
- Message exchange between a client and the server through the established connection
- The server (descendant) serving the recording client waits for the data sent by the client. After receiving the data, the server writes them into the shared memory. The synchronization of the access to the shared memory with other processes is ensured by semaphores. After writing the data in the shared memory, the server sends a signal that indicates to all other servers (descendants) that new data has been written in the shared memory.
- Immediately after establishing the connection, the recording client server sends the contents of all slots to the client. Then it goes back to sleep until it is awakened by a signal that initializes new data entries in the shared memory. Subsequently, the server gains access to the control structure (Header) of the shared memory, checks which slots contain new data, and sends this data immediately to the client.

The activities described above are carried out by both types of descendants cyclically until the connection is closed by the client or Server program operations are terminated (by the master server termination).

Shared memory structure

In the model, the shared memory was used to preserve the data obtained from the recording clients. The main reason for using this resource for inter-process communication was the possibility of simultaneously accessing several processes (descendants of the master server) to the data stored in the shared memory. Access synchronization is necessary to enable access of multiple processes to the shared memory, for example by semaphores. The shared memory is divided into two parts, as shown in Figure 5.

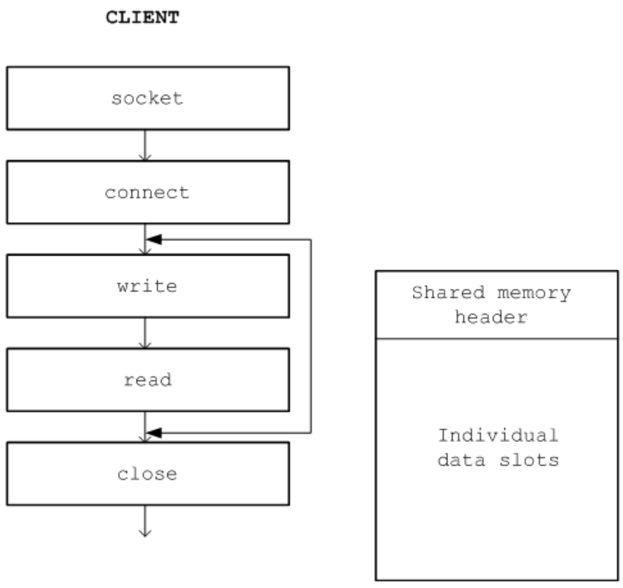

Fig 5 Shared memory structure.

The area of each data slot represents a data area into which data is written by individual recording clients, while just one slot is reserved for each recording client. The server servicing this client records only in this one slot and in the header. The server servicing the presentation client performs reading only from the shared memory whether from the slot or the header areas.

The data describing individual slots are stored in the header. These have particularly to do with entries indicating the time of the last modification to the slot. The server servicing the presentation client uses this data to decide whether the data from this slot has already been sent to the client, or if it is newly-received data, to be sent.

Process synchronization methods

In the Server program context, it is necessary to ensure synchronization of the processes which access the data in the shared memory. In order to maintain data consistency, the following rules must be observed:

- A slot cannot be modified by more than one process at any one moment
- If the slot modification is in progress, no process is allowed to read this slot
- A slot the data from which is just being read cannot be accessed by a process that wants to modify the data.

On the other hand, the situation may occur where two or more processes read data from the same slot. Semaphores and signals are used for synchronizing all processes.

Semaphores

The semaphore principle in used in the model for several purposes. The master server uses semaphores for the control of the number of connected recording clients. For this purpose, a semaphore is created whose value indicates the number of free places available for the connection of other recording clients. In the course of the start of the master server, this semaphore is initialized to a value which indicates the number of slots reserved for data from the recording clients. At the moment of connecting a recording client, the master server process reduces the semaphore value by one numeral, and the process continues in its activities.

Termination of the connection with the recording client results in the freeing-up of previously assigned semaphores. Other semaphores control the actual access of individual processes (descendants) to the shared memory area. Each process first attempts to acquire the semaphore to the header of the shared memory, and only after that, the semaphore to the corresponding data slot. This approach was chosen deliberately in the initial design of the monitoring system in order to avoid deadlocks. Deadlock is understood as a situation in which two processes mutually at the same time hold the tools they each want to use. Each data slot semaphore can exist in one of these three states:

- Containing the value 0
- Containing the value 1
- Containing the value 2, or multiples thereof.

The first state indicates a free slot that is not being used by any of the processes. The second signals an exclusively-locked slot, i.e. access to this slot is owned by a single process, which is recording the newly-received data. The last state denotes the eventuality of a slot being locked, i.e. a slot from which the data is currently being read by one or more processes. An example of synchronization using semaphores is shown in the following Figure 6.

