VERSITA

Int. J. Appl. Math. Comput. Sci., 2012, Vol. 22, No. 2, 365–377
DOI: 10.2478/v10006-012-0027-4

amcs

ADAPTIVE CONTROL OF CLUSTER–BASED WEB SYSTEMS USING NEURO–FUZZY MODELS

KRZYSZTOF ZATWARNICKI

Department of Electrical, Control and Computer Engineering
Opole University of Technology, ul. Sosnkowskiego 31, 45-272 Opole, Poland
e-mail: k.zatwarnicki@gmail.com

A significant development of Web technologies requires the application of more and more complex systems and algorithms for maintaining high quality of Web services. Presently, not only simple decision-making tools but also complex adaptation algorithms using artificial intelligence techniques are applied for controlling HTTP request traffic. The paper presents a new LFNRD (Local Fuzzy-Neural Adaptive Request Distribution) algorithm for request distribution in cluster-based Web systems using neuro-fuzzy models of Web servers in the decision-making process. The neuro-fuzzy model which is applied is discussed in detail and a design of the Web switch using the proposed solution is presented. Finally, a testbed is described and the results of a comparative simulation study on the LFNRD algorithm, and other algorithms known from the literature and used in the industry, are presented and discussed.

Keywords: neuro-fuzzy model, request distribution, Web cluster, QoWS.

1. Introduction

Nowadays, the Internet is a part of the basic media which provides entertainment, advertisement and latest news. It is also a driving force for various business activities such as, for example, running Internet stores, auction systems, Internet banking systems, and many others. The Internet constitutes a specific medium providing the user with a considerable freedom in the selection of an information source and the change of this source. This means that the users whose demands are not satisfied in a given Web service may quickly and easily change the source of information and refer to another service without any financial consequences.

The owners of Web services, who are anxious to attract clients, need to be able to offer attractive content and at the same time ensure an adequate quality of service from a technical point of view. As the users will not pay attention to the technical aspect of service when it is of high quality, a low level of quality may cause them to give up the service.

The problems of increasing and guaranteeing the quality of service were interestingly surveyed by Cardellini *et al.* (2002) and Zhou *et al.* (2007). The solutions encountered include the application of a more efficient server in the service, proper scheduling of HTTP requests on the input to the Web server (AlSa'deh and Yahya, 2008; Harchol-Balter *et al.*, 2003; Zatwarnicki, 2010), scheduling and admission control in the Web server (Borzemski and Suchacka, 2010; Elnikety *et al.*, 2004; Lee *et al.*, 2004; Quan and Chung, 2005; Wei *et al.*, 2005; Wei and Xu, 2006), the application of a locally distributed cluster-based Web system (Borzemski and Zatwarnicki, 2003; Cardellini *et al.*, 2001; 2002; Cherkasova and Karlsson, 2001; Pai *et al.*, 1998), and the application of globally distributed Web server clusters (Andreolini *et al.*, 2008; Borzemski *et al.*, 2007). Forecasting data transfer times in the Internet is of considerable significance as well (Borzemski, 2006).

The application of Web server clusters is currently the most common technique for increasing the efficiency of a Web service (Gilly *et al.*, 2011). Web clusters are used in services when the clients are spread over a limited geographic area, e.g., in one state. A Web cluster consists of a Web switch, WWW servers and back-end servers, such as application and database servers (Fig. 1). The Web switch is responsible for controlling a request flow in the cluster. The switch uses a request distribution algorithm for determining a server that will service an HTTP request. A properly constructed executor, which is a part of the switch, transfers a request obtained from a client to the WWW server and a response from the server to the client (this con-

figuration is called a two-way architecture), or the request may be transferred directly from the WWW server to the client (this configuration is called a one-way architecture) (Cardellini *et al.*, 2002).

The efficiency of the whole Web service depends, to a large extent, on the type of the request distribution algorithm applied. Among the variety of request distribution algorithms, the following can be distinguished:

(i) static algorithms, whose way of determining a server for servicing a request does not change during the operation of the algorithm;

(ii) dynamic algorithms modifying their operation based on adequate service load measures;

(iii) adaptive algorithms, which learn the behavior of the service during the work to improve the quality of the decisions made.

Adaptive algorithms can make the best decisions while achieving the assumed goals. The disadvantage of using adaptive algorithms is usually a long decision-making time, which often makes the operation of a Web switch impossible in real time.

This paper presents a new computationally simple adaptive algorithm for the control of time-varying Web traffic. An adaptive parameter estimation algorithm for a neuro-fuzzy model of a Web server is followed by an optimal decision-making process, which provides a high quality of a Web service in terms of load-sharing distribution of HTTP requests.

The application of the fuzzy approach in the two-way Web switch makes it possible to use inaccurate, uncertain, and sometimes even not up-to-date information for making decisions. The application of the approach based on neural networks provides the ability of learning and adapting to the time-varying environment. The paper describes in detail the neuro-fuzzy model proposed, which is applied in the decision-making algorithm to model the operation of a Web server.

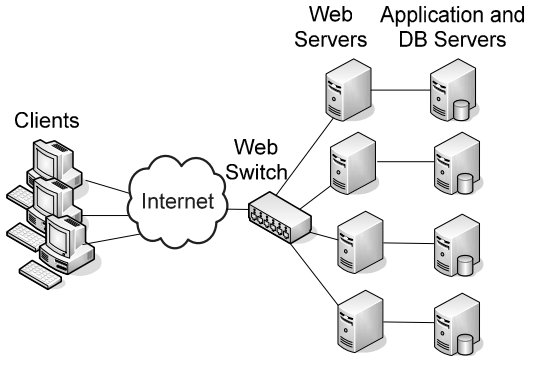

Fig. 1. Cluster-based Web system.

The paper is divided into six sections. Section 2 surveys related work referring to the artificial intelligence techniques discussed and the distribution of HTTP requests. Section 3 presents problem formulation and Section 4 contains a design of a Web switch together with a thorough description of the construction and operation of a neuro-fuzzy model for a Web server. Section 5 describes a simulation testbed used in the experiment and the research results. The final section contains concluding remarks.

2. Related work

Fuzzy and neural systems have been of great interest to scientists for a long time and they have been applied in numerous practical solutions. Fuzzy logic was introduced by Zadeh (1965) and since then has been widely used in constructing intelligent systems. Mamdani (1977) introduced fuzzy inference procedures, which resulted in numerous applications of the new solution. The main advantage of fuzzy logic is that it can cope well with the inaccurate information which may characterize physical systems. Fuzzy logic can afford abilities for building models which correspond well with a human way of understanding and perceiving reality. A complex review of the solutions used in decision-making systems can be found in the work of Zadeh (1996).

Using the solutions involving neural networks in fuzzy systems, hybrid systems can be obtained with additional learning skills. The integration of a neural network with fuzzy logic makes it possible to create a controllers characteristic of adaptation capabilities and with a simultaneous ability to make decisions in an uncertain and noisy environment (Simiński, 2010). Neuro-fuzzy systems have already been applied in dynamic load-balancing algorithms (Kwok and Cheung, 2004; Kun-Ming *et al.*, 2004), as well as in Web systems control, e.g., for scheduling the requests at the front of Web systems (Wei *et al.*, 2005; Wei and Xu, 2006), and distribution requests in local cluster based Web systems (Cherkasova and Karlsson, 2001; Riska *et al.*, 2002). The neuro-fuzzy approach was also used in the distribution of HTTP requests in globally distributed Web systems containing many local Web clusters (Borzemski *et al.*, 2007; Zatwarnicki, 2010).

