BULETINUL INSTITUTULUI POLITEHNIC DIN IAŞI
Publicat de
Universitatea Tehnică „Gheorghe Asachi” din Iaşi
Tomul LIX (LXIII), Fasc. 1, 2013
Secţia
ELECTROTEHNICĂ. ENERGETICĂ. ELECTRONICĂ

ANALYSIS OF DATA TRANSMITTED BETWEEN THE SERVER AND THE CLIENT THROUGH DIFFERENT TYPES OF COMMUNICATION

BY

LĂCRĂMIOARA IFTODE[*] and CRISTIAN FOŞALĂU

“Gheorghe Asachi” Technical University of Iaşi
Faculty of Electrical Engineering

Received: March 5, 2013
Accepted for publication: March 25, 2013

Abstract. This paper presents the analysis of transmitted data between the server and the client at network level for practical works in the electrical field. Regarding these issues, this paper presents a study that highlights the speed and efficiency of data transmission using known types of communications. To achieve this objective, in the practical applications it was used the graphical programming environment, LabVIEW, to make the interface for both server and client, thus underlining again the relatively simple and effective way of using this program to send and read data.

Key words: DataSocket technology.

1. Introduction

The Internet is the largest IT project of human interconnections, its development being downright spectacular, tens of millions of users are connected to the Internet at all times. Today, the Internet is a technological reality subsumed to the impetuous process of globalization, a force able to potentiate knowledge and communication, an interaction environment on a global scale, with some of the deepest social implications.

[*]Corresponding author: *e-mail*: lacra_iftode@yahoo.com

Communication between computer users and their applications has become an undisputed necessity nowadays.

Networking involves the transmission of data from one computer to another, and this process is divided into stages. Within each phase, the network operating system complies, as shown, a strict set of procedures called *protocols*, or *rules of behavior* that contribute to the successful completion of each operation.

By analysing the data transmitted over the network using communications protocols highlights the main features of each type of protocol, respectively the advantages and disadvantages of using a protocol instead of another one.

2. Data Transmission through the Internet

The Internet and any use of open electronic network have and will have a major impact on society and its future.

The applications of electrical field to be studied further in terms of data transmission, used as a communication protocol between server and client, the TCP/IP (Transport Control Protocol/Internet Protocol), UDP protocol (User Datagram Protocol) and the DataSocket Protocol, and all of these protocols are successfully used in LabVIEW graphical programming environment (Callaghan, 2007).

2.1. Use of TCP / IP to Transmit Data Between Server and Client

The Internet Network is based on TCP/IP protocol suite. The title actually identifies two of the most popular protocols of the protocol ensemble, namely TCP (Transport Control Protocol), which before data transmission, the transport level from source transmits a message to the transport level at destination, announcing it that it will transmit some data and specifying which software application must receive the data (Callon, 2003). After sending the first information message, source waits for confirmation of its being received and then starts transmitting packets that compose the message itself. That is why it is said that TCP is a protocol with *prior* connection (Basic TCP/IP Communication in LabVIEW). The second protocol is IP (Internet Protocol) used in the network. The specificity of this protocol is that a packet reaching the network for transmission is attached a meter leap or a lifespan. This is the maximum number of packet retransmissions in its attempt to find their way through the Internet, to their final destination. Using this information, the network can protect the Internet network from the packets that would endlessly loop through the system in an attempt to reach the destination. The value of this jump meter is usually 64.

TCP is responsible for ensuring reliable communication services between pair processes in separate host computers connected within the same network or in a set of interconnected networks. It provides connection-oriented data transfer in the transport – the same basic services as Sequenced Packet Protocol (SPP) conducted in XNS. TCP supports a wide range of ULP that need to send data to their pairs located on other host computers. TCP does not attempt to impose any structure of data sent by a higher level protocol and treats incoming data as a continuous string, leaving message structuring to the ULP (Cerf & Kahn, 2007).

Since TCP is designed to be independent with respect to the particular characteristics of the networks in which they operate, a general definition is given for the concept of *packet* (or *segment*) allowing the existence of packages with a length of up to 65 kB. TCP pair can send packets having the length up to the maximum length defined in the standard (65 kB). Usually, various implementations of TCP work with packets that have appropriate lengths for the network to which they are attached. TCP assigns a sequence number to each byte of the endless data string of its client (Douglas Comer, 2007). When changing pair segments, TCP labels the segment with sequence number of the first byte of the segment and the number of bytes contained in the package. This allows TCP to reassemble the data stream when it delivers it to higher levels. If it is necessary to retransmit a series of segments, TCP can repack data, combining two smaller segments into a larger one, for instance. This mechanism, motivated by the desire to increase the transmission efficiency in large distributed networks, where it is the question of minimizing the ratio of the number of bits of the header and the number of data bits, has as consequence that TCP is more complex than other transport protocols (Callaghan *et al.*, 2007).

TCP (Transmission Control Protocol) is a secure protocol that ensures reliable transfer of information without errors, losses or duplications between the applications on the two connected computers. In order to achieve data communication between two computers it is necessary to determine which of them will act as server and which as the client. In the case of the application to be analysed ("Study of parallel *RLC* circuit"), the server will be the computer with the IP address 192.168.1.113 and the client, with 192.168.1.108, IP addresses with which the two computers in the network can be identified.

