ISSN No. 0976-5697

Volume 3, No. 3, May-June 2012

International Journal of Advanced Research in Computer Science

TECHNICAL NOTE

Available Online at www.ijarcs.info

Active Queue Management Design Using Discrete-Event Control in Multiprocessor Environment

Shirish V. Pattalwar
Electronics and Telecommunication
Prof. Ram Meghe Institute of Tech. & Research, Badnera
Amravati, India
shirishpattalwar@rediffmail.com

Prof. Dr. Vilas M. Thakare
Deptt. Of Computer Science
SGB Amravati University, Amravati
Amravati, India
vilthakare@yahoo.co.in

Gajanan D Nagoshe
Electronics and Telecommunication
Prof. Ram Meghe Institute of Tech. & Research, Badnera
Amravati, India
name@xyz.com

Abstract: Recently, control-theoretic approaches in multiprocessor environment have been studied and employed to manage and control the performance of computing systems. Most of the existing control-theoretic approaches model computing systems as linear systems and apply feedback control. In this paper, we show discrete event modelling and control techniques can be effectively applied to performance management and control of computing systems. We use Active Queue Management design for multiprocessor environment. By modelling the logical processor as a queuing system, we formulate the problem of designing the optimal dropping strategy as an optimal queuing control problem under discrete-event control framework.

Keywords: QoS, AQM, LP, ECN

I. INTRODUCTION

Performance management has been a core research area in computer science. As computing systems become more pervasive and increasingly complex, managing computing systems by human is becoming more and more unfeasible. The demand to manage and control computing systems has grown rapidly. Traditional practices of automated resource management and control largely rely on ad-hoc techniques. As a result, changes in workloads and configurations often result in poor quality of service (QoS) or even instabilities. Recently, researchers discover AQM that feedback control schemes can be successfully used in analyzing and designing run-time IT systems [1], [2], [3]. In order to apply feedback control framework, nearly all previous work use linear models (or linearized models) to represent the underlying computing systems. However, computing systems are usually nonlinear [4] with respect to the resource allocated. In addition, workload to computing systems are usually stochastic; its parameters may change over a wide range of values. Most computing systems are discrete in nature. During the last several decades, research has shown that many computing systems can be modeled well as discrete-event systems (such as automata, petri nets, or queuing systems). [5] [6]. In computer system research, however, discrete-event models are usually used for off-line capacity planning purposes instead of online performance tuning purposes. In this paper, we explore the applicability of discrete event control to computing system applications. Specifically, we investigate the design of optimal dropping strategies for processor in multiprocessor environment by focusing on a branch of discrete event control queuing control. In our approach, we model an each processor as a single station queue, and formulate the problem of designing the optimal computing instruction strategy as an optimal queuing control problem. We then derive the optimal control strategy through uniformization and value iteration. Our solution gives the optimal computing strategy.

Based on the controller synthesis, optimal computing strategies and its parameters are given adaptively in response to different workloads. Hence it can be used in designing self-configuring. Active Queue Management (AQM) schemes. Our work opens a new perspective for studying the active queue management policies through queuing control. Since many computing systems can be modeled as discrete event systems, we believe that discrete-event control approach can be successfully applied to performance management of many computing systems.

II. SYNCRONIZATION ALGORITHMS

In a discrete-event simulation, events need be processed in a non decreasing time stamp order, because an event with a smaller timestamp has the potential to modify the state of the system and thereby affect events that happen later. This is what we call the causality constraint.

Provided that simultaneous events event with the same time stamps are sorted deterministically and consistently using certain tie-breaking rules, the causality constraint implies a total ordering of events. In parallel simulation, the global event-list in sequential simulation is replaced by a set of event-

CONFERENCE PAPER
National Conference on Information and Communication Technology for Development
Organized by PRMITR, Amravati (MS) India
http://mitra.ac.in/forthcoming.html

lists; each logical Processor (LP) maintains own simulation clock and a separate event-list that contains events that can only affect the state of the corresponding LP. Since each LP processes events on its own event-list in timestamp ordering a property also known as the local causality constraint the total ordering maintained by the original sequential discrete-event model is replaced by a partial ordering similar to Lamport's "happens before" relationship [29]. The fundamental challenge is therefore associated with the difficulty of preserving the local causality constraint at each LP without the use of a global simulation clock.

