Journal of Computers Vol. 27, No. 3, 2016, pp. 163-176
doi:10.3966/199115592016102703015

The Bandwidth Differentiated Service of Web Server

Yan-Su Hu[1] Ang Gao[2] De-Jun Mu[3]

[1] School of Electronics and Control Engineering, Chang'an University, Southern Middle Section of the Second Circular Road, Xi'an, China
huyansu@163.com

[2] School of Electronics and Information, Northwestern Polytechnical University, No. 127, West Youyi Road, Xi'an, China
gaoang@nwpu.edu.cn

[3] School of Automation, Northwestern Polytechnical University, No. 127, West Youyi Road, Xi'an, China
mudejun@nwpu.edu.cn

Received 11 March 2015; Revised 16 April 2015; Accepted 10 August 2015

Abstract. Differentiated service which belongs to the research of internet QoS means the requests with high priority should receive the quality of service “at least not lower than” the low priority. As we know, what clients most concerned is whether there is a small delay for an important request. The client-perceived delay is composed of connection delay and processing delay. So far, the QoS researches most focus on the connection delay that can be controlled by the threads / processes allocation. However, the experiments show that it is invalid when the bandwidth becomes the resource bottleneck. For this issue, the paper proposes bandwidth based QoS architecture to takes the processing delay into consideration. The self-tuning controller adjusts the bandwidth allocated to different virtual hosts to control the processing delay. The experiments show that the algorithm not only achieves the proportional delay guarantees but also emerges its advantage on the stability. Compared with the static fuzzy controller, the variance between the real delay proportion and the expect value is deceased 40% by the self-tuning control.

Keywords: bandwidth, differentiated service, proportional delay guarantees, QoS, self-tuning fuzzy control

1 Introduction

The increasing diversity of Web applications in the last decade has witnessed an increasing demand for provisioning of different levels of quality of service (QoS) to meet changing system configuration and satisfy different client requirements [1]. So many researches have been conducted focusing on the differentiated service on the application layer. Fig.1 shows the modified Apache MPM (multi-processing modules) architecture, which supports the QoS on the Web server.

(1) The single connection queue is improved to a multi-queue structure in accordance with the classified strategy. The listener monitors the network port, accepts the client TCP connections, classifies the requests based on some classified strategy, and then puts them into the appropriate waiting queue.

(2) As no request can occupy the server resource unlimitedly, the requests of different classes must be isolated. The thread per connection structure of MPM allows all the worker threads in pool to be the resource to be allocated. So we divide the pool into several sub-pools which are isolated with each other. The number of threads in each sub-pool is known as the thread quota, and the requests are serviced in the corresponding sub-pool.

(3) For N kinds of classes, let $c_i (i=1,...,N)$ be the thread quota allocated to the class i. When load varies, the size of sub-pool is dynamically adjusted to ensure the proportion between classes constant.

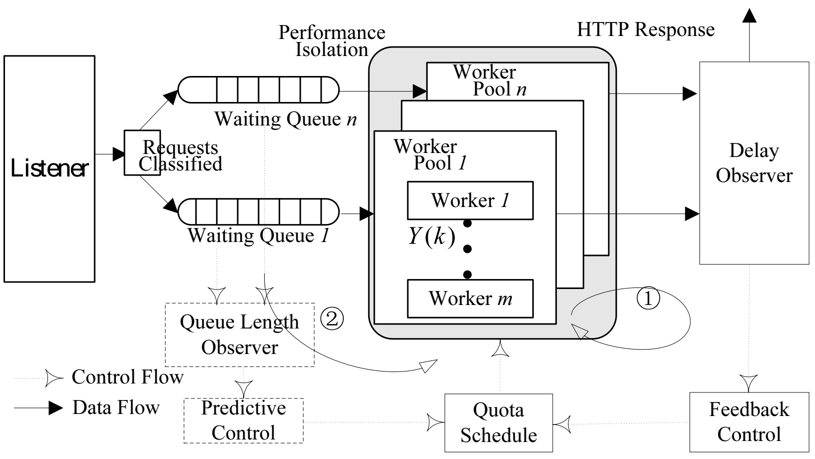


Fig. 1. The Web server architecture for differentiated service

And c_i can be calculated by the feedback control theory which should get the measured delay by the delay observer (method ① in Fig.1) [2-7], or by the queuing control theory which should get the request arrival rate by the queue length (method ② in Fig.1) [8-12].

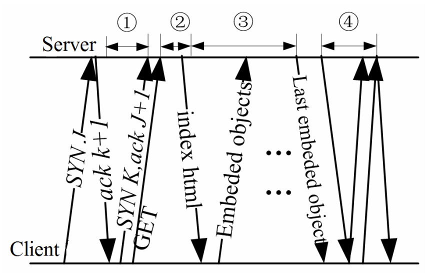
No matter what kind of method, the differentiated service is actually a scheduling problem of the bottleneck resource. For the Web server, the concurrent threads / processes and the bandwidth are the main system bottleneck. The concurrent threads / processes affect the connecting delay while the bandwidth works on the processing delay. But all the researches above aimed at the connecting delay by adjusting the threads / processes allocated to different priority clients and achieved the differentiated service. None of them focused on the impact of bandwidth. In fact, the threads / processes-based strategy only affects the connection delay but does nothing to the processing delay. When the bandwidth is the resource bottleneck, the processing delay is much bigger than the connection delay, and the traditional threads / processes-based strategy is invalid.

This paper discusses the QoS problem in the bandwidth limited Web server. The bandwidth allocated to different priorities is adjusted to guarantee a low delay for the high priority and an acceptable service for the low priority. First, experiments are operated to describe the defect of threads / rocesses-based QoS strategy in the next section. Then the third and firth section discussed about the bandwidth differentiated service architecture and the two-level fuzzy controller are proposed to realize the proportional delay differentiation service without the system model. Finally, the experiment results are shown to prove the availability and high efficiency of our method.

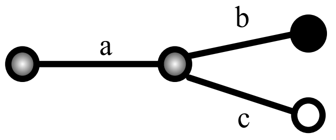
2 The defect of threads/processes-based QoS strategy

Fig.2 shows the interaction procedure between the clients and the server by HTTP 1.1 protocol: TCP connections are set up at the time period ①; the connection delay ② is the time in which HTTP requests get in the Web queue waiting to be serviced by idle threads / processes; ③ is called processing delay including the URL analyzing time and the transmission time of embedded objects; the TCP connections closed at time ④. In practice, the main factors affected the clients delay are the connection delay w_i and processing delay s_i (the total delay $l_i = w_i + s_i$).