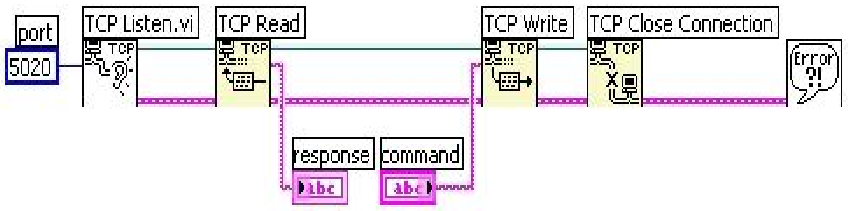


Fig. 1 – TCP/IP Protocol Functions (National Instruments – Basic TCP/IP Communication in LabVIEW).

In order to establish a connection, the TCP Listen function is used for the server and the TCP Open Conection for the client, the LabVIEW graphical programming environment featuring dedicated functions for such basic steps (Basic TCP/IP Communication in LabVIEW) (Fig. 1).

Communication is performed using specified port in the TCP Listen function, for the analysed application two ports will be used, namely port 2055 for graphics and 2056 for table data.

The client opens the connection at the address and port assigned by the server recording them in the TCP Open Conection function for the studied application connection diagram and front panel where one can notice the data presented in Figs. 2 and 3 (Basic TCP/IP Communication in LabVIEW).

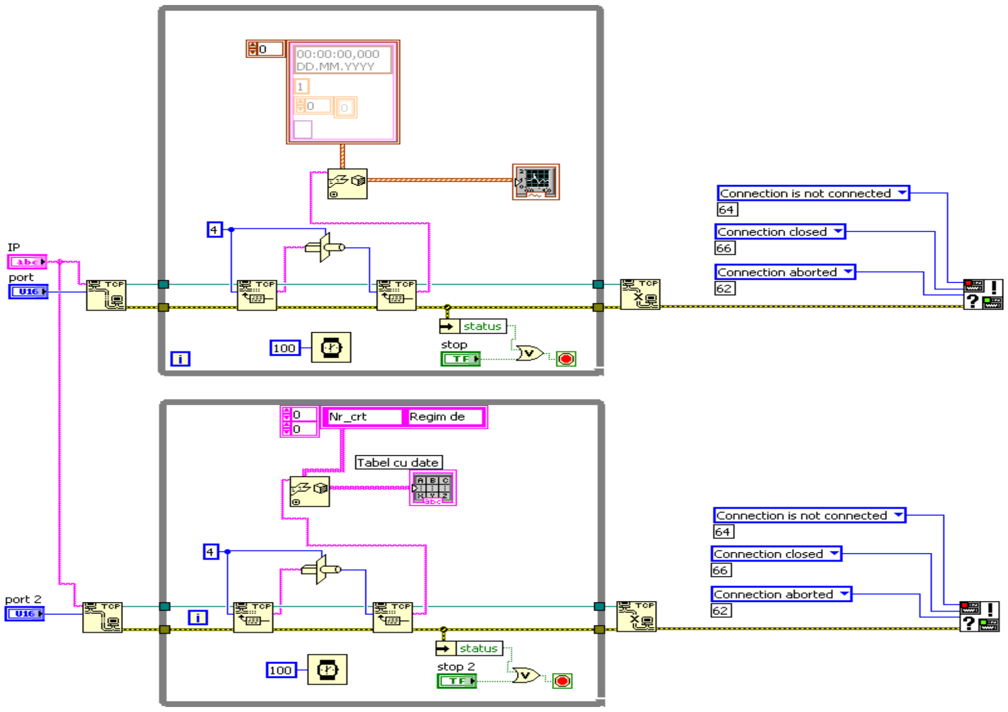


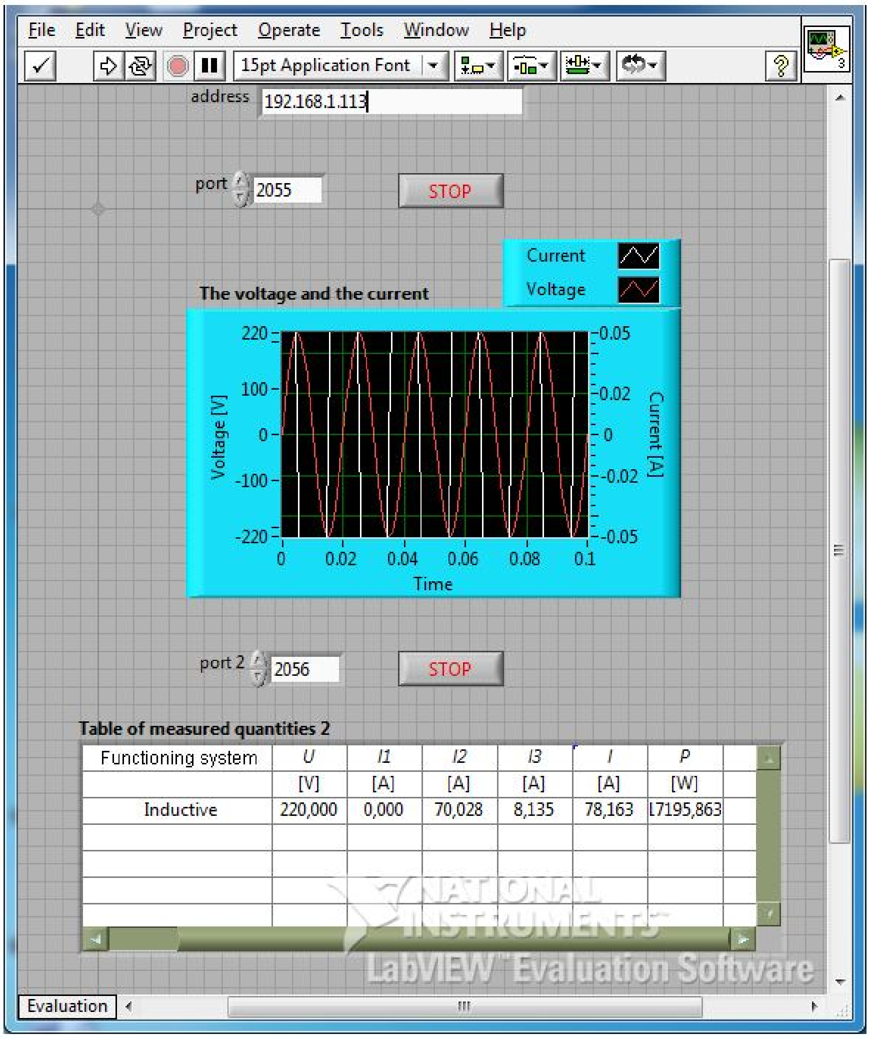
Fig. 2 – The connection diagram for the application client “Study of parallel *RLC* circuit”.

