An Admission Control Mechanism for Web Servers using Neural Network

Lahcene AID
Department of Computer Science
University of Tiaret, Algeria
Laboratoire de Communication dans les Systèmes Informatiques
École Nationale Supérieure d'Informatique
Oued Smar, Algiers, Algeria

Malik LOUDINI
Laboratoire de Communication dans les Systèmes Informatiques
École Nationale Supérieure d'Informatique
Oued Smar, Algiers, Algeria

Walid-Khaled HIDOUCI
Laboratoire de Communication dans les Systèmes Informatiques
École Nationale Supérieure d'Informatique
Oued Smar, Algiers, Algeria

ABSTRACT
Web sites are exposed to high rates of incoming requests. During temporary traffic peaks, web servers may become overloaded and their services deteriorate drastically. In this paper, we propose a method for admission control to prevent and control overloads in web servers by utilizing neural network (NN). The control decision is based on the desired web server performance criteria: average response time, blocking probability and throughput of web server. We have designed and developed a NN model able to predict web server performance metrics based on the parameters of the Apache server, the core of the Linux system and arrival traffic. The model predictor captures the complex relationship between web server performance and its configuration. This avoids an ad-hoc web server configuration, which poses significant challenges to the server performance and quality of service (QoS).

Keywords
Web server, Admission control, QoS, Neural networks.

1. INTRODUCTION
One of the major concerns of the web server administrators is to provide a fast and reliable service that meets their clients. It must indeed be capable of providing an efficient service at a given time, as the number of requests to be served simultaneously is unpredictable. However, a heavy workload may induce server thrashing and service unavailability. To cope with this, admission control can be used, which means that some requests are allowed to be served by the web server and some are rejected. In this way a reasonable response time of admitted requests can be guaranteed.

Existing approaches to admission control apply linear control theory which does not capture the system nonlinearity [1, 2] or follow a queuing theory approach where the system can be accurately modeled but at the expense of a hard model calibration process which makes it unwieldy to use [3,4]. However, it is important to develop a simple performance model that guarantees these requirements as a complicated model usually contains parameters that are difficult to estimate.

Also, web server's parameters configuration has a direct impact on server performance, availability and quality of service (QoS) [5, 6]. We propose a neuronal model able to predict the web server performance metrics; the model takes into account the effects of the effective Apache [7] web server parameters and the core parameter of the underlying operating system on which the web server runs. Furthermore, our model captures the dynamic and the nonlinear behavior of server systems.

The rest of the paper is organized as follows: The next section gives an overview of how a web server works, together with a description of the architecture of Apache [7] which is the server used in our approach. It also discusses on Neural Networks. In section 3, we describe our web server model, the procedure of obtaining data for learning and the training of the neural network. The section 4 represents the implementation of our model for predictive admission control. The last section concludes our work.

2. PRELIMINARIES
2.1 Web Server
A web server is software responsible for accepting HTTP [8] requests from clients, who are known as web browsers, and serving them HTTP responses.

A HTTP transaction consists of three steps: TCP connection setup, HTTP layer processing and network processing. The TCP connection setup is performed through a three way handshake, where the client and the server exchange TCP SYN, TCP SYN/ACK and TCP ACK messages. Once the connection has been established, the client sends a request for an object which can be for example static Hypertext Markup Language (HTML) files, image files or various script files. The server handles the request and returns the object or the results of these queries. Finally, the TCP connection is closed by sending TCP FIN and TCP ACK messages in both directions.

2.2 Apache
Apache is the most popular web server running today, accounting for more than 59% of all web domains on the Internet [9]. Apache is an open-source server, distributed and maintained by a community of developers under the Apache Software Foundation. A main focus of the Apache server is the operational stability. For these reasons, we have chosen Apache for our modeling.

Apache has a modular structure which allows the user to include or exclude different functionalities. It is a multi-task server, which means that it can serve a lot of different requests simultaneously. How the multitasking is implemented is highly dependent on what operating system the server is running on. However, On Windows the requests are handled by threads contained in a single process. The Prefork module is often used for Linux and UNIX applications. Here, each process contains only one thread, and thus, a process handles

only one request at a time. The name Prefork indicates that Apache forks processes before they are needed.

