Communication and Diagnostic Interfaces in Remote Laboratory Management Systems

http://dx.doi.org/10.3991/ijoe.v11i5.4926

M. Krbeček and F. Schauer
Tomas Bata University in Zlin, Zlín, Czech Republic

***Abstract*—There are many Remote laboratories (RLs) in the world which are created mainly by schools or some organizations. But the trend suggests that in the future all of these laboratories will be merged into large grids which will be administered by Remote Laboratory Management Systems (RLMS). Such systems are composed from many parts which communicate with each other. This communication can be crucial for correct functioning of whole system.**

The paper intends to contribute to the standardized solutions in communication schemes of RLMSs. Each part of RLMS is described in terms of communication requirements. The largest volume of data is transmitted through the Internet. The security of this communication channel based on TCP/IP protocol is described in detail. The last part of paper describes the proposal and creation of data communication and diagnostic interface for individual remote experiments included in RLMS.

***Index Terms*— Remote laboratories, remote experiments, communication interface, RLMS, remote laboratory management system, diagnostic interface.**

I. INTRODUCTION

The contemporary society is characterized by sharing of resources and assets through the Internet and growing virtualization in ICT. This approach saves the cost of expensive shared devices, available through the network. This trend can be found in a wide range of sectors of human activities in general and in science and teaching processes in particular. Teaching of natural sciences is no exception. Next we intend to describe the situation typical for universities; in other fields the situation is similar. A great deal of attention worldwide at universities and teaching institutions have been devoted to e-laboratories offering access to various remote experiments (REs) [1-2]. The ultimate goal in the creation of teaching support for a teacher are global grids of remote laboratories (RLs) and their integration into a cloud-system with an easy data retrieving, processing and storing with sufficient security.

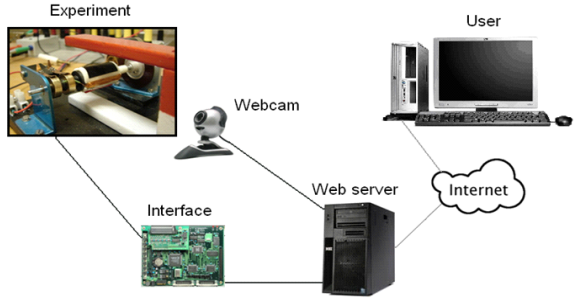

Figure 1. Schematically arrangement of the remote experiment [3]

REs are mainly designed under the auspices of the universities for the purpose of education (teaching and learning). For this reason usage of some of them is only for students of the particular university in question and access is secured by the user name and password (especially at American universities). In Europe and Australia several projects exist supporting building of open remote laboratories - with free access, either with or without registration. Some labs offer to insert client´s initials (name, country, e-mail) for voluntary statistical purposes. With the necessary exceptions (maintenance, modification, technical issues...) experiments are running on 24/7 scheme [4-8].

There are, rather unfortunately, plentiful and not compatible systems for creating of computer oriented physical experimental laboratories. But only a part of these systems allows remotely access and control of the devices via the Internet or internal network. From this group, we can select only certain systems that are suitable for the creation of remote experiments for educational and scientific purposes (iLab, Sahara, REMLABNET [9, 10 and 11]). On top of this, the first two mentioned systems are pioneering in the design and introduction of Laboratory management system (RLMS) both in USA and Australia, the last one - REMLABNET - for European use.

All described REs are based on ISES HW which is briefly described in the next section and in detail elsewhere [8].

A. Internet school experimental system (ISES)

Let us introduce ISES briefly (More detailed description is to be found in [8]) is a powerful tool for process and experiments control, acquisition, collecting and data processing in real time. Let us mention the basic features of the ISES system. The basis of the system is ISES board, which is available in several versions, differing depending on the number of inputs/outputs and also on type of communication with the control PC (by PCI card, USB connector, Wi-Fi). To this board are, by a unique connector, plugged in sensors like: ammeter, voltmeter, thermometer, position sensor, ohmmeter, load cell, anemometer, microphones, sonar, light gate, pH meter, conductivity meter, heart rate monitor, etc. Its maximum signal transfer frequency of 100 kHz allows the study of sounds or other dynamic signals, enabling simultaneous measurement, processing and displaying data via maximum eight input channels, as well as process control via two analogue and two binary output channels. The uniqueness of ISES is the used possibility of exploiting the same equipment both for experiments in the laboratory (so-called hands on experiments) and also for their remote versions - remote experiments (RE).