In our previous works, we have already presented the FNRD (Fuzzy-Neural Adaptive Request Distribution) request distribution algorithm applied in a cluster-based Web system using neuro-fuzzy models in its construction (Borzemski and Zatwarnicki, 2003; 2006). In the numerous experiments we indicated that the request distribution algorithm was better than other reference algorithms, such as CAP (Content Aware Policy) (Casalicchio and Colajanni, 2001) and LARD (Locality Aware Request Distribution) (Pai *et al.*, 1998), and also better than the popular RR (Round-Robin) algorithm often used in industrial solutions and its variation—the WRR (Weighted Round-

Robin) algorithm (Cardellini *et al.*, 2002). The FNRD algorithm uses a neuro-fuzzy model, in which only the parameters of the defuzzification membership function are estimated in the adaptation process. In this paper, we present a distribution algorithm in which the whole fuzzy model is transformed into a neural network. Owing to such an approach, it is possible to build a system which adjusts better to a time-varying operation environment.

3. Problem formulation

A key element of the Web cluster is a Web switch. It uses a proper request distribution mechanism, makes decisions based on the request distribution algorithm and contains the executor carrying out the decisions. The design of an adequate switch constitutes a basis for creating an efficient cluster-based Web system.

The main goal of the operation of the Web switch proposed in the paper is minimizing the response time for each single HTTP request. For a request service, the switch should choose a Web server with the shortest response time.

In order to describe the conception let us introduce the following notation: x_i: HTTP request, $x_i \in X$, where X is a set of HTTP requests serviced correctly in the Web service; i: index of the HTTP request, $i = 1, \ldots, I$, where I is the time-varying number of requests serviced; O_i^s: load of the s-th Web server at the moment of i-th request arrival (the load is precisely described in Section 4); S: number of Web servers in the system; $\tilde{t}_i$: response time for the i-th request, measured from the moment of sending the request from the Web switch to the server, up to receiving the HTTP response by the switch; $\hat{t}_i^s$: estimated response time of the i-th request for s-th Web server; w_i: decision, a Web server chosen to service the i-th request.

The main task is to propose such a Web switch design which for each coming request x_i, $i = 1, \ldots, I$, will determine, based on the knowledge on the load $O_i^1, \ldots, O_i^S$ of the Web servers and the knowledge on the past response times to requests $\tilde{t}_1, \ldots, \tilde{t}_{i-1}$, the Web server w_i out of S servers for which the estimated response time $\hat{t}_i^s$, $s = \{1, \ldots, S\}$, is shortest.

It is assumed that each WWW server in a local cluster may service each of the HTTP requests accepted for service within a given Web service. The Web switch initiates the request service following the FCFS (First Come First Served) strategy, i.e., according to the order they come to its input queue. All requests are treated in the same way.

Figure 2 shows an overall diagram of the decision making process in the LFNRD system according to which a basic distribution algorithm and an adaptation algorithm can be distinguished. The decision making algorithm determines the decision and the adaptation algorithm tunes the parameters of the basic distribution algorithm.

4. LFNRD switch design

The Web system proposed in the paper, containing a locally distributed Web cluster, is called the LFNRD system, and the Web switch—the LFNRD switch.

The LFNRD switch consists of the following main components: a request analysis module, server models, a decision module, an execution module and a measurement module. Figure 3 shows the LFNRD switch diagram.

The request analysis module in the LFNRD switch analyzes the i-th request and takes the HTTP address u_i of the requested object.

The server model estimates the response time to the request x_i. The switch contains S server models, that is, the number of WWW servers in the cluster. Each server model is assigned exactly to one WWW server and it estimates the response time $\hat{t}_i^s$ of the assigned server, where $s \in \{1, \ldots, S\}$. The server model estimates the response time on the basis of information on the address u_i of the object requested and the load O_i^s of the WWW server. When the request service is done, the server model updates information on the HTTP request service time based on the measured response time $\tilde{t}_i$ to request x_i. The structure and the way of operation of the server model module are discussed in the next subsection.

The decision module selects a server to service the request x_i. Decision w_i is made according to the assumption $w_i : \hat{t}_i^{w_i} = \min\left\{\hat{t}_i^1, \ldots, \hat{t}_i^s, \ldots, \hat{t}_i^S\right\}$.

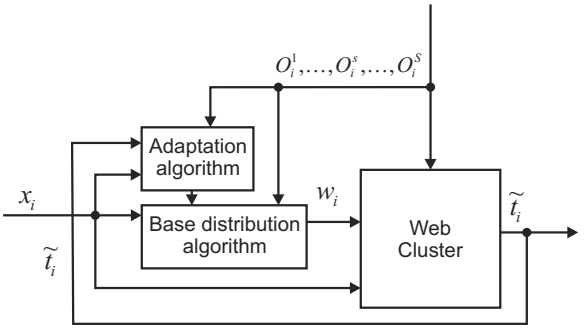

Fig. 2. Decision making process in the LFNRD system.

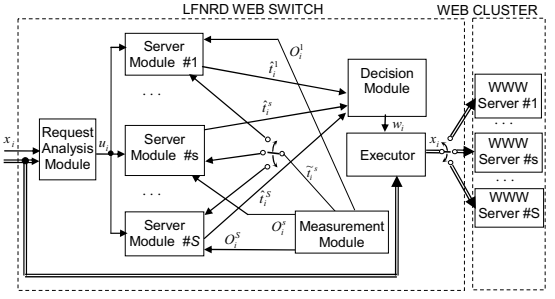

Fig. 3. LFNRD switch and the WWW server cluster.

The execution module physically transfers the request x_i to the selected server w_i. This module also supervises the process of sending a response from the server to the client. The architecture applied in the switch is the two-way architecture.

The measurement module measures an actual response time $\tilde{t}_i$ to an HTTP request. This time is transferred to the server model corresponding to the server which performed the request service. This module also measures the Web servers loads $O_i^1, \ldots, O_i^s, \ldots, O_i^S$. The term $O_i^s = [a_i^s, b_i^s]^T$ characterizes the load of the s-th WWW server and the backend server, where a_i^s is the number of all HTTP requests serviced simultaneously by the WWW server and b_i^s is the number of HTTP requests serviced simultaneously and related to dynamic objects whose content is created by the application and/or database server after receiving the requests.

4.1. Structure of the server model. The server model is a key element of the Web switch. Owing to its application, it is possible to determine the results of a decision before it is finally made. The model used should provide decision making in real time. The adopted server model consists of four functional modules: a classifying module, an estimation mechanism, an adaptation mechanism and a server parameters module. Figure 4 shows a diagram of the server model.

In what follows, we will drop the index s in the pertaining formulae, thus assuming that all calculations will refer to the s-th Web server.

The classification module automatically classifies all objects requested by a client. Response times for the objects belonging to the same class should be similar. The address u_i of the requested object is transferred to the input to the module. The classification module possesses information on the sizes and types of HTTP objects available in the service. For static objects (files on the Web server disks), requests are classified based on the size of the objects requested. For each dynamic object, the content of which is created at the moment of the request arrival, a separate class is determined. At the output of the mechanism, a class k_i of the requested object is obtained, where $k_i \in \{1, \ldots, K\}$, and K is the number of classes determined.

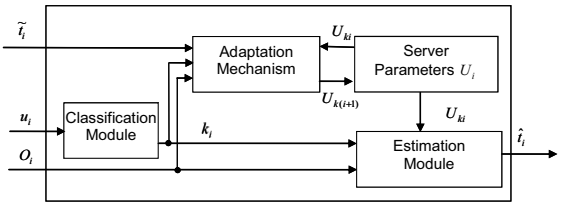

Fig. 4. WWW server model.

The server parameters module stores information U_i on parameters used by the estimation mechanism to designate the time $\hat{t}_i$. The vector $U_i = [U_{1i}, \ldots, U_{ki}, \ldots, U_{Ki}]$ contains information concerning parameter sets for the k-th class requests, where

$$U_{ki} = [A_{ki}, B_{ki}, Y_{ki}],$$

$$A_{ki} = \left[\alpha_{1ki}, \ldots, \alpha_{lki}, \ldots, \alpha_{(L-1)ki}\right],$$

$$B_{ki} = \left[\beta_{1ki}, \ldots, \beta_{mki}, \ldots, \beta_{(M-1)ki}\right],$$

$$Y_{ki} = [t_{1ki}, \ldots, t_{jki}, \ldots, t_{Jki}]$$

are the parameters of neuro-fuzzy model to be defined later on, $k \in \{1, \ldots, K\}$.