III. ACTIVE QUEUE MANAGEMENT

Recent measurements have shown that the growing demand for high speed data computation has driven parallel computing up exponentially. It is important to achieve low data loss and delay, and optimum utilizations of each computational unit. Active Queue Management (AQM) policies are intended to help achieving both optimum utilizations and low delays. The basic idea behind AQM queue management scheme is to detect incipient congestion in instruction handling with delay early and to convey congestion notification to the event handler processor. Hence event handler processor will AQM use their transmission rates before queues in the network overflow and delay occur AQM. To do this, AQM maintains an exponentially-weighted moving average of the queue length which it uses to detect congestion. When the average queue length exceeds a minimum threshold, instruction are randomly dropped or marked with an explicit congestion notification (ECN) bit. When the average queue length exceeds a maximum threshold, all instruction are dropped or marked. While AQM is certainly an improvement over traditional drop-tail queues, the performance of the AQM algorithm depends significantly upon the setting of each of its parameters, which was shown to be a not easy task [22].

IV. PROBLEM STATEMENT AND FORMULATION

We assume the event handler processor enforces instruction-dropping-based. AQM scheme are used to achieve the performance goals of small delay, low dropping rate, and high utilization. Let the average arrival rate of incoming instruction to the processor be b (inst/sec).The processor enforces some "optimal" instruction dropping strategy by selecting appropriate instruction dropping probabilities. Let us use $p(t)$ to denote the processor's dropping probability at time t, then the outgoing instruction execution rate is $b(1 - p(t))$ at time t.

Different instruction dropping strategies have different impacts on the performance of a processor, including instruction delays, number of dropped instructions, and link utilizations. Generally speaking, under a given AQM scheme, if a processor drops instructions more aggressively, less instructions will be admitted and go through the processor, hence the outgoing link's utilization may be lower; but in return, the admitted instructions will experience smaller delays. On the other hand, if under an AQM scheme which drops instructions less aggressively, the admitted instructions may be queued up at the processor, hence the admitted instructions will experience larger delays. But in this case the outgoing link's utilization may be higher, since more instructions are admitted and transmitted by the processor.

Though the goal of maintaining small instruction delay usually contradicts to the goal of admitting more requests and maintaining high link utilizations, a good AQM scheme tries to make intelligent tradeoffs in an optimal way. For example, in order to achieve both high link utilizations and low instruction delays, it is desired that the processor's service queue is maintained at a small but steady value [24]. This is because a small but steady queue ensures small queuing delay; at the same time, the steadiness of the queue allows that there are always instructions to be processed in the outgoing link, hence help to maintain a high link utilization. In the following section, we formulate the "optimal" dropping strategy as an optimal queuing control problem.

A. Problem Formation:

In this section, we formally formulate the optimal dropping strategy problem for an AQM processor. For simplicity, we assume that the outgoing interface of the processor is an M/M/1 queue: the incoming instructions to the processor follows a Poisson arrival with rate b; the service rate of the processor for the instructions is exponentially distributed with rate [1]. As it has been discussed above, in order to achieve both low instruction delay and high link utilization, we would like the (equilibrium) processor queue to be small but steady. Let us use S to denote the target queue length of the processor. At the same time, it is desired for the processor to minimize the number of dropped instructions.

For a specific AQM policy π, let us define the admitting instructions rate at a specific time t as $\lambda\pi(t)$, where $0<=\lambda\pi(t)<=b$. Then $b - \lambda\pi(t)$, represents the dropping rate at the processor for this policy. Let $S\pi(t)$ denote the queue length at time t under policy π, so the difference between the current queue length and the target queue length is $S\pi(t) - S$. We define an objective function with respect to policy π as

$$V\pi = \lim_{T\to\infty} \frac{1}{T} E[\int_0^T ((S\pi(t) - S^2) + W.\big(b - \lambda\pi(\mathrm{t})\big)^2 dt]$$

Where $E[.]$ is the expectation function.