Apache uses a single configuration file in which directives can be written to control Apache's behavior. A directive is like a command for the server. Several directives are directly related to the server's performance, the most important ones are MaxClients, ListenBacklog, and KeepAlive. The MaxClients directive sets the limit on the number of simultaneous clients that will be served. A lower value of MaxClients decreases the number of child processes allowed in the system. This will free up memory space but reduce the throughput. If the MaxClients value is set too high it may cause the server to start swapping when the server load increases. Connection requests from clients collect in a queue until an Apache process or thread becomes free to service them. The maximum length of the queue is controlled with the ListenBacklog directive. Many operating systems reduce this value to a system limit. On Linux this limit is the kernel variable somaxconn [10]. When the queue starts growing the response time for a client's request might become very long, because the average time for each request spends in the queue will increase. If this queue is limited and made very small, the server will not be able to handle a load peak without rejecting some of the client requests. The KeepAlive directive allows multiple sequential HTTP requests to be made by a client on the same connection.

2.3 Neural Network Basics

A neuron is a processing unit in a neural network (NN). It is a node that processes all fan-in from other nodes and generates an output according to a transfer function called the activation function [11].

The multilayer perceptron (MLP) is a Feedforword Neural Network (FNN) with one or more layers of units between the input and output layers. The architecture of an MLP can be represented as an acyclic graph, so that neurons in any layer are connected only to neurons in the next layer and no feedback between layers.

The training of neural networks can be viewed as a nonlinear optimization problem in which the goal is to find a set of network weights that minimize the cost function for given examples [12]. This kind of parameters (network weights) estimation is also called learning or training algorithm. The backpropagation [13] learning algorithm is the most frequently used method in training the networks.

The powerful of neural networks is due to their strong learning and generalization capability. After a neural network learns the unknown relation from given examples, it can then predict, by generalization, outputs for new samples that are not included in the learning sample set.

3. WEB SERVER MODEL

3.1 Proposed model

We model the web server using the multilayer perceptron (MLP). The MLP is having three layers as shown in Figure 1, one hidden layer between the input and output layers. The model inputs are MaxClients, queue length and arrival rate. The queue length represents at the same time the two parameters ListenBacklog and the limit system somaxconn. The output layer has 3 output nodes which are the performance metrics: throughput, average response time and blocking probability.

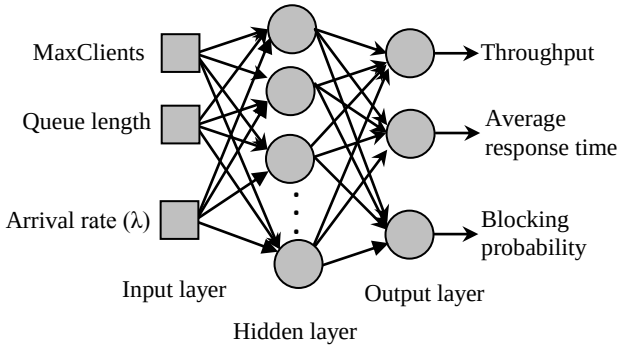

Fig 1: An MLP model of web servers.

The throughput is the maximum number of requests served in a given time period. The response time is the average time required for the system to respond to a client's request. The blocking probability is defined as the ratio between the number of requests not allowed to be served by the web server and the total number of requests received by a server in a measurement period. The number of hidden nodes is fixed at the training phase.

3.2 Training set

Our goal is to study the web server behavior at various arrival rates: low, high and even when the arrival exceeds the server's capacity.

However, to measure server performance it is necessary to run a tool on the clients' computers that generates an HTTP workload. Httperf [14] is a tool for measuring web server performance. It provides a flexible facility for generating various HTTP workloads and for measuring server performance. We have chosen Httperf tool, which generates HTTP workload according to Poisson distribution.

In order to automate the process of load testing of a web server, we chose Autobench tool which is a perl wrapper around Httperf. Autobench runs Httperf a number of times against each host, increasing the number of requested connections per second on each iteration, and extracts the significant data from the Httperf output. Distributed autobench comprises two programs, autobenchd and autobench_admin. Autobenchd is a daemon, which should be installed on each of the client machines. It listens to instructions from autobench_admin which is used for controlling the cluster of test servers.

The configuration of the laboratory is illustrated by Figure 2. One server computer and five client computers connected through a 100 Mbits/s Ethernet switch.