The estimation mechanism estimates the response time $\hat{t}_i$ for the i-th request. The response time is estimated based on the server load O_i and current data $U_{ki}|_{k=k_i}$ for the k_i-th class of the object requested.

The adaptation mechanism updates information $U_{ki}|_{k=k_i}$ based on the system load O_i, as well as the estimated and measured response times $\hat{t}_i$ and $\tilde{t}_i$, respectively.

The model of the HTTP request service system can work in estimation and adaptation modes. In the estimation mode the system model estimates a response time to a request. In this mode the adaptation mechanism does not participate in calculations. When operating in the adaptation mode, the Web server model adjusts to the time-varying environment, thus improving the quality of operation in the estimation mode. Adaptation is carried out by the adaptation mechanism after the client's request service is completed and the measured response time $\tilde{t}_i$ to the request is obtained.

The estimation and adaptation mechanisms form a neuro-fuzzy model of the Web system, whose the parameters for the particular classes of objects are stored in the server parameters module. Later in the paper we show how to estimate the response time and how to update the parameters of the model for the particular k_i-th class. For the clarity of pertaining formulae, we assume that $k = k_i$.

4.2. Estimation of the HTTP request response time. The operation of the Web system model in the estimation mode is equivalent to the operation of the fuzzy model, whose diagram is shown in Fig. 5. The model consists of fuzzification, rule base, inference and defuzzification blocks.

Let us assume that loads a and b are linguistic variables. The real physical domain of the linguistic variables a and b is the set $[0, \infty)$. Let us also assume that a and b are not only the denotations of linguistic variables but also of the elements from the real physical domain of variables (such an approach is often taken in practical issues (Driankov *et al.*, 1996)). The set of linguistic values of the linguistic variable a is $\{Z_{a1}, \ldots, Z_{al}, \ldots, Z_{aL}\}$, where L is the number of fuzzy sets of the linguistic variable a.

For the linguistic variable b, the set of linguistic values is $\{Z_{b1}, \ldots, Z_{bm}, \ldots, Z_{bM}\}$, where M is the number of fuzzy sets of the linguistic variable b.

In the process of fuzzification, the values of degrees of membership in the input fuzzy sets are calculated. It is assumed that the particular fuzzy sets are denoted in the same way as their linguistic values. The value of the degree of membership is contained in the range $[0, 1]$ (Zadeh, 1996).

It has been assumed that membership functions for all input fuzzy sets are triangular functions. The functions are piece-wise linear and have a limited support, and therefore the process of calculating the degrees of membership is not time-consuming. Also the shape of the function is described with a low number of parameters, which may be tuned in the adaptation process. In addition, the author's experience strengthens the understanding of the linguistic space in terms of triangular membership functions.

Figure 6 shows a graphic representation of the membership functions for the input a. The membership functions for the input b can be presented in a similar way.

Parameters $\alpha_{1ki}, \ldots, \alpha_{lki}, \ldots, \alpha_{(L-1)ki}$ determine the supports and shapes of the membership functions for the input a. The membership functions for the input b look similarly, and the parameters are denoted by $\beta_{1ki}, \ldots, \beta_{mki}, \ldots, \beta_{(M-1)ki}$. The membership functions $\mu_{Zal}(a_i)$, $l = 1, \ldots, L$, for the input a, whose current value at the moment of arrival of the request x_i is a_i, are relegated to Appendix.

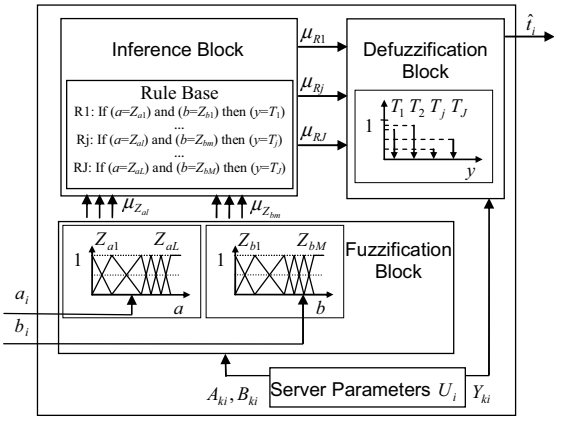

Fig. 5. Fuzzy model of the WWW server.

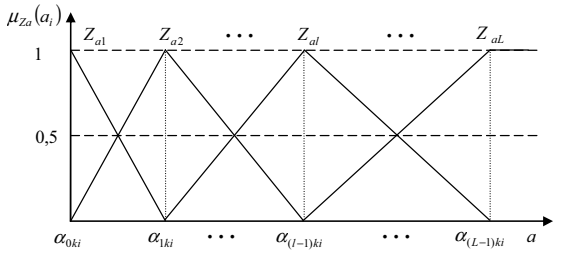

Fig. 6. Membership functions of fuzzy sets to input a.

Membership functions for the input b can be described with similar formulae, and the functions are denoted by $\mu_{Z_{bm}}(b_i)$, where $m = 1, \ldots, M$. The triangular membership functions applied meet the condition of the unit division for both inputs, hence

$$\sum_{l=1}^{L} \mu_{Z_{al}}(a_i) = 1, \quad \forall a_i \in [0, \infty),$$

$$\sum_{m=1}^{M} \mu_{Z_{bm}}(b_i) = 1, \quad \forall b_i \in \langle 0, \infty).$$

The values of the degrees of membership in the particular fuzzy sets are obtained at the output of the fuzzification block and they are transferred to the inference block.

Further on, the membership functions for the output will be discussed. A linguistic variable t will appear in the conclusion of fuzzy rules to follow. Its real domain is in the range $[0, \infty)$ and the set of its linguistic values is $\{T_1, \ldots, T_j, \ldots, T_J\}$, where J is the number of fuzzy sets for the output t. Output fuzzy sets are singletons indicating some values $t_{1ki}, \ldots, t_{jki}, \ldots, t_{Jki}$, which are equal to request service times assigned to the same k_i-th class for various loads of the HTTP request service system. The membership functions for the output can be expressed as

$$\mu_{T_j}(t_i) = \begin{cases} 1 & \text{if} \quad t_i = t_{jki}, \\ 0 & \text{if} \quad t_i \neq t_{jki}, \end{cases} \tag{1}$$

where $j = 1, \ldots, J$, and t_i is the value of variable t.

In the process of adaptation, the parameter values of the membership functions for inputs and outputs may be time-varying.

The rule base of the model discussed is linguistically and numerically complete and contains $J = LM$ rules of the following forms:

$$R1: \text{IF } (a = Z_{a1}) \text{ AND } (b = Z_{b1}) \text{ THEN } (t = T_1),$$
$$R2: \text{IF } (a = Z_{a1}) \text{ AND } (b = Z_{b2}) \text{ THEN } (t = T_2),$$
$$R3: \text{IF } (a = Z_{a1}) \text{ AND } (b = Z_{b3}) \text{ THEN } (t = T_3),$$
$$\vdots$$
$$Rj: \text{IF } (a = Z_{al}) \text{ AND } (b = Z_{bm}) \text{ THEN } (t = T_j),$$
$$\vdots$$
$$RJ: \text{IF } (a = Z_{aL}) \text{ AND } (b = Z_{bM}) \text{ THEN } (t = T_J),$$

where $l = 1, \ldots, L$, $m = 1, \ldots, M$, $j = 1, \ldots, J$, $R1, \ldots, Rj, \ldots, RJ$ denote consecutive rules.