As we talked above, both the feedback control and the queuing theory are based on the threads / processes-based QoS strategy. For the former, the improved MPM is equivalent to a control model as Fig.3. The plant is the worker threads. The system desired output is the inherent priority parameter ratio between class i and i+1. The controller output $X(k)$ is the thread quota ratio and $Y(k)$ is the measured output ratio between the class i and i+1. The plant's mathematical model (order and parameters) is obtained by system identification. Then the controller is designed based on the classical control theory. The latter method tries to correct the deviation before the system output being affected with the help of queuing theory. In the predictive control, the queue length observer measures the request arrival rate λ_i and



(a) The interaction of clients and server



(b) The test topology

Fig. 2. The interaction of clients and server and test topology

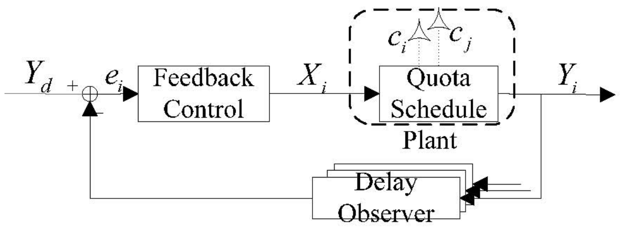


Fig. 3. The feedback control mode

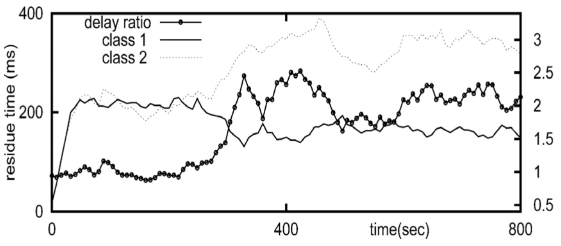
service rate μ_i. Then the predictive controller reallocates the thread quota c_i for respective class. In the reference paper, the Web server performance mainly subjects to the system bottlenecks, so each class in Apache can be considered as a $M/M/1/\infty$ or a $M/G/1/\infty$ queuing model.

Here are the experiments to prove the defect of threads / processes-based QoS strategy. As shown in Fig. 1(b), Apache 2 (Http-ver.2.2.63) serves two kinds of clients, i.e. client 1 and client 2. The clients are priority classified by their IP address. Take SURGE [13] be the workload generator. The PI controller in paper [12] is used to adjust the threads / processes allocation for different clients. The object is the delay ratio between the two kinds of clients constant. Set the expected delay ratio be 0.5, which means the delay of high class is half of the low one. In order to illustrate the effect of the bandwidth, two experiments are designed at the bandwidth of 10 Mbps and 2 Mbps. Link b and c are non-bottleneck links at 100Mbps bandwidth.

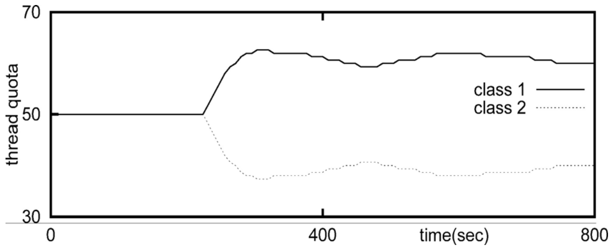
(1) *10 Mbps bandwidth:* Set the total number of Apache threads be 100. Two client machines running SURGE operate 100 concurrent UE respectively. Now the server threads are the bottleneck resources. The controller allocates more threads to high priority for less delay. In Fig. 3, the controller operates at 200 s, and the delay ratio achieves 0.5 (Fig.4(a)) when the number of threads for client 1 is increased to 65 (Fig.4(b)). At the moment, the processing delay of two clients is roughly equal $s_1 = s_2 = 100ms$ (Fig.4(c)), but their connection delay is obviously different $w_1 = 50ms$, $w_2 = 250ms$ (Fig.4(d)). So the threads / processes-based QoS strategy is valid in this situation.

(2) *2 Mbps bandwidth:* Set the total number of Apache threads be 200. Two client machines also operate 100 concurrent UE respectively. Fig.5(b) shows that no matter how to allocate the threads, the delay ratio is still far away from the expected value (Fig.5(a)). Compared with the results above, the processing delay greatly increases (Fig.5(c)), but the connection delay is almost zero (Fig.5(d)). Obviously, the thread / process-based QoS strategy is invalid (Fig.4(a)). That is because when the bandwidth is the bottleneck, the processing delay for transmitting pages becomes the main element instead of the connection delay and the thread allocation affects little to it.

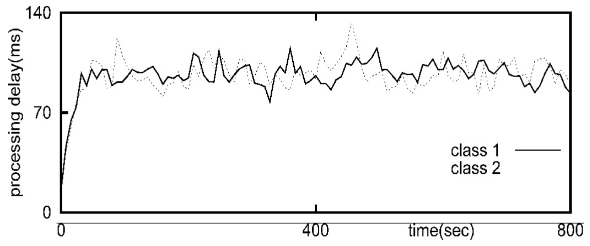
In the internet, the bandwidth is usually the bottleneck resource because the transmission for huge files takes plenty of bandwidth and occupies the threads / processes for a long time. In this case, the processing delay that is much greater than the connection delay cannot be ignored. So the threads / processes-based QoS strategy fails in the second experiment.



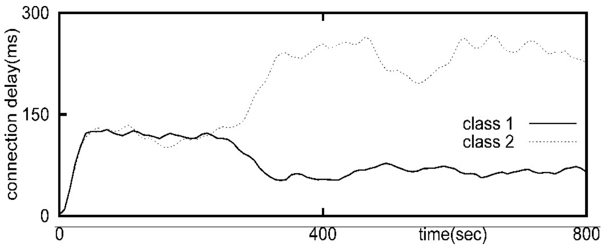
(a) The total delay and the delay ratio



(b) The thread quota



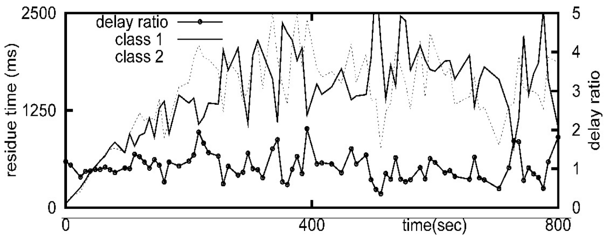
(c) The processing delay



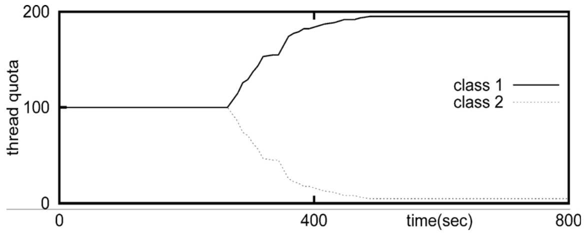
(d) The connection delay

Fig. 4. 10 Mbps, 100 threads, 100 UE

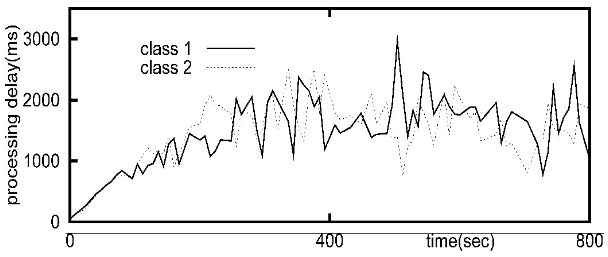
Note. the connection delay is the main element, and threads / processes-based QoS strategy is valid