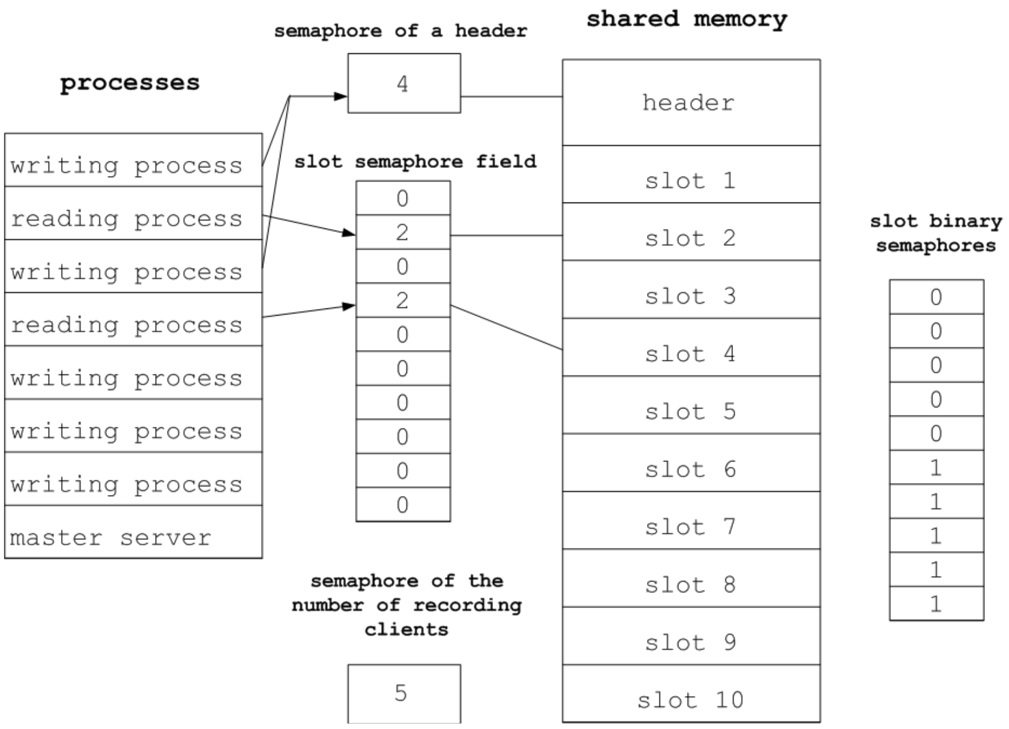

Fig 6 Synchronization of process access to the shared memory.

Figure 6 depicts the situation where two presentation clients gain access to the shared memory. Both have the header of the shared memory locked, and each of them is reading a different slot. From the field of binary semaphore slots and the semaphore of the number of recording clients, it follows that a total of 5 recording clients, assigned to slots 1 – 5, are simultaneously connected. Nevertheless, at the moment captured in the picture, they are sleeping; i.e. waiting for data from a recording client or to receive a semaphore. The slot semaphore field, which synchronizes access to individual slots, contains a value of 2 in two semaphores – the slot is locked and shared for reading only. The other semaphores from this field contain the value of 0; i.e. the slot is not being accessed by any process.

Signals

The processes that ensure operation of the recording clients use the SIGUSR signal (First User-Defined Signal) to notify the new data recordings to the slots of the processes servicing presentation clients. The master server processes and the servers servicing the recording clients ignore the reception of this signal. The processes servicing the presentation clients use this signal for their wakeup. This method is more efficient than periodic wake-ups using the sleep() function.

Program configuration

The server is configured in a way similar to the clients, i.e. using a configuration file. The name of the file - including its full path, is specified in the variable of the SMONCONFIG shell. In the event that this variable is not set, the program tries to open the default /etc/monconfig.server configuration file. If even this file does not exist, then the program server reports an error.

The configuration file contains three parameters. The first of them - USERS, determines the number of slots that are allocated at startup. Thereby, it also determines the maximum value of recording clients which can be connected simultaneously to the server. The other two parameters specify the ports, on which the server is waiting for connections from individual types of the clients.