In the inference block, the degrees of the activation levels for the particular fuzzy rules are calculated. Initially, it is necessary to calculate the degree of a rule premise for the particular rules. The degree of activation for rule Rj is calculated according to

$$\mu_{R_j}(a_i, b_i) = T\left(\mu_{Z_{aL}}(a_i), \mu_{Z_{bm}}(b_i)\right), \tag{2}$$

where a_i and b_i are load values at the moment of arrival of the i-th request, and T is the T-norm operator, adopted here as the PROD operator, i.e., the product. Therefore, (2) assumes the form

$$\mu_{Rj}(a_i, b_i) = \mu_{Z_{al}}(a_i)\,\mu_{Z_{bm}}(b_i). \tag{3}$$

Having calculated the degrees of membership of the rule premises, the degrees of membership of fuzzy rule conclusions should be computed, that is, an inference is carried out. In the model presented, the inference is based on the implication using the PROD operator, i.e., an algebraic product, which is also called the Larsen rule. The membership function of the fuzzy rule conclusion is obtained through limiting the full membership function of the rule conclusion to the level determined by the degree of the rule activation level. The formula (4) shows the inference-modified membership function of the fuzzy rule conclusion obtained:

$$\mu_{T_j^*}(t_j) = \begin{cases} \mu_{Rji} & \text{if} \quad t_i = t_{jki}, \\ 0 & \text{if} \quad t_i \neq t_{jki}, \end{cases} \tag{4}$$

where $\mu_{Rji} = \mu_{Rj}(a_i, b_i)$. In practice, a value of the function $\mu_{T_j^*}(t_i)$ is equal to $\mu_{Rj}(a_i, b_i)$. At the output of the inference block (Fig. 5), the values of the membership degrees for the premises of the particular rules μ_{Rji}, $j = 1, \ldots, J$, are obtained.

In the defuzzification block, a crisp value of the model output is calculated. There are many defuzzification methods; for the fuzzy model discussed the height method was chosen, which requires an insignificant amount of calculation and thanks to which the model is 'sensitive' to input changes. The defuzzification result, that is, an estimator of the HTTP request response time, is obtained from the formula

$$\hat{t}_i = \frac{\sum\limits_{j=1}^{J}\left(t_{jki}\mu_{T_j^* i}\right)}{\sum\limits_{j=1}^{J}\mu_{T_j^* i}}, \tag{5}$$

where $\mu_{T_j^* i} = \mu_{T_j^*}(t_{jki})$. Because $\mu_{T_j^*}(t_{jki}) = \mu_{Rji}$ and accounting for the unity division condition for the membership functions for both inputs

$$\sum_{j=1}^{J}\mu_{Rji} = 1,$$

(5) may be rewritten in the following form:

$$\hat{t}_i = \sum_{j=1}^{J} t_{jki}\mu_{Rji}. \tag{6}$$

In the process of estimation, the following data are used:

$$A_{ki} = [\alpha_{1ki}, \ldots, \alpha_{lki}, \ldots, \alpha_{Lki}],$$

$$B_{ki} = [\beta_{1ki}, \ldots, \beta_{mki}, \ldots, \beta_{Mki}],$$

$$Y_{ki} = [t_{1ki}, \ldots, t_{jki}, \ldots, t_{Jki}],$$

which are the parameters of the membership functions for inputs and outputs. The parameters are weights in the neuro-fuzzy model, which can be determined after transformation of the fuzzy model described into the neuro-fuzzy model.

4.3. Adaptation of the server model. Let us now discuss the model operation in the adaptation mode. The essence of adaptation is that, within each class of HTTP objects, the consecutive request will be renumerated with a new subindex g (corresponding to but not substituting for i). We now have the reindexed requests x_g within each class $k_g \in \{1, \ldots, K\}$, with $g = 1, \ldots, G_k$. Note that $\sum_{k=1}^{K} G_k = I$. Now, adaptation will be performed separately within each class $k_g \in \{1, \ldots, K\}$. Specifically, the incoming request will be classified to a particular class, within which adaptation will be solely proceeded. Thus we can say that we have separate neural networks for each of S servers. Possible (minor) cross-adaptation in all the remaining classes, the task being conceivable in some Web-related applications, will be a subject of the author's future research.

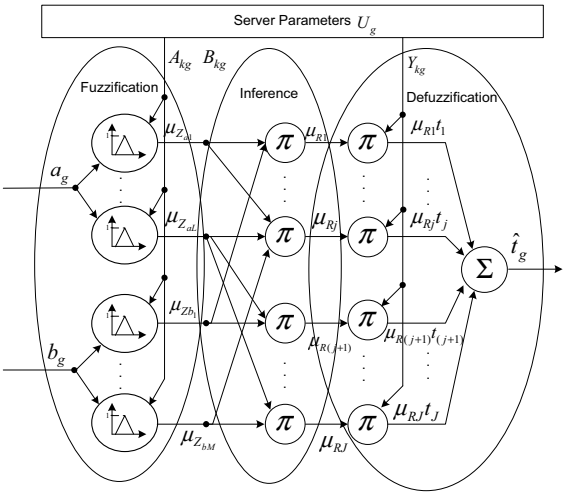

Fig. 7. Neuro-fuzzy model of a WWW server.

Figure 7 shows an overall diagram of the neuro-fuzzy model created as a result of the fuzzy model transformation.

In the neuro-fuzzy network presented, the input layer of neurons consists of two parts, each of which is responsible for fuzzification of a different input value. Each part contains the number of neurons which is equal to the number of fuzzy sets for a given input, hence the input layer contains $L+M$ neurons. The value of the degree of membership in the particular inputs of fuzzy sets is obtained at the output of each neuron of the fuzzification layer.

The degrees of membership $\mu_{T_j^*}(t_g) = \mu_{Rj}(a_g, b_g)$, $j = 1, \ldots, J$, being the inference modified function of membership in the output fuzzy sets, are calculated in the inference layer of the neuro-fuzzy network described. The number of neurons in this layer corresponds to the number of rules in the rule base and is equal to J. The structure of the fuzzification layer in the neuro-fuzzy model depends on the fuzzification method adopted. As has been mentioned before, the height method was applied in the model discussed. The defuzzification layer contains $J+1$ neurons, of which J neurons calculate the product of time t_{jkg} taken from the server parameters module for the k_g-th class and the activation level μ_{Rjg} for the j-th fuzzy rule, $j = 1, \ldots, J$. The remaining neuron calculates the sum of the products over $j = 1, \ldots, J$. This sum is the output value, that is, the estimated request response time $\hat{t}_g$.

Adaptation in the neuro-fuzzy model is carried out through the adjustment of its parameters based on information on the estimation error $e_g = \hat{t}_g - \tilde{t}_g$ (Horikowa *et al.*, 1992). The parameters of the fuzzy sets of inputs and outputs will be subjected to the process of adaptation. The backpropagation method developed by Werbos (1974) is used in the adaptation method. The adaptation algorithm updates the parameters to minimize the value of the mean square error $E_g = (e_g)^2/2$. A gradient descent rule developed by Widrow and Hoff (1960) was used in the minimization process for the mean square error, according to which parameters are tuned based on the formula $\delta_{(g+1)} = \delta_g - \eta \partial E_g / \partial \delta_g$, where $\delta_{(g+1)}$ is the updated value of the parameter δ_g, η is the learning rate, and $\partial E_g / \partial \delta_g$ is the partial error.

Partial errors for the parameters of the fuzzy sets for the output are calculated according to

$$\begin{aligned}\frac{\partial E_g}{\partial t_{\nu kg}} &= \left(\hat{t}_g - \tilde{t}_g\right) \frac{\partial \hat{t}_g}{\partial t_{\nu kg}} \\ &= \left(\hat{t}_g - \tilde{t}_g\right) \frac{\partial}{\partial t_{\nu kg}} \sum_{j=1}^{J} \mu_{Rjg} t_{jkg} \\ &= \left(\hat{t}_g - \tilde{t}_g\right) \mu_{R\nu g},\end{aligned} \tag{7}$$

where $\nu \in \{1, \ldots, J\}$. New parameter values of the fuzzy sets of the output are calculated according to

$$t_{\nu k(g+1)} = t_{\nu kg} + \eta_y \mu_{R\nu g} \left(\tilde{t}_g - \hat{t}_g\right), \tag{8}$$

where η_y is the learning rate for fuzzy output parameters.

