Controlling the Response Time of Web Servers

Mohamed Ghazy Shehata[1], **Navid Mohaghegh**[2], **and Mokhtar Aboelaze**[2]
[1]Department of Electrical Engineering, Effat University, Jeddah, Saudi Arabia
[2]Department of Computer Science and Engineering, York University, Toronto, ON, Canada

Abstract - *Internet server provisioning is a very challenging problem for content providers and large server farms. In this paper we investigate the control theoretic approaches of managing servers in order to satisfy a required Quality of Service (QoS). We compare between two widely used models in the literature and our proposed simple technique. First model is a queueing based M/G/1 model with a PI controller. The second technique represents the server as a second order system and estimates the system parameters on line every sampling period. The estimated parameters are used in the design of a first order filter/controller in order to track the required QoS. Finally we present a simple technique based on the Additive Increase Multiplicative Decreases used in TCP congestion avoidance. We use simulation to compare these three techniques. Surprisingly, the AIMD performs the best among these three and it requires the least computation overhead among the three.*

Keywords: Quality of Service, web server provisioning, response time control, performance guarantees.

1 Introduction

The Internet is growing with an unprecedented rate and is infiltrating every aspect of our lives. E-commerce sites, mail servers, file servers, content servers, and search engines are few examples of applications that we use almost every day of our lives. It is difficult to imagine our lives without the *heavy* use of the Internet.

These sites are powered by powerful servers (or in many cases a large server farms where hundreds and may be thousands of servers are used) that receive users' requests, process them and send back the response. One of the major problems facing providers is how to condition the servers in order to produce the agreed-upon quality of service (QoS) and at the same time minimize their cost.

The Qos is either an agreed-upon contract between the servers owners and the content provider that must be maintained by the server owners, or a generally accepted criterion that is enforced by the server owners in order not to drive clients away. The consumers are known to be impatient, if the response is not within a specific period of time (that period varies greatly according to the application) the customer will probably terminate the session and navigate away to another site (the just-a-click-away syndrome).

Throwing hardware at the problem (also known as over provisioning) is not the optimal solution. Designing the system to work at the peak capacity wastes a lot of resources that most of the time would be unused. What is required is a policy that achieves the required QoS without wasting a lot of hardware. That is usually achieved by using admission control, where requests will be turned down if accepting the request results in a longer response time than what is required in the QoS agreement. If the system is overloaded, accepting a request not only means that this request will suffer more than usual response time, but also it means that all requests arriving after that request will suffer a longer than normal response time. By turning down one request, we lost one request but the following ones will be served according to the required QoS agreement.

Recently, there have been a lot of studies that suggests the use of classical feedback control theory in order to control access to the server and maintain the required response time. The argument is just as in the case of a controller controlling the gas rate going into a furnace in order to maintain the output (temperature) at a specific level, a controller to control the request rate delivered to the server can maintain the required output (response time) at a specific level.

However, the main differences between systems where classical control showed a lot of promise and internet servers are:

- Systems where classical control is very promising are very well understood; usually its behavior is governed by differential equations (difference equations in case of discrete systems). This lends itself very nicely to classical control theory. Where controllers are designed in the continuous time (discrete time) case using Laplace (Z) transform.
- Servers work in a highly unpredictable environment with probabilistic inputs (at best) and a lot of randomness in both arrival pattern and service time. Queuing theory has been successfully used to describe such a system. However almost all queuing theory results are based on a stable system and are valid at the steady state (average).

In this paper, we study the problem controlling the arrival rate in order to maintain the required QoS. We use 2 methods from classical control theory and one method that has proved itself to be successful in controlling congestion in TCP/IP traffic. We use simulation to compare these three methods.

The remainder of the paper is organized as follows: Section II describes our motivation and surveys previous work in this area. Section III describes the setting and the proposed solutions. Section IV shows the result of our work and compares it with previous solutions. Section V concludes the paper and describes our future work.

