Response Time Control for Web Servers Under Realistic Traffic Patterns

Mokhtar Aboelaze[1] Navid Mohaghegh
Dept. of Computer Science and Engineering
York University
Toronto, ON CANADA

Mohamed Ghazy Shehata
Department of Electrical Engineering
Effat University
Jeddah, Saudi Arabia

Abstract— **Service provisioning for Internet servers, as well as many other servers, plays a very important role in the cost and performance of the server. Service/content providers are required to maintain a certain Quality of Service (QoS) either contractually or implicitly in order not to lose customers. In this paper we investigate the control theoretic approaches for service provisioning. We compare between 2 widely used approaches and a simple technique we propose for admission control in order to maintain the agreed upon QoS. We show using simulation that our proposed technique is more stable, needs less overhead, and produce better results than these 2 previously proposed techniques.**

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

I. INTRODUCTION

Service providers depend on large powerful servers in server farms to provide services to clients. Service provides are required to maintain a certain quality of service. Usually, the QoS is expressed as *x*% of the requests are completed within *y* seconds. This QoS could be implemented as a contract between the service provider and its clients, or could be implied in order to keep clients from moving to another site because of late response. The service providers need to implement and enforce a policy in order to guarantee the QoS and at the same time minimize their cost. The cost is not only the hardware required, but also power consumption which plays a major role in their cost model.

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 QoS violation. 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 this request, we lost one request but the following ones will be served according to the required QoS agreement. Another technique that is related to admission control is adding or waking up servers. In that context, instead of rejecting the request, a new server is added to the pool, or a server is awaken from the sleep state. The objective is to maintain the agreed upon QoS with minimum resources (minimum number of active servers). However, we will not cover these issues in this paper.

Recently, there have been many studies that suggest the use of classical feedback control theory in order to control access to the server and to 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). These systems lend itself very nicely to classical control theory. Where controllers are designed in the continuous time (discrete time) domain 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 case of sudden overload, the system response time is not represented by the simple queueing theory based formula that is used to derive the controller for a stable queue. In that case the controller is using a formula that is not valid anymore. However, that is exactly the time when we need the controller to stabilize the system. That partially explains that queueing theory based results are not widely used in practice.

In this paper, we propose a simple and low overhead admission strategy for web servers in order to maintain a required average response time. Our admission technique is simple and requires very low overhead. We compare our technique with 2 previously proposed queueing theory based admission techniques. We simulate our technique under 2

978-1-4673-5157-7/13/$31.00 © 2013 IEEE

different scenarios. First we consider a simple exponential distribution for incoming jobs service time, then we use a more realistic model with a long tail distribution (lognormal). Our simulation results show that our proposed technique is more stable, require less overhead, and produce better results than the other 2 queueing theoretic based techniques.

The remainder of the paper is organized as follows: Section 2 describes our motivation and surveys some of the previous work in this area. Section 3 describes the traffic patterns we used in our simulation. Section 4 describes the setting and the proposed solutions. Section 5 shows the result of our work and compares it with previous solutions. Then we end the paper with a conclusion of our work and our future work.

II. MOTIVATION AND RELATED WORK

A. Motivation

Our main concern here is to propose an admission based technique to satisfy the required QoS using minimum resources. Our objective is two folded. First we want a minimum overhead. it is not practical for a busy server to spend a lot of time to run the admission control protocol, after all that is time that could have been used to serve customers. The second objective is good performance. All admission based control depends on rejecting incoming requests at time of overload. The difference between 2 protocols is, are they rejecting the right request? By the right requests we mean the requests that have the most impact on relieving the overload.

B. Related Work

A self-tuning controller is proposed in [10]. 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

$$T_{RT} = \frac{E[X]}{1 - \lambda E[X]} \qquad (1)$$

Where λ is the arrival rate (assumed to be Poisson arrival) in requests per second, and E [X] is the Expected value of the service time in seconds. 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.

Lu et al in [13] used queueing predictor and feedback control in order to guarantees relative delay. Relative delay is the difference in delay experienced by different classes of requests. In their model they assume an M/M/1 system. The relative delay guarantees are achieved by allocating server resources among the different classes. In [5], the authors proposed an admission control to guarantee the required QoS by web servers. They estimated the resource requirements of the incoming request and then reject requests that would exceed server capacity.

Zhang et al in [19] used a regression based approximation of the CPU demands of the incoming client transactions in multi-tier systems. They used this approximation to model the system as a network of queues and used capacity planning in order to improve the performance of the system.