II. COMMUNICATION SCHEME OF REMOTE EXPERIMENTS

As we can see in Figure 1, the remote experiment consists of several constituent blocks, which have to communicate with each other for the proper operation of the RE. This communication takes place at different level of complexity, depending on the block of the experiment. It is clear that the communication among the physical hardware (measurement modules) and the D/A-A/D converters on the PCI transducer board will be completely different to control communication between the user and the server of experiment. In the same way the demands for reliability of each communication will be different accordingly. Demands vary depending on the importance of transmitted data and signals. Communication scheme of RE based on the ISES can be seen in the following Figure 2. According to the diagram, communication can be divided into two parts: communication of HW components (HW communication) and SW components (SW communication through the Internet).

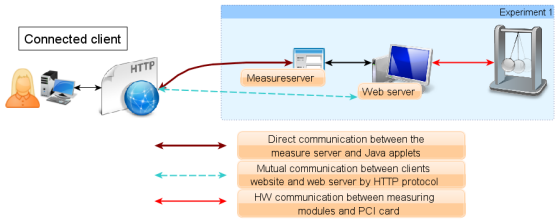

Figure 2. Communication scheme of the ISES RE

A. Hardware communication

As the HW communication we can denote all the signals and information transferred directly from the physical HW of the RE to PC. It comes mainly from individual measuring ISES modules via the ISES board into the PCI transducer board in case of the ISES system attached via the PCI bus. All this communication takes place via an analogue signal (0-5V). Since this communication takes place at very short distances, we don't have to take any precautions to maintain signal integrity. We are also limited directly by the hardware design and its possible disturbances and outages.

B. Software communication

To the SW communication group here we can include all data transmitted in the digital form. The majority of SW communication goes through the Internet towards the client. This route is mainly for bidirectional information exchange between the client and the server of the RE. This communication forms is the basis of every RE and thus is very important, because it transfers not only the measurement data from the experiment to the client but also the control commands from the client according to which the experiment performs. Therefore there must be secured real-time communication with reasonable time-delay (in order of ms) and with superior reliability with respect to the bidirectional data transfer.

We can also include the communication of experiments with RLMS to the group of SW communication. RLMS is a new concept in the field of remote laboratories, which ensures their integration into larger grids. With this concept we can meet first in USA in the project iLab [9] and in Australia in the project LabShare [10]. The similar project is Remlabnet [11] is at the European level. Here occurs another communication branch providing both the diagnostics of current working experiment, the transmission and storage of configurations of individual experiments and the measured data of individual clients for further use. Obviously even this communication must be very reliable and there mustn't be disturbance of transmission.

1) TCP/IP communication

Since all SW communication takes place via the Internet, we are talking here about TCP/IP (Transmission Control Protocol / Internet Protocol) architecture. TCP is one of the core protocols of the Internet. Applications can establish connections among themselves on the networked computers through which they can transmit data by using TCP. It is a connection-oriented protocol transmitted a stream of bytes with a reliable delivery. This is documented in IETF RFC 793 specification [12]. TCP/IP protocol is built as a three-layer formation, composed of the applications, TCP and IP parts. The communication itself is ensured by the IP part, the error correction by the TCP one.