The objective function $V\pi$ represents the average cost for policy π under infinite time horizon. The first term within the integration represents the deviation of current queue length from the target queue length at time t. The second term represents the total rate of dropped instructions at time t under policy π. W is a weight of the valuation between these two terms. It represents the relative cost of dropping instructions. The sum of these two terms within the integration represents the total quadratic cost under policy π at time t. It reflects our desired goal of low instruction delay, high utilization, and low instruction loss in AQM policy design. In the following discussion, for the ease of notation, we move the policy subscript π out from the terms in the square brackets to the expectation function in Eq. (1) without incurring ambiguity, that is

$$V\pi = \lim_{T\to\infty} \frac{1}{T} E\pi[\int_0^T ((S\pi(t) - S^2) + W.\big(b - \lambda\pi(\mathrm{t})\big)^2 dt]$$

CONFERENCE PAPER
National Conference on Information and Communication Technology for Development
Organized by PRMITR, Amravati (MS) India
http://mitra.ac.in/forthcoming.html

Our design objective is then to find a policy under which the average cost function is minimized, i.e.

Minπ $V\pi$

suject to $0<=\lambda\pi(t)<=b$

This Equation defines an interesting optimal control problem. The control input to the system is in terms of the rate of admitted incoming instructions, i.e. $\lambda(t)$. Note the dynamics of the system in our formulation are governed by the queuing system instead of a traditional linear system represented by differential/difference equations, hence our formulation is not a typical optimal control problem. In the following section, we give a solution to this optimal queuing control problem.

V. SOLUTION METHOD VIA VALUE ITERATION

In this section, we present a method to solve the above mentioned optimal queuing control problem. First, we note the queue length $S(t)$ forms a continuous time Markov chain(CTMC). Let N denote the processor's maximum buffer size, then there are N +1 states for this continuous time Markov chain. State i $\epsilon\{0,\ldots..N\}$ of the Markov chain corresponds to the case where there are i instructions in the queue. In our solution method, we first convert the CTMC into a discrete time Markov chain (DTMC) to facilitate the usage of value iteration algorithms. This is done through a technique called uniformization [25].

A. Conversion from CTMC to DTMC:

For a continuous time Markov chain(CTMC), state transitions are allowed to occur at any time instant. Therefore CTMC is widely used to model a large number of real-world stochastic systems. On the other hand, discrete time Markov chain (DTMC) models are easy to handle. Fortunately, we can convert a CTMC to a DTMC, making the two chains stochastically equivalent through uniformization. To this end, we select a uniform rate γ= μ+ b. The transition probabilities among states for the stochastically equivalent DTMC are obtained by dividing the original transition rate in the CTMC by γ. This procedure is shown in the following equations.

$$Poo = 1 - \frac{\lambda}{\mu + b}$$

$$Poo = \ 1 - \frac{\lambda}{(\mu + b)}$$

$$Po1 = \ \frac{\lambda}{(\mu + b)}$$

$$P_{i,i+1} = \ \frac{\lambda}{(\mu + b)}$$

$$P_{i,i-1} = \ \frac{\mu}{(\mu + b)}$$

$$P_{i,i} = P_{i,i+1} - P_{i,i-1} \ = \frac{b - \lambda}{(\mu + b)}$$

$$Pn, n-1 = \ \frac{\mu}{(\mu + b)}$$

$$Pn, n = 1 - \ \frac{\mu}{(\mu + b)}$$

Figure 1

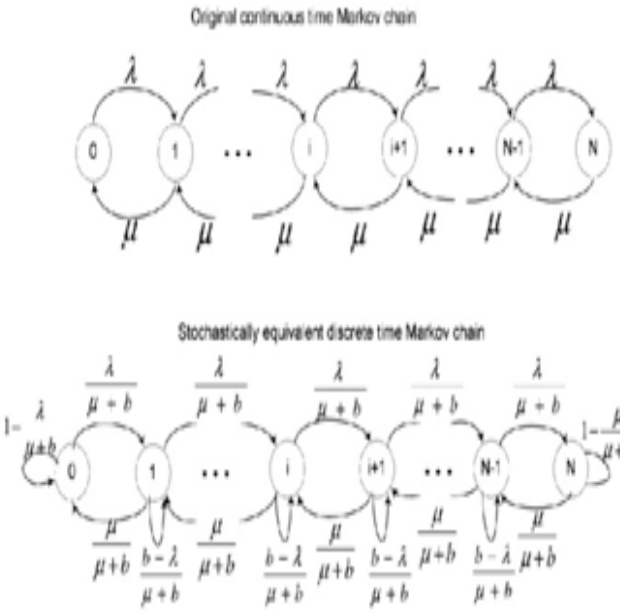

Figure: 2 illustrates the CTMC and the stochastically equivalent DTMC.

