

Control Computing Systems as Queueing Systems: The Case of Optimal Dropping Strategy for Internet Routers

Xue Liu

Hewlett Packard Laboratories
Palo Alto, CA, 94304
Email: xue.liu@hp.com

Wenbo He

University of Illinois at Urbana-Champaign
Champaign, IL, 61801, United States
Email: wenbohe@uiuc.edu

Abstract—The rapid development and pervasive deployment of information technology (IT) has created a need to enforce service and resource management policies automatically. Recently, there is a surge in using control-theoretic approach to manage and control computing systems. Unlike most of the existing control-theoretic approaches which model computing systems as linear systems, this paper shows that queueing control techniques can be effectively applied to computing system's performance management and control. We use Active Queue Management for Internet routers as a working example. By modeling the router as a single station queue, we formulate the problem of designing the optimal dropping strategy as an optimal queueing control problem. We derive the optimal controller design using uniformization and value iteration. Through numerical evaluations, we discuss the effect of various design parameters and workload characteristics on the optimal dropping strategies.

I. INTRODUCTION

Performance management has been a core research area in computer science. As today's IT systems, networks, and services become more pervasive and increasingly complex, the demand to manage and control them has grown rapidly. Managing computing systems by human is becoming more and more unfeasible due to economic and fault-tolerance considerations. During the last 50 years, computing has brought great benefits of speed and automation in our everyday life, but now there is an overwhelming need to automate their maintenance by computing systems themselves.

Traditional practice of automated run-time resource management and control largely relies 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 discovered that feedback control, which was very successful in other engineering disciplines (e.g., mechanical, electrical, aeronautic), can also be successfully used in analyzing and designing run-time IT systems [1], [2], [3], [4]. These control-theoretic approaches have advantages such as self-adaptation to load changes and robustness to model inaccuracies, hence they can help to achieve better self-management for complex computing systems. Indeed, in the past five years, there has been considerable success with applying control theory to computing systems. Areas addressed include: Internet services, Web servers, database systems, and power management etc. Chapter 1 in [5] gives an extensive summary of related work in this area.

In order to use feedback control framework, nearly all previous work use linear models to represent the underlying computing system. However, computing systems are usually nonlinear [6] with respect to the resource allocated. In addition, workload to computing systems are usually stochastic; its parameters could change over a wide range of values. As a result, the differential/difference equation model used by classical control theory does not model real-world computing system well, except in the limited case of heavy workload that allows for fluid approximations.

A good-quality mathematical model of the plant is critical to any control system design. In spite of the tradition, we believe that the key to success of applying feedback control to manage computing systems is not to force-fit a computing system into linear models. Rather, we should model computing systems as what they are.

During the last several decades, research has shown that queueing models represent typical computing systems well. For a detailed review of this area, please refer to references [6][7]. However, in computer system research, queueing models are usually used for off-line capacity planning purposes instead of online performance tuning purposes.

In control theory, over the last two decades, along with the large body of research on control of discrete event systems, control techniques for queueing systems have been emerged. In this paper, we explore the applicability of queueing control techniques to computing system applications. We illustrate this approach through a working example of designing optimal dropping strategies for Internet routers. In our approach, we model an Internet router as a single station queue, and formulate the problem of designing the optimal dropping strategy as an optimal queueing control problem. We then derive the optimal control strategy through uniformization and value iteration. Our solution gives the optimal dropping strategy similar to the Random Early Detection (RED) policy and its variants. Our controller synthesis gives different optimal dropping strategies in response to different workloads. Hence it can be used in designing self-configuring Active Queue Management (AQM) schemes.

Since many computing systems can be modeled as queueing systems, we believe that queueing control approach has its own advantages over classical control-theoretic approaches in these systems and deserve being further investigated. We hope this work can help to bring new ideas and

tools in feedback control of computing systems.

The remainder of this paper is organized as follows. Section II briefly reviews related work in feedback control of computing systems. Then it gives a background of Active Queue Management and introduces optimal packet dropping problem which serves as a working example throughout the rest of the paper. Section III presents our model and formulates the optimal packet dropping problem as an optimal queueing control problem. Section IV describes the solution method to the optimal queueing control problem. Section V evaluates the method and discuss the effect of various design parameters and workload characteristics on the optimal dropping strategies. Finally Section VI concludes the paper with summaries of the work.