At the Internet protocol suite TCP is the middle layer between applications above and the IP protocol below him. It ensures reliability by repeatedly sending of unreliable or lost packets and correct order rearranging of received packets. TCP plays the role of transport layer in the ISO/OSI [13] network model. The application sends a stream of bytes to TCP protocol. It divides the bytes into segments then passes the resulting packets to IP protocol to be transferred through the Internet to TCP part on the receiving side of the connection. TCP verifies that the packets are not lost by assigning the serial number to each packet, which is also used to verify that the data was received in the correct order. TCP module on the receiving side sends back an acknowledgment for packets that have been successfully received. If the acknowledgment does not return to the sender within a specified time (round-trip time, RTT), the packet is sent again. Scheme of this check is shown in the Figure 3 below. Individual packets are labelled as PDU (protocol data unit) in the Figure. The ACK then stands for acknowledgement that the packet was received.

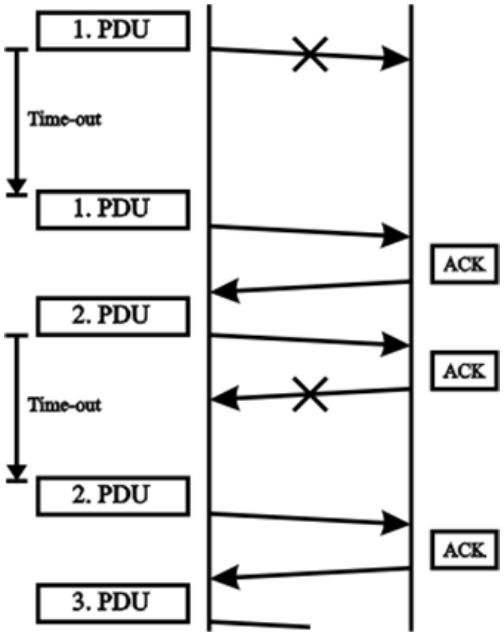

Figure 3. Scheme of the check of the receiving packet within the TCP/IP communication protocol

TCP protocol verifies that the transmitted data has not been corrupted by the noise. Before sending it calculates a checksum which is saved as a part of the packet. The recipient calculates a checksum of received data again and verifies that they match.

TCP protocol uses port numbers to distinguish between communicating applications. Each side of a TCP connection has an associated 16-bit unsigned port number (there are 65535 ports) assigned to the application. Ports are divided into three groups: well known, registered and dynamic/private. List of well-known ports are assigned by the Internet Assigned Numbers Authority (IANA) and are typically used by system processes. Well-known applications running as servers and passively accepting connections use these ports typically. Some examples: FTP (port 21 and 20), SMTP (port 25), DNS (port 53) and HTTP (port 80). Registered ports are typically used in applications of end users as an arbitrary number of source ports when opening the connection to the servers but can also identify services. Dynamic/private ports can also be used in end applications most often in non-commercial sphere.

This scheme provides access to sockets, which are a mechanism for inter-process communication. TCP sockets are supported on all operating systems. TCP sockets have two distinct interfaces: one interface to implement a client and another to implement a server. The most basic protocol is that the servers create a listening port and wait for the connection from clients.

The implementation of client´s side is a simpler one. The socket procedures accept two special arguments, called host-name and service. Host-name is a string which must be the name of an internet host. It is looked up using the ordinary lookup rules for your computer. For example, if your host is “ises.info” and host-name is "exp1", then it specifies “exp1.ises.info”. This service specifies the application type to which one wants to be connected. A networked computer normally provides several different services, such as telnet or FTP. Each service is associated with a unique port number. The service argument specifies the port number, either as a string, or directly as an exact non-negative integer. Port strings are decoded by the operating system using a table, for example, on UNIX the table is in /etc/services. Numerical record of addresses containing the number of the host computer and designation its services (port) can look like this: 195.178.94.33:8000.

After establishing a socket connection between two computers (server-client) that is performed directly in the application that uses this connection the I/O port is available to which you can read and write characters using ordinary scheme I/O procedures such as read and write (depends on programing language). The other parameters can be further implemented for each opened socket connection such as buffer-size (specifies the size of the read and write buffers used by the port), etc. These parameters depend on the application itself, which used connection.

So we can say that with proper implementation of socket connections on TCP protocol and sufficient bandwidth of the Internet connection we reach sufficient reliable data transmission for both control and information data from REs.