The second step is converting the objective function from CTMC to the equivalent DTMC counterpart. The cost function governed by the original CTMC is

$$C(t,\lambda) = (s(t) - s)^2 + W.\left(b - \lambda(t)\right)^2$$

Therefore correspondingly, the DTMC's cost function can be derived as

$$C'(i,\lambda) = \frac{1}{\lambda} c(i,\lambda) = \frac{1}{\mu + b}\ C(i,\lambda)$$

The corresponding DTMC's optimal queuing control problem is as follows

Min$_\pi$ V_π

$$= \lim_{T\to\infty} E\pi \frac{1}{T}[\sum_{0}^{T}[C'(i,\lambda)]]$$

$$= \lim_{T\to\infty} E\pi \frac{1}{T}[\sum_{0}^{T} \frac{1}{(\mu + b)} ((S(i) - \ S^2) + W.\left(b - \lambda(i)\right)^2)$$

Subject to $0<= \lambda(i)<= b$

In the optimal queuing control problem of DTMC, the dynamics of the system are governed by the DTMC,

B. Solution to the Optimal Queuing Control Problem of DTMC:

The optimal queuing control problem for the DTMC can be solved using the following value iteration algorithm [25].

C. Value iteration algorithm:

Step 0: Set n=0 and $V_0(0) = 0$; Set iteration stop criteria, i.e. the maximum number of iterations M, and accuracy tolerance threshold ε>0;

Step 1: Choose a state x (0 <= x <= N) as a baseline (distinguished) state;

Step 2: Set $Vn(i) = = min\lambda\{ C(i,\lambda) + Pij(\lambda)\mu n(j))$

Step 3: Set $un+1(i) = Vn(i) - Vn(x)$;

Step 4: Goto Step 2, until the maximum number of iterations M is reached for δ= max(iεs) | V(i) + ∑ Pij(λ)μn(j)|

CONFERENCE PAPER
National Conference on Information and Communication Technology for Development
Organized by PRMITR, Amravati (MS) India
http://mitra.ac.in/forthcoming.html

Step 5: Output $Vn(x)$ and the stationary policy realizing
$\text{Min} \mid C(I, \lambda) + \sum Pij(\lambda)\mu n(j)\mid$

It is worth noting that the theoretic solutions given in this section may not be directly applicable to real-world applications. This is because: First, modeling the processor queue as an M/M/1 queue is based on a simplified assumption. Real world instructions may not follow Poisson arrival, and the service rate of instructions may not be exponential; Second, the complexity of value-iteration (and other dynamic-programming based solution methods) is usually high. However, for these real-world applications, we can use techniques such as Q learning[26] and neuro-dynamic programming [27] to derive near-optimal solutions.

D. Evaluation:

To evaluate the queuing control based approach, we implemented the uniformization and value iteration algorithm to get the optimal dropping strategies for the processor under different set-ups. In this section, we report these results and discuss the effect of various design parameters. Note the value iteration algorithm gives the optimal strategy in terms of admitting instructions rate ¸ with respect to processor's queue length. Here we also report the optimal dropping probability of the processor. The dropping probability is expressed as

$$\frac{b - \lambda(i)}{b}$$

Where b is the incoming instructions rate, and i is the Processor's current queue length.

In the first set of experiments, the incoming instructions rate is set to b = 120 (inst/sec); the service rate of the processor is set to μ= 100 (inst/sec); processor's buffer size is set to N = 200,and the target queue length is set to S = 50. Fig. 3–Fig. 5 report the optimal admitting instructions rate and the optimal dropping probability (p) with respect to the queue length in the processor. For Fig. 3, the weight W representing the relative cost of dropping instructions is set to 0:01. For Fig. 4 and Fig. 5, we set W = 0:1, and W = 1:0 respectively.