If the connection is sucessful, the function involved in establishing the connection will return a single reference name, *refnum*, which will enable the identification of the connection made.

It is important to mention that in order to connect between two workpoints it is necessary that the server virtual instrument be in execution when the client virtual instrument launches the application.

The virtual instruments with server function have as main feature the pre-established allocation. Thus, for connecting a single client, the function chain that makes the communication will start with TCP Listen to which is assigned the port to use. When more customers are connected to the server, the virtual instrument will contain a communication loop for each client.

The transmission of information between server–client and client–server *via* TCP/IP, for both ports 2055 or 2056, can be seen by analysing records of network shown in Fig. 4 (http://wireshark.en.softonic.com).



address 192.168.1.113

port 2055

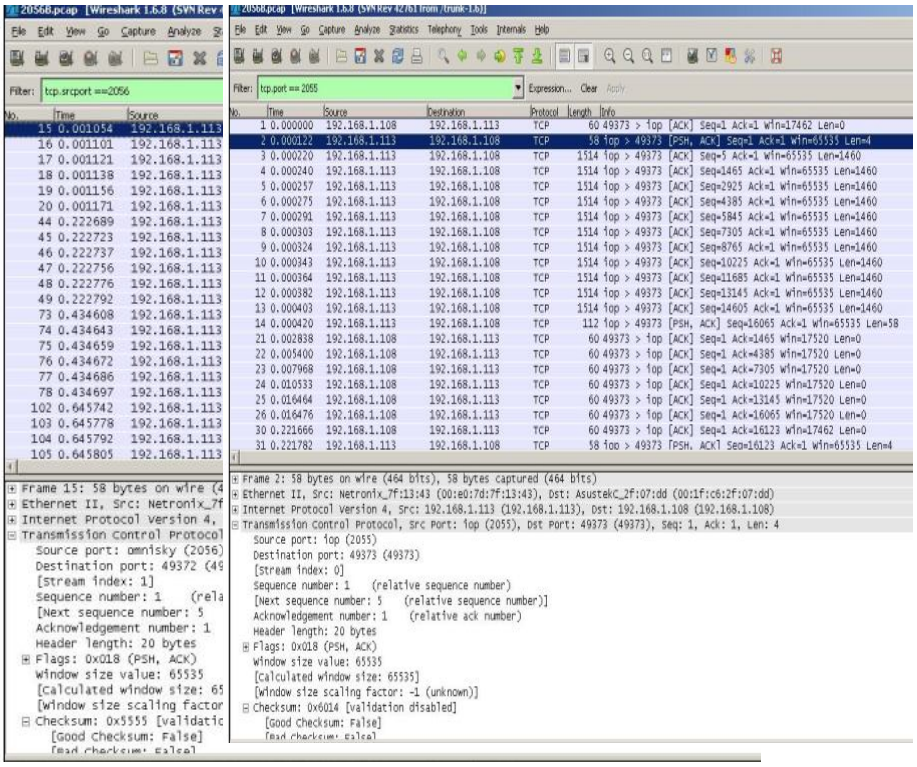
port 2 2056

Table of measured quantities 2

Functioning system	*U*	*I1*	*I2*	*I3*	*I*	*P*
	[V]	[A]	[A]	[A]	[A]	[W]
Inductive	220,000	0,000	70,028	8,135	78,163	17195,863

Fig. 3 – Front panel of the application client “Study of parallel *RLC* circuit”.