The authors in [11] considered a system where the incoming traffic is either high priority or low priority jobs. The high priority jobs are assigned a share of the processing power such that the average response time is the required response time with low variance; the rest is assigned to the low priority jobs. They used a combination of feed forward and feedback control to adjust the share of high priority jobs of the system resources. However, they assumed a Poisson arrival and exponential service time distribution which is not practical for real servers.

Amani, Kihl, and Robertsson in [2] proposed a nonlinear autoregressive neural network as a response time predictor. Their results show that the system can reasonably predict response time. Also, they used in their simulation an M/M/1 system with Poisson arrival and exponential service time distribution. As we explain in section 3, this is not a realistic assumption for a real server.

The authors in [12] proposed another admission control based technique to control the response time of the server. In their model they also used Eq. (1) to represent the system. They also proposed an adaptive control scheme where the model parameters are estimated on line and is used to modify the controller parameters. They used a recursive least squares technique [3] in order to estimate the model parameters at run time.

In [18] 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.

Guitart in [8] proposed a session based admission control for secure environment. In their technique, they gave preference for requests that could use the 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 [4] 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]

III. TRAFIC PATTERNS

Most of the previous work considered simple traffic patterns. The job interarrival time (time between 2 consecutive arrivals) is considered to be exponentially distributed (also known as Poisson arrival). The service time is also considered exponentially distributed. The exponential distribution assumption of service time makes calculation tractable in case of representing the server as a queueing system; however actual service time requirements is more complicated than that as mentioned in [16].

The results from [6], [15] show that exponential distribution is a bad fit for actual web traffic. They also suggest that lognormal distribution and Weibull distribution are the best fit for web traffic. In this paper, we use both exponential distribution and lognormal distribution to test our proposed technique. The reason for including exponential distribution is to show that some techniques works well for exponential distribution service time, but are not effective for long tail distributions.

In our simulation for both exponential and lognormal we assume an average page size of 30,297 bytes. For lognormal distribution the standard deviation is 184,749 bytes as was suggested in [15]. In section 5, we show the results of our simulation.

IV. SYSTEN SETUP

One of the major problems in 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 rate with this probability to meet the required service performance agreement. The major question here is how should we select the sampling period?

The requests arrivals to a server are usually in the milliseconds range (for powerful servers it could be in the microseconds range). 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. In this case, the server will respond fast, but results in a huge overhead. For example, the online recursive estimator in [17] requires a number of large matrix multiplications every sample period in order to estimate the system parameters. That puts a limit on the sampling time.

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 [12] and considering the response time only. Then we consider a general model for the system. We assume a second order model and we estimate the model parameters on line. Then a first order controller/filter is used to control the average response time. Finally we consider a third 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]. That technique is used to admit or turn down requests based on the response time.

A. PI Controller

This is the method proposed in [10]. Eq. 1 is assumed 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

In Fig 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 un-modulated arrival rate, and Δp is the correction produced by the controller to P_0 in order to guarantee the required τ_{ref}. Linearizing Eq. 1 using Taylor series around the operating point λ_0 we can solve for the PI controller parameters. The results of this scheme together with some comments about the scheme are discussed in the next section.

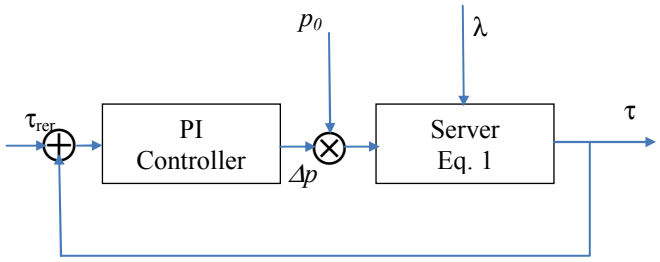


Figure 1. Schematic diagram of the PI controller

B. Estimating The System parameters

Similar to [12] 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 as the average response time. We tried several models for the system and found that the best fit is a second order system. In this case, we assume that the system output y can be represented as a

second order system where the input u is the arrival rate as follows.

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

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 [17]. Figure 3 shows the entire system (the estimator and the controller).