2 Motivation and Related Work

2.1 Motivation

Our motivation is to control the response time of an Internet server. Usually, and specifically in E-commerce applications the service is structured as 3-tier server. The first tier deals with static contents. The server gets a request to send a specific page, and it responds by sending the page. The second tier deals with dynamic contents. Requests arrive to the second tier server that fetches and calculate dynamic contents and sends it back. Third tier deals with database accesses.

Although 3-tier architecture is quite common in E-commerce applications, in this paper Our motivation is to control the response time of an Internet server. Usually, and specifically in E-commerce applications the service is structured as 3-tier server. The first tier deals with static contents. The server gets a request to send a specific page, and it responds by sending the page. The second tier deals with dynamic contents. Requests arrive to the second tier server that fetches and calculate dynamic contents and sends it back. Third tier deals with database accesses.

Although 3-tier architecture is quite common in E-commerce applications, in this paper we deal with single tier services only. The reason for that is we are concentrating on comparing the different approaches of admission control. Currently we are in the process of building a low power server with a 3-tier architecture. Once this system is built, we will test it using the three approaches mentioned here. Our objective is to implement a low power server that could achieve the same performance as bigger, more powerful, and more power hungry servers.

2.2 Previous Work

A lot of work is done in improving the performance of web servers and achieving a specific QoS. Earlier work in this area was mainly either service differentiation [3] or using data prefetching [4]. In service differentiation customers (requests) are treated differently giving a priority for one type of requests (more important customers) over the others. In prefetching, data we think will be requested soon is prefetched ahead of being actually requested. Both of these 2 techniques can improve the performance of the system (for only one group of requests in service differentiation case) but there are no guarantees that a specific level of performance is met.

Service differentiation is combined with admission control in [12]. They classified incoming requests into two categories, and admission control is based on the queue size of each category and some real time system measurements. They tested their system using Apache with static contents and some basic form of dynamic content.

A self tuning controller is proposed in [11]. They used a queuing model known as processor sharing model and a proportional integral (PI) controller to satisfy a target response time. Their queuing mode is M/G/1 where the response time is given by the equation

$$T_{RT} = \frac{E[X]}{1-\lambda E[X]} \tag{1}$$

Where λ is the arrival rate (assumed to be Poisson arrival) and E [X] is the Expected value of the service time. They also linearize the model around the operating point. Equation (1) that describes the system is valid only in the steady state and for stable queues (by stable we mean average arrival rate is less than average service rate). For short time periods and in heavy traffic the arrival rate may be greater than the service rate. Although the objective of the controller is to avoid such a case, but when it happens the equation used to model the system is not valid anymore.

The authors in [13] proposed an admission control to control the response time of the server. In their model they used Eq. (1) to represent the system. They also proposed an adaptive control scheme where the model parameters are estimated on line (using RLS technique) and is used to modify the controller parameters [2]. While in [16] the authors proposed an adaptive architecture that performs admission control. Their technique depends on using TCP SYN policer and an HTTP header-based connection control in order to maintain the required response time limit.

Malarait et al in [14] proposed a nonlinear continuous time model using fluid approximation for the server. They used this model to obtain an optimal configuration of the server in order to achieve maximum availability with performance constraints and maximum performance with availability constraints. They also validated their design using TPC-C benchmark.

Elnikety et al in [7] proposed a proxy called Gatekeeper to perform admission control as well as user-level request scheduling. Their idea depends on estimating the cost of the request, then deciding if admitting that request will exceed the system capacity or not (system capacity is determined by using offline profiling). they noted that since the proxy is external to the server, no server modification is required for their gatekeeper.

Guitart in [8] proposed a session based admission control for secure environment. In their technique, they gave preference for connections that could use existing SSL connections on the server. They also estimated the service time for incoming requests in order to prevent overloading the server.

Blanquer et al [5] proposed a software solution for QoS provisioning for large scale Internet servers. They proposed the use of traffic shaping and admission control together with monitoring response time and weighted fair queuing in order to guarantee the required QoS. For an excellent review of performance management for internet applications, the reader is referred to [9].