III. COMMUNICATION NEEDS OF LABORATORY MANAGEMENT SYSTEM (RLMS)

Remote laboratories are usually not affordable to the outside world and confined to the use of their own university owing to the complexity of their interoperation. The inter-institutional sharing of remote laboratories is now more and more considered to be an important objective of their employment. Thus, creating remote laboratories able to be integrated in a generic and inter-institutional framework rather than limited to the individual academic institution is currently the trend in remote laboratories development. Exactly this task solve the RLMSs, giving answer to these needs by providing a common portal for the accessing and administrating a wide pool of heterogeneous but functioning remote laboratories that might be distributed among several universities in order to span their dissemination and inter-institutional cooperation. The pioneer RLMSs that have been adopted at many universities include the iLab Shared Architecture (ISA - USA) [9] and Sahara (Australia) [10].

Depending on the new trends in the field of remote experiments arises the need to create RLMS for European area. Since the two above mentioned systems provide interconnection of American and Australian laboratories and at present are not affordable for European participants. For this reason we decided to create such RLMS at our university first for the internal and testing purposes but subsequently expandable to the Czech Republic and later serving to the whole Europe. RLMS was primarily built on the ISES platform however it will allow the inclusion of any remote experiment. Such RLMS working in a cloud raises a number of new channels of communication which must be built. The scheme our proposed RLMS is shown in the Figure 4 below.

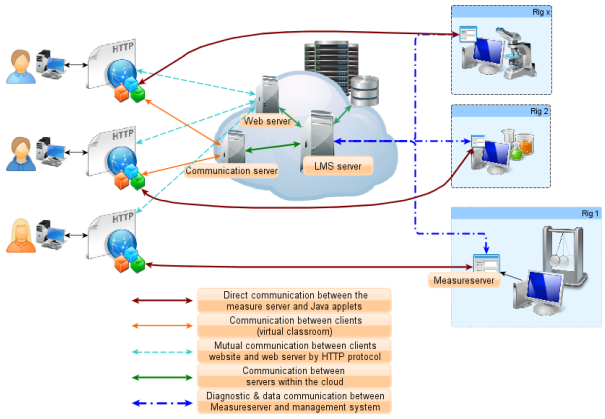

Figure 4. Communication scheme of designed LMS

Compared to the original communication scheme of remote experiments, in RLMS there are three new types of communication. The first is the communication of the servers in the cloud (green line in Figure 4). This communication is ensured in the design of the cloud structure and takes place among the participating data centres. The second new channel of communication is communication among individual clients themselves via the communication server (orange line in Figure 4). On the client’s side will be used a simple messaging client most likely based on the JS language. Communication server will provide connection and transmission of video, audio and text messages between connected users. The last new channel of

communication is the transfer of information between the "MeasuringServer" (MS) and the management server in the cloud (blue line in Figure 4). Transfer of measurement data and its storing for later use will take place there. This channel will also provide diagnostics of individual experiments and remote repair of their malfunctioning.

Next parts of the paper will be focused on creating of a communication interface for the data transmission and a communication between MS and cloud.

IV. COMMUNICATION AND DIAGNOSTIC INTERFACES OF RLMS

Let us focus on the transfer and storage of the measured data between the experiment-site (rig) and the management server of the RLMS. Data output from the MS are implemented in the form of a text file. This text file is always compiled each time at measurements end. The first task is to transfer this file with measured data and store it in the RLMS storage. For this purpose communication interface was created. It consists of a part on the rig-side which controls data-file compiling and ensures its sending and the second part on the RLMS server that saves the file on the RLMS storage and possibly parses the received data into the database.

A. Creation of communication interface at the experiment-side

The experiment side communication interface is created in the Java programming language, primarily because of its portability among platforms. The application starts at the same time as MS. It allows the client to set the basic parameters of communication, later it runs as a background service. The parameters of the communication interface are pre-set according to the type of experiment and its installation.