In order to relate the training set to reality, we have used one of the servers of the University of Tiaret [15], which was a PC Pentium dual core 1.8 GHZ with 1GB of RAM. The server used Debian Linux as operating system and Apache 2.2.9 prefork. The five clients were PCs Pentium IV 1.8 GHz with 512 MB RAM and used Ubuntu Linux 9.4 as operating system. So, we have installed autobench_admin on one client computer and autobenchd on each of the others client machines. The document file was stored in the server's local hard disk.

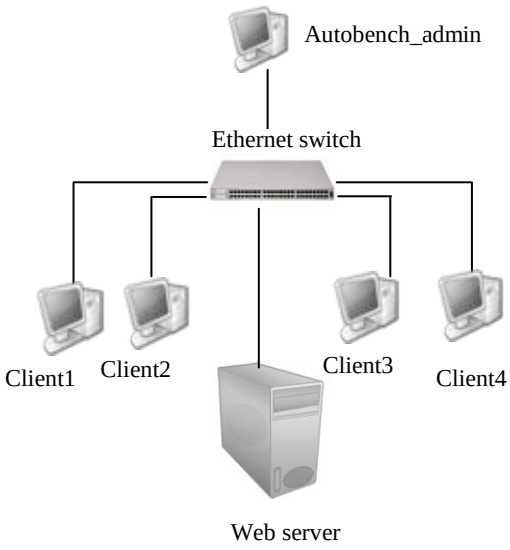

Fig 2: Configuration of the laboratory.

We examined the effects of Linux kernel parameter and Apache parameters configuration on performance metrics of the web server. Indeed, we changed the values of two parameters: MaxClients from 25 to 300 with step size 25 and the queue length from 25 at 500 with step size 50. Autobench_admin contacts the set of daemons on all 4 machines, and instructs them to generate simultaneously HTTP workload according to Poisson distribution with a rate parameter λ on the target web server. We changed λ (in number of requests/second) from 25 to 500 with step size 25. For each values of λ autobench_admin collates the results (performance metrics). This allowed us to have a training set of 4800 samples. We were interested in the following performance metrics: average response time, throughput, and blocking probability.

3.3 Training the neural model

Before training, data were normalized by subtracting the mean and dividing by the standard deviation.

An Hyperbolic tangent function was used on the hidden layer while a linear function was used on the NN output. Initial weights were randomly chosen and the NN training was programmed using MATLAB.

A proper selection on the number of hidden neurons has significant effect on the network performance. For this, we increased gradually the complexity of the model from 10 to 34 hidden neurons by adding 4 neurons each time. We found that over 34 hidden neurons, network becomes too complex and generalization ability becomes low. A cross-validation method [16] was used to estimate network generalization capability rather than rigidly partitioning the data into a separate training set (used to build the network) and validation set (used to test the network). We chose 5-fold cross-validation [16, 17].

The data were divided into five approximately equal partitions, and each partition was tested using networks built using the four remaining partitions as the training data.

The network with a 3-34-3 architecture was trained five times with different initializations of weights and biases. The solution with the lowest error was chosen.

The training was performed using backpropagation algorithm with adaptive learning rate. This one was increased if the new iteration error was smaller than the previous iteration error. On the other hand, it was decreased if the new iteration error was larger than the previous iteration error. A momentum parameter was used to accelerate learning.

The model obtained after the learning phase was tested over 20% of the data (i.e., 960 samples) that were not served in training. It has been found that the trained neural network model is able to generalize and give a meaningful output for new samples that are not included in the learning sample set. The below graphs show the testing results. The graphs were plotted between the test sample set and the corresponding outputs. It is seen from the graph that the actual output approaches to the desired output.

Figures 3, 4 and 5 show the average response time, the throughput and the blocking probability depending on the arrival rate. While Figures 6, 7 and 8 show successively the same metrics but depending on the queue length. Figures 9, 10 and 11 show the same metrics depending on MaxClients.

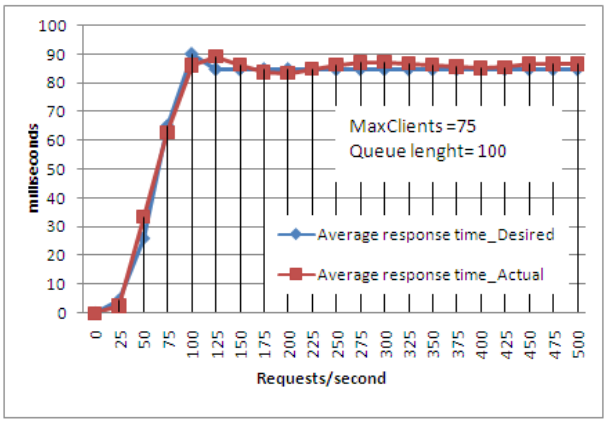

Fig 3: Average response time depending on arrival rate.