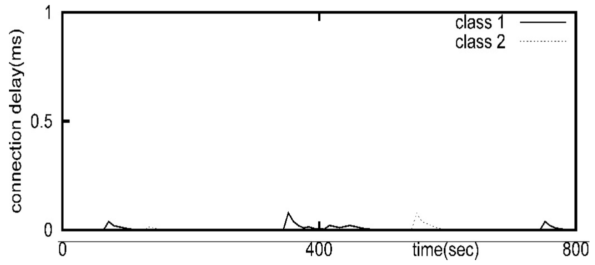
(a) The total delay and the delay ratio



(b) The thread quota



(c) The processing delay



(d) The connection delay

Fig. 5. 2 Mbps, 200 threads, 100 UE

Note. the processing delay is the main element, and threads / processes-based QoS strategy is invalid

3 The bandwidth differentiated service architecture

For the Web server, the processing delay s_i is proportional to the page size $size_i$, but inversely proportional to the bandwidth b_i, which is $s_i \propto size_i/b_i$ [14]. Fig.6 shows the differentiated service architecture based on the bandwidth allocation. According to the Service-Level Agreement (SLA), the terminal routers classify the HTTP requests into $N(N \geq 2)$ queues and then send them to the related ports of Web server. Each port represents a Virtual Host [15]. The QoS observer submits the feedback information to the fuzzy controller where the processing time of different priority is controlled by adjusting the available bandwidth b_i of the Virtual Host. Due to the physical bandwidth is limited to W, there is the constraint condition:

$$\sum_{i=1}^{N} b_i(k) = W \text{.} \tag{1}$$

The control objective is:

$$\begin{cases} \dfrac{\delta_{i+1}}{\delta_i} = Y_{i_{desire}}, \; i = 1, \ldots, N-1 \\ y_i(k) = \dfrac{l_{i+1}(k)}{l_i(k)} \\ y_i(k) = Y_{i_{desire}} \end{cases} \tag{2}$$

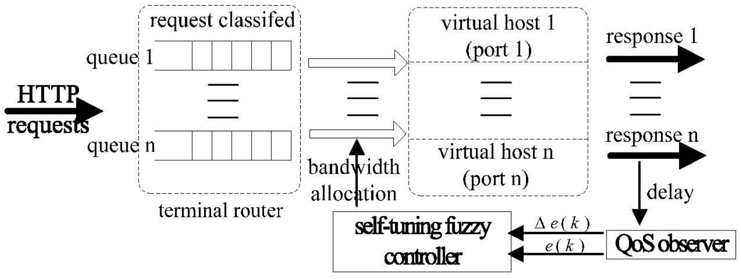


Fig. 6. The bandwidth differentiated service architecture

Where δ_i is the SLA inherent priority factor, the smaller value means the higher priority. $Y_{i_{desire}}$ is the expected delay ratio, and $y_i(k)$ is the measured delay ratio between two adjacent priority. So the control objective can be summarized as that the delay ratio between two adjacent priority equals to a specific value set by the SLA.

The features of different requests can be described by a two-tuples $[\lambda, \overline{\rho}]$, where λ is the requests arrival rate and $\overline{\rho}$ is the average size of request page. Hence, the bandwidth for the HTTP requests can be described by Equation (3) [16]:

$$\text{Bandwidth Needed}=F([\lambda, \overline{\rho}], y_{desire}, \varepsilon) \text{.} \tag{3}$$

Where ε is the uncontrollable stochastic disturbance.

At present, there are two methods to establish the mathematical model for the concerned problem. One is the traditional linear feedback control and the other is the queuing model based predictor.

(1) Traditional linear feedback control has been applied as an analytic method for QoS guarantees in web servers because of its self-correcting and self-stabilizing behavior. For this method, the nonlinear relationship between the allocated resource of a class and its received service quality is linearized at a fixed operating point. It is well known that linear approximation of a nonlinear system is accurate only within the neighborhood of the point where it is linearized. In fast changing web servers, the operating point changes dynamically, thus, simple linearization is inappropriate.

Further more, in web servers, resource allocation must be based on an accurately measured effect of previous resource allocation on the client-perceived response time of web pages. According to HTTP, a client should fist sends a request for the base page to retrieve a web page. The server then schedules the request according to its resource allocation. At this point, it is impossible to measure the client perceived response time of the web page because the server needs to handle the request and the response needs to be transmitted over the networks. An accurate measurement of resource-allocation effect on response time thus is delayed. Consequently, the resource allocation is significantly complicated because it has to be based on an inaccurate measurement.

(2) The process delay has also been addressed using queuing model based predictor in [5-8]. They integrated the predictor into a linear feedback controller to react to incoming performance degradation according to predicted server workloads. Without an appropriate model to describe the server behaviors

with respect to web pages, the performance of their approach is limited.

So in this paper, we propose the fuzzy control to avoid the complicated modeling of bandwidth consumption [17]. The "allocation-and-see-what-happens" [18] method updates the bandwidth of each class $b_i(k)$ periodically based on the deviation $e_i(k) = y_i(k) - y_{desire}$ to hold the Equation (2).

4 The self-tuning fuzzy controller

Now we will present the model independent two-level self-tuning fuzzy controller (STFC) and the implementation issues. The bandwidth resource controller (BRC) on the first level takes advantage of fuzzy control theory to address the issue of lacking accurate server models. The scaling factor controller (SFC) is to compensate the effect of process delay by adjusting the resource controller's output scaling factor according to transient server behaviors.

4.1 The architecture design

The STFC based on the "Class-Per-Loop" structure is proposed as Fig.7 [9]. It is composed of two level fuzzy controllers [19]: the BRC and the SFC Aimed at the uncertainty of controlled objects, the BRC allocates the host resource by fuzzy inference and outputs the bandwidth variation quality $\Delta u_i(k)$. The SFC is to compensate the processing delay jitter caused by size diversity of request files, and the output is the scaling factor $\alpha_i(k)$.

