DOI: 10.5604/01.3001.0010.5215

WEB SERVER LATENCY REDUCTION STUDY

Fatma Mbarek, Volodymyr Mosorov, Rafał Wojciechowski
Lodz University of Technology, Institute of Applied Computer Science

Abstract. *This paper investigates the characteristics of web server response delay in order to understand and analyze the optimisation techniques of reducing latency. The analysis of the latency behavior for multi-process Apache HTTP server with different thread count and various workloads, was made. It was indicated, that the insufficient number of threads used by the server handling the concurrent requests of clients, is responsible for increasing latency under various loads. The problem can be solved by using a modified web server configuration allowing to reduce the response time.*

Keywords: web server, latency, thread

BADANIA REDUKCJI OPÓŹNIEŃ SERWERA WWW

Streszczenie. *W artykule opisano badania charakterystyk czasowych serwera WWW w celu zrozumienia i analizy technik optymalizacyjnych powodujących redukcję opóźnienia. Dokonano analizy czasów opóźnień dla wieloprocesowego serwera Apache dla różnej liczby wątków i obciążeń. Wskazano, że niewystarczająca liczba wątków wykorzystywanych przez serwer, obsługujących jednoczesne żądania klientów, wpływa znacząco na zwiększenie opóźnień dla różnych obciążeń. Problem może być rozwiązany za pomocą modyfikacji ustawień serwera WWW, pozwalających na skrócenie czasu reakcji.*

Słowa kluczowe: web server, opóźnienie, wątek

Introduction

The World-Wide Web (WWW or Web) is an information space that is used by many people. Variety of information can be accessed quickly and easily from different remote locations. With the explosive growth of the World-Wilde Web, in both of clients' numbers and the volume of information, a heavy workload is placed on servers [2]. As a result, users observe long retrieval times for web pages and they complain about web latency (or response time).

Latency is the time that it takes to set up a connection between two endpoints and transmit a request to the server for providing services. It comes from various sources such as client or server slowness, as well as network bottlenecks. When the web servers are overloaded or have insufficient resources, they can take long time to handle a request. Furthermore, the web retrieval delay causes can be resolved by using faster computers, modifying request, handling algorithms or providing cache mechanisms [16].

Considerable efforts of previous researches conclude that web servers spend more time in kernel. Hu *et al.* [13] studied the behavior of popular Apache web server performance. They found that Apache reported about 30-50% of execution time on kernel system and 20-25% of total CPU time on user code. Almeida *et al.* [2] found that up to 90% of time is spent in the kernel for handling HTTP requests in the case of saturated web server. In addition, the work of Boyed-Wickizer *et al.* [8] studied Linux scalability and it reported about 60% of execution time of Apache process in the kernel. Based on these above results, we are interested to understand causes of network latencies and we focus on the research of finding a proposition that can improve server performance. Basic goal of this study is to provide capability of multi-processing WWW server to handle a large amount of concurrent connections in Linux [1].

To understand network server latency, we have simulated Apache Web Server v2.4.10 depending on its Multi-Process architecture which uses multiple processes with multiple threads in each one to treat incoming HTTP requests [3]. We have examined server response time in term of various data sizes and numbers of threads. This means, we must avoid network latency through augmentation of the number of worker threads in each server process. As a result, server performance is improved whatever the resources size and our measured results confirm the significant reduction of response time. The rest of this paper is organized as follows. Section 1 gives a review of previous work. Section 2 describes web components. Section 3 explains latency measurement methodology for a web page. Section 4 explains the role of threads in Apache Server architecture. In section 5, we evaluate the experimental setup and its results. Finally, section 6 provides concluding remarks and future works.

1. Related work

Several researches have enhanced many works for optimizing network servers, particularly in regard to communication protocols handling such as HTTP and TCP protocols. Faber *et al.* [12] discussed the overloading of busy web servers and proposed a modification to HTTP and TCP that shifts the TIME-WAIT state to clients. In [16], simple modifications to the HTTP protocol were proposed which consist in eliminating unnecessary network round-trip time (RTT) in order to improve web server latency. Chandranmenon *et al.* [9] proposed a paradigm to reduce round trip time (RTT) using reference points caching of documents. Other research proposed by Dodge *et al.* [11] consists caching technique in conjunction with prefetching to decrease user perceived response time.

These considerable studies have been interested in improving server performance. However, replication and caching techniques may make a busy web server because a big number of requests still charge the original web server. Furthermore, dynamic web pages cannot be cached, they must fetched from original servers [8].