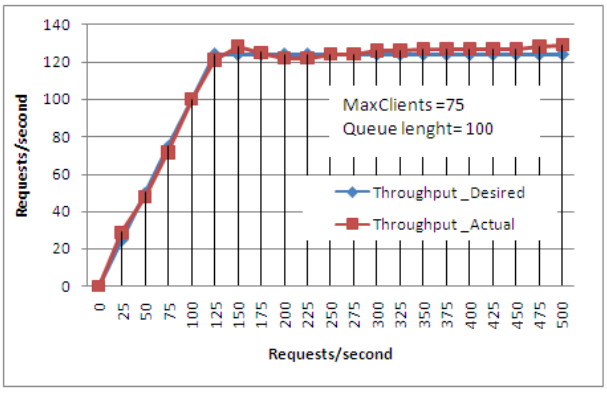

Fig 4: Throughput depending on arrival rate.

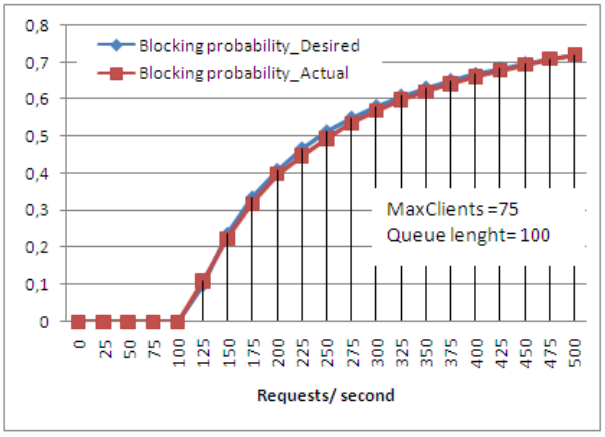

Fig 5: Blocking probability depending on arrival rate.

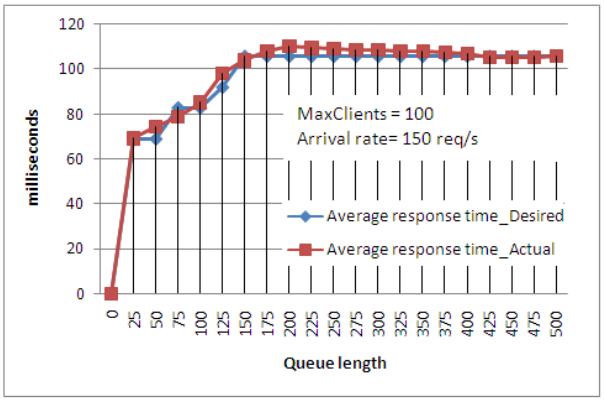

Fig. 6: Average response time depending on queue length.

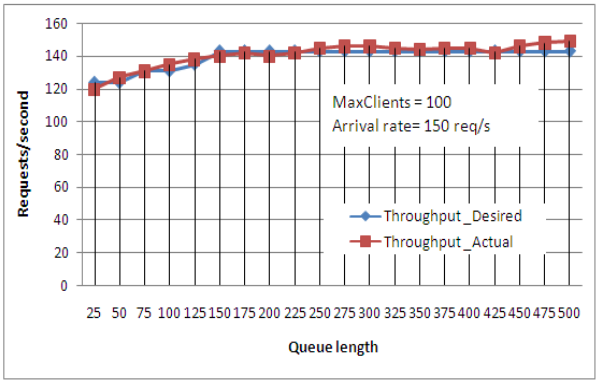

Fig 7: Throughput depending on queue length.

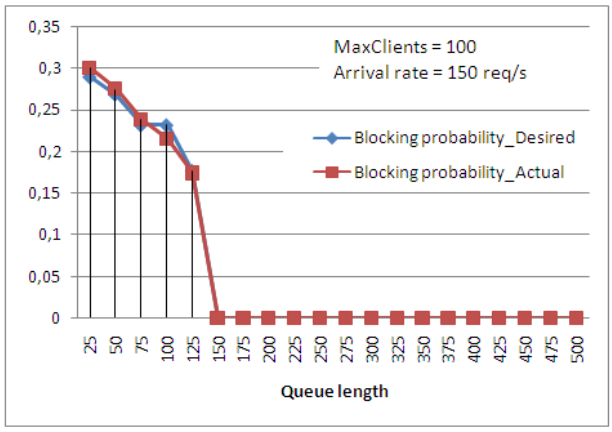

Fig 8: Blocking probability depending on queue length.