To calculate the partial errors for the parameters of the input fuzzy sets (6), is converted to the following form:

$$\begin{aligned}&\hat{t}(a_g, b_g) \\ &\quad = \sum_{m=1}^{M} \sum_{l=1}^{L} \mu_{Z_{al}}(a_g)\, \mu_{Z_{bm}}(b_g)\, t_{((m-1)L+1)kg},\end{aligned} \tag{9}$$

where L is the number of fuzzy sets for the input a and M is the number of fuzzy sets for the input b. The error introduced to the neuro-fuzzy network by the parameters of the fuzzy sets of input a can be expressed with

$$\begin{aligned}&\frac{\partial E_g}{\partial \alpha_{\phi kg}} \\ &= \left(\hat{t}_g - \tilde{t}_g\right) \frac{\partial \hat{t}_g}{\partial \alpha_{\phi kg}} \\ &= \left(\hat{t}_g - \tilde{t}_g\right) \\ &\quad \times \frac{\partial}{\partial \alpha_{\phi kg}} \sum_{m=1}^{M} \sum_{l=1}^{L} \mu_{Z_{al}}(a_g)\, \mu_{Z_{bm}}(b_g)\, t_{((m-1)L+l)kg} \\ &= \left(\hat{t}_g - \tilde{t}_g\right) \\ &\quad \times \sum_{m=1}^{M} \left(\mu_{Z_{bm}}(b_g) \frac{\partial}{\partial \alpha_{\phi kg}} \sum_{l=1}^{L} \mu_{Z_{al}}(a_g)\, t_{((m-1)L+l)kg} \right) \\ &= \left(\hat{t}_g - \tilde{t}_g\right) \\ &\quad \times \sum_{m=1}^{M} \left(\mu_{Z_{bm}}(b_g) \sum_{l=1}^{L} t_{((m-1)L+l)kg} \frac{\partial \mu_{Z_{al}}(a_g)}{\partial \alpha_{\phi kg}} \right),\end{aligned} \tag{10}$$

where $\phi \in \{1, \ldots, L-1\}$.

Updated parameter values of the fuzzy sets for the input a can be calculated as

$$\begin{aligned}&\alpha_{\phi k(g+1)} \\ &\quad = \alpha_{\phi kg} + \eta_a \left(\tilde{t}_g - \hat{t}_g\right) \\ &\quad \times \sum_{m=1}^{M} \left(\mu_{Z_{bm}}(b_g) \sum_{l=1}^{L} t_{((m-1)L+l)kg} \frac{\partial \mu_{Z_{al}}(a_g)}{\partial \alpha_{\phi kg}} \right).\end{aligned} \tag{11}$$

Updated parameter values of the fuzzy sets for the input b

can be calculated in a similar way,

$$\beta_{\gamma k(g+1)} = \beta_{\gamma kg} + \eta_b \left(\tilde{t}_g - \hat{t}_g\right) \times \sum_{l=1}^{L} \left(\mu_{Z_{al}}(a_g) \sum_{m=1}^{M} t_{((l-1)M+m)kg} \frac{\partial \mu_{Z_{bm}}(b_g)}{\partial \beta_{\gamma kg}} \right), \quad (12)$$

where $\gamma \in \{1, \ldots, M-1\}$.

The elements $\partial \mu_{Z_{al}}(a_g) / \partial \alpha_{\phi kg}$ and $\partial \mu_{Z_{bm}}(b_g) / \partial \beta_{\gamma kg}$ in (11) and (12) are the partial derivatives calculated for triangular membership functions of the inputs a and b. The unfavorable feature of the adopted membership functions is the fact that they are piecewise linear and thus there are points for which a partial derivative cannot be calculated. For non-differentiable points, such as peaks, it is assumed after Pilinski (1996) that a derivative in these points is equal to the mathematical average of the two neighboring derivatives. The parameters of the fuzzy sets are updated online every time the request service is completed.

The number of the fuzzy sets for the inputs and the assumed learning rates η_a, η_b and η_t influence the adaptation rate and the accuracy of the described model for the system modeled. The higher the number of the fuzzy sets for the input, the more accurately the model describes the system behavior; however, at the same time it decreases the convergence rate to the actual system. The adaptation rate also depends, to a large extent, on the learning rate adopted. The higher the values of the rate, the faster the adaptation proceeds, but at the same time the model loses the properties of generalization. After numerous preliminary investigations, not discussed in this paper, it was assumed that the number of fuzzy sets for both inputs will be the same, $L = M = 4$, for each of the experiments to be presented further in the paper. Based on these investigations and experiments similar to those of Lee *et al.* (1995), also the learning rates $\eta_a = \eta_b = 0.1$ and $\eta_t = 0.3$ were determined. Since the adaptation algorithm operates online, the values of the learning rates are constant during the operation of the Web switch. It is adopted that the initial values of parameters t_{jki}, where $j = 1, \ldots, J$, $k = 1, \ldots, K$, $g = 1$, should be zero. Parameters α_{lkg}, and β_{mkg} of membership functions for the input fuzzy sets, where $l = 1, \ldots, L-1$, $m = 1, \ldots, M-1$, $k = 1, \ldots, K$, $g = 1$, are adopted to uniformly cover the actual span of the inputs a_g and b_g, where the values of the parameters of terminal fuzzy sets are adopted as $\alpha_{0kg} = 0$, $\alpha_{3kg} = 100$, $\beta_{0kg} = 0$, $\beta_{3kg} = 100$.

5. Simulation model and experiment results

In order to determine the quality of the operation of the switch working under the control of the LFNRD algorithm, an adequate testbed was prepared and simulation experiments were carried out. For the development of the simulation program, the CSIM19 packet (CSIM, 2008) was used. The simulator included the following modules: a request generator, an LFNRD switch, a WWW server, and a database server. The simulator scheme is shown in Fig. 8.

Table 1. Workload model parameters.

Category	Distribution	Parameters
Number of page views per session	Reverse Gaussian	$\mu = 3.86$, $\lambda = 9.46$
User's think time	Pareto	$\alpha = 1.4$, $k = 1$
Number of objects on site	Pareto	$\alpha = 1.33$, $k = 2$
Browser think time	Weibul	$\alpha = 7.640$, $\sigma = 1.705$
Size of HTML page frame	Lognormal Pareto	$\mu = 7.630$, $\sigma = 1.001$, $k = 10240$, $\alpha = 1$
Embedded object size	Lognormal	$\mu = 8.215$, $\sigma = 1.46$

The widely used, and well known from the literature, HTTP request generator model, which made modeling the clients' behavior possible, was adopted in simulation experiments (Barford *et al.*, 1999; Casalicchio and Colajanni, 2001; Xia *et al.*, 2005). Advantages of this model were confirmed by Williams *et al.* (2005). Owing to the model applied, the generated request traffic complied with the actual traffic observed on the Internet, which is characterized by bursts and self-similarity. Such a traffic may be generated using long-tail distributions, such as Pareto and lognormal distributions. During its operation, the request generator created a given number of simulation clients within one second. Table 1 shows probability distributions and distribution parameter values used in the HTTP request generator.

The simulated WWW service was able to service

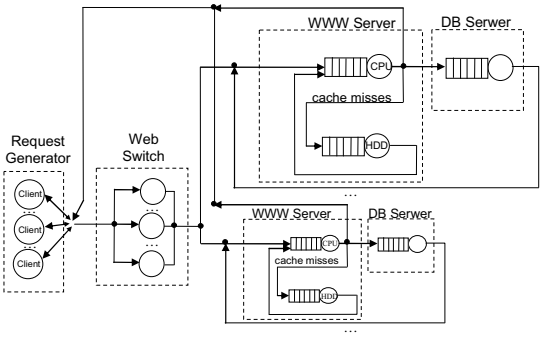

Fig. 8. Simulation model

both static (referring to static objects) and dynamic requests (referring to dynamic objects) in the simulation program. The latter were serviced by the WWW server and the database server whereas the former were serviced by the WWW server only (Menasce and Almeida, 1998; Pai *et al.*, 1998). It was assumed for testing purposes that 80% of the requests referred to static resources and 20% to dynamic ones.