In addition, Nahum *et al.* [15], Aron *et al.* [6] have proposed implementation optimizations for web servers in regards to reduce system overhead. While Ruan *et al.* [17] found that the origin of network server latency has come from the negative interactions between the server application and the locking and blocking in the operating system. They proposed web server optimization in regards to request scheduling.

Our approach focuses on studying and avoiding the latencies from server side. Thus, with modifying server configuration, the number of worker threads is increased which allows to handle more and more requests.

2. Web components

As shown in Fig. 1a, the main web components are clients, network communication and server. Formally, a client is a requester of services that initiates the network communication, while server is a provider of services and which passively waits for contact.

A typical web access identifies the requested HTML document by *Uniform Resource Locator (URL).* A given URL contains a host name and a file name on that machine [9]. An URL indicates the HTTP protocol that allows for exchange of hypertext information using GET method. HTTP is a native client-server protocol for the web [20], which functions as a request-response protocol. For accessing to the web, a client browser establishes a TCP connection to the server using its IP address and exchanges SYN packets of TCP's three-way handshake procedure [16].

According to Fig. 1b, a web server follows six steps of processing client request [10]. Its first step is *accepting the client connection*. Then, the client submits a HTTP request message to the server. The web server *reads the incoming request* of client and checks its file system in order to find the requested file. When it *finds the file*, it *sends a response header* to the client. Next step of processing request is *reading the file* from file system or memory cache. In the last step, the server replies to the client by *sending data* as response message. Furthermore, the server may repeat the read file and send data steps for larger files until it is has transmitted all of the requested document. This operation is shown in Fig. 1b by the self-loop.

To display a web page, a client browser needs to launch many HTTP transactions to fetch different web components (images, HTML sources, and links) of the page [7].

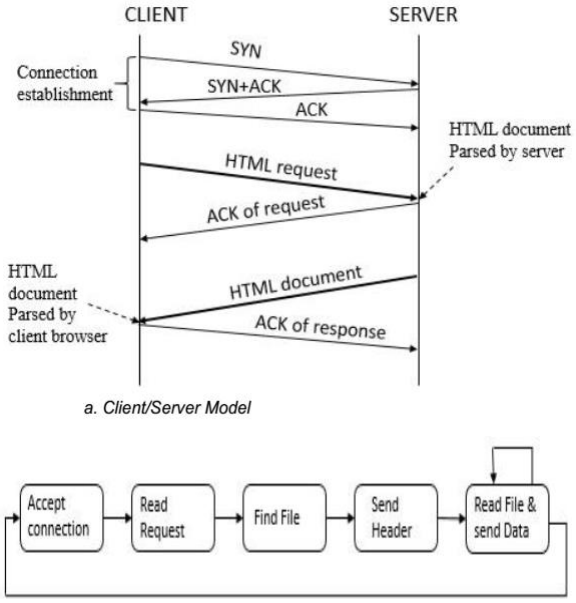


Fig. 1. Typical web access

3. Methodology for measurement of latency

Latency or response time is the amount of time required by a packet to traverse the transmission endpoints. The response time is measured by calculating the difference between the time of sending the request by the client and the time of receiving the last byte of server response. We define L to be latency and T_r, T_s to be respectively receiving time and sending time. The general formula of web latency could be defined as:

$$L = T_r - T_s \quad (1)$$

Web response time comes from several sources such as network bottlenecks, big payloads, insufficient bandwidth and weak client (low or busy CPU).

It depends on six following parameters [18]:

- *Page Size (resource size):* is measured in Kbytes or Mbytes. Its impact is obvious and is illustrated in our results.
- *Minimum Bandwidth:* is defined as bit-rate of consumed information capacity between two end points.
- *Round-Trip Time (RTT):* is the time required for a packet to travel from source to destination and back again. In the context of a web page, the source is user's browser and the destination is web server.
- *Turns:* a web page contains an additional objects such as several graphics or applets which are not transmitted with the base HTML page. These objects need an additional connection between the web server and the user. So, turns are considered as the fair number of communication cycle between two endpoints for the web page objects.
- *Server processing time:* the processing time required by the server itself. That means the time required in the kernel to handle incoming requests. This time can vary for different types of web pages, for example creating dynamic web pages needs more server effort, computing time and introduces delay while pages with static content need negligible processing time.
- *Client processing time:* it is insignificant time. For example, if the requested page contains a Java applet, the client's browser can take several seconds to load and run the Java interpreter.