As it can be seen in Fig. 4 data are sent as data packets, assigning to each port a number based on the submitted information, the time allocated to the data transmission being on the two ports.



```
20568.pcap [Wireshark 1.6.8 (SVN Rev
Filter: tcp.srcport ==2056
No.   Time      Source
 15 0.001054  192.168.1.113
 16 0.001101  192.168.1.113
 17 0.001121  192.168.1.113
 18 0.001138  192.168.1.113
 19 0.001156  192.168.1.113
 20 0.001171  192.168.1.113
 44 0.222689  192.168.1.113
 45 0.222723  192.168.1.113
 46 0.222737  192.168.1.113
 47 0.222756  192.168.1.113
 48 0.222776  192.168.1.113
 49 0.222792  192.168.1.113
 73 0.434608  192.168.1.113
 74 0.434643  192.168.1.113
 75 0.434659  192.168.1.113
 76 0.434672  192.168.1.113
 77 0.434686  192.168.1.113
 78 0.434697  192.168.1.113
102 0.645742  192.168.1.113
103 0.645778  192.168.1.113
104 0.645792  192.168.1.113
105 0.645805  192.168.1.113

⊞ Frame 15: 58 bytes on wire (4
⊞ Ethernet II, Src: Netronix_7f
⊞ Internet Protocol Version 4,
⊟ Transmission Control Protocol
    Source port: omnisky (2056)
    Destination port: 49372 (49
    [Stream index: 1]
    Sequence number: 1    (rela
    [Next sequence number: 5
    Acknowledgement number: 1
    Header length: 20 bytes
  ⊞ Flags: 0x018 (PSH, ACK)
    Window size value: 65535
    [Calculated window size: 65
    [Window size scaling factor
  ⊟ Checksum: 0x5555 [validatic
      [Good Checksum: False]
      [Bad Checksum: False]
```

```
20568.pcap [Wireshark 1.6.8 (SVN Rev 42761 from /trunk-1.6)]
Filter: tcp.port == 2055
No.  Time      Source         Destination    Protocol Length Info
 1 0.000000  192.168.1.108  192.168.1.113  TCP      60 49373 > iop [ACK] Seq=1 Ack=1 Win=17462 Len=0
 2 0.000122  192.168.1.113  192.168.1.108  TCP      58 iop > 49373 [PSH, ACK] Seq=1 Ack=1 Win=65535 Len=4
 3 0.000220  192.168.1.113  192.168.1.108  TCP    1514 iop > 49373 [ACK] Seq=5 Ack=1 Win=65535 Len=1460
 4 0.000240  192.168.1.113  192.168.1.108  TCP    1514 iop > 49373 [ACK] Seq=1465 Ack=1 Win=65535 Len=1460
 5 0.000257  192.168.1.113  192.168.1.108  TCP    1514 iop > 49373 [ACK] Seq=2925 Ack=1 Win=65535 Len=1460
 6 0.000275  192.168.1.113  192.168.1.108  TCP    1514 iop > 49373 [ACK] Seq=4385 Ack=1 Win=65535 Len=1460
 7 0.000291  192.168.1.113  192.168.1.108  TCP    1514 iop > 49373 [ACK] Seq=5845 Ack=1 Win=65535 Len=1460
 8 0.000303  192.168.1.113  192.168.1.108  TCP    1514 iop > 49373 [ACK] Seq=7305 Ack=1 Win=65535 Len=1460
 9 0.000324  192.168.1.113  192.168.1.108  TCP    1514 iop > 49373 [ACK] Seq=8765 Ack=1 Win=65535 Len=1460
10 0.000343  192.168.1.113  192.168.1.108  TCP    1514 iop > 49373 [ACK] Seq=10225 Ack=1 Win=65535 Len=1460
11 0.000364  192.168.1.113  192.168.1.108  TCP    1514 iop > 49373 [ACK] Seq=11685 Ack=1 Win=65535 Len=1460
12 0.000382  192.168.1.113  192.168.1.108  TCP    1514 iop > 49373 [ACK] Seq=13145 Ack=1 Win=65535 Len=1460
13 0.000403  192.168.1.113  192.168.1.108  TCP    1514 iop > 49373 [ACK] Seq=14605 Ack=1 Win=65535 Len=1460
14 0.000420  192.168.1.113  192.168.1.108  TCP     112 iop > 49373 [PSH, ACK] Seq=16065 Ack=1 Win=65535 Len=58
21 0.002838  192.168.1.108  192.168.1.113  TCP      60 49373 > iop [ACK] Seq=1 Ack=1465 Win=17520 Len=0
22 0.005400  192.168.1.108  192.168.1.113  TCP      60 49373 > iop [ACK] Seq=1 Ack=4385 Win=17520 Len=0
23 0.007968  192.168.1.108  192.168.1.113  TCP      60 49373 > iop [ACK] Seq=1 Ack=7305 Win=17520 Len=0
24 0.010533  192.168.1.108  192.168.1.113  TCP      60 49373 > iop [ACK] Seq=1 Ack=10225 Win=17520 Len=0
25 0.016464  192.168.1.108  192.168.1.113  TCP      60 49373 > iop [ACK] Seq=1 Ack=13145 Win=17520 Len=0
26 0.016476  192.168.1.108  192.168.1.113  TCP      60 49373 > iop [ACK] Seq=1 Ack=16065 Win=17520 Len=0
30 0.221666  192.168.1.108  192.168.1.113  TCP      60 49373 > iop [ACK] Seq=1 Ack=16123 Win=17462 Len=0
31 0.221782  192.168.1.113  192.168.1.108  TCP      58 iop > 49373 [PSH, ACK] Seq=16123 Ack=1 Win=65535 Len=4

⊞ Frame 2: 58 bytes on wire (464 bits), 58 bytes captured (464 bits)
⊞ Ethernet II, Src: Netronix_7f:13:43 (00:e0:7d:7f:13:43), Dst: AsustekC_2f:07:dd (00:1f:c6:2f:07:dd)
⊞ Internet Protocol Version 4, Src: 192.168.1.113 (192.168.1.113), Dst: 192.168.1.108 (192.168.1.108)
⊟ Transmission Control Protocol, Src Port: iop (2055), Dst Port: 49373 (49373), Seq: 1, Ack: 1, Len: 4
    Source port: iop (2055)
    Destination port: 49373 (49373)
    [Stream index: 0]
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 5    (relative sequence number)]
    Acknowledgement number: 1    (relative ack number)
    Header length: 20 bytes
  ⊞ Flags: 0x018 (PSH, ACK)
    Window size value: 65535
    [Calculated window size: 65535]
    [Window size scaling factor: -1 (unknown)]
  ⊟ Checksum: 0x6014 [validation disabled]
      [Good Checksum: False]
      [Bad Checksum: False]
```

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Expression... Clear Apply

Fig. 4– Information exchange between client and server *via* TCP/IP.