The dynamic requests were divided into three classes (Cardellini *et al.*, 2001):

- high intensive, applying a heavy load on the database server, constituting 1% of dynamic requests;
- medium intensive, applying a medium load on the database server, constituting 14% of dynamic requests;
- low intensive, applying a small load on the database server, constituting 85% of dynamic requests;

It was assumed that the size of all the static objects provided by the service was 400 MB. The size and structure of the simulated Web object were generated according to distributions presented in Table 1.

Clients send requests to the module of the LFNRD switch in the simulation program. It is assumed that the link between the clients and the switch is of an infinite transfer capacity.

In order to compare the quality of service for the cluster being under control of the LFNRD algorithm with the quality of service under other reference algorithms and the algorithms used most often in industrial solutions, the FNRD, LARD, CAP, WRR and RR algorithms were also implemented in the switch module. The LARD algorithm assigns incoming requests taking into account the content of the Web servers caches. The CAP algorithm assigns requests according to the RR policy separately for different types of requests. The WRR algorithm assigned a different number of clients' requests to Web servers according to the weights adopted. Weights for the particular servers were calculated with a definite interval based on the number of HTTP requests actively serviced by the particular servers. This measure reflects a server load well and is often used in request distribution algorithms (Casalicchio and Colajanni, 2001; Cardellini *et al.*, 2002). The algorithm redirected more requests to the less-loaded servers.

The WWW server module included a processor, a hard disk and cache. This model is recognized in the literature and was introduced by Pai *et al.* (1998).

The processor and the disk were modeled as queue systems with a single queue and one service. Request service times were determined in our experimental tests for the server with an Intel Pentium 4, 2 GHz processor and a Seagate ST340810A 80GB IDE hard disk. Linux Fedora Core 6 the operating system and the WWW server software Apache 2.2.4 with modules PHP 5.1 were installed on the server (Zatwarnicki, 2011).

Costs connected with the request service time can be described in the following way:

- service cost on the processor:
  - TCP/IP connection cost is 0.10097 ms;
  - request analysis and HTTP response preparation cost is 0.14533 ms;
  - data transfer cost is $0.004291z$ ms;
- service cost on the disk was modeled according to Eqn. (13)

$$S^D(z) = \begin{cases} 4.5 & \text{if} \quad z \in [0, 128], \\ 0.03813125z - 0.3808 & \text{if} \quad z \in (128, \infty), \end{cases} \tag{13}$$

where the service time on disk S^D is given in milliseconds, and z is the size of the requested object given in KB.

It is assumed after Pai *et al.* (1998) that the WWW server cache in the server model operates according to the LRU (Least Recently Used—the least recently used objects are the first to be removed) policy. The database server was modeled as a queue system with a single queue to a resource and one service. Dynamic requests service times were modeled according to the hyperexponential distribution after Cardellini *et al.* (2001), as well as Casalicchio and Colajanni (2001). Dynamic requests service times depended on the request type (high, medium and low intensive). Table 2 shows the assumed dynamic requests service times on the database server. During the simulation experiments the request response time and the 95-th percentile of the Web page response time were measured. The response time of the page was calculated as a sum of response times for objects of the page $T_{page} = \sum_{n=1}^{N} \tilde{t}_i$, where N is the number of objects embedded in the page plus one object of the HTML page frame, $\tilde{t}_i$ is the response time to an HTTP request referring to the page object (Cardellini *et al.*, 2002). The indicated service quality measure reflects the client's perspective of work with a given Internet service. Often the clients may not register the fact of downloading a single HTTP object by a browser but

Table 2. Request service times on the database server.

Dynamic request type	Average service time [ms]
Low intensive	10
Medium intensive	50
High intensive	100

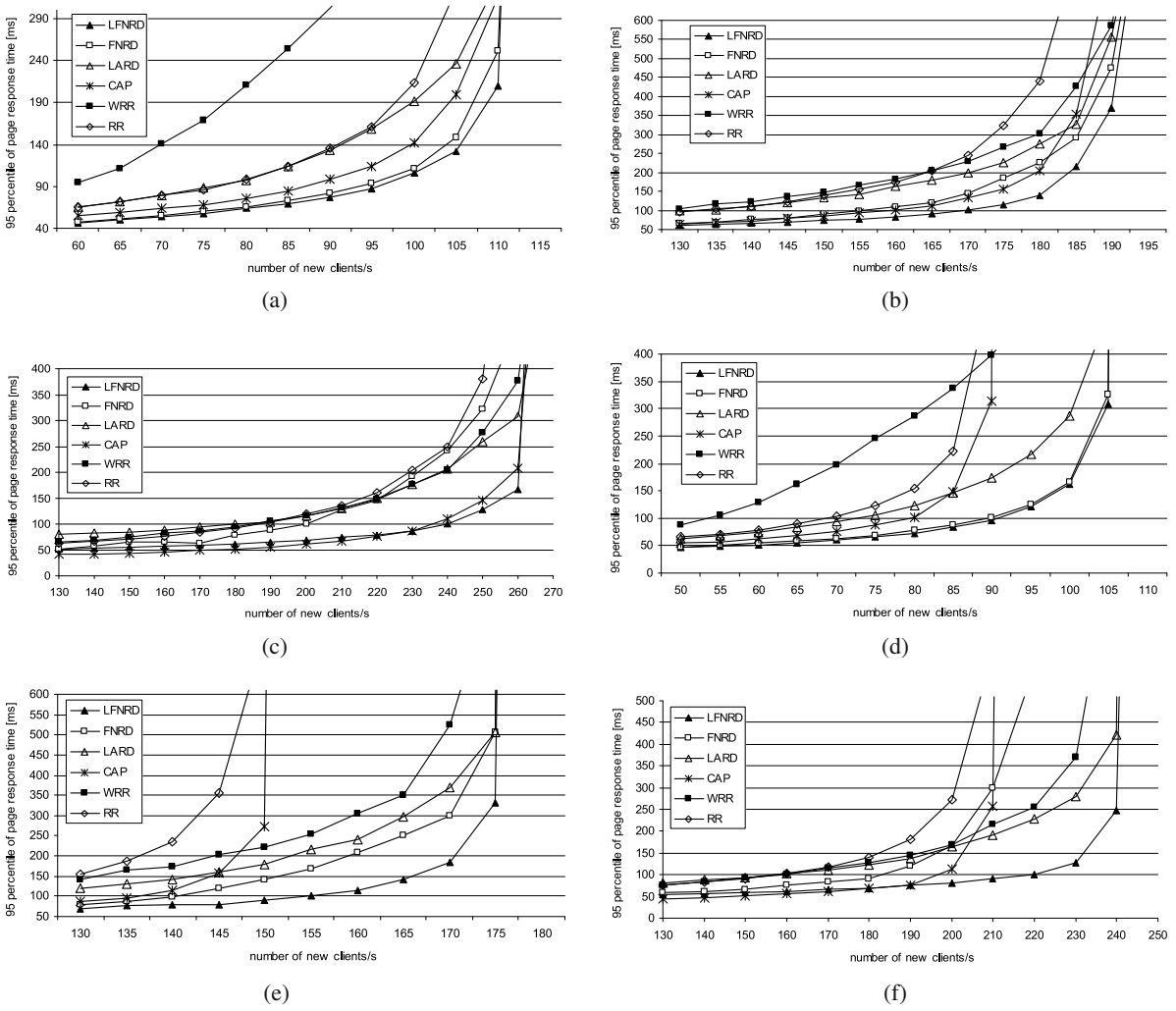

Fig. 9. 95-th percentile of the page response vs. the number of new clients for configurations: Hom3s (a), Hom5s (b), Hom7s (c), Het1s/2s (d), Het2s/3s (e), Het3s/4s (f).

they will pay attention to the loading time of the whole Web page consisting of a larger group of objects.

