# Computer networks in education

*Milan* Adamek [1,a] *Miroslav* Matysek[1] and *Michaela* Barinova[1]

[1] *Tomas Bata University in Zlin, Faculty of Applied Informatics, Nam. T. G. Masaryka 5555, 760 05 Zlin, Czech Republic*

**Abstract.** In order to deepen practical knowledge of students within the subject "Computer Networks", a monitoring network system has been developed which is able to read, collect and distribute data describing real technological process. The monitoring system design is based on a client/server architecture. The system was verified on a model chemical reactor used for hydrolysis of chrome-tanned wastes from tanneries.

## 1 Introduction

University students educated in the field of Informatics attend either during their bachelor or master study in various forms the subject "Computer Networks". The basic content of the courses is to acquaint the students with the problems and administration of computer networks from a user`s viewpoint. Usually, the courses more or less copy the following structure: explanation of the types of computer networks, topology and architecture of computer networks, communication protocols, and inseparable problems of computer networks security. However, computer networks represent broad and dynamically developing area which is difficult to compass in the limited time dedicated to this subject so that students could acquire practical skills through practical tasks and applications.

A client – server architecture was chosen for the system design, which meets the requirements for the separation of the parts executing data reading, collecting, distribution and presentation [1]. The said system architecture works on the assumption that 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 in the selection of the environment, in which any part of the system can be realized.

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

1. Support of inter-process communication (IPC),
2. simultaneous processing of multiple programs,
3. means of network communication based on the TCP/IP protocol,
4. reasonable hardware demands.

As a result, for the realization of both the server and individual clients we gave preference to Linux operating system. This operating system showed best match with the above listed criteria [2, 3].

## 2 ARCHITECTURE OF THE MONITORING SYSTEM

In network applications, the model client - server is a standard [4]. The term server in this case 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. In the developed monitoring system, the behavior of the server and client can be described as follows:

- Process server is running on a PC. It initializes itself, then goes into a sleep mode and waits for a contact from the client.
- Process client is running on the same or another PC. In the event that the client and server are running on the same PC (under one OS), the communication between them is performed by means of the OS. In the case that the client and server run on different PCs, the communication between them is mediated through a computer network. The process client sends its request toward the server and waits for the result of the processing 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.

### 2.1 Client

The role of the client in the proposed monitoring system is reading technological data and their visualization. A standard client works with a known port,

[a] Corresponding author: adamek@fai.utb.cz

which in this case is referred to as the reserved port. The reserved ports are installed in order to establish a 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 (for example 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 (it is even not desirable due to liberation of the port for further communication) and the communication is performed on randomly selected and mutually conflict-free ports from the range of dynamic and private ports. The reserved port is after establishing the connection released immediately for possible communication with another client.

Reading of technological data is ensured by the recording client that is realized as a controller of the device reading the relevant technological process quantity, which it subsequently sends to the server. It is possible to run multiple recording clients under one operating system (PC). In this case, each client ensures reading and transfer of different technological quantity. There are no specific requirements of the environment, in which this part is realized, except for the ability to use the TCP/IP protocol for the connection with the server program.

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

### 2.2 Server

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

1. Receiving data from the recording clients
2. Maintaining the current copies of the read data
3. Sending the current data to the presenting clients.

The proposed system uses two kinds of servers depending on the time needed to perform the required processes. Their function can be characterized as follows:

1. In the event the server is able to handle the clients` requests in a short period of time, the server processes these requests itself. This type of processes in the model is called iterative server. A typical example of an iterative server is a service providing date and time.

2. If the time for the processing of a client`s request is unlimited, the server processes the request in a competitive way. This type of server is called competitive. The competitive server - parental process (master server) creates its copy – the descendant. The descendant then further processes the client`s request, so the original (parental) process of the server may again go into a sleep mode and wait for the next client`s request. This type of server of course requires an operating system, which allows simultaneous running of multiple processes. The competitive servers are widely used for example in ftp and Telnet services. The established connection between the client and the server usually contains multiple requests – in other words, multiple exchanges of queries and answers take place between the client and the server.

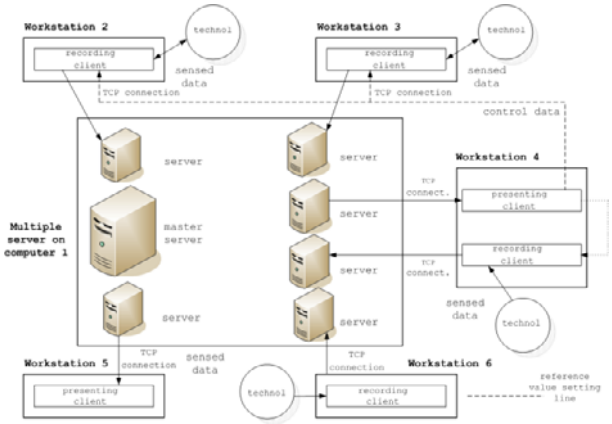


**Fig. 1.** The designed model of the monitoring and controlling system.

## 3 SOFTWARE

### 3.1. Program Client

Communication between the client and server through the TCP protocol is carried out according to the following algorithm:

1. Finding the IP address of the server from the specified name or address input in dot notation.
2. Creation of a mailbox for communication - function socket()
3. Connection to the server - function connect()
4. Data exchange with the server - functions read() or write()
5. Termination of communication with the server.