Almost everyone who used Control theory used the average response time as the parameter to control. One major problem with that is the QoS requested is not on the form of average response time or average delay. The QoS is usually on the form x% of requests have a response time better than y msec. Guarantees to average response time do not solve this problem.

In this paper we investigate this problem. We compare between using average response time and the actual QoS percentage of the requests that satisfied the required response time guarantees. We also present a simpler technique that does require much less overhead compared with control theoretic approaches and produces better results in our simulation.

3 System Setup

As we mentioned before, most of the work done using control theoretic approach to control quality of service considered the average response time as the parameter to control. The problem of that approach is that most of the required QoS is not about the average response time [11]. Usually, the QoS requirement is described as 90% of the requests face a processing delay of not more than 150 milliseconds. Controlling the average response time will not lead us to a specific condition such as the one described above.

The reason for that is the Internet traffic is highly volatile and unpredictable. Classic queuing theory deals with such scenarios if we know the distribution of the incoming traffic and service time (or at least know some parameters about the underlying distribution such as the average and the standard deviation). For example consider the simplest type of queues known as M/M/1. The cumulative probability distribution of the response time is [10].

$$F(\tau) = 1 - e^{-\tau\mu(1-\rho)} \qquad (2)$$

Where, μ is the service rate and ρ is the server utilization. Since the average time spent in the system for M/M/1 is 1/(μ(1-ρ)) = 1/(μ-λ), where λ is the arrival rate. By simple substitution of τ to be the average time spent in the system, we find that the probability that any customer meets a response time more than the average to be 36.8%.

Another problem with using control theoretic approach is how to handle the overhead in calculating and adjusting the system and the controller parameters. Consider for example admission control. The system output should be monitored to collect statistics about the parameter to be controlled. Every sampling period, the collected data are used in order to calculate the admission probability and modulate the arrival with this probability to meet the required service performance measure. The major question here is how should we select the sampling period?

The requests arrivals to a server are usually in the milliseconds range, or even less for very powerful servers. Now, what should be the sampling period? If we consider the sampling period to be on order of seconds, that is good from the overhead point of view. After all we do not want to overload the server with control calculation since that time is taken from serving incoming requests. However, since the web traffic is highly volatile and unpredictable, what happened few seconds ago might not have an impact on the current operation of the system. For examples 3-5 seconds ago we received a rush of requests that resulted in prolonging the response time and not meeting the required QoS, Now we reduce the admission probability in order to slow down the arrival, but there is very little arrival now. That leads to wasting CPU cycles because of the time difference between the sampling rate and traffic variations.

The second choice is taking the sampling time in the order of milliseconds. Although the response will be much faster than the previous case, however that is too much overhead for the CPU. Even the online recursive estimator in [15] requires a number of large matrix multiplications every sample period in order to estimate the system parameters.

In this paper, we present three different techniques for QoS guarantees in a web server. First we consider a simple M/G/1 model similar to the one proposed in [13]. This model predicts the response time, so the only controllable parameter here is the average response time. Then we consider a general model for the system. We assume a second order model and we estimate the system parameters on line. Then a first order controller/filter is used to control the average response time. Our third model is a variation of the previous one where the parameter to control is the percentage of the requests that failed the QoS requirements. Finally we consider a fourth model where we used a simple variation of the Added

Increase Multiplicative Decrease AIMD that was successfully implemented in congestion avoidance for TCP/IP protocols [1].