The investigations were carried out for six configurations of the servers in a cluster. Homogeneous WWW and database servers, whose description and configuration have been shown above, were used in the first three configurations. The first configuration (denoted as Hom3s) consisted of three WWW and database servers. The second configuration (Hom5s) was composed of five sets of servers. In the third configuration (Hom7s), seven sets of servers were used. Subsequent configurations comprised WWW and database servers described above, and the servers set having all request service times extended by 33%. In the fourth configuration (Het1s/2s), the cluster consisted of three WWW and database servers, with one set having all request service times extended by 33%. The fifth configuration (Het2s/3s) was composed of five sets of servers, and two sets were slower then others. In the last configuration (Het3s/4s), seven sets of servers were used, and two sets of servers had extended service times.

Figure 9 presents diagrams of the 95-th percentile of the Web page response time in a function of the load (number of new clients created per second) for four configurations of the server cluster. In Fig. 10, the diagrams of mean request response time for three of the configurations are presented.

As can be noticed, the tendencies on the graphs in Figs. 9 and 10 are the same. The diagrams show that the best results, that is, the lowest values for the 95-th percentile of the page response time and the mean request response time, are obtained for the LFNRD algorithm both in the case of homogeneous and heterogeneous clusters. Good results for this algorithm are obtained for the smaller three-server and five-cluster system, as well as for the bigger seven-server cluster. In the case of heterogeneous server clusters, the LFNRD algorithm results are signifi-

cantly better than those for the other algorithms, with response times at heavy load being almost twice shorter for this algorithm than for the other ones.

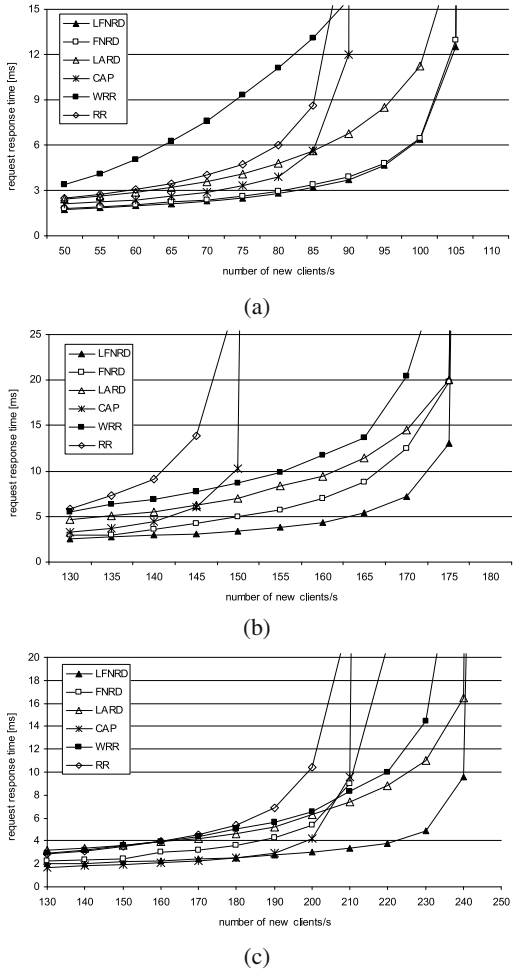

Fig. 10. Mean request response time vs. the number of new clients for configurations: Het1s/2s (a), Het2s/3s (b), Het3s/4 (c).

In the clusters with three sets of servers, similar results are obtained for the FNRD algorithm as for the LFNRD algorithm. The FNRD algorithm worked better for smaller clusters because parameters which were not tuned in this algorithm were initially set for three and four Web server clusters. For homogeneous cluster configurations quite good results are obtained for the CAP algorithm. The above results indicate that the LFNRD-controlled Web switch provides the service of the highest quality in relation to all other algorithms comparatively analyzed.

In addition to the experiments described above, an analysis of the request capacity of the Web switch was made, in terms of the number of requests redirected per second and mean decision time. The results were obtained for the server with two Intel Dual-Core Xeon 5160 processors with non-optimized software and performed for four concurrently operating processes, and for the cluster model containing three Web servers. The obtained results show that the Web switch working under the LFNRD has a relatively low request capacity as compared to the remaining algorithms. However, the maximum number of requests redirected per second is equal to 616000 for the LFNRD algorithm, which is sufficiently high even for a heavily loaded modern Web cluster. Also, the mean time to make an LFNRD decision equals 0.000001623 s, and is three times of magnitude smaller than the response time. This proves that the highest quality LFNRD Web switch can indeed operate in real time. However, the decision making time (including the adaptation time) depends on the number of Web servers operating in the cluster. The time to estimate the response time for one Web server is constant and is equal for each of the servers. Consequently, the time worst-case complexity depends linearly on the number of Web servers in the cluster $O(S)$. For this reason the LFNRD Web cluster should not contain too many Web servers—its number should not exceed 7. If the number of Web servers is bigger, then a different architecture of the cluster should be used. In such a case a Web switch with a simple distribution algorithm should be placed in the front-end of the cluster. This switch should receive all requests destined for the Web cluster and distribute them between two or more LFNRD Web switches (Cardellini *et al.*, 2002).

6. Conclusion

The paper has presented a design of a high quality Web switch applying a new LFNRD request distribution method using neuro-fuzzy models of Web servers. The application of these models has made it possible to (i) essentially improve the Web cluster operation quality as compared with the other reference algorithms, and (ii) forecast the decision making effects well before realization. The simulation experiments have shown that the approach proposed is adequate, and owing to the application of the algorithm developed it is possible to shorten request response times significantly. Further research has also shown that the proposed distribution method is computationally efficient and the operation of a Web switch in real time is possible.

In our future work we are going to apply neuro-fuzzy models in Web systems guaranteeing service quality both in locally and globally distributed systems. We suppose that the presented model can estimate the service time not only of single request but also of all of the Web page. This could be used to derive a method that helps to keep the Web page response time within established boundaries in such a way that, at a heavy workload, page response time both for small and complex pages would not exceed the imposed time limit.

References

AlSa'deh, A. and Yahya, A.H. (2008). Shortest remaining response time scheduling for improved web server performance, *in* J.Filipe and J. Cordeiro (Eds.), *Web Information Systems and Technologies*, Lecture Notes in Business Information Processing, Vol. 18, Springer-Verlag, Berlin/Heidelberg, pp. 80–92.

Andreolini, M., Casolari, S. and Colajanni, M. (2008). Autonomic request management algorithms for geographically distributed internet-based systems, *Proceedings of the 2nd IEEE International Conference on Self-Adaptive and Self-Organizing Systems, Venezia, Italy*, pp. 171–180.

Barford, P., Bestavros, A., Bradley, A. and Crovella, M. (1999). Changes in web client access patterns: Characteristics and caching implications, *World Wide Web* **2**(1): 15–28.

Borzemski, L. (2006). The use of data mining to predict web performance, *Cybernetics and Systems* **37**(6): 587–608.

Borzemski, L. and Suchacka, G. (2010). Business-oriented admission control and request scheduling for e-commerce websites, *Cybernetics and Systems* **41**(8): 592–609.

Borzemski, L. and Zatwarnicki, K. (2003). A fuzzy adaptive request distribution algorithm for cluster-based web systems, *11th Euromicro Workshop on Parallel, Distributed and Network-Based Processing, PDP 2003, Genoa, Italy*, pp. 119–126.

Borzemski, L. and Zatwarnicki, K. (2006). Fuzzy-neural web switch supporting differentiated service, *in* B. Gabrys, R.J. Howlett and L.C. Jain (Eds.), *Knowledge-Based Intelligent Information and Engineering Systems*, Lecture Notes in Artificial Intelligence, Vol. 4252, Springer-Verlag, Berlin/Heidelberg, pp. 195–203.

Borzemski, L., Zatwarnicki, K. and Zatwarnicka, A. (2007). Adaptive and intelligent request distribution for content delivery networks, *Cybernetics and Systems* **38**(8): 837–857.

Cardellini, V., Casalicchio, E., Colajanni, M. and Mambelli, M. (2001). Web switch support for differentiated services, *ACM Performance Evaluation Review* **29**(2): 14–19.

Cardellini, V., Casalicchio, E., Colajanni, M. and Yu, P.S. (2002). The state of the art in locally distributed web-server systems, *ACM Computing Surveys* **34**(2): 263–311.