Considering the above latency parameters, the total response time of web page can be defined as:

$$Latency = \frac{Page\ size}{\text{Bandwidth}} + (Turns \times Round\ Trip\ Time) + Server\ Processing\ Time + Client\ Processing\ Time \quad (2)$$

To simplify the equation, we define L to be the total latency of web page, RTT to be round trip time, T to the number of turns, P to be page size, B to be bandwidth, Cc to be the client processing time and Cs to be server processing time.

The implicit formula of web page response time is:

$$L = \frac{P}{B} + (T \times RTT) + Cs + Cc \quad (3)$$

4. Threads

As part of the validation stage of web latency evaluation study, we needed to understand the implications of web server configurations. For this study, we used the most common web server on the Internet - Apache. Apache web server is a free open source code that has been ported over to many platforms such as Linux and allows anyone to make modification to the server [3, 5].

With its multi-process architecture, Apache may several processes working simultaneously and in each process may be made up of multiple threads [1].

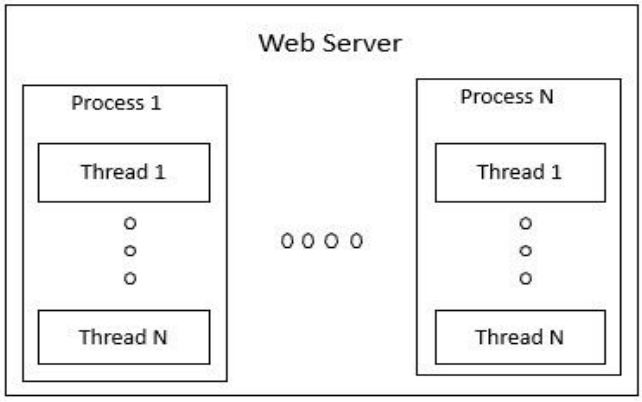


Fig. 2. Apache Web Server Architecture (Multi-Process Model)

In figure 2, a multi-process paradigm is based on two independent concepts: a process and a thread. Process is used to group related resources together. These resources include child processes, signal handlers, open files and much more of other information. Putting resources in the form of a process may ease their management.

Thread is the entity scheduled for execution on the CPU. The threads allow various executions to take pace in the same process. That means having multiple threads running in parallel in one process [19].

Table 1. Process and Thread properties

Items per process	Items per thread
Address space	Registers
Global variables	Program counter
Open files	Stack
Child processes	State
Pending alarms	
Accounting information	
Signals and signal handlers	

In table 1, we can see the properties for each process and thread. A thread has its registers that hold the current running variables. It has a program counter which gives information about next executing instruction while its stack allows to store the execution history.

Although these properties are private for each thread, the process properties are shared among the existing threads in one process. For example, if one thread opens a file, the other threads that belong to the same process can see, read and write this file. Furthermore, the decomposition of application into several threads working in parallel makes the programming model simple and optimizes system performance.

There are many reasons for having threads. First, the time needed for creating a new thread is less than for a process because the new thread shares the same address spare with other threads. Second, the time of terminating a thread is less than of a process. Third, the communication between threads of one process is simple and which causes less communication overheads [14].

Apache HTTP server is based on threads that have direct impact on server performance. In our study, we evaluated the number of threads and its impact on web latency. So, Apache server configuration module allows us to alter on server's behavior using the following factors presented in table 2 [4, 5].

Table 2. Apache server configuration factors

Factor	Description
StartServers	Initial number of server processes to start
MinSpareThreads	Minimum idle server processes
MaxSpareThreads	Maximum idle server processes
ThreadLimit	The upper limit of the configurable number of threads per child process
ThreadsPerChild	Number of worker threads per server process
MaxRequestWorkers	Maximum simultaneous requests in service

5. Experimental evaluation

In this section, we describe our Local Area Network (LAN) environment testbed including the hardware and software used. Then we present our results with discussion.

5.1. Testbed description

Our experiments are carried out on 71 virtual machines using Oracle VM VirtualBox. One virtual machine is acting as the server, and 70 others as clients connected to the server via 100 Mbits/s Ethernet switch. Each machine has a single 2.2 GHz Intel processor with 1GB of RAM and running under Ubuntu v15.04. The client machines generate workload as HTTP requests by executing a bash script code. We use Apache HTTP server v2.4.10 listening on port 80 and that uses a separate process to handle the incoming clients' requests.