3.1 Using PI Controller

This is the method proposed in [11]. Eq 1 is considered to represent the system where T_{RT} represents the response time. The schematic diagram of the controller is shown in Figure 1 where the server is represented by Eq. 1

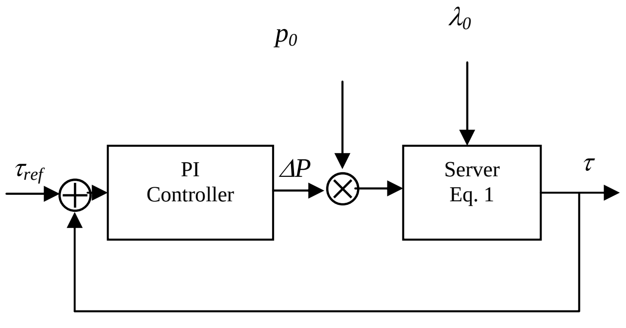

Fig. 1 Schematic diagram of the controller

In Fi.g 1 τ represents the response time, τ_{ref} is the required average response time, P_0 is the admission probability derived from Eq. 1 in order to make $\tau = \tau_{ref}$. λ_0 is the unmodulated arrival rate, and Δp is the correction produced by the controller to p in order to guarantee the required τref.

Linearizing Eq. 1 using Taylor series around the operating point λ_0 we can solve for the PI controller. The results of this scheme together with some comments about the scheme is discussed in the next section.

3.2 Estimating the System Parameters

Similar to [13] we assume no knowledge of the system under control (the server). By monitoring the input and output of the server we derive the model parameters. Here the system under control is the server with input λ_0 and output either the average response time or the percentage of the requests that confirm to the required QoS. We tried several models for the system and found out that the best fit is a second order system. In this part, we consider two solutions one that adjusts the average response time and one that adjusts the percentage of requests conforming to QoS.
In this case, we assume that the system output y (no matter what the output is, it could be response time, or the percentage of packets that missed the service time threshold) can be represented as a second degree system where the input u is the arrival rate as follows.

$$\frac{y}{u} = \frac{1 + a_1 Z^{-1} + a_2 Z^{-2}}{b_0 + b_1 Z^{-1} + b_2 Z^{-2}} \tag{3}$$

Where Z^{-1} is the delay operator. The parameters a_i and b_i are estimated on line by measuring the output y and the input u and averaging them over the sampling period. We use a well known recursive least square estimator [15].

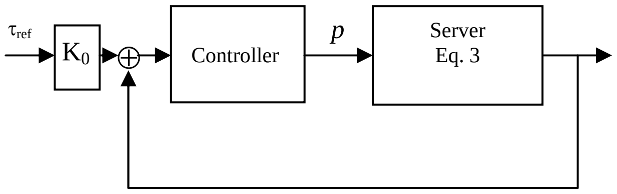

Fig. 2 Schematic diagram of the controller where we estimate the system parameters.

The controller was designed as a PI controller on the form

$$G_c = \frac{\beta_0 Z^{-1} + \beta_1}{\alpha_0 Z^{-1} + \alpha_1} \tag{4}$$

α_0, α_1, β_0, β_1, and K_0 are chosen in order to achieve a reasonable overshot, settling time within the sampling period and the proper tracking of the output for K0

3.3 Additive Increase Multiplicative Decrease

This idea came from the sliding window control in TCP [1]. In TCP the window size is decreased (by a multiplicative factor) if there is a lost packet and is increased (by an additive factor) for successful transmission of a packet [6].

The proposed scheme works as follows. If the queue size grows beyond a high threshold N_{high} the probability of accepting a new packet is multiplied by γ, where $0 < \gamma < 1$. If the queue size drops below N_{low}, the admission probability is increased by η, where $0<\eta<1$. The probability is bounded in the interval [0.1, 1] for practical purposes.

The obvious question is how to choose the values of N_{high}, N_{low},γ, and η, The proper choice of these parameters depend on the service time and inter-arrival time distributions. In our simulation, we tried different values for the parameters and found that the best choice for N_{high} is the target delay divided by the average service time, while $N_{low} = N_{high}/2$. We also found the best values for $\eta = 0.4$, and γ=0.8-0.9. Clearly the optimal values for these parameters depend on the arrival and service distribution and can be fine tuned online.

4 Results and Discussion

In this section we show the results of our simulation using Matlab for the four cases proposed in Section III.