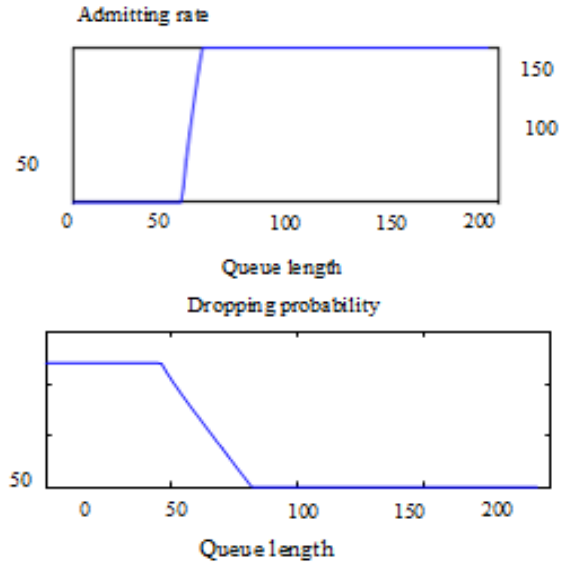

Figure. 3 Optimal admitting rate and dropping probability v.s. queue length, when W = 0.01

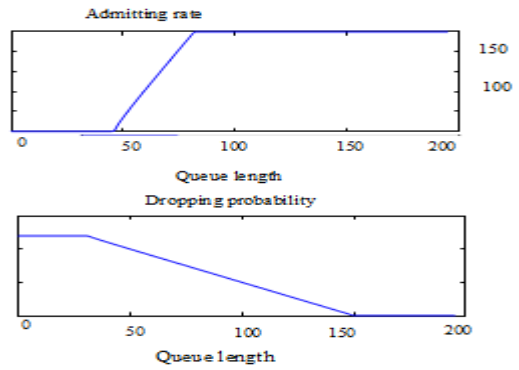

Figure. 4. Optimal admitting instructions rate and dropping probability v.s. queue length, when W = 0.1.

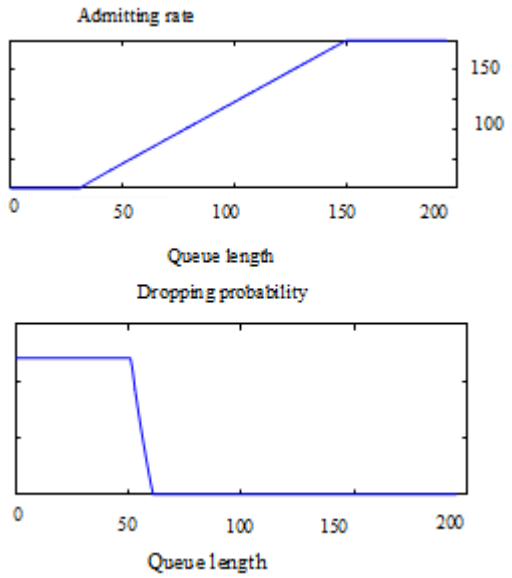

Figure. 5. Optimal admitting instructions rate and dropping probability v.s. processor's queue length, when W = 1.0.

We get four observations from the results reported in Fig. 3–Fig. 5.

a. The control law of optimal strategies is event-driven, since the control action is a function of the number of instructions in the current processor queue. This contrasts to the classical time-driven control, where control action is usually trigger AQM at constant time intervals.

b. Optimal strategies are closed-loop-based feedback controls. The feedback measurement for each optimal controller is the number of instructions in the processor's queue.

c. When the weight W (which represents the relative cost of dropping instructions) changes, the optimal dropping strategy also changes. The first experiment with W =0:01 reflects the situation where dropping instructions incurs negligible cost. When we gradually increase the value of W, we anticipate the optimal dropping probability curve to become more flat since a larger W means larger cost will be incur AQM for the dropped instructions. This can be clearly observed from Fig. 3 – 5.

CONFERENCE PAPER
National Conference on Information and Communication Technology for Development
Organized by PRMITR, Amravati (MS) India
http://mitra.ac.in/forthcoming.html

d. The optimal control strategies derived in this paper give similar results to other AQM schemes including Random Early Detection (AQM) and its variants [28],[22]. Our approach is based on solid theoretical designand synthesis through queuing systems control. As a result, it gives a natural way to calculate the controller parameters (i.e. AQM parameters). Unlike previous control-theoretical approach [23] which uses linearized fluid-approximation of the system as the plant model, Our control design is based on the nonlinear queuing model.