II. BACKGROUND AND RELATED WORK

A. Control of Computing Systems

Classical control theory has recently been applied to a variety of performance-related computing systems. Chapter 1 in [5] gives an extensive summary of related work in this area. The systems being controlled include Lotus Notes email server [8], Apache Web server [1][9][2], Squid proxy server [10], Lustre file system [11], as well as multi-tier e-commerce site [12]. The output metrics include system-level metrics, such as CPU and memory utilization [9], cache hit ratio [13], and server queue length [14]; application level metrics such as response time, throughput [11][2] and server power consumptions [15], or business level metrics such as profits [3]. Control mechanisms used include admission control or request throttling [11], application parameter tuning [9][3], resource allocation [2][13], and middleware [16][17]. The types of control algorithms used include variations of proportional, integral, and derivative (PID) control [1][14][16], pole placement [9], linear quadratic regulator (LQR) [9] and adaptive control [13][11] etc.

Whereas classical control deals with linear systems whose dynamics are described by differential/difference equations, a majority of computing systems' states are discrete and their dynamics follow event-driven dynamics. Those systems are more suitably be represented by discrete event systems [18]. Discrete-event control theory has been applied in domains such as manufacturing [19]. Recently, it has been studied by Wang et.al. to improve the safe execution of IT automation workflows [20].

Computing systems are shown to be modeled well by queueing models [6][7]. The approach presented in [21] combines queueing model and feedback control for controlling computing systems. However, the controller design is still based on classical control theory. In contrast, this paper applies queueing control directly to computing system's management.

B. Active Queue Management

Recent measurements have shown that the growing demand for network bandwidth has driven traffic up exponentially in the Internet. It is important to achieve low packet loss and delay, and high link utilizations in the Internet.

Active Queue Management (AQM) policies on Internet routers are intended to help achieving both high link utilization and a low packet delay. It has been one of the most active areas of Internet congestion control research in the past few years. RED [22] (Random Early Detection) is a well-known AQM scheme. The basic idea behind RED queue management scheme is to detect incipient congestion early and to convey congestion notification to the end-hosts. Hence end hosts will reduce their transmission rates before queues in the network overflow and packets are dropped. To do this, RED 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, packets are randomly dropped or marked with an explicit congestion notification (ECN) bit. When the average queue length exceeds a maximum threshold, all packets are dropped or marked. While RED is certainly an improvement over traditional drop-tail queues, the performance of the RED algorithm depends significantly upon the setting of each of its parameters, which appears to be a difficult task. Feng et al. [23] proposed a mechanism for adaptively varying one of the RED parameters, max_p , with the aim of reducing the packet loss rates across congested links. Floyd et al. in [24] discuss the algorithmic modifications to the self-configuring RED algorithm [23] for tuning max_p adaptively.

Recently, in [25], Hollot et al. have studied the problem of tuning RED parameters from a control-theoretic view point. In this approach, a linearized version of a nonlinear dynamic model of TCP is used to analyze and design AQM systems using RED. The linearized model interconnects TCP and the bottlenecked router queue in discussion. Classical control theory was used in designing the control system, where RED parameters are embodied in. The model used for controller design was based on fluid-flow approximation.

In this paper, a router is modeled as a single station queue which captures the internal dynamics of the router. Optimal dropping strategy and its associated threshold values are derived through optimal queueing control formulation and controller design.

III. PROBLEM STATEMENT AND FORMULATION

Fig. 1 illustrates an Internet router and shows its relationship with incoming traffic, outgoing traffic and dropped traffic. We assume the router enforces packet-dropping-based AQM scheme to achieve the performance goals of small delay, low dropping rate, and high utilization. Assume the traffic arrives at the router with an average arrival rate of b (reqs/sec). The router enforces some "optimal" packet dropping strategy by selecting appropriate packet dropping probabilities. Let us use $p(t)$ to denote the router's dropping probability at time t , then the outgoing traffic rate is $b(1 - p(t))$ at time t . The exact meaning of "optimal strategy" will be discussed in details in Section III-A.