The goal of our study is to improve web server performance through reducing web latency and to understand the implications of different server's configuration on the latency profiles. We examine the performance characteristics of web server under varying number of clients various workloads, different number of threads used by the Apache server and. Our tests are focused on two key metrics: Load size and Number of threads. Loads are categorized into small (11 KB), medium (28 MB) and big (389 MB) sizes of resources. Our purpose of choosing load size is to simulate the connection behavior and to trace a tractable analysis.

5.2. Experimental results

In this subsection, we present our results. These results show the impact of such factors as load size and the used number of threads on latency characteristics.

A. Latency vs Load

Varying size of workload can measure the capacity of our server and studies how latency profiles change under load. Figure 3 shows that server latency increases when the size of resources increases. Conserving the same server configuration, Apache server spends more time submitting a large resource to clients. The causes of this latency depend on repetition of reading file and sending data steps until receiving the last byte of requested file by clients.

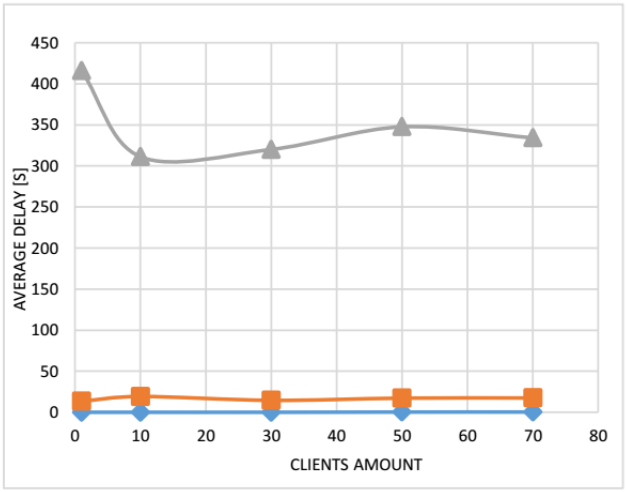


Fig. 3. Apache Server latency for handling requests for different size of resources (♦ – 11 KB, ■ – 28 MB, ▲ – 389 MB)

B. Latency vs Threads

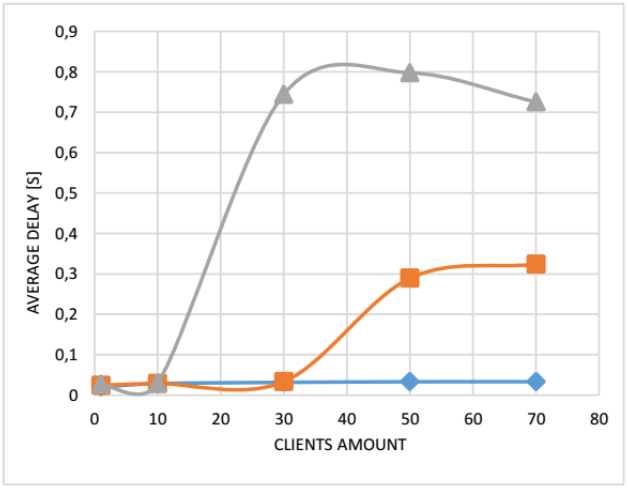
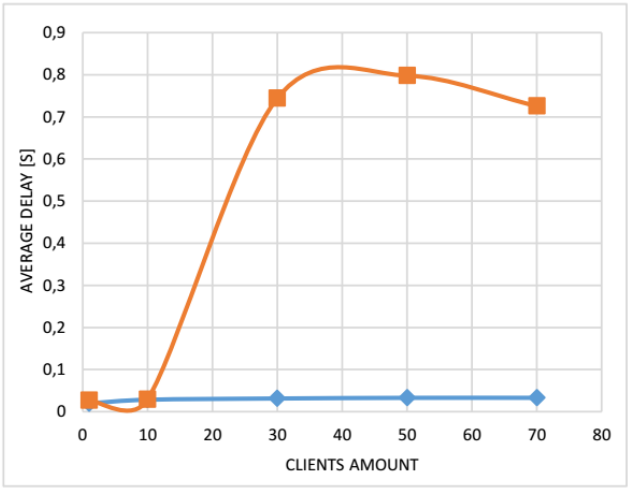