Resolving critical states

The basic critical conditions include: termination of the connection between client and server, termination of work, or system failure. The Server program responds to any of the above-mentioned conditions by its proper termination, including freeing-up all tools being used. The method primarily used for the determination of a failure or for disconnection with the Client program is to send data out-of-range, which the client evaluates and subsequently correctly closes the connection. In cases when the server process does not manage to inform a client about a failure by sending the data out-of-range, the clipboard flag SO KEEPALIVE is used, which notifies the client process of the situation. Another state where it is advantageous to resolve in the master server process is to terminate any descendant. In this case, it is necessary to execute the calling the kernel wait() function in order to avoid the origin of zombie processes in the system.

4 NETWORK (OVER)LOAD

When implementing network applications, it is essential to evaluate the network load from two perspectives. It is necessary to consider the extent to which the deployed applications will load the overall operation of the network, and – vice versa, to evaluate the effect of network load on application performance. This has to do with the basic viewpoints on the performance issues of both the application and network.

The design of the monitoring system took into consideration the minimum network load and the method chosen for data-transfer between the server and clients was one that uses small packets, which are sent in longer time intervals. This prevents overload stress on the network, since it could be busy with other user applications.

To assess the effect of long-term network (over)load, it is necessary to consider what time interval the data in the given monitoring system must be presented in. In the monitoring of a technological process, this mainly concerns acquiring data for further calculations or for the verification of the course of the process control. The proposed monitoring system will not be significantly affected by sudden network overload crashes, which may be caused by a short-term failure of the file – server, for instance.

5 IMPLEMENTATION OF THE MONITORING SYSTEM

The monitoring system was developed and implemented on a real technological process model, for educational purposes and for a more-detailed study of the monitoring system´s properties and behavior. Students are provided with the opportunity to verify the functioning of the monitoring system on a real technological device, and thereby, to familiarize themselves with the data-transfer process, server functions, and the recording and presentation clients. The study of real processes is still an important part of engineering education in many fields, and in many cases, it cannot be fully substituted for only mere simulations of individual physical/chemical phenomena [7]. The need for, and importance of, practical experience with real systems, e.g. in the form of remote laboratories, is also repeatedly reflected in technical literature oriented on the field of the role of computer technology in university education [e.g. 8].

Technological process monitoring

The realistic model was represented by a chemical reactor, which was constructed for model studies on the enzymatic hydrolysis of potentially hazardous chrome-tanned wastes generated during leather manufacturing. The disposal of chrome-tanned waste in open landfills - where it is subjected to climatic and other external conditions is not suitable, since there is a risk of oxidation of trivalent chromium present in this waste into carcinogenic hexavalent chromium salts [9]. The processing of chrome-tanned wastes - the most typical representative of which is chromium shavings and their conversion into valuable products is therefore advantageous not only from the environmental, but above all, the health protection point of view.

One of the most advanced industrially implemented procedures so far is two-step hydrolysis under alkali conditions in the presence of a proteolytic enzyme [10]. The process is performed in two stages, namely - alkali treatment, and consecutive enzymatic hydrolysis. The results of the reaction after the first step is a gelatable protein and an intermediate filter cake.

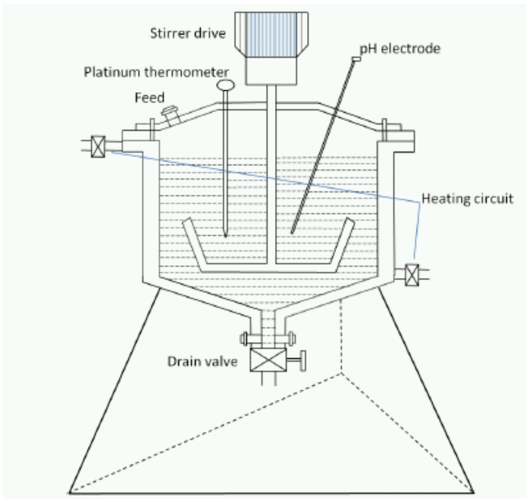

Fig 7 Simplified structure of the chemical reactor.

The cake is then subjected to enzymatic hydrolysis, that generates an insoluble filter cake with concentrated chromium and soluble protein hydrolysate - which, in the terminal phase, is dried in a vacuum evaporator. Depending on the way in which the reaction is controlled, the reaction generates protein solution of various qualities, given

specifically by gel strength (Bloom value). The model reactor allows the studying of the hydrolysis conditions at which it is possible to obtain desired properties of final products.
The monitoring system is implemented in the first stage of the technological process, i.e. alkaline treatment (denaturation) of chrome shavings in the chemical reactor. Simplified models of the used chemical reactor used, and its control, are depicted in Figures 7 and 8, respectively.