Different packet dropping strategies have different impacts on the performance of a router, including packet delays, number of dropped packets, and link utilizations. Generally speaking, under a given AQM scheme, if a router drops

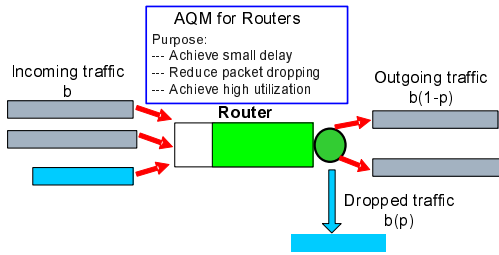


Fig. 1. An AQM router with incoming traffic, outgoing traffic, and dropped traffic.

packets more aggressively, then less packets will be admitted and go through the router, hence the outgoing link's utilization may be lower; but in return, the admitted packets will experience smaller delays. On the other hand, if under an AQM scheme which drops packets less aggressively, the admitted packets may be queued up at the router, hence the admitted packets will experience larger delays. But in this case the outgoing link's utilization may be higher, since more packets are admitted and transmitted by the router.

Though the goal of maintaining small packet 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 packet delays, it is desired that the router's service queue is maintained at a small but steady value [26]. This is because a small but steady queue ensures small queueing delay; at the same time, the steadiness of the queue allows that there are always packets 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 queueing control problem. Then in Section IV, we show how to use uniformization and value iteration techniques to derive the optimal packet dropping strategies.

A. Problem Formulation

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

For a specific AQM policy π , let us define the admitting traffic rate at any time t as $\lambda_\pi(t)$, where $0 \leq \lambda_\pi(t) \leq b$. Then $b - \lambda_\pi(t)$ represents the dropping rate at the router 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

$$V_\pi = \lim_{T \rightarrow \infty} \frac{1}{T} E \left[\int_0^T \left((S_\pi(t) - S)^2 + W \cdot (b - \lambda_\pi(t))^2 \right) dt \right], \quad (1)$$

where $E[\cdot]$ is the expectation function.

The objective function V_π 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 packets at time t under policy π . W is a weight of the valuation between these two terms. It represents the relative cost of dropping packets. 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 packet delay, high utilization, and low packet loss in designing the AQM policy.

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

$$V_\pi = \lim_{T \rightarrow \infty} \frac{1}{T} E_\pi \left[\int_0^T \left((S(t) - S)^2 + W \cdot (b - \lambda(t))^2 \right) dt \right]. \quad (2)$$

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

$$\begin{aligned} & \text{Min}_\pi \quad V_\pi \\ & \text{subject to: } 0 \leq \lambda(t) \leq b. \end{aligned} \quad (3)$$

Equation (3) defines an interesting optimal control problem. The control input to the system is in terms of the rate of admitted incoming packets, i.e. $\lambda(t)$. Note the dynamics of the system in our formulation are governed by the queueing system instead of a 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 method to this optimal queueing control problem.

IV. SOLUTION METHOD VIA VALUE ITERATION

In this section, we present a method to solve the above-mentioned optimal queueing control problem. First, we note the queue length $S(t)$ forms a continuous time Markov chain (CTMC). Let N denote the router's maximum buffer size, then there are $N + 1$ states for this continuous time Markov chain. State $i \in \{0, \dots, N\}$ of the Markov chain corresponds to the case where there are i packets 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 [27].

A. Conversion from CTMC to DTMC

For a continuous time Markov chain (CTMC), state transitions may occur at any time instant. Therefore CTMC is considered more realistic in modeling of a large number of

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 $\gamma = \mu + b$. The transition probabilities between states for the stochastically equivalent DTMC are obtained by dividing the original transition rate in the CTMC by γ . This procedure is shown in Equation (4).

$$\begin{aligned}
P_{00} &= 1 - \frac{\lambda}{\mu+b}, \\
P_{01} &= \frac{\lambda}{\mu+b}, \\
&\vdots \\
P_{i,i+1} &= \frac{\lambda}{\mu+b}, \\
P_{i,i-1} &= \frac{\mu}{\mu+b}, \\
p_{i,i} &= 1 - p_{i,i+1} - p_{i,i-1} = \frac{b-\lambda}{\mu+b}, \\
&\vdots \\
P_{N,N-1} &= \frac{\mu}{\mu+b}, \\
P_{N,N} &= 1 - \frac{\mu}{\mu+b}.
\end{aligned} \tag{4}$$

Fig. 2 illustrates the CTMC and the stochastically equivalent DTMC.