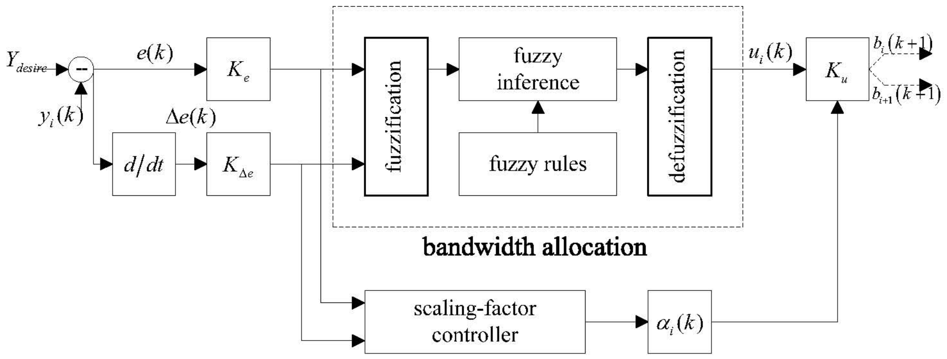


Fig. 7. The two level self-tuning fuzzy controller

As shown in the Fig.7, the inputs of the STFC are the system deviation $e_i(k)$ and its gradient $\Delta e_i(k)$. According to the inputs at the k^{th} sampling period, the STFC calculates the output for the next sampling period $\alpha_i(k) \times \Delta u_i(k)$, which is the bandwidth variation quality needed to adjust.

$$\begin{cases} \tilde{b}_i(k+1) = b_i(k) + \alpha_i(k)\Delta u_i(k) \\ \tilde{b}_{i+1}(k+1) = b_{i+1}(k) - \alpha_i(k)\Delta u_i(k) \end{cases} \quad \textbf{(4)}$$

$\tilde{b}_i(k+1)$ is the expected bandwidth at the next sampling time. But since the finite physical bandwidth, the actual bandwidth allocated to class i denoted as $b_i(k+1)$ is not equal to the expected value. In this paper, we are only concerned with the delay proportional relationship, so what we required is just the actual bandwidth ratio $m_i(k)$ of two adjacent class is in accord with the expected ratio, which is

$$\frac{\tilde{b}_i(k+1)}{\tilde{b}_{i+1}(k+1)} = \frac{b_i(k+1)}{b_{i+1}(k+1)} = m_i(k) \quad \textbf{(5)}$$

Combined with the Equation (1):

$$(1 + \sum_{i=1}^{N-1} \prod_{j=i}^{N-1} m_j(k)) b_N(k+1) = W \quad \textbf{(6)}$$

Solve Equation (6), the bandwidth quota $b_i(k+1)$, $(i=1,...,N-1)$ allocated to each virtual host is obtained.

4.2 The BRC

First fuzzify the input and output variables. Specifically, linguistic variables E , EC and U are used to describe $e_i(k)$, $\Delta e_i(k)$ and $\Delta u_i(k)$ respectively. The corresponding basic universes are $\left[-|e_{\max}|,|e_{\max}|\right]$, $\left[-|ec_{\max}|,|ec_{\max}|\right]$ and $\left[-|\Delta u_{\max}|,|\Delta u_{\max}|\right]$. Each linguistic variable includes 7 linguistic value, which is {*Negative Large (NL), Negative Medium (NM), Negative Small (NS), Zero (ZE), Positive Small (PS), Positive Medium (PM), Positive Large (PL)*}(Table 1). As shown in Fig.8(a), choose "*Trimf*" (?) be their membership function. While the universes are all defined as $\{-3,-2,-1,0,1,2,3\}$. The quantification factors are $K_e = 3/|e_{\max}|$, $K_{\Delta e} = 3/|ec_{\max}|$, and the proportional factor is $K_u = |\Delta u_{\max}|/3$. Let $\hat{e}_i(k)$, $\hat{\Delta e}_i(k)$ and $\hat{\Delta u}_i(k)$ be the quantified values:

$$\hat{e}_i(k) = K_e e_i(k), \hat{\Delta e}_i(k) = K_{\Delta e}\Delta e_i(k), \Delta u_i(k) = K_u \hat{\Delta u}_i(k) \tag{7}$$

Table 1. The variables fuzzying illustration of BRC

Linguistic variable	Basic Universes	Linguistic value	Universes (7 levels)	Quantification / Proportional Factor
E	$\left[-\|e_{\max}\|,\|e_{\max}\|\right]$	{NL, NM, NS, ZE, PS, PM, PL}	$\{-3,-2,-1,0,1,2,3\}$	$K_e = 3/\|e_{\max}\|$
EC	$\left[-\|ec_{\max}\|,\|ec_{\max}\|\right]$	{NL, NM, NS, ZE, PS, PM, PL}	$\{-3,-2,-1,0,1,2,3\}$	$K_{\Delta e} = 3/\|ec_{\max}\|$
U	$\left[-\|\Delta u_{\max}\|,\|\Delta u_{\max}\|\right]$	{NL, NM, NS, ZE, PS, PM, PL}	$\{-3,-2,-1,0,1,2,3\}$	$K_u = \|\Delta u_{\max}\|/3$

Then set the fuzzy rules. As shown in Fig.8(b), we can set the fuzzy rules based on the 5 zones with different characteristic:

(1) For Zone 1 and Zone 3, $e_i(k)$ and $\Delta e_i(k)$ are with opposite signs. Now the controller is self-correcting, and the achieved value is moving towards to the expected value. Thus, the bandwidth variation quality $\Delta u_i(k)$ should get a smaller value to avoid the system overshoot.

(2) For Zone 2 and Zone 4, $e_i(k)$ and $\Delta e_i(k)$ are with the same signs. Now the achieved value is moving away from the expected value. Thus, a large value of $\Delta u_i(k)$ is needed to reverse the trend.

(3) For Zone 5, there is a small $|e_i(k)|$ and $|\Delta e_i(k)|$. The system is approximate stability. So we only need a tiny adjustment of $\Delta u_i(k)$.

Fig.8(c) shows the fuzzy rules table, and Fig.8(d) is the surface of fuzzy rules. The fuzzy rule bases are described as:

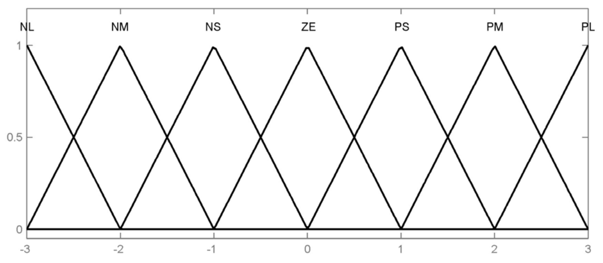
$$F = \{F_1, \cdots F_L\}, L = 49$$
$$F_L : if\ E\ is\ A_l, and\ EC\ is\ B_l, then\ U\ is\ C_l, which\ is\ rule(A_l, B_l) \to C_l$$

Where A_l , B_l and C_l are the linguistic value of E , EC and U in the l^{th} rule. For example, when $e(k) = 1/4$, $\Delta e(k) = 1$, based on the membership function as Fig.8, there is:

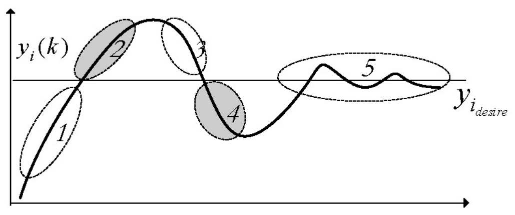
$$\mu_{ZE}(e(k)) = 0.75, \mu_{PS}(e(k)) = 0.25, \mu_{PS}(\Delta e(k)) = 1 \tag{8}$$

μ is the condition for membership. The fuzzy inference adopts minimal rule, which is:

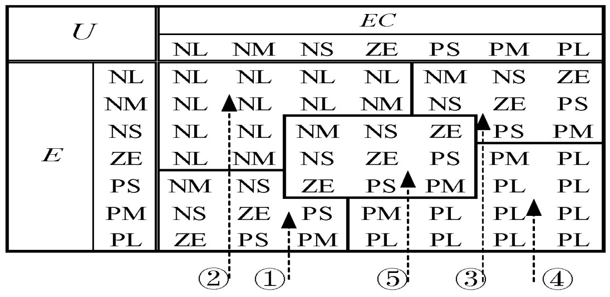
$$\mu_{F_l}(e, \Delta e) = \min(\mu_{A_l}(e), \mu_{B_l}(\Delta e)) \tag{9}$$



(a) The membership functions ("*Trimf*") of $e_i(k), \Delta e_i(k), \Delta u_i(k)$

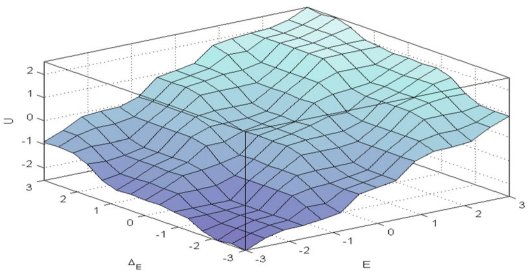


(b) Control effect in 5 zones



U		*EC*						
		NL	NM	NS	ZE	PS	PM	PL
E	NL	NL	NL	NL	NL	NM	NS	ZE
	NM	NL	NL	NL	NM	NS	ZE	PS
	NS	NL	NL	NM	NS	ZE	PS	PM
	ZE	NL	NM	NS	ZE	PS	PM	PL
	PS	NM	NS	ZE	PS	PM	PL	PL
	PM	NS	ZE	PS	PM	PL	PL	PL
	PL	ZE	PS	PM	PL	PL	PL	PL

(c) The rule-base of the resource controller



(d) The surface of fuzzy rules

Fig. 8. Fuzzy control rules in BRC

Obviously, only $rule(ZE,PS) \to NS$ and $rule(PS,PS) \to NM$ are valid at present:

$$\mu_{(ZE,PS)}(3/4,1) = \min(0.75,1) = 0.75, \mu_{(PS,PS)}(1/4,1) = \min(0.25,1) = 0.25 \quad \textbf{(10)}$$

Defuzzify the output by the "center average" method. Let $b(F_l)$ denote the center of membership function where it reaches its peak. Then the fuzzy control output is:

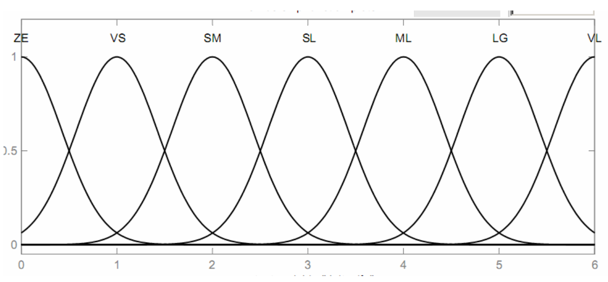
$$\Delta u(k) = \frac{\sum_{l=1}^{L} b_{F_l}[\min(\mu_{A_l}(e(k)), \mu_{B_l}(\Delta e(k)))]}{\sum_{l=1}^{L}[\min(\mu_{A_l}(e(k)), \mu_{B_l}(\Delta e(k)))]} \quad (11)$$

4.3 The SFC

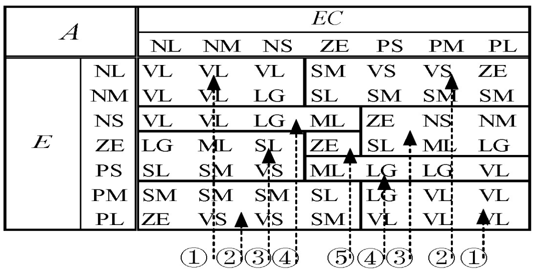
The SFC compensates the processing delay jitter caused by size diversity of request files. $\alpha_i(k)$ is denoted by the linguistic variables A, and the basic universe is $[0, \alpha_{\max}]$, $\alpha_{\max} = 1$. The linguistic value consists of *{Zero (ZE), Very Small (VS), Small (SM), Small Large (SL), Medium Large (ML), Large (LG), Very Large (VL)}* (Table 2), and the membership function is "*Gaussian*" type (Fig.9 (a)). The universe is $\{0,1,2,3,4,5,6\}$. The proportional factor is $K_\alpha = |\alpha_{\max}|/6$, $\alpha_i(k) = K_\alpha \hat{\alpha}_i(k)$.

Table 2. The variables fuzzying illustration of SFC

Linguistic variable	Basic Universes	Linguistic value	Universes(7 levels)	Quantification / Proportional Factor
A	$[0, \alpha_{\max}]$	{ZE, VS, SM, SL, ML, LA, VL}	$\{0,1,2,3,4,5,6\}$	$K_\alpha = \lvert\alpha_{\max}\rvert/6$



(a) The membership function (“*Gaussian*”) of $\alpha_i(k)$



A		EC						
		NL	NM	NS	ZE	PS	PM	PL
E	NL	VL	VL	VL	SM	VS	VS	ZE
	NM	VL	VL	LG	SL	SM	SM	SM
	NS	VL	VL	LG	ML	ZE	NS	NM
	ZE	LG	ML	SL	ZE	SL	ML	LG
	PS	SL	SM	VS	ML	LG	LG	VL
	PM	SM	SM	SM	SL	LG	VL	VL
	PL	ZE	VS	VS	SM	VL	VL	VL

(b) The rule-base of the SFC

Fig. 9. Fuzzy control rules in SFC

Similar with the BRC, the fuzzy rules can also be described by 5 zones (Fig.9(b)):

(1) When $|e_i(k)|$ is large, $e_i(k)$ and $\Delta e_i(k)$ are with the same signs (Zone 1), the delay is not only far away from the expected value, but also moving far away. Now need a large α to prevent the deviation trend. But if $e_i(k)$ and $\Delta e_i(k)$ are with opposite signs (Zone 2), α should be small to reduce the over-shoot and the setting time without sacrificing of the bandwidth allocation sensitivity.