2.2. Use of the User Datagram Protocol for Data Transmission between Server and Client

UDP (User Datagram Protocol) is the other protocol in the IP Host-to-Host level. UDP provides a simple type of data transmission with low load: datagrams (UDP Communication in LabVIEW).

The simplicity of datagrams makes UDP to be inadequate for some applications, but perfect for sophisticated applications, which have their own connection-oriented functions. Other possible uses of UDP include data changes such as routing tables, system messages, network monitoring data, etc. These types of data exchanges require no flow control, acknowledging or rearranging of packages.

UDP was designed to be a minimal and efficient transport protocol. These attributes are reflected directly in the structure of its header.

UDP does not provide any of the advanced features that TCP offers. There are no synchronization mechanisms, flow control, congestion management, confirmation, etc. UDP does its best to deliver a datagram. If it fails, the datagram is removed and there is no attempt to retransmit it. The UDP Protocol (User Datagram Protocol) is a means of transmitting small (about 65 kB), not critically important data packets to one or more recipients. Unlike TCP, UDP protocol does not guarantee that the data will reach their destination or the fact that different packages will arrive in the order they were sent (UDP Communication in LabVIEW).

Not having control communication facilities, the data transmission or reception *via* UDP protocol do not require explicit specification of the other "end" of the connection. A client only needs to listen on the specified UDP port and he will receive any data packet sent to that port of his.

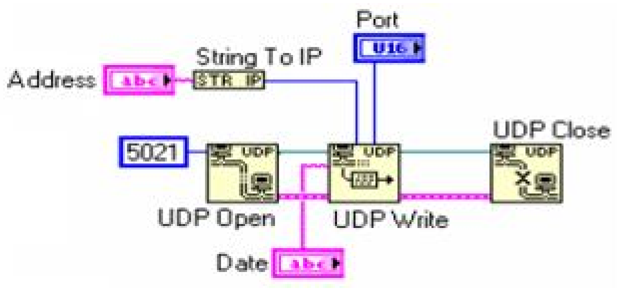


Fig. 5 – UDP Connection.

For a server to send a data packet through this protocol, it is sufficient to open a connection on an UDP port (Fig. 5), to access the UDP Write function, specifying the IP address of the recipient, its UDP port and the data to be sent, and then to possibly close that connection (UDP Communication in LabVIEW).

As most routers recognize IP addresses where a byte with value of 255 has the meaning of "send to all in the field", one can use addresses like 36.122.32.255. Trying to send to 255.255.255.255 address to the "whole Internet" will result in sending data packet only in the subnet because subnetworks avoid to transmit their data to addresses that are outside of them.

In order to read data sent to one of its ports after opening a connection on that port, a client must access the UDP Read function that also offers information about the IP address and the server port from which the package data was transmitted (Calloni, 2003). In Figs. 6 and 7, which are diagrams of client and server connections, one can see the UDP protocol functions used for transmitting information from server to client.

Thus, for the client we used UDP Open functions to open the application and for reading the information sent by the server we used the UDP Read function which provides information regarding the IP address and the server port from which the packet was sent. It can also be seen that the data sent by the server are received in bundle format, and the function used for the reception is Unflatten From String which is a function used to convert a string into a format required by the user.

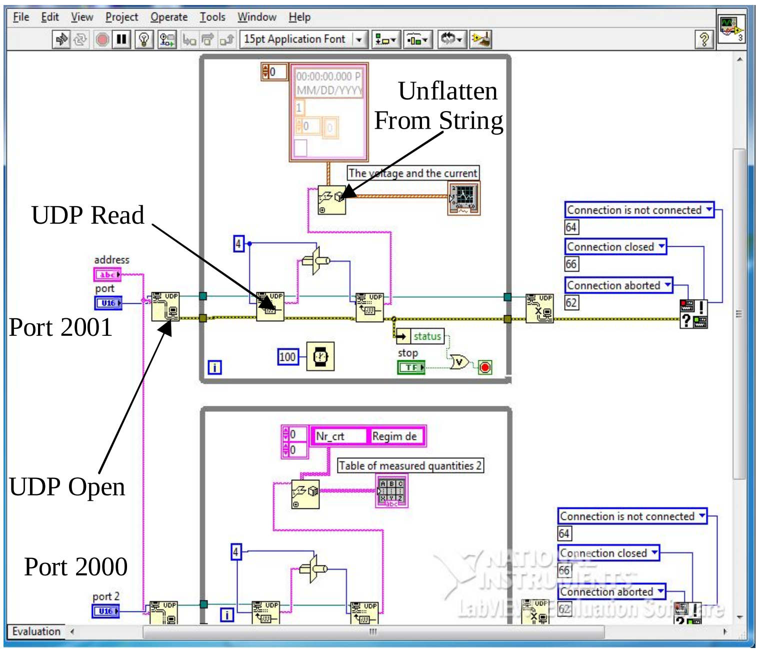


Fig. 6 – The connection diagram for the client for the application “Study of parallel *RLC* circuit”.