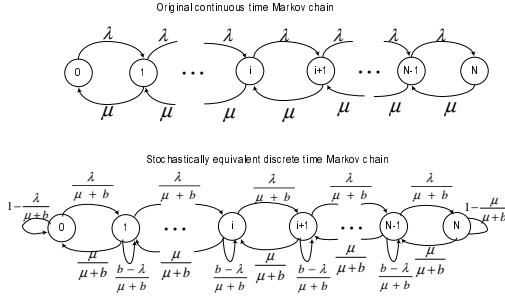


Fig. 2. The CTMC and the equivalent DTMC.

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

$$C(t, \lambda) = (S(t) - S)^2 + W \cdot (b - \lambda(t))^2. \tag{5}$$

Correspondingly, the DTMC's cost function can be derived as [27]

$$C'(i, \lambda) = \frac{1}{\gamma} C(i, \lambda) = \frac{1}{\mu + b} C(i, \lambda). \tag{6}$$

Therefore, the corresponding DTMC's optimal queueing control problem is as follows

$$\begin{aligned}
\text{Min}_{\pi} \quad V_{\pi} &= \lim_{T \rightarrow \infty} E_{\pi} \frac{1}{T} \left[\sum_{i=0}^T [C'(i, \lambda)] \right] = \\
&\lim_{T \rightarrow \infty} E_{\pi} \frac{1}{T} \left[\sum_{i=0}^T \frac{1}{\mu+b} ((S(i) - S)^2 + W \cdot (b - \lambda(i))^2) \right]
\end{aligned} \tag{7}$$

subject to $0 \leq \lambda(i) \leq b$.

In the optimal queueing control problem of DTMC, the dynamics of the system are governed by the DTMC, and its transition probabilities are given in Equation (4).

B. Solution to the Optimal Queueing Control Problem of DTMC

The optimal queueing control problem for the DTMC can be solved using the following value iteration algorithm [27].

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 ;
- Step 1:** Choose any state x ($0 \leq x \leq N$) as a baseline state;
- Step 2:** Set $V_n(i) = \min_{\lambda} \{C(i, \lambda) + \sum_j P_{ij}(\lambda) u_n(j)\}$;
- Step 3:** Set $u_{n+1}(i) = V_n(i) - V_n(x)$;
- Step 4:** Goto Step 2, until the maximum number of iterations M is reached or $\delta = \max_{i \in S} |V_n(i) - V_{n-1}(i)| \leq \varepsilon$;
- Step 5:** Output $V_n(x)$ and the stationary policy realizing $\min_{\lambda} \{C(i, \lambda) + \sum_j P_{ij}(\lambda) u_n(j)\}$.

V. EVALUATION

To evaluate the queueing control based approach, we implemented the uniformization and value iteration algorithm to get the optimal dropping strategies for the router 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 traffic rate λ with respect to router's queue length. Here we also report the optimal dropping probability of the router. The dropping probability is expressed as $p(i) = \frac{(b-\lambda(i))}{b}$, where b is the incoming traffic rate, and i is the router's current queue length.

In the first set of experiments, the incoming traffic rate is set to $b = 120$ (reqs/sec); the service rate of the router is set to $\mu = 100$ (reqs/sec); router's buffer size is set to $N = 200$, and the target queue length is set to $S = 10$. Fig. 3–Fig. 5 report the optimal admitting traffic rate (λ), and the optimal dropping probability (p) with respect to the queue length in the router. For Fig. 3, the weight W representing the cost of dropping packets is set to 0.01. For Fig. 4 and Fig. 5, we set $W = 0.1$, and $W = 1.0$ respectively.

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

- 1) The control law of optimal strategies is event-driven, since the control action is a function of the number of packets in the current router queue. This contrasts to the classical time-driven control, where control action is usually triggered at constant time intervals.
- 2) Optimal strategies are closed-loop-based feedback controls. The feedback measurement for each optimal controller is the number of packets in the router's queue.
- 3) When the weight W (which represents the relative cost of dropping packets) changes, the optimal dropping strategy also changes. The first experiment with $W = 0.01$ reflects the situation where dropping packets 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 incurred for the dropped packets. This can be clearly observed from Fig. 3 – 5.