A predetermined set of AQM parameters under a "typical" workload may not render good performance under a different workload. For example, it has been shown in [22] that the effectiveness of AQM depends, to a large extent, on the appropriate parameterization of the AQM queue when load changes. A good control policy should adapt its parameters in response to workload dynamics.

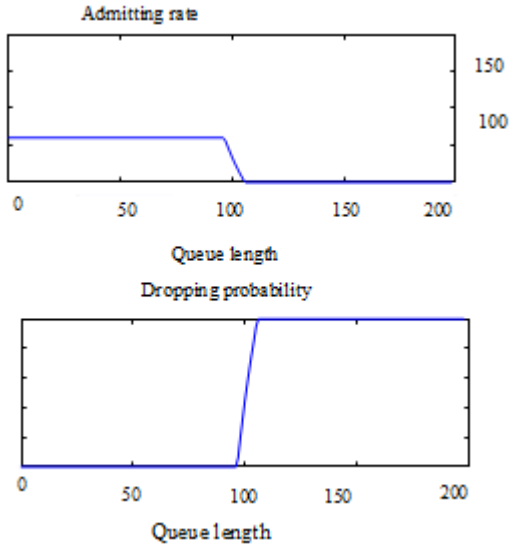

Figure. 6. Optimal admitting instructions rate and dropping probability v.s. processor's queue length, when b = 60 (inst/sec).

Fig. 6 show the optimal admitting instructions rate (λ), and optimal dropping probability (p) with respect to the queue length in the processor when incoming instructions's intensity changes. In these experiments, the service rate of the Processor is μ = 100 (ints/sec), total buffer size is N = 200, target

From the results shown in Fig. 6 , we see as the workload intensity increases, the corresponding optimal policies begin dropping instruction more aggressively. For ex- ample, when λ = 60(inst/sec), the optimal policy does not begin dropping packets until the processor's queue length reaches 95 (Fig. 6); but when under high workload of λ = 180(inst/sec), the optimal policy begins dropping instructions when processor's queue length reaches only 27 (Fig. 8). In this setup, when the processor's queue length reaches 82, all incoming instructions are dropped under the optimal policy, as compatible AQM to the value of 106 for the case when λ = 60(inst/sec).

VI. CONCLUSIONS AND FUTURE WORK

The rapid development and pervasive deployment of in information technology(IT) has created a need to enforce service and resource management policies automatically.

In this paper we have described how queuing control techniques can be effectively applied to computing system's performance control and management. Specifically, we study how to design optimal dropping strategies for multiprocessor using this approach. We formulate the problem of designing the optimal dropping strategy as an optimal queuing con-troll problem. We then derive the optimal controller using uniformization and value iteration. Through numerical evaluation, we also discussed the effect of various design parameters and workload characteristics on the optimal dropping strategies.

Since many computing systems can be modeled as discrete-event systems, we believe that discrete-event con-tro l approach has its own advantages over many classical control-theoretic approaches for these systems. We hope this work can bring new ideas and tools to feedback control of computing systems.

VII. REFERENCES

[1]. T. F. Abdelzaher and C. Lu, "Modeling and performance control of internet servers," in 39th IEEE Conference on Decision and Control2000.

[2]. C. Lu, T. Abdelzaher, J. Stankovic, and S. Son, "A feedback control approach for guaranteeing relative delays in web servers," in IEEE Real-Time Technology and Applications Symposium, 2001.

[3]. Y. Diao, J. Hellerstein, and S. Parekh, "Using fuzzy control to maximize profits in service level management," IBM Syst. J., vol. 41, no. 3, pp. 403-420, 2002., 2002.

[4]. L. Kleinrock, Queuing Systems, Vol. 2, Applications. John Wiley, 1976.

[5]. P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," SIAM Journal of Control and Optimization, vol. 25, no. 1, pp. 206–230,

[6]. C. G. Cassandras and S. Lafortune, Introduction to Discrete Event Systems. KLUWER ACADEMIC PUBLISHERS, 1999.

[7]. J. L. Hellerstein, Y. Diao, S. Parekh, and D. M. Tilbury, FeedbackControl of Computing Systems. Wiley-IEEE Press, 2004.