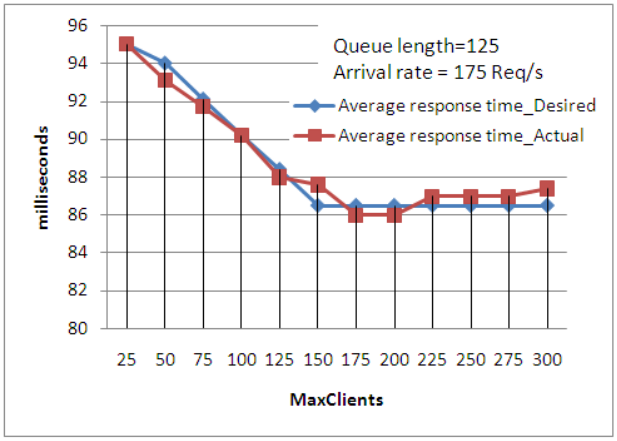

Fig 9: Average response time depending on MaxClients.

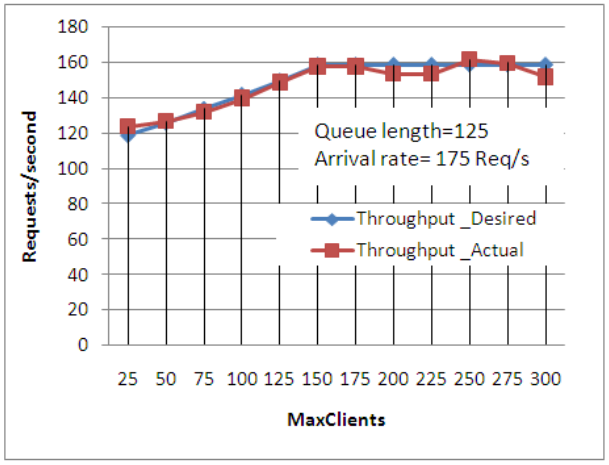

Fig 10: Throughput depending on MaxClients.

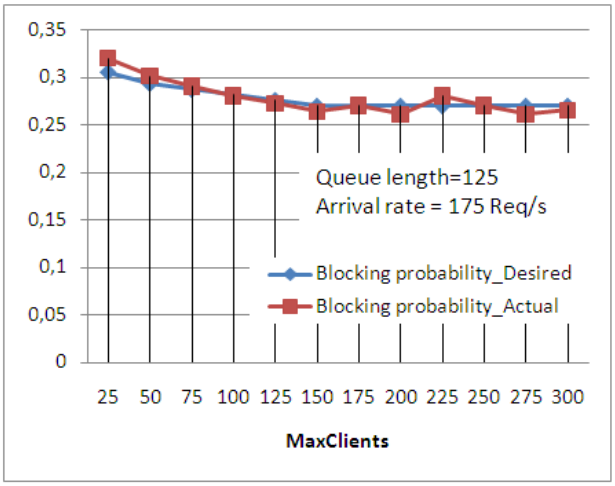

Fig 11: Blocking probability depending on MaxClients.

4. IMPLEMENTATION

The trained neural network model was converted in C++ language on Linux platform. The administrator has only to keyboard the values of the input parameters (MaxClients, queue length and arrival rate), the NN model predicts instantly the performance metrics (average response time, the throughput and the blocking probability). According to the prediction done, the administrator can change the values of the following parameters: MaxClients, ListenBacklog and the kernel variable somaxconn. It is possible to reconfigure the Apache server during runtime by doing a graceful restart from the command line. The MaxClients and ListenBacklog values are first edited in the configuration file and then the command line is issued to make the running server update its configuration. Arrival rate is limited using the Linux firewall iptables [18]. Administrators can use iptables to set the maximum limit on the number of admitted TCP connections per second.

However, when the administrator has changed the parameters, he has done an admission control based on a neural prediction.

If the administrator knows the desired performance metric values and needs to know the values of the corresponding parameters (model inputs) in order to configure the web server and limit the accepted http requests, the program changes the three values of the Neural Network inputs by iteration and the estimated outputs are compared with the desired performance metrics. Then, it retains the input values whose outputs are near or equal to the desired performance metrics.