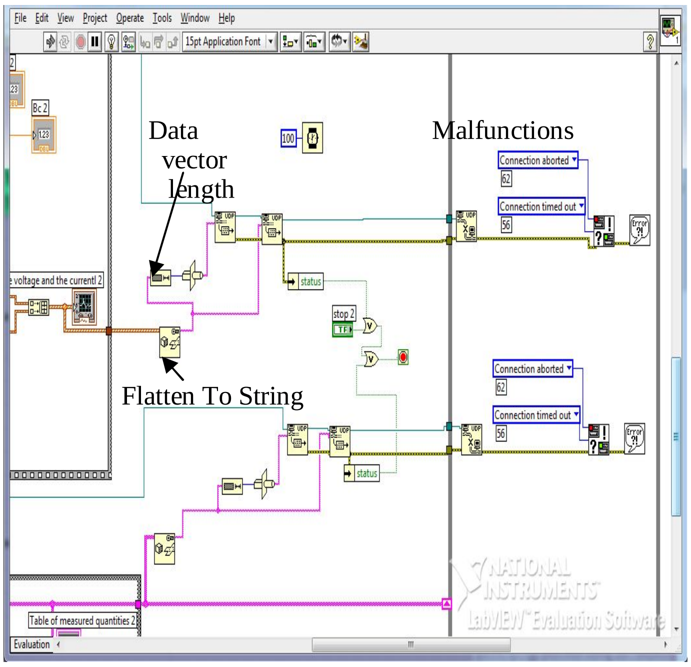


Fig. 7 – The connection diagram for the server for the application “Study of parallel *RLC* circuit”.

For a server to send information through this protocol, it is sufficient to open a connection on an UDP port (Fig. 5), to access the UDP Write function, specifying the IP address of the recipient, its UDP port and the data to be sent, and then to possibly close that connection (UDP Communication in LabVIEW). In Fig. 7 it can be seen that for the data transmission in bundle format it was used the Flatten-To-String conversion function, which is designed to convert data into a string of binary elements. In the same figure it can be seen that the data vector length is specified and also the operating malfunctions that may occur during data transmission.

3. Results

The various features that LabVIEW graphical programming environment provides for the transfer of information enable the development of applications for virtual laboratories with varying degrees of complexity and different levels of user access.

LabVIEW is the most powerful software and the simplest tool for acquiring, analysing and presenting data from real world, and it includes tools with which data can be analysed for more than 400 analysis functions.

The way the information is transferred using TCP/IP functions allows the development of communication without using a Web browser, but it requires a more laborious activity for the application development, particularly in relation to the development of a specific protocol to organize the transmitted data. Thus, in the following, it can be seen how by using Wireshark program the information is transmitted from server to client and *vice-versa*. (http://wireshark.en.softonic.com).

Wireshark is an open source application that monitors data packets. It is used to solve problems in the network, for traffic analysis, software development and communication protocols for educational purposes. Originally, the application was named *Ethereal*, but in May 2006 the project was renamed *Wireshark* due to trademark issues. A very important tool for Wireshark is libcap library with which the network packets are captured (Fig. 4). As shown by analysing the information from network for the paper "Study of parallel *RLC* circuit", a set of data is transmitted from the server to the client by means of 19 packages grouped as follows: for transmitting table data port 2056 was used and a total of 6 packets, the data being transmitted in a very short period of time as seen in the graph from Fig. 8.

For the transfer of information in a table in Fig. 9, the transfer time was represented for packets in the case of the first 5 series. It appears that the time elapsed between the end of the first package transfer and the second package (at the beginning of the graph transfer) the transfer time is much higher (about

0.1 ms), after which it oscillates around 0.02 ms (the time of transfer is 5 times lower).

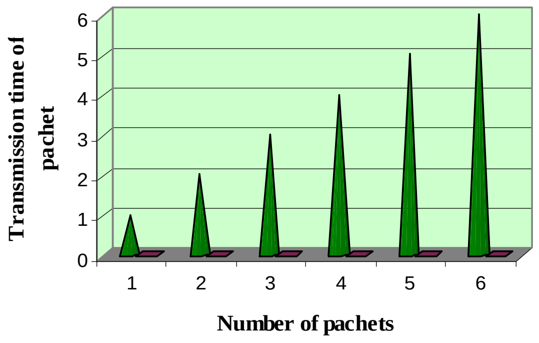


Fig. 8 – Graph for transmission *vs*. time dependency of a packet and the number of packets for port 2056 for the application “Study of parallel *RLC* circuit”.

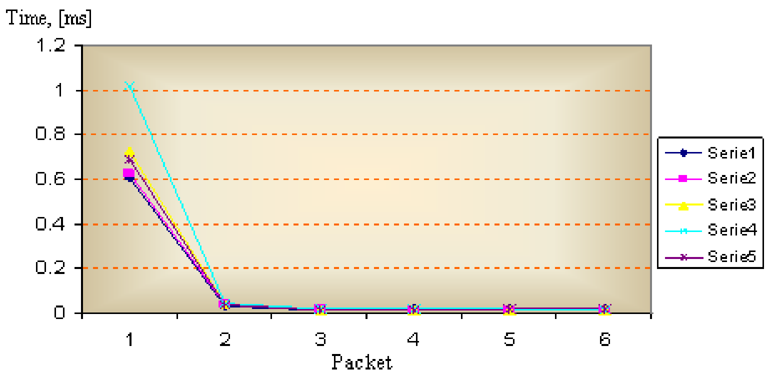


Fig. 9 – Evolution of the transfer time for the information designed for the table on each packet in the case of the first 5 series (the case of the 6 packets) for the application “Study of parallel *RLC* circuit”, port 2056.

When transmitting data to define the graph a total of 13 packages were used, the time allocated for their transmission being higher, which it can be seen in the graph from Fig. 10.

In Fig. 11 a large difference in the transfer times can be observed at the two ports.

The transfer of information *via* UDP protocol for the application “Study of parallel *RLC* circuit”, is done as with TCP / IP on two ports, namely: on port 2000 used for the transfer of information for graph, data transmitted through 10 packets, or port 2001 used the information in table transmitted through 4 packets (http://wireshark.en.softonic.com).