Requirements for the communication service:

- The service repeatedly checks for a new data file,
- When the new file occurs the service checks if the file is readable (to avoid collision during creation of file),
- Service establishes socket connection with the opposite service running on the server,
- It sends the file in the byte form,
- After completion of dispatch service waits for confirmation of the correct reception of a file otherwise it sends the file again,
- It closes the connection,
- It deletes the uploaded file,
- Then service continues in cyclic checks of the newly created files.

Adjustable parameters of communication service:

- The communication address and port No,
- The interval of cyclic control of new file (depending on the time of measurement),
- The path to the new data files.

All these parameters will be present to correct values automatically.

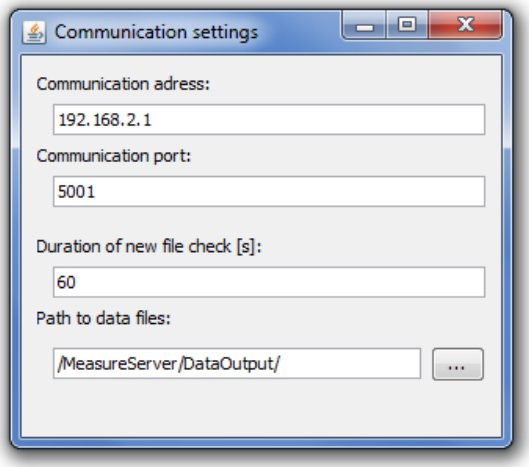

Figure 5. Communication settings of experiment-side service

B. Creation of communication interface at the server-side

The server-side communication interface is created in the Java programming language, primarily because of its portability among platforms. It allows administrator to set the basic parameters of connection and saving of data at start. Then it runs as a background service. It is designed as stand-alone application but it can be easily included into management system.

Requirements for the communication service:

- Service listens on a particular port and waits for connection,
- Service establishes socket connections to the opposite service at the experiment-side,
- It receives a file in a byte form,
- After completion of reception service checks the integrity of the file and sends confirmation of correct reception of the file otherwise receiving will start again,
- Service ends the connection,
- It saves the received file to the selected directory / reads the received file and stores the data into the appropriate place of the database.

Adjustable parameters of communication service:

- The communication address and port No,
- The path to the directory for file storage / access settings to database.

All these parameters will be present to correct values automatically.

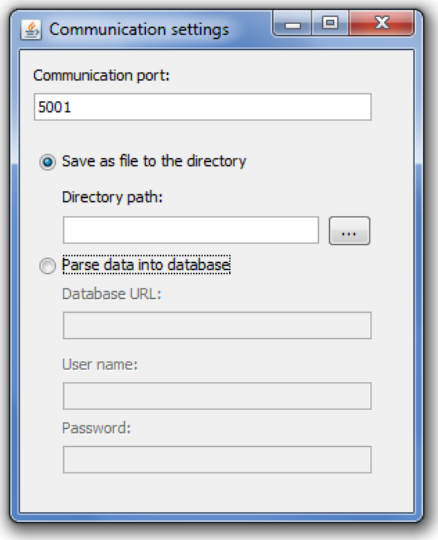

Figure 6. Communication settings of server-side service

C. *Integration of communication services into a current system*

Created communication services were used for a testing purposes and optimisation of transferred data format. But an aspect of user simplification calls for an integration of this services into MS. Because of that whole created solution was transformed as a block of code and integrated directly into a source code of MS. All parameters of original solution were added to settings of MS. When the server is started, it checks connection with RLMS and informs user if his/her data are backed up or not. MS sill creates files with measured data which are stored on local hard drive. But since the backup is operable this function can be turned off.

Transferred data are labelled with MAC addresses of experiment PC and a time mark. This serves as a unique identification of experiment and ensures correct linking with current user of experiment. Each user of REMLABNET has access to his/her historical measurements on each experiment connected into the system. This data can be further processed directly in our system or exported as a data file.