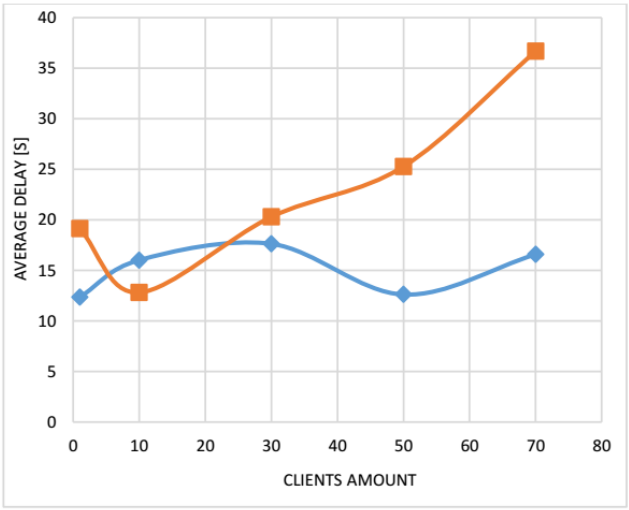
Fig. 4. Apache Server latency for handling requests for different number of threads (▲ – 30 threads, ■ – 50 threads, ♦ – 150 threads)

To understand the implications of Apache server's configurations on the latency profiles, we used 30, 50 and 150 threads in each small file size (11 KB) – test as shown in Figure 4. The experiment demonstrates that the web latency decreased when the number of threads increased.

C. Latency vs Load vs Threads



(a) Apache Server latency for handling requests for small size resource – standard text file, size 11 KB (■ – 30 threads, ♦ - 150 threads)



(b) Apache Server latency for handling requests for meduim size resource – binary file, size 28 MB (■ - 30 threads, ♦ - 150 threads).

(c) Apache Server latency for handling requests for big size resource – binary file, size 389 MB (■ - 30 threads, ♦ - 150 threads).

Fig. 5. Response time of workloads sizes vs. threads' number

Figure 5 displays the average of observed latencies for an Apache web server with 30 and 150 threads according to workloads category. Fig. 5a, b present the server response time in case of handling requests for small and medium size resources. The average delay is increasing with the growth of client count. The web server latency depends on the number of threads used. Increasing the number of threads that are used by the multi-process server can improve the overall performance and decrease latency. However it is limited by requests load / client amount. In Fig. 5c, the average delay for handling requests for big size resource, is shown. As expected, the delay is related to network bandwidth / capacity. The improvement requires, in that case, a change of the network connection.

5.3. Discussion

We summarize our observations study as follows:

- Web latency depends on two key metrics: load size and number of worker threads in the kernel.
- The optimization of Apache features is possible.
- Apache HTTP server is efficient at creating additional processes if needed.
- Misconfiguration of a server may have an impact on its performance.
- Web server latency is reduced when we use a large number of threads.

For different number of threads assigning configurations, latency profiles are changed. A server can handle many concurrent connections in the same time by increasing the amount of its worker threads. That provides the availability of services. In our experiments, we used small amount of threads because our LAN network is smaller. However, in the case of wide area network (WAN), the server needs numerous threads to process many thousands of requests.

6. Conclusion

To improve web servers' performance, many techniques were explored in both web applications and servers' kernels. This paper explores a performance study of web server at LAN network. In fact, we focused to increase network server availability through reducing web latency on server kernel. We experimentally studied Apache web server behavior under several loads and with different server configurations and we observed their impact on the server response time. To optimize web latency, the modification of web server configuration in process / threads handling module, was made. For the same conditions, the improvement of the server performance and the change of the latency profiles were obtained.

This research is made to evaluate performance of one web server. In the future, we are going to analyze the performance of web servers' cluster and to understand their behavior issues. We plan also to analyze load balancing system which is responsible for dispatching requests to servers following certain load balancing algorithms.

Acknowledgement

Work partially funded by the European Commission under the Erasmus Mundus E-GOV-TN project (Open Government data in Tunisia for service innovation and transparency) – EMA2; Grant Agreement nº 2013-2434/001-001.

References

[1] Aaqib S.M., Sharma L.: Analysis of Delivery of Web Contents for Kernel-mode and User-mode Web Servers. International Journal of Computer Applications 12(9), 2011, 37–42
[2] Almeida J.M., Almeida V., Yates D.J.: Measuring the behavior of a World Wide Web server. High Performance Networking VII. IFIP — The International Federation for Information Processing. Springer, Boston 1997.
[3] Apache Software Foundation 2016. Apache HTTP server version 2.4, https://httpd.apache.org/docs/2.4/mpm.html
[4] Apache Software Foundation 2016. Apache MPM Event. Available from https://httpd.apache.org/docs/2.4/mod/event.html
[5] Arlitt M., Williamson C.: Understanding Web server configuration issues. Software: Practice and Experience 34(2) 2004, 163–186, [DOI: 10.1002/spe.575].