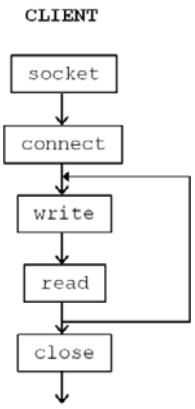


**Fig. 2.** The basic algorithm for the communication between a client and the server

### 3.2. Program Server

Processing of the clients` requirements is performed by the server in a competitive way. In most cases it is an apparent competition, which is achieved by sharing the process in time.

The basic algorithm of the Server program is as follows:

1) Immediately after starting, the Master server program creates mailboxes for communication and these are associated with the reserved ports - functions socket()

and bind(). The ports for the connection with the clients are defined in the server configuration file.

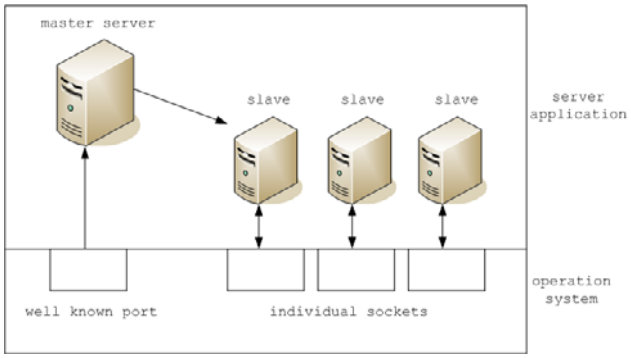


**Fig. 3** Competitive TCP server.

2) After they have been created, the mailboxes are set into a passive state in which they are waiting for connections from the clients - functions listen() and select(). Using the select() function, the kernel can put the master server process into a standby mode. In this mode it waits until an event occurs, then it is activated. In the proposed model, the kernel notifies the master server process only in the case when data for connection initialization (i.e. data for reading) appear on any of the mailboxes.

3) At the occurrence of the event "connection request", the master server receives the request on the related port - function connect(). If the case of a recording client, the master server makes a checkout of the number of the connected recording clients. If a new client exceeds the maximum number of the recording clients, the connection is closed and the master server again puts itself into a sleep mode. It the event there is enough room for the connection of a new recording client, a call to the kernel function fork() is executed, resulting in the creation of a descendant to see to the operator's requirements. If a presentation client is connected, the way of the processing of the connection request is the same except for the checkout of the maximum number of connected recording clients. After that, the master server closes the no more necessary mailbox that was created by the prior calling of the connect() function.

4) 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 in the shared memory. The synchronization of the access to the shared memory with other processes is ensured by semaphores. After entering 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.

• Server serving the recording client sends immediately after establishing the connection the contents of all slots to the client. Then it goes to sleep until it is awakened by a signal that initializes entering new data in the shared memory. Consequently, 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 described activities are carried out by both types of descendants cyclically until the connection is closed by the client or the Server program operation is terminated (by termination of master server).

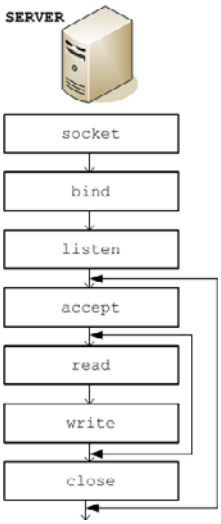


**Fig. 4** Basic structure of the server program

### 3.3. Structure of the shared memory

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

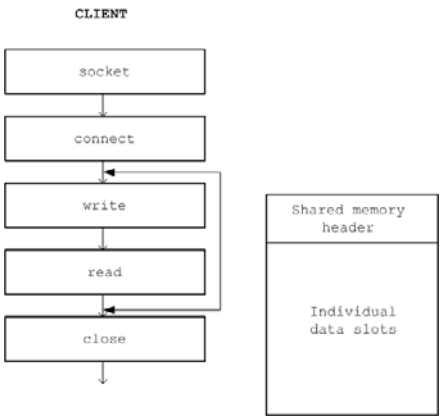


**Fig. 5** Shared memory structure

The area of each data slot represents a data area into which data is written by individual recording clients, with just one slot reserved for each recording client. The server serving this client records only in the one related slot and in the header. The server serving the presenting client performs only reading from the shared memory from both the slot and the header areas.