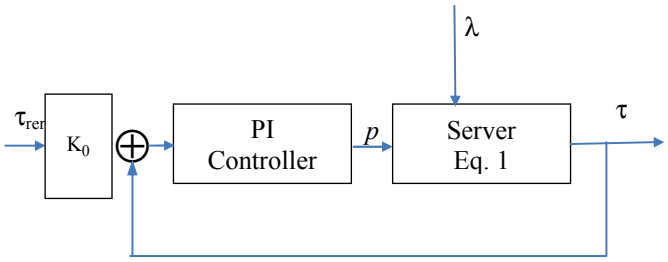


Figure 2. Schematic Diagram of the parameters estimation controller

The controller is designed as a PI controller on the form

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

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 K_0.

C. 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 [7].

One of the major problems with the two previous techniques is the sampling rate. To completely capture all the information you have to sample at a rate more than the maximum frequency in the signal. For a server we cannot sample at a rate equal to or more than twice the maximum frequency (in this case it is the packet arrival). That will lead to the server spending more time in calculating the admission probability than in actually serving the incoming requests.

By using the queue size at the server in order to control the admission probability the server can react very quickly to changes in the queue size and can prevent server overloading and by proper adjustment of the system parameters, we can reject packets that would have missed the target response time anyway and thus free server time to respond to other packets.

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 $\gamma = 0.8\text{-}0.9$. Clearly the optimal values for these parameters depend on the arrival and service distribution and can be fine-tuned online.

V. Results and Discussion

In this section we show the results of our simulation using Matlab for the three cases we consider. PI controller, predicting system parameters, and our proposed AIMD technique.

For all the experiment we ran the simulation for 3200 seconds. We considered 2 traffic patterns; the first is M/M/1 with exponential arriving time and exponential service rate. Then we consider a lognormal distribution for service time according to [15]. In both types, the average traffic was the same. The average request length 30,297 bytes, while the standard deviation for the lognormal distribution is 184,749 bytes [15]. The average request length was normalized to 0.0303 seconds of service time, while the sampling time is normalized to 2 seconds. The actual service time is of little importance since the determining factor is the ratio between request rate and service time (within a limit of course since the file transmission rate is a lower bound on the response time).

The interarrival rate varied from 0.035 to 0.2 seconds (arrival rate of 5 to 28.5 requests per second). That corresponds to a utilization of 15-85% which is the normal range of operating a server. The conforming packets are the percentage of the packets with a response time of 150 msec. or less.

Figures 3,4, and 5 show the probability of acceptance, the average response time, and the percentage of packets with response time less than or equal to the required 150 msec. under an exponential distribution of request size For a PI based controller, a second order system with parameters estimation/prediction, and our proposed AIMD technique respectively.

Figure 3 shows the acceptance probability as a function of arrival rate. From the Figure we see that the PI based technique starts dropping requests at a rate of 10 requests per second (utilization of 30%) and drops dramatically with increasing arrival rate. while the others maintain a reasonable acceptance rate. The response time of the PI (Figure 4) is almost identical to our proposed AIMD albeit at a much less acceptance rate.

Figure 5 shows that our proposed protocol has the highest percentage of requests within the required 150 msec. response time. The Figures also show that the parameter estimation/prediction technique have a reasonably high acceptance rate, however the average response time increases exponentially with increasing the system load (Figure 4)