In Fig. 12 are represented the transfer times for each packet sent on both ports for 20 sets of data transmitted from server to client.

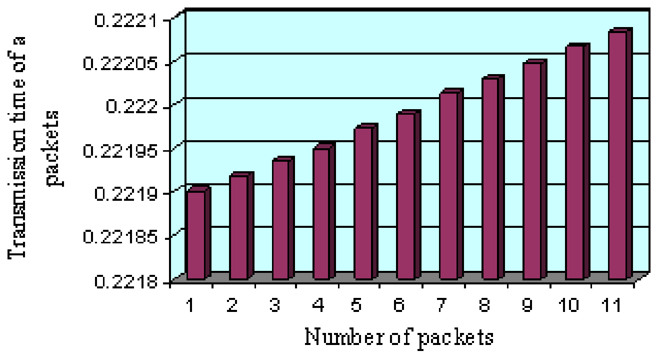


Fig. 10 – Evolution of the transfer time for the information designed for the graph on each packet, port 2055, for the application “Study of parallel *RLC* circuit”.

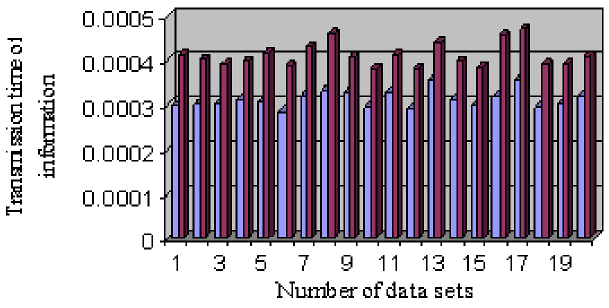


Fig. 11 – Evolution of the transfer time for the information on each packet on ports 2055 and 2056 for the application “Study of parallel *RLC* circuit”.

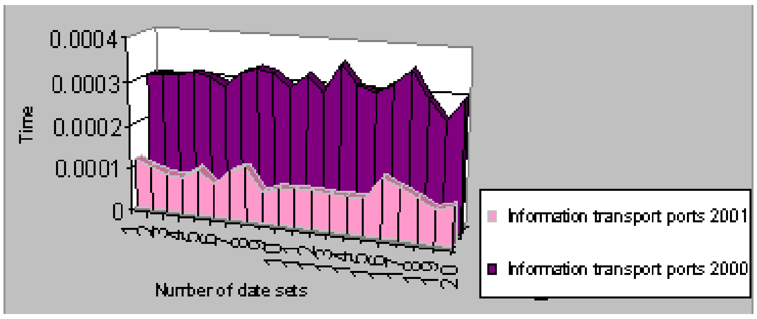


Fig. 12 – Evolution of the transfer time for the information on each packet on ports 2000 and 2001 for the application “Study of parallel *RLC* circuit”.

It is important to note that in the case of this protocol the information sent from the server to the client get truncated, therefore this protocol is used less in providing information for applications in the electrical field. In Fig. 13 it can be seen that the number of packets sent by the server is different from the one that reaches the client.

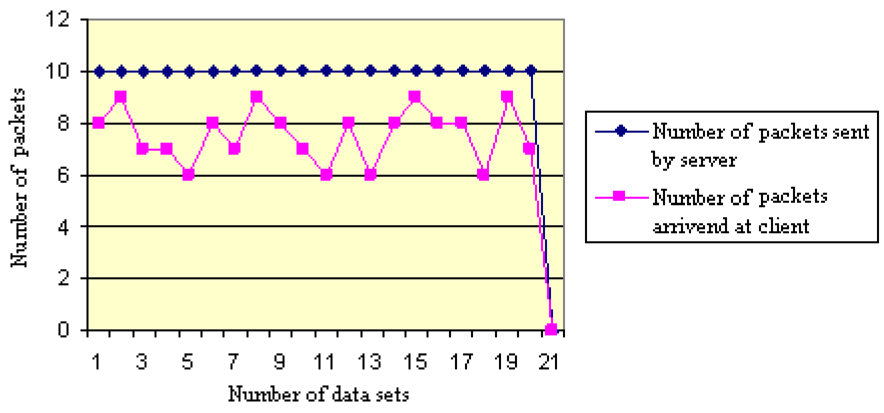


Fig. 13 – Information sent by the server reaching the client for the 2000 port for the application "Study of parallel *RLC* circuit".

The observations highlighted in the chart above lead to the conclusion that the protocol that is used to efficiently transmit information, in full size without truncation is the TCP/IP protocol used by most people who need quick and efficient data transmission.

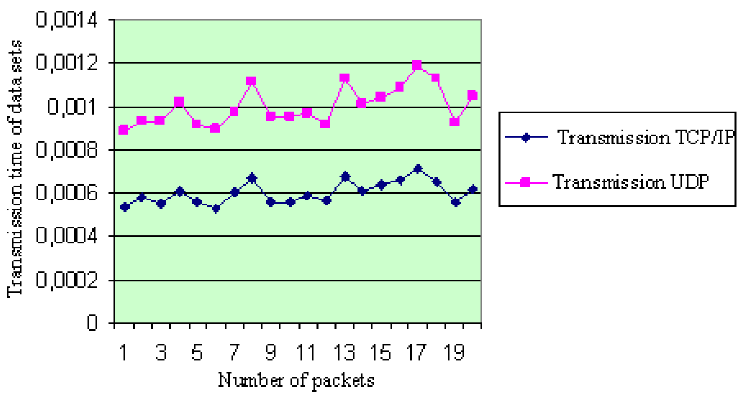


Fig. 14 – Information transport on protocol TCP/IP and UDP for the application "Study of parallel *RLC* circuit".