The data describing individual slots are stored in the header. These particularly include entries indicating the time of the slot last modification. This data is used by the server operating the presentation client to decide whether the data from this slot has already been sent to the client or whether it is newly received data to be sent.

### 3.4. Process synchronization methods

Within the Server program it is necessary to ensure synchronization of the processes by which the data in the shared memory are accessed. The following rules must be observed in order to maintain data consistency:

1. In one moment a slot cannot be modified by more than one process.
2. If the slot is being modified, no process is allowed to read this slot.
3. The slot from which the data are just being read cannot be accessed by a process that wants to modify the data. On the other hand, a situation may occur in which two or more processes read data from the same slot.

Synchronization of all processes is based on methods using semaphores and signals.

The basic critical conditions include: termination of the connection between the client and the server, termination of work or system failure. To any of the named conditions the program Server responds by a proper termination, including the release of all tools used. For the determination of a failure or disconnection of the client program uses the primary method of sending data out of range, which evaluates the client and closes the connection. For cases where the process server does not manage to inform the client about the failure data outside the range is used symptom clipboard SO KEEPALIVE that this situation highlights the client process. Another condition which should be treated in the process of the master server, the termination of a child. In this case it is necessary to call the kernel function wait() system to avoid zombie processes.

## 4 NETWORK LOAD

When implementing network applications, the network load must be evaluated from two perspectives. It is necessary to consider to which extent the deployed applications will load the overall operation of the network and vice versa, to evaluate the effect of the network load on the application performance. These are the basic viewpoints on the performance issues of both the application and network.

With respect to the minimum network load, a method for the data transfer between the server and clients was selected that uses small packets which are sent in longer time intervals. This prevents the network from a stress load, since the network could be busy with other user applications.

To assess the effect of a long-term network load it is necessary to consider the time interval in which the data in the monitoring system must be presented. In the monitoring of a technological process it mainly concerns obtaining data for further calculations or verification of the course of the process control. The proposed monitoring system will not be significantly affected by crash network loads, which may be caused for instance by a short-term failure of the file - server.

## 5 IMPLEMENTATION OF THE MONITORING SYSTEM

For educational purposes and more detailed study on the monitoring system properties and behavior, the developed monitoring system was implemented on a real model of a technological process. This gives students the opportunity to verify the functioning of the monitoring system on a real technological device and get closer insight into the data transfer process, the functioning of the server and both the recording and presenting client. The study on real processes is still an important part of engineering education in various fields and in many cases it cannot be fully substituted by 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 repeatedly reflected in technical literature in the field of the role of computer technology in university education [e.g. 8].

### 5.1. Monitoring of a technological process

The realistic model was represented by a chemical reactor, which was constructed for model studies on enzymatic hydrolysis of potentially hazardous chrome-tanned wastes generated during leather manufacturing. Disposal of chrome-tanned waste at 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]. Processing of chrome-tanned waste, the most typical representative of which is chromium shavings, and its conversion into valuable products is therefore advantageous not only from environmental, but mainly from health protection point of view.

One of the most advanced and industrially implemented procedures so far is the technique of two-step hydrolysis under alkali conditions in the presence of a proteolytic enzyme [10]. The process is done in two stages, namely alkali treatment and consecutive enzymatic hydrolysis. The results of the reaction after the first step is gelatable protein and intermediate filter cake. The cake is then subjected to enzymatic hydrolysis, giving insoluble filter cake with concentrated chromium and soluble protein hydrolyzate, which is in the terminal phase dried in vacuum evaporator. Dependent on the way in which the reaction is controlled, the reaction gives protein solution of various qualities, given specifically by gel strength (Bloom value). The model reactor allows studying 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

and its control are depicted in the following Figs. 6 and 7, respectively.