For all the experiment we ran the simulation for 1600 seconds using Matlab. We collected the percentage of the requests accepted, the percentage of the requests that required less than 150 msec. and the average response time. For every experiment we considered two traffic scenarios.

- **Traffic A**: This is the baseline system, we assumed an average exponential interarrival time of 55 *msec*. and an average exponential service time of 35 *msec*. (64% utilization). The target response time is 150 *msec*.
- **Traffic B**: In this scenario, we start as in Traffic A. At the simulation midpoint (after 800 seconds) we increase the arrival rate by decreasing the interarrival time to 45 *msec*. (utilization of 78%). The objective here is to see how the controller reacts to increasing the arrival rate in order to satisfy the QoS requirements.

Fig 3 shows that response time for a 20 minutes simulated run under traffic B. We can see that after 800 msec. the average response time increases. The main function of any controller is to adjust the admitting probability in order to avoid such a scenario.

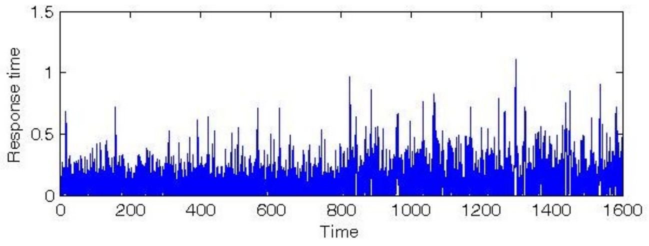

Figure 3. Response time without a PI controller under traffic B

Fig. 4 (same setting as Fig. 3) shows the response time and the acceptance probability as a function of time. It is obvious that the controller suppressed the input in the second half of the simulation leading to a more consistent (equal) response time. One thing that is very noticeable here is the rapid changes in the admitting probability compared to the other method we used. Although the probability changes very rapidly, the performance is the worse compared to the other techniques. One possible explaination this is that Eq. (1) does not describe the system when the traffic increases beyond stability even for a short period of time

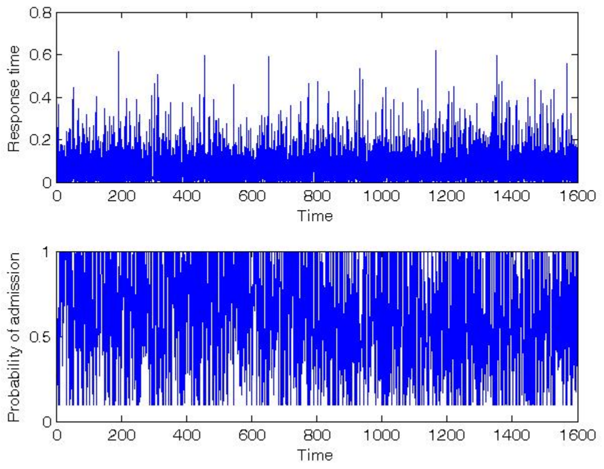

Fig. 4. Response time and admitting probability under traffic B for a PI controller

Then we consider our technique where we assume a second order system and estimate system parameters on line. Once the parameters are estimated on line, the parameters are used to choose the parameters of a first order controller/filter in order to track the required QoS criterion either directly by controlling the percentage of conforming packets, or indirectly through controlling the average response time.

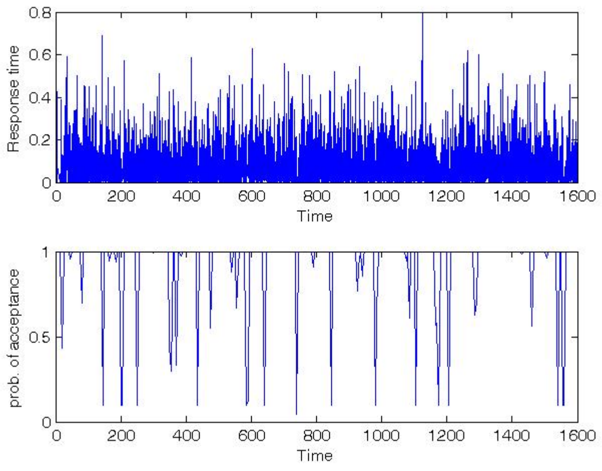

Fig 5. Response time and admitting probability assuming a second order system and a first order filter under traffic A (c0ntrolling averge response time).