[6] Aron M., Druschel P.: TCP implementation enhancements for improving Web server performance. Technical Report TR99-335, Rice University, July 1999.
[7] Banga G., Druschel P.: Measuring the capacity of a Web server under realistic loads. Baltzer Science Publishers BV, 1999, 69–83.
[8] Boyed-Wickizer S., Clements A.T., Mao Y., Pesterev A., Frans-Kaashoek M., Morris R., Zeldovich N.: An Analysis of Linux Scalability to Many Cores. OSDI'10 Proceedings of the 9th USENIX conference on Operating systems design and implementation, 2010, 1–16.
[9] Chandranmenon G.P., Varghese G.: Reducing Web Latency Using Reference Point Caching. Proceeding of IEEE INFOCOM 2001.
[10] Choi G.S., Kim J., Ersoz D., Das C.R.: A Multi-threaded Pipelined Web Server Architecture for SMP/SoC Machines. International World Wide Web Conference Committee (IW3C2), Chiba, Japan, 2005, 730–739.
[11] Dodge R., Menascé D.A.: Prefetching inlines to improve web server latency. Proc. of the Computer Measurement Group Conference, Anaheim, 1998.
[12] Faber T., Touch J., Jue W.: The TIME-WAIT state in TCP and its Effect on Busy Servers. Proceedings of IEEE INFOCOM, 1999.
[13] Hu Y., Nanda A., Yang Q.: Measurement, analysis, and performance improvement of the Apache Web server. The 18th IEEE International Performance, Computing, and Communications Conference (IPCCC'99), Phoenix/Scottsdale, Arizona, 1999.
[14] Marshall D.: Threads: Basic Theory and Libraries. 5/1999. Available: https://www.cs.cf.ac.uk/Dave/C/node29.html
[15] Nahum E., Barzilai T., Kandlur D.: Performance Issues in WWW Servers. IEEE/ACM Transactions on Networking Conference, 2/2002.
[16] Padmanabhan V.N., Mogul J.C.: Improving HTTP Latency. Computer Networks and ISDN Systems, 12/1995, 25–35.
[17] Ruan Y., Pai V.S.: The Origins of Network Server Latency & the Myth of Connection Scheduling. SIGMETRICS/Performance, New York, 2004.
[18] Savoia A.: Web Page Response Time 101. The software testing and quality engineering magazine STQE, 2001, 48–53.
[19] Tanenbaum A.S.: Modern Operating Systems, 2nd Edition, 2002, 81–100. Available: http://www.cs.vu.nl/~ast/books/mos2/sample-2.pdf
[20] Viles C.L., French J.C.: Availability and Latency of World Wide Web Information Servers. The USENIX Association, Computing Systems 8(1)/1995, 61–91.

M.Sc. Fatma Mbarek
e-mail: fmbarek@iis.p.lodz.pl

She received her Bachelor degree in Computer Science in 2012 and her M.Sc. degree in Computer Systems and Network Security in 2014 from Faculty of Sciences of Gabes, Tunisia.
She is currently a Ph.D. Student at Institute of Applied Computer Science of Lodz University of Technology under Erasmus Mundus EGOV-TN Project. Her research focus on Network Security, Network and Server performance.

D.Sc. Eng. Volodymyr Mosorov
e-mail: w.mosorow@kis.p.lodz.pl

He received the M.Sc. and the Ph.D. degrees in telecommunication from the Lviv Polytechnic National University, Ukraine in 1983 and 1998, respectively. In 2009 he received the DSc. Degree (Habilitation) in Computer Science from AGH University of Science and Technology in Cracow, Poland.
He has been working in the Institute of Applied Computer Science (previously the Computer Engineering Department), at the Faculty of Electrical, Electronic, Computer and Control Engineering, TUL since 2000, currently as a Professor of Lodz University of Technology.

Ph.D. Eng. Rafał Wojciechowski
e-mail: r.wojciechowski@kis.p.lodz.pl

In 2001, he received the M.Sc. degree in Software Engineering and Networking Systems from the Lodz University of Technology. In 2008, he received the Ph.D. degree in Computer Science from the Lodz University of Technology.
He is working as an Assistant Professor at Institute of Applied Computer Science of Lodz University of Technology.

otrzymano/received:25.10.2016 *przyjęto do druku/accepted: 14.08.2017*