5. CONCLUSION

In this paper, we have presented a neural network model able to predict web server performance metrics.

The adopted NN model inputs were chosen as MaxClients, queue length and arrival rate. The corresponding outputs were the performance metrics: throughput, average response time and blocking probability.

The artificial NN training phase was performed using backpropagation algorithm with adaptive learning rate.

The experimentations have been carried out on a real web server at the computer center of the University of Tiaret (Algeria), in order to evaluate our neural model prediction performances.

The obtained results show that the designed neural model ensures a predictive admission control with optimal configuration of the considered web server. Indeed, it can predict the performance metrics at both lighter loaded and overloaded regions.

We can say that we have designed an efficient artificial NN model with real abilities of performance metrics prediction. Indeed, this predictor captures the complex relationship between web server performance and its configuration. This avoids an ad-hoc web server configuration, which poses significant challenges to the server performance and quality of service (QoS).

This model could be used to perform admission control of a web server incoming requests on the basis of desired criteria. It could, also, be used for the monitoring of the web server performances.

As a future work, it would be very interesting to design a real-time tuning of the web server parameters for a more efficient neuronal prediction based admission control. It could be carried out by a feedback control strategy.

6. ACKNOWLEDGMENTS

We would like to thank Youcef Meslem for his support at the computer center of the university of Tiaret and Abdelkader Maatoug for his useful help.

7. REFERENCES

[1] Parekh S., Gandhi N., Hellerstein J., Tilbury D., Jayram T. and Bigus J. 2002. Using Control Theory to Achieve Service Level Objectives In Performance Management. Real-Time Syst., 23(1-2), 127–141.

[2] Diao Y., Gandhi N., Hellerstein J., Parekh S. and Tilbury D. 2002. Using MIMO feedback control to enforce policies for interrelated metrics with application to the Apache Web server. In Proceedings of the Network Operations and Management Symposium.

[3] Cao J. and Nyberg C. 2002. On overload control through queue length for web servers, in Proceedings of the 16th Nordic Teletraffic Seminar NTS16, Helsinki University of Technology, Espoo, Finland.

[4] Robertsson A., Wittenmark B., Kihl M. and Andersson M. 2004. Admission control for web server systems - design and experimental evaluation. In Proceedings of the 43rd IEEE Conference on Decision and Control, (Dec. 2004).

[5] Elnikety S., Nahum E., Tracey J. and Zwaenepoel W. 2004. Method for Transparent Admission Control and Request Scheduling in E-CommerceWeb Sites. In 13th international conference on World Wide Web, New York, NY, (May. 2004).

[6] Heiss H.-U. and Wagner R. 1991. Adaptive Load Control in Transaction Processing Systems. In 17th International Conference on Very Large Data Bases, San Francisco, CA, USA.

[7] Apache web server, http://www.apache.org.

[8] Stallings W. 2000. Data & Computer Communications. Prentice Hall, sixth Edition.

[9] December 2010 Web server survey by Netcraft, http://news.netcraft.com/.

[10] Linux manual page, http://linux.die.net/man/2/listen

[11] Du K. -L. and Swamy M. N. S. 2006. Neural Networks in a Softcomputing Framework, Springer-Verlag London Limited.

[12] Leondes C. T. 1998. Neural Network Systems Techniques and Applications. Vol. N°1. Academic Press, California.

[13] Rumelhart D. E., Hinton G. E. and Williams R. J. 1986. Learning internal representations by error propagation. In: Rumelhart DE, McClelland JL (eds) Parallel distributed processing: Explorations in the microstructure of cognition, 1:Foundation, 318–362. MIT Press, Cambridge, USA.

[14] HTTPerf – Mosberger D. and Jin T. 1998. A Tool for Measuring Web Server Performance. HP Research Labs. (December. 1998), volume 26 issue 3, ACM Sigmetrics Performance Evaluation Review.

[15] Tiaret university website, http://www.univ-tiaret.dz.

[16] Hastie T., Tibshirani R. and Friedman J. 2002. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer-Verlag, New York.

[17] Setiono R. 2001. Feedforward neural network construction using crossvalidation. Neural Comput. 13, 2865–2877.

[18] Purdy G. N. 2004. Linux iptables Pocket Reference, O'Reilly Media Inc, Sebastopol, California, USA.