Figure 5 shows the response time and admitting probability for our system under traffic A assuming a 2nd order system with on-line parameters estimation. While Fig. 6 shows the same system under traffic B scenario. The parameter to be controlled in this case is the average response time.

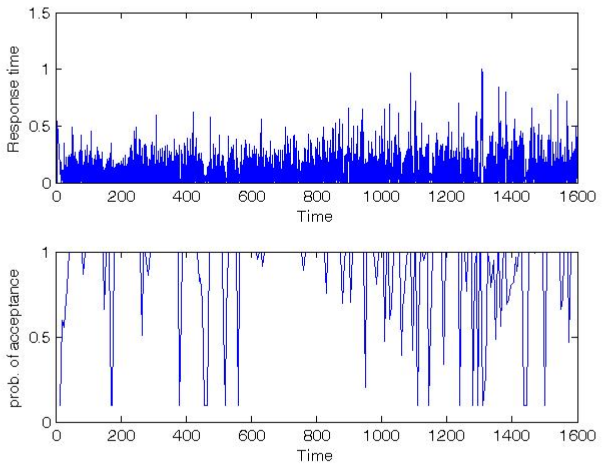

Fig 6. Response time and admitting probability assuming a second order system and a first order filter under traffic B (controlling average response time).

The changes in the admitting probability for this system is much less than the case of a PI controller. That is by itself is not an advantage (however it might give an indication that the system is stable and does not oscillate) but as we will see in Table 1, the performance here is much better than the PI controller case.

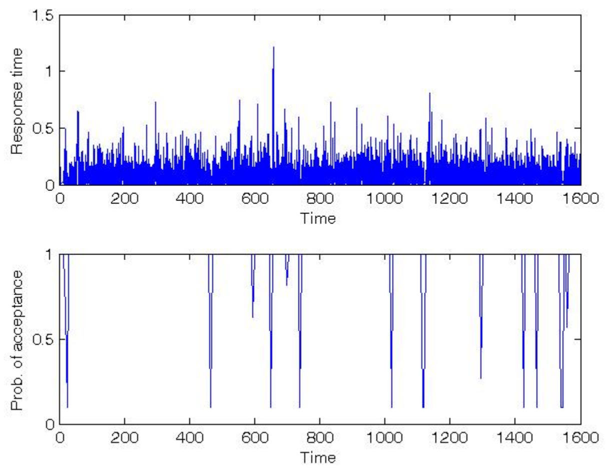

Fig 7. Response time and admitting probability assuming a second order system and a first order filter under traffic A (controlling QoS).

Figure 7 and 8 shows the same results as Figures 5 and 6 but in this case we use the percentage of the conforming requests as the parameter to control. Although it is difficult to see, directly from the Figures, which one is better in maintaining the required QoS, controlling the response time, or the percentage of conforming packets directly from the Figures, Table 1 shows that in fact controlling the percentage of conforming packets produces better results.

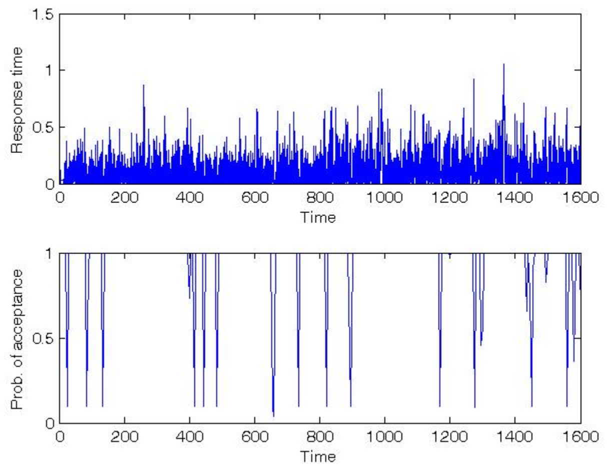

Fig 8. Response time and admitting probability assuming a second order system and a first order filter under traffic B (controlling QoS).