In the Fig. 14 we have biglelighted the fact that, in the case of TCP/IP protocol, the data transmission time is shorter than in the case of UDP protocol, which is slower and has no clear structure as far as data transmission is concerned.

We can also mention the fact that the TCP/IP protocol is safer, the data transmission between the server and the client is safe, complete and orderly; we cannot say the same about the UDP protocol, where the data is transmitted in a fragmentary way, and there is no way we can correct or add missing data.

This leads to the conclusion that this protocol which can be used to transfer data efficiently and quickly, in a complete format, with no repetitions or corrections of the data which haven't reached their destination safely (the client), is the TCP/IP protocol, which is the choice protocol of most users, specifically thanks to the characteristics we have mentioned above. These characteristics make this protocol (TCP/IP) a safe, quick, efficient way of data transmission.

4. Conclusions

The various features that LabVIEW graphical programming environment provides for the transfer of information enable the development of applications for virtual laboratories with varying degrees of complexity and different levels of user access.

Regarding the analysis of information transmitted through different types of communication, it can be seen that the information transmitted *via* TCP/IP protocol is considered reliable, fast, and easy to use depending on the type of data to be processed, analysed and transmitted; this cannot be said about the UDP protocol which most often does not faithfully transmit the information and in this case the client is not be able to solve interpretations or imposed requirements.

Therefore, for the development of applications such as virtual laboratory with varying degrees of complexity and different levels of user access, the TCP/IP protocol is used and through its available functions it is able to analyse demands, to transmit, to process and to analyse the required data.

REFERENCES

Abu-Aisheh A., Farahmand F., *LabVIEW-based Integrated Virtual Learning Platform* (*IVLP*). 10th IEEE Internat. Conf. on Adv. Learn. Technol., Sousse, Tunisia, 2010.

Callaghan M.J., Harkin J., McColgan E., McGinnity T.M., Maguire L.P., *Client–Server Architecture for Collaborative Remote Experimentation*. J. of Network a. Comp. Appl., **30**, 1295-1308 (2007).

Callon R., *Internetwork Protocol*. Proc. of the IEEE, **71**, *12*, 1388-1392 (2003).

Cerf V., Kahn R., *A Protocol for Packet Network Intercommunication*. IEEE Trans. Commun., **22**, *5*, 637-648 (2007).

Douglas Comer E., *Internetworking with TCP/IP – Principles, Protocols and Architecture*. In *Internetworking with TCP/IP*. T. **I**, Prentice Hall, NY, 2006.

Grimaldi D., Lamonaca F., *Dynamic Configuration of Measurement Procedures on PDA by Using Measurement Application Repository Server*. Proc. of IEEE Internat. Workshop on Intell. Data Acq. a. Adv. Comp. Syst.: Technol. a. Appl., Dortmund, Germany, 2007.

Munteanu A., Greavu V., *Reţele de calculatoare, proiectare şi administrare*. Edit. Polirom, Iaşi, 2006.

Ursuţiu D., Cotfas P., Samoilă C., *Self Growing Remonte Controlled Laboratory*. Internat. J. of Online Engng., JOE, **2**, *1* (2006).

* * * *Integrating the Internet into Your Measurement System-DataSocket Tehnical Overview*. National Instruments, http://www.ni.com/pdf/wp/wp1680.pdf.

* * * *Basic TCP/IP Communication in LabVIEW*. LabVIEW User Manual, National Instruments.

* * * *DataSocket Overview*. http://wireshark.en.softonic.com.

* * * http://zone.ni.com/devzone/taskdoc.nsf/webmain/24F68725CB69FBE586256802007B8D43?opendocument&node=DZ52052_US

* * * *UDP Communication in LabVIEW*. http://zone.ni.com/devzone/taskdoc.nsf/webmain/BB41313D7AB28CBD86256802007B8DC5?opendocument&node=DZ52052_US

* * * http://zone.ni.com/devzone/conceptd.nsf/webmain/14F3B2BC11811DC18625A9D0068B2D1?opendocument&node=DZ52047_US

ANALIZA DATELOR TRANSMISE ÎNTRE SERVER ŞI CLIENT PRIN DIFERITE TIPURI DE COMUNICAŢII

(Rezumat)

Laboratoarele virtuale, întâlnite în toate programele ştiinţifice şi tehnice, reprezintă o parte importantă din experienţa educaţională. Ele nu doar arată concepte şi idei de la cursuri dar şi pun în practică teoria prezentată la cursuri astfel încât utilizatorii să poată observa cum evenimente şi fenomene neaşteptate pot afecta măsurătorile reale şi cum pot controla algoritmii. Se prezintă analiza informaţiilor din reţea transmise între server şi client şi invers, a unei aplicaţii din domeniul electric cu aplicaţia wireshark, mediului de programare grafică LabVIEW fiind utilizat pentru implementarea grafică a serverului şi clientului. Astfel, utilizatorii au posibilitatea de a observa modul de variaţie a unor parametri fără a fi prezenţi în faţa unui stand cu aparate reale; mai mult, prin utilizarea unor programe speciale pot vedea modul de transmitere a datelor cu timpul alocat fiecărui set de date, respectiv numărul de pachete utilizat pentru transmiterea unui set de date complet de la server la client şi invers. Prin analizarea acestor date se pot defini modalităţile cele mai simple, avantajoase şi rapide, utilizate pentru transmiterea informaţiilor corecte, netrunchiate şi ordonate.