(2) Because of the page size diversity and the load high dynamic features, $|e_i(k)|$ is small when the system at the neighbor of equilibrium point. When $\Delta e_i(k)$ is large and $e_i(k)$ is with the same signs, the system is departing from the expected value rapidly (Zone 4). Now α should be large to compensate the deviation. When $\Delta e_i(k)$ is small and $e_i(k)$ is with the opposite sign, the system is moving towards to the expected value (Zone 3). A small α can avoid the overshoot.

(3) When $|e_i(k)|$ and $|\Delta e_i(k)|$ are both small, α only need a fine tuning at zero.Obviously, the fuzzy rule bases are:

$$G = \{G_1, \cdots G_M\}, M = 49$$
$$G_M : if\ E\ is\ A_m, and\ EC\ is\ B_m, then\ U\ is\ D_m, which\ is\ rule(A_m, B_m) \rightarrow D_m$$

The surface of fuzzy rules is similar to Fig. 8(d), so we do not give it repeatedly. Supposed the membership function reaches its peak value at the point of b_{G_m}. Then calculate the defuzzification value of α by the “center average” method:

$$\hat{\alpha}(k) = \frac{\sum_{m=1}^{M} b_{G_m}[\min(\mu_{A_m}(e(k)), \mu_{B_m}(\Delta e(k)))]}{\sum_{m=1}^{M}[\min(\mu_{A_m}(e(k)), \mu_{B_m}(\Delta e(k)))]} . \quad \textbf{(12)}$$

5 Experimental Results

This section evaluates the validity of the proposed strategy by experiments and compares the performance of different controllers. The test topology is shown in Fig.1(b). The experimental parameters are shown in Table 3. The test bed consists of a Web sever machine and two client machines, each is equipped with a 3.0 GHz Pentium 4 professor, 521 MB RAM and 100 Mbps network card. The bandwidth of physical host is set to be 2 Mbps. The Web server is Apache 2 (Httpd-ver2.2.63) running on Windows NT and the total number of server processes is configured to be 200. Liunx-2.6.27 is applied for the two client machines. Let SURGE (ver 1.00a) as the workload generator and each operates 100 concurrent UE. Requests are classified according to their IP address. All the experiments are under HTTP1.1 pipeline and the number of maximum concurrent clients in SURGE is 1.

Table 3. The experimental parameters configuration

Name	Configuration
CPU	3.0G Hz Pentium 4
RAM	512 M (DDR3)
Web server	Apache 2(Httpd-ver2.2.63) with 200 threads
Bandwidth of Web sever	2 Mbps
Operating System of Web sever	Windows NT
Workload generator	SURGE (ver 1.00a) each with 100 concurrent CE
Operating System of client machine	Liunx-2.6.27

Set $Y_{desire} = y_{1_{desire}} = \delta_1/\delta_2 = 1/2$, which means the delay of high class is half of the low one. Limited $|e_{\max}| = |ec_{\max}| = 0.5$ and $|u_{\max}| = 25000$ which means the bandwidth can be adjusted no more than 25 Kbps in a sampling period. The relative variance $\Psi(e)$ is defined as a control performance metric. A smaller $\Psi(e)$ indicates a better stability that controller can keep $\boldsymbol{Y}(k)$ at $\boldsymbol{Y}_{desire}$.

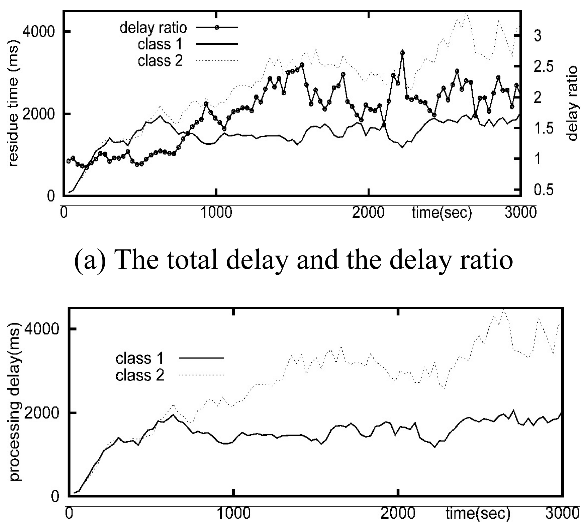
$$\Psi_i(e) = \frac{\sqrt{\sum_{k=1}^{n} e_i(k)^2 / n}}{y_{i_{desire}}} \quad . \tag{13}$$

Three experiments are designed to evaluate the bandwidth-based differentiated service model in different controller.

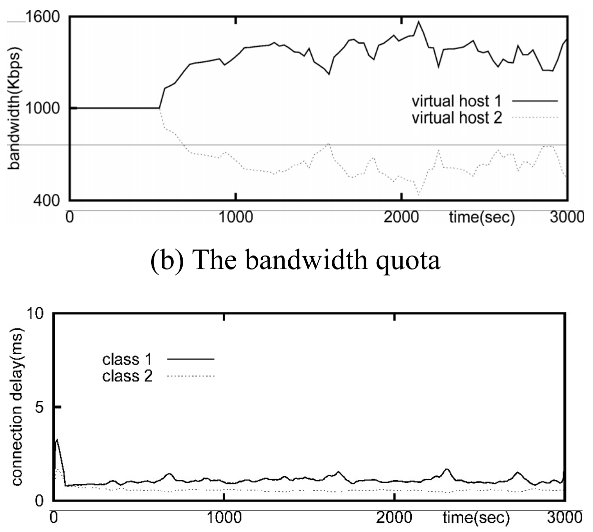
Experiment 1: Set the sampling time $T = 25s$ under the self-tuning fuzzy control. As shown in Fig. 10, we can see that:

(1) At the first 500s, the STFC is closed and the bandwidth quota allocated to different class is the same. So correspondingly there is little difference at the total delay (Fig.10(a)). But when the controller operates after 500s, the bandwidth allocated to different class is adjusted (Fig.10(b)), and the delay ratio is gradually settled at the expected value (Fig.10(a)). This proves the validity of the self-tuning fuzzy controller.

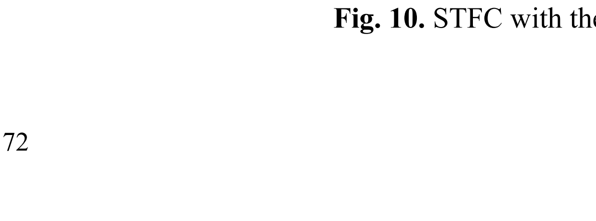
(2) When bandwidth is the system bottleneck, the processing delay is the main delay element, while the connection delay with no difference of two classes can be ignored for their small value (Figs. 10(c) and 10(d)). Compared with the experiments in Section 2, we again find that the proportional delay guarantee is realized by adjusting the processing delay (bandwidth). This proves the validity of the bandwidth differentiated service architecture.