Figures 9 shows the system using AIMD with Nhigh $=T_{target}/\tau_{av}$, $N_{low} = N_{high}/2$. η =0.4, and γ=0.8. Where Ttarget is the target delay and is set to 150 msec. and τ_{av} is the average response time under traffic A. Figure 10 shows the same setting under traffic B. The admitting probability varies very quickly compared to Fig 5,6,7,8. However that is expected since the changes in the admitting probability is calculated every time the queue size grows beyond a specific threshold, or decreases below another threshold. However as we will see in Table 1, this is the best performance among the three techniques.

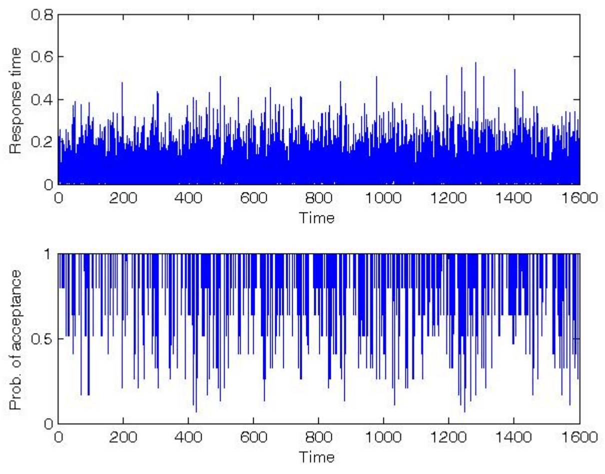

Fig 9 Response time and admitting probability using Additive Increase Multiplicative Decrease AIMD under traffic A

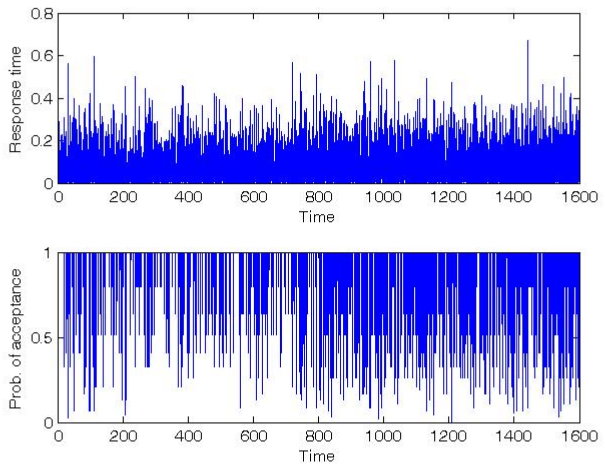

Fig 10 Response time and admitting probability using Additive Increase Multiplicative Decrease AIMD under traffic B

We summarize the results in Table 1. The first column shows the 4 different techniques we used. The second column shows for every technique the results under traffic A and traffic B.

The actual results are shown in columns 3, 4, and 5. Column 3 shows the percentage of admitted requests. Column 4 shows the percentage of the requests that is conforming to the required QoS. The first number shows the percentage of conforming requests to all arrived requests, while the number in parenthesis shows the percentage of conforming requests to admitted requests only. Finally column 5 shows the average response time for all admitted packets.

From Table 1 we can also see that AIMD has the highest admitting policy under traffic A (97%), while PI has the lowest (61%). It also shows that AMD has the highest conforming percentage under traffic A. Under traffic B assuming a 2nd order system with online parameters estimation has slightly higher admitting probability than AIMD (94% vs. 93%), however the percentage of conforming requests is much higher for AIMD (compared to all arriving requests or admitted requests). basically that states that the AIMD rejects a very small percentage of incoming requests, but it rejects the tight ones.