D. *Design of diagnostic communication interface*

Diagnostic services should ensure keeping track of the status of the experiments connected to the RLMS. Depending on the activity of the experiment, its breakdown or failure the status will be displayed at the access portal of RLMS. Diagnostic should also allow sending commands to the experiment in case of detected faults. From the description of the functionality it's clear that we need to communicate directly with MS at the experiment-side because only there can be found the latest information about the availability and status of the experiment. It is necessary to intervene in the source code of MS and to program diagnostic communication interface directly here. Depending on this is necessary to establish a communication interface on the server-side (cloud) which will communicate with MS. For this reason only a server part will be proposed and described in detail, the solution of MS part will be subject of the following paper.

Because the RLMS will include many experiments, it is crucial that the interface will allow connection and supervision of many experiments, and, on the other hand, will be able to identify the individual experiments and communicate with them individually. As there will be the IP and MAC address of each experiment stored in the server database, it will be appropriate to use these addresses for identification. For this purpose a service on the server-side will be created, which will allow connection of arbitrary number of experiments. Service will maintain a list of currently connected IP/MAC addresses and will allow communication with them.

The basic functionality of diagnostics is to determine whether the experiment is at present available or not. This is accomplished in several ways. The basic feature of the experiment´s availability is the list of connected IP addresses. When you start the experiment the socket connection is created with the service on the server-side. This connection remains active until it is closed by one of the parties or one of them is disconnected. In case of malfunction occurring and experiment turning off, the service will be closed, connection will be lost which indicating experiment inaccessibility.

However an error may occur even with the experiment, which has not terminated the connection but it is not proper for measurements. For this purpose another precaution is operative, as the server service will send a regular broadcast request for the status report to all connected experiments after a certain time frame. They will replay its current status or be considered as out of function. In this way not only fault of experiment will be detected, but its occupancy as well. Furthermore, the experiments themselves report their status change automatically. Log of experiment reports at diagnostics server is shown in (Figure 7). All these measures should ensure the availability of the actual table of experiments status at any time.

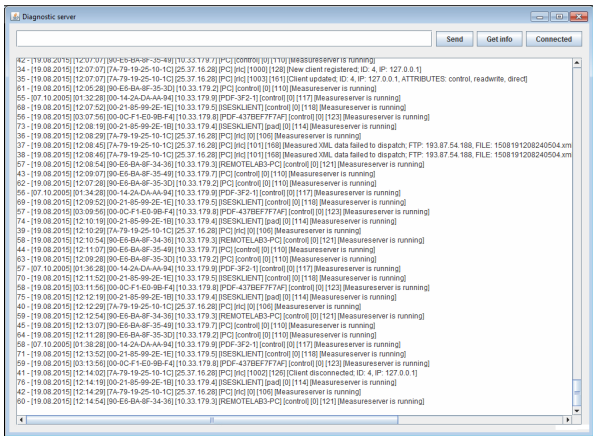

Figure 7. Diagnostic server report log

Besides, the diagnostic system allows so-called automatic self-repair of an experiment. In case of the experiment´s failure and communication running (experiment answers on the broadcast by error) the server sends the instruction to perform a certain actions according to the type of error. One of them may be the restart of the experiment or the whole computer to which experiment is attached, which makes experiment operational in most cases.

If these or other measures don't bring the experiment back to operation, the system will contact the administra-

tor of the experiment automatically. System will notify about found errors and will request the check of experiment. The system supports not only the diagnostics of the experiment as a whole but its parts as well to relieve the supervising load of the remote experiment administrator as much as possible. From the readings transmitted from the individual sensor, the system will be able to determine its malfunction easily giving the clue for its repair to administrator.

E. *Current state and future development of diagnostic system*

The first part of diagnostic interface at experiment-side is already created at this time. It was implemented directly into a code of MS. Current version provides connection with diagnostic server, responding on server broadcast and sending of actual experiment state. It already contains processing of received commands like a software or PC restart. We are still working on the deep data diagnostic which will indicate malfunction of measuring modules. Server-side of diagnostic interface is also under development. Current version is sufficient for testing and provides basic diagnostic functions. But advanced decision algorithms for experiments problem solving were not created yet.