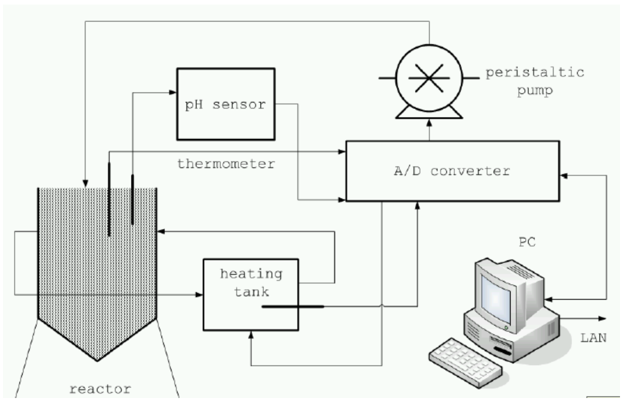

Fig 8 Block diagram of the chemical reactor control.

The reactor, a cylindrical tank of a fermentation reactor type, represents a non-flow isothermal heterogeneous reactor with a hollow shell through which the heating liquid flows which is heated in a separated tank outside the reactor. The alkaline treatment takes place at a pH of 12 and a temperature of 70°C. If the pH decreases below the required level (due to the buffering effect of protein), it is adjusted by the addition of an alkali. Hydrolysis is most efficient at the said pH and temperature; an increase of either value above or below this optimal level reduces the reaction rate, or affects the resulting products.

6 CONCLUSIONS

TCP protocol-based communications guarantee the sufficient integrity of the transferred data. The system architecture makes possible the dynamic connection and disconnection of both the monitoring places and the programs for monitored processes' data presentation. The modularity of the system has prepared good preconditions for the implementation of the client application in the environment of any operating system equipped with TCP/IP protocol-based communication support. 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 environments and verified on the laboratory hydrolysis/fermentation reactor installed within the premises of the Faculty of Applied Informatics.
The system has been successfully implemented at the Faculty of Applied Informatics of Tomas Bata University in Zlín, Czech Republic within the context of the "Computer Networks" subject. Extending practical knowledge contributes to higher employment opportunities of the graduates on the labor market - especially in technical branches. Students can also use the system for their Bachelor's and Master's theses; in addition to the educational benefits, these theses contribute to the gradual upgrading and development of the monitoring network system - including its extension for example to mobile applications.

7 AKNOWLEDGEMENT

The work was performed with the financial support of Research Project NPU I No. MSMT-7778/2014 by the Ministry of Education of the Czech Republic, by the European Regional Development Fund under the Project CEBIA-Tech No. CZ.1.05/2.1.00/03.0089, and by the European Social Fund under the Project No. CZ.1.07/2.3.00/30.0035.

References:

[1] Tanenbaum, A: *Modern operating systems*, 2nd edition, Prentice Hall, Englewood Cliffs (2001).
[2] Stevens, W. R.: *Advanced Programming in the UNIX Environment*, 2nd Ed. Addison-Wesley, Reading (2005)
[3] Stevens, W. R.: TI - *UNIX network programming,* Prentice-Hall, Englewood Cliffs (1998)
[4] Tanenbaum, A. S., Woodhull, A. S.: *Operating Systems: Design and Implementation,* Prentice-Hall, Englewood Cliffs (2006)
[5] Silberschatz, A., Galvin, P. B., Gagne, G.: *Operating systems concepts,* 7th Ed. John Wiley & sons, Inc. (2002)
[6] Bigelow, S. J.: *Troubleshooting, Maintaining & Repairing Networks,* 1st Ed. Osborne/McGraw-Hill (2002)
[7] Nedic, Z., Machotka, J., Nafalski, A.: *Remote laboratories versus virtual and real laboratories,* Frontiers in Education, FIE 2003. 33rd Annual 1, T3E-1-T3E-6 (2003)
[8] Domínguez, M., Fuertes, J. J., Prada, M. A., Alonso, S., Morán, A.: *Remote laboratory of a quadruple tank process for learning in control engineering using different industrial controllers*, Comput. Appl. Eng. Educ., Vol. 22, No. 3, 375-386, (2011)

[9] Kolomazník, K., Adámek, M., Anděl, I., Uhlířová, M.: Leather waste - Potential threat to human health, and a new technology of its treatment. J. Haz. Mat., Vol. 160, No. 2-3, 514-520, (2008)

[10] Kolomaznik, K., Mladek, M., Langmaier, F., Janacova, D., Taylor, M. M.: Experience in industrial practice of enzymatic dechromation of chrome shavings, J. Am. Leather Chem. As., Vol. 94, No. 2, 55-63, 1999