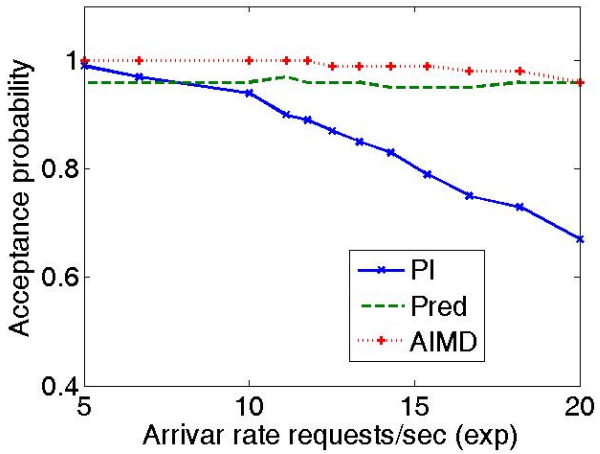


Figure 3. Acceptance probability for exponential request size

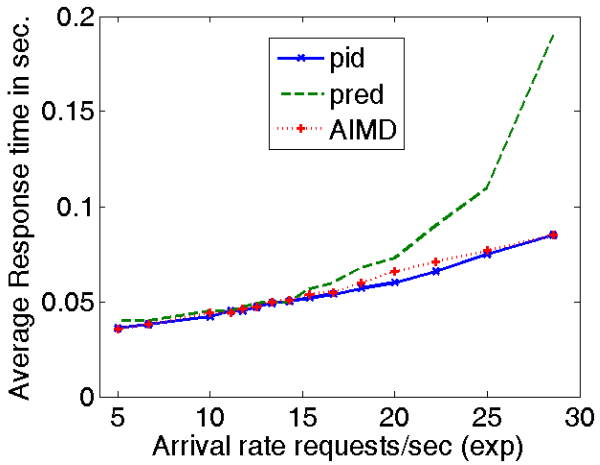


Figure 4. Response time for exponential request size

Another factor that is not clear from the simulation results is the overhead. AIMD have the minimum overhead, it rejects or accepts any request based on a probability generated from the queue length. The overhead in the prediction model is more severe. Every sampling period the protocol performs the prediction followed by the probability generation which requires matrix inversion and multiplication. That is a time we are not serving any requests in, and at the same time not producing good performance measures.

Then, we shift our attention to a more realistic traffic pattern. We use lognormal distribution [15] for request size. Figures 6, 7, and 8 show the same performance measures as Figures 3, 4, and 5. However, in this case we use a long tail (lognormal) distribution for request size.

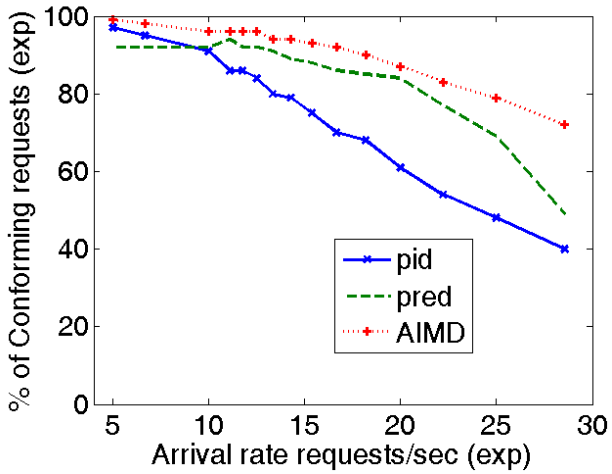


Figure 5. Percentage of conforming requests for exponential request size

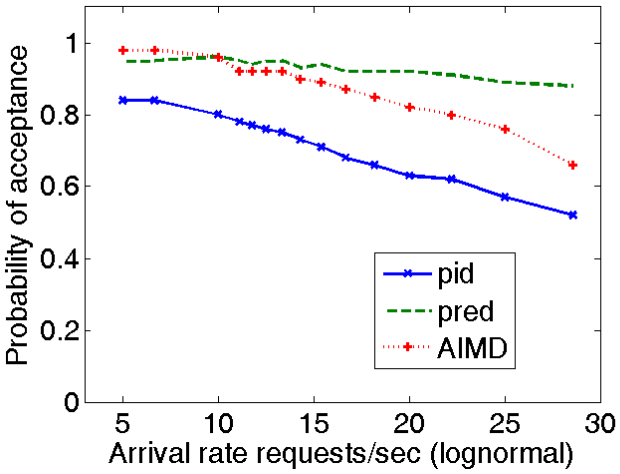


Figure 6. Acceptance probability for lognormal request size

From Figure 6 we see that the prediction method has the highest admission rate followed by our proposed AIMD and then the PI. However if we consider Figure 7, the prediction technique has the worst average response time. Basically the prediction technique admits everything but the average response time and the percentage of the conforming requests are very low. Also from Figure 7 the response time of the prediction techniques varies unpredictably with increasing the system load. Finally Figure 8 shows that under a wide range of utilization, the AIMD technique has the highest percentage of the conforming requests. Coupled with its simplicity and very low overhead, the AIMD outperforms the other two techniques by a very wide margin.

One thing that is worth mentioning here and is very clear in Figure 7, the prediction technique although it has the highest overhead behaves very erratically in medium and high utilization range. We repeated the simulation many times, and it always behaves very erratically outside of low utilization area. Changing the sampling period did very little for the stability. By taking a closer look at the simulation, at medium to high utilization, and especially with a long tail request size

distribution, the matrices are singular and its inversion result in a non-stable system.