The first step in the future development will be the integration of the server-side diagnostic interfaces into one robust unit, which will be built on the Apache Tomcat. Apache Tomcat (or simply Tomcat, formerly Also Jakarta Tomcat) is an open source web server and servlet container developed by the Apache Software Foundation (ASF). Tomcat implements the Java Servlet and the Java Server Pages (JSP) specifications from Sun Microsystems, and provides a "pure Java" HTTP web server environment for Java code to run in [14]. Since the source code of the created diagnostic interfaces are also written in Java, the implementation into this system will be very easy (just copied). We can say that created applications are a kind of preparation for allowing the development and testing of RLMS system but also part of the final solution of its diagnostic interface.

The RLMS server will certainly undergo a series of changes and get many additional features, which will make its use even more attractive. Quite sure it will be on request data mining from the database with subsequent sending. The data flux will of course occur in the opposite direction as well concerning the experiment itself (physical theory, hardware design ...). RLMS server will also be ready for a possible extensions and communication with other RLMS using access to our experiments and vice versa.

V. CONCLUSION

RL as acute trend in education and science brought the need for further improvement in terms of integration of RL into large units by RLMS, which provide access to many experiments, provide their continuous availability and other services for teamwork and cooperation, while working with experiments. At the global level there are already several RLMS existing, but they are currently unavailable for general use. This was the reason to attempt at FAI UTB in creating and testing REMLABNET. Such an extensive project consists of many layers, which must seamless communicate, ensuring secure data transmission. Outside communication security of RLMS is to be achieved by using the TCP / IP protocol, which declares 100% transmission of transmitted data packets. Currently proposed inner RLMS communication system contains different types of communication, depending on the layers in where the communication takes place. The paper is focussing on the data communication between the experiment and the RLMS system, which ensures the transmission of the measured data and preserves them for later use. By the Java programming language the solution (service) was developed, which ensures the transmission of these data and their subsequent storage in a database. The service consists of a server-side and an experiment-side among them the transmission of data files takes place. Service was designed and created according to its functionality and adjustability, on the both sides of the communication channel.

The diagnostics communication interface was also created providing information about the current status of the individual experiment allowing the management of any number of REs. It consists of three levels. First level is based on a "keepalives" feature of TCP protocol. When the remote experiment is turn on socket communication channel is created with RLMS server. There is very low bit rate which do not affect other connections. When the malfunction occurs and this connection is lost it is indication for RLMS that there is something wrong. Second level is based on a broadcast from server and reporting of status. Sometimes the server send request for the status report. Each of experiment has to reply. Also when the experiment state is changed the new state is sent to server automatically. This keeps actual information about the experiment state on server. Third level provides a deep data diagnostic. It monitors values from measuring modules and compare them with reference values. A malfunction of sensors can be detected by this. This level can also react on the fault by restarting of required software or whole computer which can make the experiment operable again. Information from each level is collected in the RLMS and the experiment administrator is informed based on them.

All designed and created solutions will be implemented into a large robust system that will ensure transmission of all data in system and will be able to communicate with any number of experiments without performance loss. This compatibility is ensured by the selected programming language (Java) and Apache Tomcat application platform for building back-end part of the system.

REFERENCES

[1] A. A. Humos, B. Alhalabi, M. Hamzal, E. Shufro, and W. Awada, "Remote labs environments (RLE): A constructivist online experimentation in science, engineering, and information technology". *31st Annual Conference of IEEE Industrial Electronics Society, 2005. IECON 2005*. IEEE, 2005, vol. 2, issue 4, pp. 6. http://dx.doi.org/10.1109/IECON.2005.1569238

[2] M.E. Auer and C. Gravier,"The many facets of remote laboratories in online engineering education". *IEEE Transactions on Learning Technologies*. 2009, vol. 2, issue 4, p. 260-262. http://dx.doi.org/10.1109/TLT.2009.53

[3] M. Krbecek, "Creation of multimedia interactive teaching tool with utilisation of remote experiments. Zlín", 2011. 108 p. Diploma thesis. University of Tomas Bat'a in Zlín.