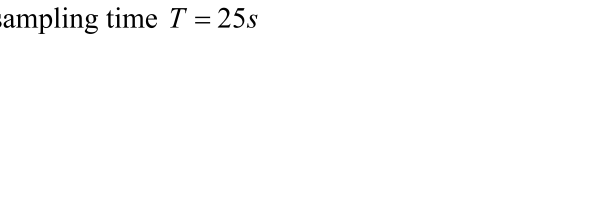
(a) The total delay and the delay ratio



(b) The bandwidth quota



(c) The processing delay



(d) The connection delay

Fig. 10. STFC with the sampling time $T = 25s$

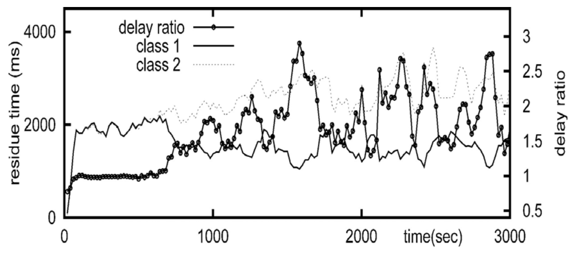
Experiment 2: Compare the control effect between the STFC and the static fuzzy controller with the sampling time $T = 25s$. For the static fuzzy controller, close the scaling factor which means $\alpha^{Static}(k) \equiv 1$. Set the proportional factor of bandwidth variation quality be one-third of the original, which is $3K_{\Delta u}^{Static} = K_{\Delta u}^{STFC}$ [20]. The other variables keep the same. So we get the results as Fig.11:

(1) Similarly to Experiment 1, the static fuzzy controller is closed before 500s and the bandwidth quota allocated to different class is the same (Figs. 11(a) and 11(b)). But when the controller operates, the static fuzzy controller also can achieve the differentiated service by adjusting the bandwidth (Figs. 11(a) and 11(b)) which proves the validity of the bandwidth differentiated service architecture once again.

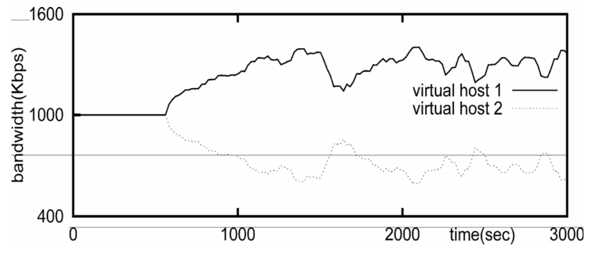
(2) Now the bandwidth is also the limited resource. So when the controller operates, the processing delay (Fig.10 (c)) is much larger than the connection delay (Fig. 11(d)) which can be ignored.

(3) Compared with Fig.10, there is little difference for the setting time. But obviously, the delay ratio jitter of static fuzzy controller is much bigger than that of self-tuning fuzzy controller. This situation can be specified by calculating the relative variance $\Psi(e)$. Take class 1 for example. As shown in the Equation (14), the relative variance of STFC is just 40% of the static fuzzy controller. The reason is the scaling factor can effectively compensate the delay jitter caused by the size diversity of request files. The STFC maintains a better system performance in the "heavy-tailed" workload.

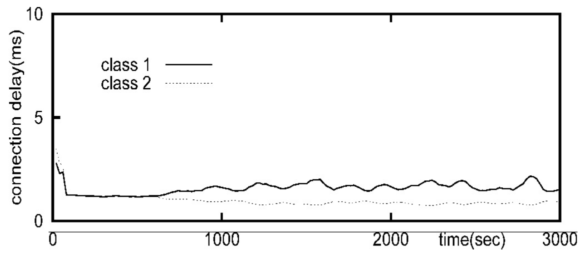
$$\begin{gathered}\Psi_1^{Static,25s}(e) = 0.2608, \Psi_1^{STFC,25s}(e) = 0.1870, \\ \frac{\Psi_1^{Static,25s}(e) - \Psi_1^{STFC,25s}}{\Psi_1^{STFC,25s}} = 39.5\% \end{gathered} \quad . \qquad \textbf{(14)}$$



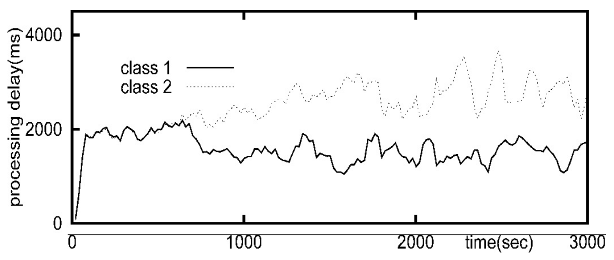
(a) The total delay and the delay ratio



(b) The bandwidth quota



(c) The processing delay

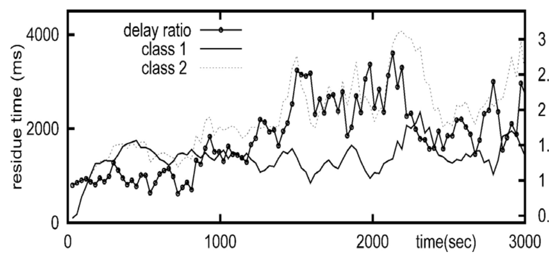


(d) The connection delay

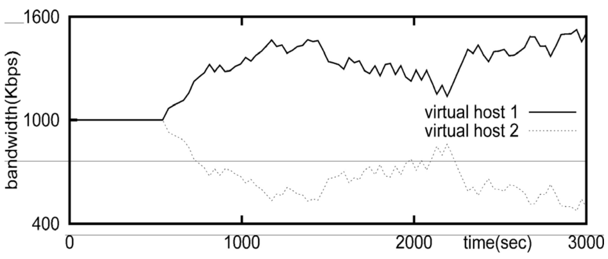
Fig. 11. Static fuzzy controller with the sampling time $T = 25s$

Experiment 3: Compare the control effect of STFC under different sampling time. Fig.12 shows the results for $T = 15s$. By comparing Fig.10 and Fig.12, it is found that both of them achieve the proportional delay guarantees, but the delays jitter of a large sampling time is smaller which can be described by the Equation (15). The possible reason is the delay jitter caused by large files is more apparent in small sampling time.