[8]. N. Gandhi, S. Parekh, J. Hellerstein, and D. Tilbury, "Feedback control of a lotus notes server: Modeling and control design," in American Control Conference, 2001

[9]. Y. Diao, N. Gandhi, J. L. Hellerstein, S. Parekh, and D. M. Tilbury, "Mimo control of an apache web server: Modeling and controller design," in American Control Conference, 2002.

[10]. Y. Lu, A. Saxena, and T. F. Abdelzaher, "Differentiated caching services: A control-theoretical approach," in International Conference on Distributed Computing Systems,

CONFERENCE PAPER
National Conference on Information and Communication Technology for Development
Organized by PRMITR, Amravati (MS) India
http://mitra.ac.in/forthcoming.html

2001.

[11]. X. Liu, J. Heo, L. Sha, and X. Zhu, "Adaptive control of multi-tieAQM web application using queuing pAQMictor," in 10th IEEE/IFIP Network Operations and Management Symposium (NOMS 2006), Vancouver, Canada, 2006

[12]. M. Karlsson, C. Karamanolis, and X. Zhu, "Triage: Performance isolation and differentiation for storage systems," in The Twelfth IEEE International Workshop on Quality of Service (IWQoS 2004), 2004.

[13]. Y. Lu, C. Lu, T. Abdelzaher, and G. Tao, "An adaptive control framework for qos guarantees and its application to differentiated caching services," in IWQoS, 2002.

[14]. B. Li and K. Nahrstedt, "A control-based middleware framework for quality-of-service adaptations," IEEE Journal on Selected Areas in Communications, vol. 17, no. 9, pp. 1632–1650, 1999.

[15]. R. Zhang, C. Lu, T. F. Abdelzaher, and J. A. Stankovic, "Control- ware: A middleware architecture for feedback control of software performance," in International Conference on Distributed Computing Systems, 2002.

[16]. S. Parekh, N. Gandhi, J. L. Hellerstein, D. Tilbury, T. S. Jayram, and J. Bigus, "Using control theory to achieve service level objectives in performance management," Real Time Systems Journal, vol. 23, no 1 -2 ,2001

[17]. B. A. Brandin, "The real-time supervisory control of an experimental manufacturing cell," IEEE Transactions on Robotics and Automation, vol. 12, no. 1, pp. 1–14, 1996.

[18]. Y. Wang, T. Kelly, and S. Lafortune, "Discrete control for safe execution of it automationworkflows," in Eurosys 2007, 2007.

[19]. R. Jain, The Art of Computer Systems Performance Analysis. John Wiley and Sons, 1991.

[20]. L. Sha, X. Liu, Y. Lu, and T. Abdelzaher, "Queuing model based network server performance control," in 23rd IEEE Real-Time Systems Symposium (RTSS02). IEEE Computer Society, 2002, p. 81.

[21]. S. Floyd and V. Jacobson, "Random early detection gateways for con- gestion avoidance," IEEE/ACM Transactions on Networking, vol. 1, no. 4, pp. 397–413, 1993.

[22]. W.-C. Feng, D. D. Kandlur, D. Saha, and K. G. Shin, "A self- configuring AQM gateway," in IEEE INFOCOM 99, 1999, pp. 1320–1328.

[23]. C. V. Hollot, V. Misra, D. F. Towsley, and W. Gong, "A control theoretic analysis of AQM," in IEEE Infocom, 2001, pp. 1510–1519.

[24]. S. Athuraliya, V. H. Li, S. H. Low, and Q. Yin, "REM: Active queue management," IEEE Network, vol. 15, no. 3, pp. 48–53, 2001.

[25]. V. Kulkarni, Modeling and Analysis of Stochastic Systems. Chapman and Hall, 1996.

[26]. T. M. Mitchell, Machine Learning. McGraw-Hill, 1997.

[27]. D. P. Bertsekas and J. Tsitsiklis, Neuro-Dynamic Programming. Athena Scientific, 1996.

[28]. S. Floyd, R. Gummadi, and S. Shenker, "Adaptive AQM: An algorithm for increasing the robustness of AQM's active queue management," 2001

CONFERENCE PAPER
National Conference on Information and Communication Technology for Development
Organized by PRMITR, Amravati (MS) India
http://mitra.ac.in/forthcoming.html