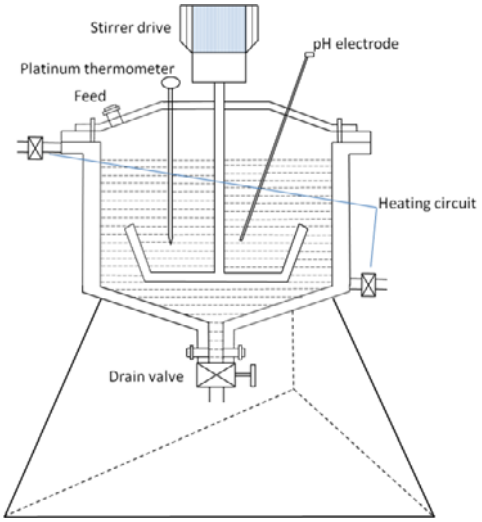


**Fig. 6** Simplified structure of the chemical reactor.

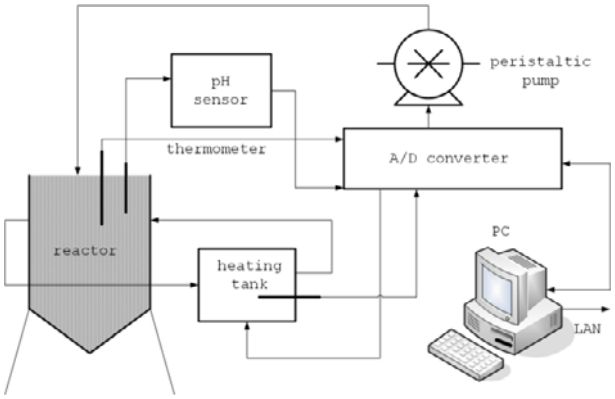


**Fig. 7** Block diagram of the chemical reactor control.

The reactor, a cylindrical tank of a type of fermentation reactor, represents a non-flow isothermal heterogeneous reactor with hollow shell, through which flows the heating liquid 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 addition of an alkali. At the said pH and temperature, the hydrolysis is most efficient; an increase of either value above or below this optimal level reduces the reaction rate or affects the resulting products.

The peristaltic pump (type PS 10) is used for dosing of the reaction components. Sensor PHR01 for the measurement of pH consists of a measuring glass electrode and a reference silver-chloride electrode, which are conjugated in a common stem. The potential of the measuring electrode after it is immersed in the solution is dependent mainly on hydrogen ion concentration. The platinum thermometers Pt 100 are used for the measuring of the reaction mixture temperature and the temperature of heating liquid in the heating tank. The measured data are converted from the analogue to digital form via the ADVANTECH PCI 1716 technological card and subsequently used as input data for the control circuit.

## 6 CONCLUSIONS

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 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. 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 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 framework of the subject “Computer Networks”. Extended 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 and master theses; in addition to the educational benefits, the theses contribute to gradual upgrading and development of the monitoring network system including its extension for example to mobile applications.

## 5 Conclusion

Online software tools for designing CCTV systems are tools that are easier to use and in most cases solved by means of calculators. The tools can usually calculate the basic parameters of a CCTV system, which are often not sufficient for modeling of objects in which the CCTV system is to be located. Second type is represented by software tools which, once installed on a PC, are much more sophisticated environment with support of a wide range of libraries. They allow creating of 3D models of buildings and their interior equipment. Cameras with defined parameters are subsequently placed into the models. The tools can then simulate the images captured by the cameras and show previews of real footage. The last type of software are mobile applications. They predominantly operate on the same principle as the online tools.

## Acknowledgement

This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic within the National Sustainability Programme project No. LO1303 (MSMT-7778/2014) and also by the European Regional

Development Fund under the project CEBIA-Tech No. CZ.1.05/2.1.00/03.0089

## References

1. Tanenbaum, *Modern operating systems*, 2nd edition, Prentice Hall, Englewood Cliffs, NJ, 2001.
2. W. R. Stevens, *Advanced Programming in the UNIX Environment*, 2nd Ed. Addison-Wesley, Reading, MA, 2005.
3. W. R. Stevens, *TI - UNIX network programming*, Prentice-Hall, Englewood Cliffs, NJ, 1998.
4. S. Tanenbaum & A. S. Woodhull, *Operating Systems*: Design and Implementation, Prentice-Hall, Englewood Cliffs, NJ, 2006.
5. Silberschatz, P. B. Galvin, G. Gagne, *Operating systems concepts*, 7th Ed. John Wiley & sons, Inc., NY, 2002.
6. S. J. Bigelow, Troubleshooting, *Maintaining & Repairing Networks*, 1st Ed. Osborne/McGraw-Hill, NY, 2002.
7. Z. Nedic, J. Machotka, A. Nafalski, *Remote laboratories versus virtual and real laboratories*, Frontiers in Education, 2003. FIE 2003. 33rd Annual 1 (2003), T3E-1-T3E-6.
8. M. Domínguez, J. J. Fuertes, M. A. Prada, S. Alonso, A. Morán, *Remote laboratory of a quadruple tank process for learning in control engineering using different industrial controllers*, Comput. Appl. Eng. Educ., Vol. 22, No. 3, 2011, pp. 375-386.
9. K. Kolomazník, M. Adámek, I. Anděl, M. Uhlířová, *Leather waste - Potential threat to human health, and a new technology of its treatmen*t. J. Haz. Mat., Vol. 160, No. 2-3, 2008, pp. 514-520.
10. K. Kolomaznik, M. Mladek, F. Langmaier, D. Janacova, M. M. Taylor, *Experience in industrial practice of enzymatic dechromation of chrome shavings*, J. Am. Leather Chem. As., Vol. 94, No. 2, 1999, pp. 55-63.