Casalicchio, E. and Colajanni, M. (2001). A client-aware dispatching algorithm for web clusters providing multiple services, *Proceedings of the 10th International World Wide Web Conference, Hong Kong, China*, pp. 535–544.

Cherkasova, L. and Karlsson, M. (2001). Scalable webserver cluster design with workload-aware request distribution strategy ward, *Proceedings of the 3rd International Workshop on Advanced Issues of e-Commerce and Web-Based Information Systems (WECWIS), Washington, DC, USA*, p. 212.

CSIM (2008). Mesquite software 2008: Development toolkit for simulation and modeling, `http://www.mesquite.com`.

Driankov, D., Hellendoorn, H. and Reinfrank, M. (1996). *An Introduction to Fuzzy Control*, Springer, New York, NY.

Elnikety, S., Nahum, E., Tracey, J. and Zwaenepoel, W. (2004). A method for transparent admission control and request scheduling in e-commerce web sites, *WWW'04: Proceedings of the 13th International Conference on World Wide Web, New York, NY, USA*, pp. 276–286.

Gilly, K., Juiz, C. and Puigjaner, R. (2011). An up-to-date survey in web load balancing, *World Wide Web* **14**(2): 105–131.

Harchol-Balter, M., Schroeder, B., Agrawal, M. and Bansal, N. (2003). Size-based scheduling to improve web performance, *ACM Transactions on Computer Systems* **21**(2): 207–233.

Horikowa, S., Furuhashi, T. and Uchikawa, Y. (1992). On fuzzy modeling using fuzzy neural networks with the backpropagation algorithm, *IEEE Transactions on Neural Networks, Los Alamitos, CA, USA*, pp. 801–806.

Kwok, Y.-K. and Cheung, L.-S. (2004). A new fuzzy-decision based load balancing system for distributed object computing, *Journal of Parallel and Distributed Computing* **64**(2): 238–253.

Kun-Ming, V., Chou, Y. and Wang, Y. (2004). A fuzzy-based dynamic load-balancing algorithm, *Journal of Information, Technology and Society* **4**(2): 55–63.

Lee, K.M., Kwak, D.H. and Leekwang, H. (1995). Tuning of fuzzy models by fuzzy neural networks, *Fuzzy Sets and Systems* **76**(1): 47–61.

Lee, S.C.M., Lui, J.C.S. and Yau, D.K.Y. (2004). A proportional-delay diffserv-enabled web server: Admission control and dynamic adaptation, *IEEE Transactions on Parallel and Distributed Systems* **15**(5): 385–400.

Mamdani, E. H. (1977). Application of fuzzy logic to approximate reasoning using linguistic synthesis, *IEEE Transactions on Computers* **C-26**(12): 1182–1191.

Menasce, D. and Almeida, V. (1998). *Capacity Planning for Web Performance. Metrics, Models, and Methods*, Prentice-Hall, New York, NY.

Pai, V.S., Aron, M., Banga, G., Svendsen, M., Druschel, P., Zwaenepoel, W. and Nahum, E. (1998). Locality-aware request distribution in cluster-based network servers, *ACM SIGPLAN Notices* **33**(11): 205–215.

Pilinski, M. (1996). Universal network trainer, *Proceedings of the 2nd Conference on Neural Networks and Their Applications, Częstochowa, Poland*, Vol. 2, pp. 383–391.

Quan, Z. and Chung, J.-M. (2005). Statistical admission control for real-time services under earliest deadline first scheduling, *Computer Networks* **48**(2): 137–154.

Riska, A., Sun, W., Smirni, E. and Ciardo, G. (2002). Adaptload: Effective balancing in clustered web servers under transient load conditions, *22nd International Conference on Distributed Computing Systems (ICDCS 2002), Vienna, Austria*, pp. 103–111.

Simiński, K. (2010). Rule weights in a neuro-fuzzy system with a hierarchical domain partition, *International Journal of Applied Mathematics and Computer Science* **20**(2): 337–347, DOI: 10.2478/v10006-010-0025-3.

Wei, J. and Xu, C.-Z. (2006). Provisioning of client-perceived end-to-end QoS guarantees in web servers, *IEEE Transactions on Computers* **55**(12): 1543–1556.

Wei, J., Zhou, X. and Xu, C.-Z. (2005). Robust processing rate allocation for proportional slowdown differentiation on internet servers, *IEEE Transactions on Computers* **54**(8): 964–977.

Williams, A., Arlitt M., Williamson, C. and Barker, K. (2005). Web workload characterization: Ten years later, *in* X. Tang, I. Xu and S.T. Chanson (Eds.), *Web Content Delivery*, Web Information Systems Engineering and Internet Technologies, Vol. 2, Springer-Verlag, Berlin/Heidelberg, pp. 3–21.

Xia, C.H., Liu, Z., Squillante, M.S., Zhang, L. and Malouch, N. (2005). Web traffic modeling at finer time scales and performance implications, *Performance Evaluation* **61**(2): 181–201.

Zadeh, L.A. (1965). Fuzzy sets, *Information and Control* **8**(3): 338–353.

Zadeh, L. A. (1996). Fuzzy logic-computing with words, *IEEE Transactions on Fuzzy Systems* **4**(2): 104–111.

Zatwarnicki, K. (2010). Neuro-fuzzy models in global HTTP request distribution, *in* J. Pan, S. Chen and N.T. Nguyen (Eds.), *Computational Collective Intelligence*, Lecture Notes in Computer Science, Vol. 6421, Springer-Verlag, Berlin/Heidelberg, pp. 1–10.

Zatwarnicki, K. (2011). Identification of the Web server, *in* A. Kwiecień, P. Gaj and P. Stera (Eds.), *Computer Networks*, Communications in Computer and Information Science, Vol. 160, Springer-Verlag, Berlin/Heidelberg, pp. 45–54.

Zhou, X., Wei, J. and Xu, C.-Z. (2007). Quality-of-service differentiation on the internet: A taxonomy, *Journal of Network and Computer Applications* **30**(1): 354–383.

Krzysztof Zatwarnicki received his M.Sc. and Ph.D. degrees in computer science from the Faculty of Computer Science and Management, Wrocław University of Technology, in 1998 and 2003, respectively. He is an assistant professor at the Institute of Control and Computer Engineering, Opole University of Technology, Poland. He has authored or co-authored some 50 papers. His research interests concentrate on the problem of improving the quality of the Web service.

Appendix
Membership functions for input fuzzy sets

$$\mu_{Z_{a1}}(a_i) = \begin{cases} \dfrac{a_i - \alpha_{1ki}}{\alpha_{0ki} - \alpha_{1ki}} & \text{if } 0 \le a_i < \alpha_{1ki}, \\ 0 & \text{otherwise,} \end{cases}$$

$$\vdots$$

$$\mu_{Z_{al}}(a_i) = \begin{cases} \dfrac{a_i - \alpha_{(l-2)ki}}{\alpha_{(l-1)ki} - \alpha_{(l-2)ki}} & \text{if } \alpha_{(l-2)} < a_i \le \alpha_{(l-1)ki}, \\ \dfrac{a_i - \alpha_{lki}}{\alpha_{(l-1)ki} - \alpha_{lki}} & \text{if } \alpha_{(l-1)ki} < a_i < \alpha_{lki}, \\ 0 & \text{otherwise,} \end{cases}$$

$$\vdots$$

$$\mu_{Z_{aL}}(a_i) = \begin{cases} \dfrac{a - \alpha_{(L-2)ki}}{\alpha_{(L-1)ki} - \alpha_{(L-2)ki}} & \text{if } \alpha_{(L-2)ki} < a_i < \alpha_{(L-1)ki}, \\ 1 & \text{if } \alpha_{(L-1)ki} \le a_i, \\ 0 & \text{otherwise,} \end{cases} \tag{14}$$

where $l = 1, \ldots, L$ and $\alpha_{0ki} = 0$.

Received: 3 March 2011
Revised: 12 August 2011
Re-revised: 28 September 2011