[4] M. Cooper, "Remote laboratories in teaching and learning – issues impinging on widespread adoption in science and engineering education". *International Journal of Online Engineering* 2005*, vol.*1 issue 1, ISSN 1861-2121.

[5] M. Krbeček, F. Schauer and R. Jašek. "Security aspects of remote e-laboratories". *International Journal of Online Engineering (iJOE)*. 2013-07-11, vol. 9, issue 3, p. 34-39. Available from: http://online-journals.org/i-joe/article/view/2586. http://dx.doi.org/10.3991/ijoe.v9i3.2586

[6] Library of Labs. LiLa - Library of Labs [online]. [cited 2014-04-19]. Accessible from: http://www.lila-project.org/home.html.

[7] M. Krbeček, F. Schauer, and I. Zelinka. "Possible utilization of the artificial intelligence elements in the creation of remote experiments". *International Journal of Online Engineering (iJOE)*. 2014, vol. 10, issue 1, p. 46-52. http://dx.doi.org/10.3991/ijoe.v10i1.3110

[8] F. Schauer, I. Kuřitka, and F. Lustig, "Creative laboratory experiments for basic physics using computer data collection and evaluation exemplified on the intelligent school experimental system (ISES)". *World Innovations in Engineering Education and Research iNEER*, USA, Special Volume, 2006, p. 305-312.

[9] V. J. Harward, J. A. del Alamo, S. R. Lerman, P. H. Bailey, J. Carpenter, K. DeLong, C. Felknor, J. Hardison, B. Harrison, I. Jabbour, P. D. Long, M. Tingting, L. Naamani, J. Northridge, M. Schulz, D. Talavera, C. Varadharajan, W. Shaomin, K. Yehia, R. Zbib, and D. Zych, "The iLab Shared Architecture: A web services infrastructure to build communities of internet accessible laboratories". *Proceedings of the IEEE*. IEEE, 2008, vol. 96, issue 6, p. 931-950. http://dx.doi.org/10.1109/JPROC.2008.921607

[10] M. Tawfik, D. Lowe, S. Murray, M. de la Villefromoy, M. Diponio, E. Sancristobal, M. José Albert, G. Díaz, and M.Castro, "Grid remote laboratory management system - Sahara reaches Europe". *10th REV International Conference on Remote Engineering and Virtual Instrumentation*, Sydney, 2013. ISBN 978-1-4673-6345-7.

[11] F. Schauer, M. Krbeček, P. Beňo, M. Gerža, L. Pálka, and P. Špilakova. "REMLABNET - open remote laboratory management system for e-experiments". *2014 11th International Conference on Remote Engineering and Virtual Instrumentation (REV)*. IEEE, 2014, vol. 2, issue 4, p. 268-273. http://dx.doi.org/10.1109/REV.2014.6784273

[12] IETF RFC 793. *Transmission Control Protocol RFC 793*. California: Information Sciences Institute University of Southern California, September 1981. Available from: http://www.ietf.org/rfc/rfc793.txt

[13] Wikipedia contributors. OSI model [online]. Wikipedia, The Free Encyclopedia; 2014 Jul 18, 18:46 UTC [cited 2014-04-18]. Available from: https://en.wikipedia.org/wiki/OSI_model

[14] The Apache Software Foundation. Apache [online]. 2014 [cited 2014-04-18]. Accessible from: http://www.apache.org/

AUTHORS

M. Krbeček and **F. Schauer** are with the Tomas Bata University in Zlín, Faculty of Applied Informatics, Nad Stráněmi 4511, Zlín, CZ- 760 05, Czech Republic (e-mail: krbecek@fai.utb.cz, fschauer@fai.utb.cz).

This work was supported by the Internal Grant Agency (IGA) of Tomas Bata University in Zlín. Submitted, 19 August 2015. Published as resubmitted by the authors 20 August 2015.