TABLE 1 COMPARISON BETWEEN THE FOUR RPOPOSED METHODS

Technique		% accepted	% conforming	Av. delay msec.
PI	A	61%	53%(86%)	74
	B	57%	48%(84%)	80
Est.(t_{resp})	A	92%	73%(79%)	90
	B	88%	63%(72%)	119
Est. (T_{resp})	A	95%	75%(79%)	95
	B	94%	66%(70%)	125
AIMD	A	97%	83%(86%)	78
	B	93%	77%(83%)	85

5 Conclusions

In this paper we investigated 3 different techniques for controlling admitting probability in an internet server in order to conform to a required QoS. Using simulation we show that a simple AIMD technique outperforms more complicated control-theoretic approaches, and it requires much less overhead compared to the control-theoretic approaches.

For future work, we are building a low power server using small embedded microprocessors. We will be testing these proposed methods under realistic traffic when the server is up and running

6 References

[1] M. Allman, V. Paxson, and W. Stevens "TCP Congestion Control" RFC2581 April 1999. IETF. Available at *tools.ietf.org/html/rfc2581* Checked March 2011.

[2] K. Astrom, and B. Wittenmark "*Adaptive control*" 2nd Edition. Prentice-Hall 1994.

[3] J. Almeida, M. Dabu, A. Manikutty, and P. Cao G. O. "Providing differentiated levels of service in web content hosting". *Workshop on Internet Server Performance.* Madison, WI June 1998. pp 92-101

[4] M. Banatre, V. Issamy, F. leleu, and B. Charpiot. "Providing quality of service over the Web: A newspaper-based approach". *Proc. of the 6th International World Wide Web Conference*. April 1997.

[5] J. Blanquer, A. Batchelli, K. Schauser, and R. Wolski. Quorum: Flexible quality of service for internet *services. Second Symposium on Networked Systems Design and Implementation (NSDI'05)*, Boston, MA, U.S.A., 2–4 May 2005; 159–174.

[6] D. Comer *Internetworking with TCP/IP* 5th Edition. Prentice-Hall Upper Saddle, N.J. 2006

[7] S. Elnikety, E. Nahum, J. Tracey, and W. Zwaenepoel . " Method for transparent admission control and request scheduling in e-commerce web sites". *Proc. of the 13th International Conference on World Wide*

[8] J. Guitart, D. Carrera, V. Beltran, J. Torres, E. Ayguade . "Designing an overload control strategy for secure e-commerce applications". *Computer Networks* 2007; **51**(15):4492–4510.

[9] J. Guitart, J. Torres, and E. Ayguade. "A survey on performance management for Internet applications". Concurrency and Computation: Practice and Experience. Vol-22 No. 1. 2010 pp 68-106.

[10] R. Jain. *The art of computer system performance analysis: techniques for experimental design, measurements, simulation and modeling*. Wiley Interscience. 1991.

[11] A. Kamra, V. Misra, and E. Nahum "Yaksha: A self tuning controller for managing the performance of 3-tiered web sites". *Proc. of the 12th IEEE International Workshop on Quality of Service IWQOS.* June 2004. pp 47-56.

[12] K. Li, and S. Jamin. "A measurement-based admission control web server". *Proc. of IEEE Infocom*. March 2000.

[13] X. Liu, J. Heo, L. Sha, and X. Zhu "adaptive control of multi-tiered web application using queueing predictor". *Proc. of 10th IEEE/IFIP Network Operations and Management Symposium (NOMS)* 2006

[14] L. Malrait, S. Bouchenak, and N. Marchand "Experience with ConSer: a system for server control through fluid modeling". *IEEE Transactions on Computers* Vol ? No. ? 2010

[15] P. Paraskevopoulos *Digital control system* Prentice-Hall 1996.

[16] T. Voigt, and P. Gunningberg. "Adaptive resource-based web server admission control". *Proc. of the 7th International Symposium on Computers and Communication*. 2002