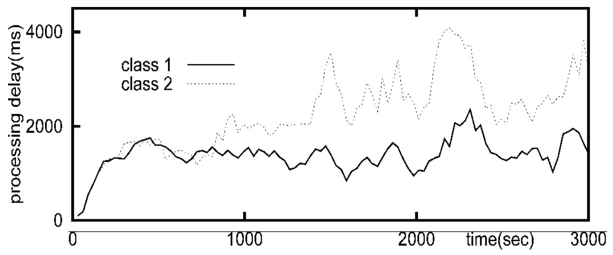
$$\begin{gathered}\Psi_1^{STFC,15s}(e) = 0.2362, \Psi_1^{STFC,25s}(e) = 0.1870 \\ \frac{\Psi_1^{STFC,15s} - \Psi_1^{STFC,25s}}{\Psi_1^{STFC,25s}} = 26.3\% \end{gathered} \quad . \qquad \textbf{(15)}$$



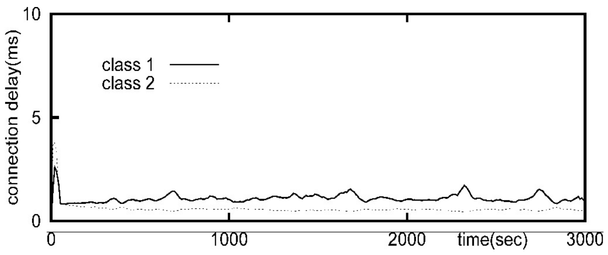
(a) The total delay and the delay ratio



(b) The bandwidth quota



(c) The processing delay



(d) The connection delay

Fig. 12. Self-tuning fuzzy controller with the sampling time $T = 15s$

6 Conclusion and future work

Finally, the authors conclude the paper and provide plan of future work. The threads / processes-based strategy reduces the connection delay by providing more threads / processes to the high priority which should enjoy a better service for the SLA. But when the bandwidth becomes the bottleneck, the processing delay plays a major role in the service quality. This paper first designs experiments to illustrate the defects of threads / processes-based strategy. Then a two level STFC is proposed: the first level is the BRC, which aims at the object uncertainty without the mathematical model. The second level is the SFC which compensates the processing delay jitter caused by huge files. Finally, the experiments show that the controller not only realizes the proportional delay guarantees, but also reduces the delay ratio jitter compared with the static fuzzy controller.

But we only talked about the situations of single resource being the bottleneck. For example, we deploy enough bandwidth for the files transmitting when the threads / processes strategy is adopted. Vice versa, there are sufficient threads / processes in the bandwidth limited condition. So in the future, we will focus on the limitation of mix resources. We will also study how to combine the two strategies to fit different internet conditions.

Acknowledgement

This work was supported by the Young Scientists Fund of the National Natural Science Foundation of China (Grant Nos. 61203233) and the China Postdoctoral Science Foundation (Grant No. 2013M540772).

References

[1] Tsai, T. T.-H., Lin, A. J., & Chin, K-Y. (2012). Study on web users loyalty effected by different computer expertise levels, *Journal of Computers, 23*(1), 25-34.

[2] Gao, A., Mu, D. & Hu, Y. (2012). The research of differentiated service and load balancing in web cluster. *International Journal of Computers, Communications & Control, 7*(4), 660-672.

[3] Zhou, X., Wei, J., & Xu, C.-Z. (2006). Resource allocation for session-based two-dimensional service differentiation on e-commerce servers. *IEEE Transactions on Parallel and Distributed Systems, 17*(8), 838-850.

[4] Lama, P., & Zhou, X. (2009, July). *Efficient server provisioning with delay guarantee on multi-tier clusters*. Paper presented at the 2009 17th international workshop on quality of service, Charleston, SC.

[5] Chan, K. H., & Chu, X. (2008, May). *Design of a cluster-based web server with proportional connection delay guarantee*. Paper presented at the IEEE International Conference on Communications, Beijing, China.

[6] Lu, C., Abdelzaher, T. F., Stankovic, J., & Son, S.-H. (2001). A feedback control approach for guaranteeing relative delays in web servers. In A. D. Williams (Ed.), *IEEE real-time technology and applications symposium* (pp. 51-62). Piscataway, NJ: IEEE.

[7] Wei, J., & Xu, C. (2006). eQoS: Provisioning of client-perceived end-to-end Qos guarantees in web servers. *IEEE Transactions on Computers, 55*(12), 1543-1566.

[8] Liu, X., Heo, J., Sha, L., & Zhu, X. (2006, April). *Adaptive control of multi-tiered web application using queuing predictor*. Paper presented at the Proceedings of the 10th IEEE / IFIP network operations and management symposium, Vancouver, CA.

[9] Lu, Y., Abdelzaher, T., Lu, C., Sha, L., & Liu, X. (2003, May). *Feedback control with queuing theoretic prediction for relative delay guarantees in web servers*. Paper presented at the 9th IEEE Real-Time and Embedded Technology and Applications Symposium, Washington, DC.

[10] Zhou, X., Cai, Y., & Chow, E. (2006). An integrated approach with feedback control for robust web QoS design. *Computer Communications, 29*(16), 3158-3169.

[11] Wei, J., Zhou, X., & Xu, C. (2005). Robust processing rate allocation for proportional slowdown differentiation on internet servers. *IEEE Transactions on Computers, 54*(8), 964-977.

[12] Hu, Y.-S., Mu, D.-J., Gao, A., & Dai, G.-Z. (2001). The research of QoS approach in web servers. *International Journal of Computers Communications & Control, 6*(4), 636-647.

[13] Barford, P., & Crovella, M. (1998, June). *Generating representative web workloads for network and server performance evaluation*. Paper presented at the Proceedings of the 1998 ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems, Madison, WI.

[14] Abdelzaher, T. F., & Bhatti, N. (1999). Web server QoS management by adaptive content delivery. In *7th international workshop on quality of service* (pp. 216-225). Piscataway, NJ: IEEE.

[15] Xue, P.-Y., Zhang, X.-Y., Bai, J., & Jiao, S.-J. (2013). Research of disk migration technology for virtual machine in cloud computing. *Journal of Computers, 24*(3), 3-10.

[16] Lluch-Lafuente, A., & Montanari, U. (2005). Quantitative μ-calculus and CTL defined over constraint semirings. *Theoretical Computer Science, 346*(1), 135-160.

[17] Li, Z., Wang, W., & Jiang, Y. (2009). Managing quality-of-control and requirement-of-bandwidth in networked control systems via fuzzy bandwidth scheduling. *International Journal of Control, Automation and Systems*, 7(2), 289-296.

[18] Juttner, A., Szabo, I., & Szentesi, A. (2003). On bandwidth effciency of the hose resource management model in virtual private networks. *IEEE Infocom, 1*(1), 386-395.

[19] Wei, J., & Xu, C. Z. (2005). A self-tuning fuzzy control approach for end-to-end Qos guarantees in web servers. In H de Meer & N. Bhatti (Eds.), *Quality of Service—IWQoS 2005* (pp. 123-135). Berlin, Germany: Springer Berlin Heidelberg.

[20] Mudi, R. K., & Pal, N. R. (1999). A robust self-tuning scheme for PI-and PD-type fuzzy controllers. *IEEE Transactions on Fuzzy Systems, 7*(1), 2-16.