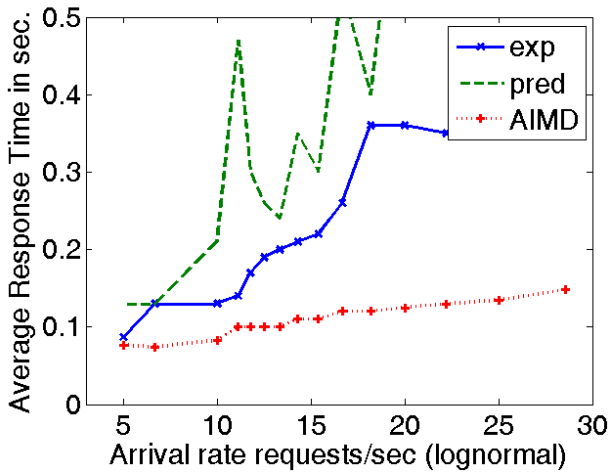
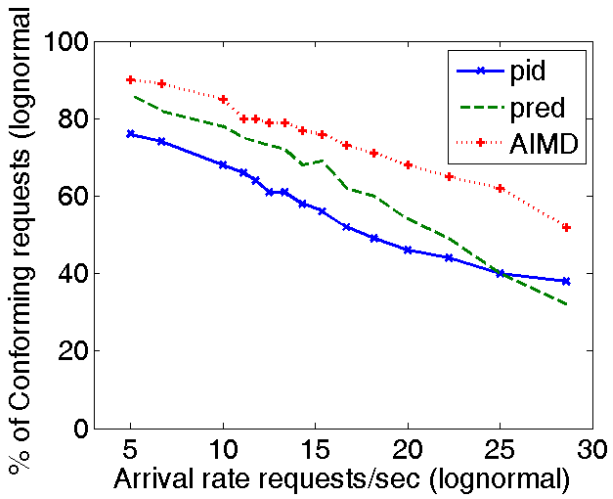


Figure 7. Average response time for lognormal request size



Percentage of conforming requests for lognormal request size

Interestingly enough, although AIMD has a reasonably high acceptance ratio, it also has the least average response time and the highest percentage of conforming requests. That means although AIMD rejects the least number of requests, however it rejects the right requests, thus resulting in an acceptable response time. While PID and prediction based protocols rejects more requests, and also result in a higher response time. The prediction method have a very good acceptance ratio, but at the cost of more response time.

CONCLUSION

In this paper we investigated 3 different techniques for controlling admission 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

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] P. Amani, M. Kihl, and A. Robertsson. "Multi-step ahead response time prediction for single server queueing system" Proc. of the IEEE Symp. on Computers and Communication (ISCC). 2011 pp 950-955

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

[4] 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.

[5] X. Chen, H. Chen, and P. Mohapatra "Aces: an efficient admission control for QoS-aware web servers", Computer Communication, vol. 26, no. 14 2003.

[6] H.-K Choi, and J. Limb "A behavioral model of the web". Proc. of the International Conference on Network Protocols. 1999, pp 327-334

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

[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] 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.

[11] M. Kjaer, M. Kihl, and A. Robertsson. "Response time control of a single server queue". Proc. of the IEEE Conference on Decision and Control. New Orleans, LA Dec. 12-14, 2007

[12] 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

[13] Y. Lu, T. Abdelzaher, C. Lu, L. Sha, and X. Liu "Feedback control with queueing-theoretic prediction for relative delay guarantees in web servers" in Proc. IEEE Real-Time and Embedded Technology and Application Symposium (RTAS'03), Toronto, Canada 2003, pp208-217

[14] L. Malrait, S. Bouchenak, and N. Marchand "Experience with ConSer: a system for server control through fluid modeling". IEEE Transactions on Computers Vol. 60 Issue 7 2010

[15] M. Molina, P. Castelli, and G. Foddis "Web traffic modeling exploiting TCP connections' temporal clustering through HTML-REDUCE". IEEE Network May 2000. pp 46-55.

[16] V. Pacson, and S. Floyed "Wide area traffic: the failure of the Poisson modeling". IEEE/ACM Transactions on Network. Vol. 3, issue 3 Jun 1995, pp 226-244.

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

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

[19] Q. Zhang, L. Cherkasova, and E. Smirnu, "A regression-based for dynamic resource provisioning of multi-tier application" in *Proc. of the 4th International Conference on Automatic Computing*. Washington, DC. 2007.