- 4) The optimal control strategies derived in this paper give similar results to other AQM schemes including Random Early Detection (RED) and its variants [24], [23]. Our approach is based on solid theoretical design and synthesis through queueing systems control. As a result, it gives a natural way to calculate the controller parameters (i.e. AQM parameters). Unlike previous control-theoretic approach [25] which uses linearized fluid-approximation of the system as the plant model, our control design is based on the nonlinear queueing model. Queueing models usually capture the dynamics of a computing system better than approximated linear models. Hence the controllers derived are also expected to give better control performances compared with those derived from approximated linear models.

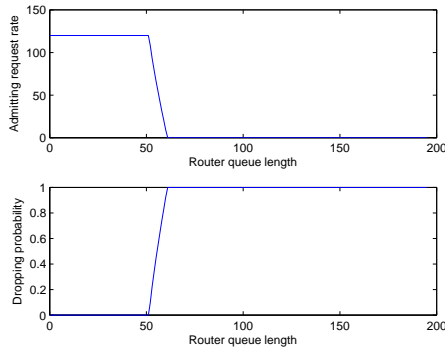


Fig. 3. Optimal admitting traffic rate and dropping probability v.s. router's queue length, when $W = 0.01$.

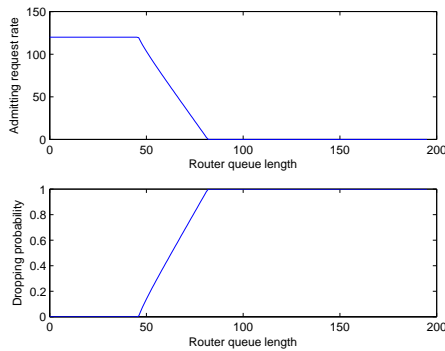


Fig. 4. Optimal admitting traffic rate and dropping probability v.s. router's queue length, when $W = 0.1$.

Now, let's study the effect of incoming traffic's workload on the optimal control policies. Real-world traffic in the Internet changes at a wide scale. 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 [23] that the effectiveness of RED depends, to a large extent, on the appropriate parameterization of the

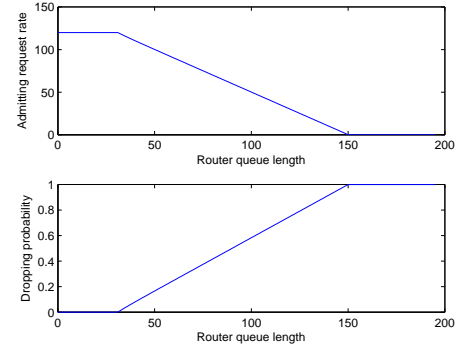


Fig. 5. Optimal admitting traffic rate and dropping probability v.s. router's queue length, when $W = 1.0$.

RED queue when load changes. A good control policy should adapt its parameters in response to workload dynamics.

Fig. 6 – Fig. 8 show the optimal admitting traffic rate (λ), and optimal dropping probability (p) with respect to the queue length in the router when incoming traffic's intensity changes. In these experiments, the service rate of the router is $\mu = 100$ (reqs/sec), total buffer size is $N = 200$, target queue length is $S = 50$. The weight for the cost of dropping packets is $W = 0.1$. Fig. 6, Fig. 7, and Fig. 8 correspond to the cases when incoming traffic's arrival rate (b) is 60 (reqs/sec), 100 (reqs/sec), and 180 (reqs/sec) respectively.

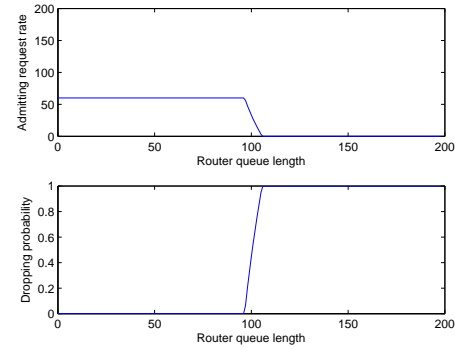


Fig. 6. Optimal admitting traffic rate and dropping probability v.s. router's queue length, when $\lambda = 60$ (reqs/sec).

From the results shown in Fig. 6 to Fig. 8, we see as the workload intensity increases, the corresponding optimal policies begin dropping packets more aggressively. For example, when $\lambda = 60$ (reqs/sec), the optimal policy does not begin dropping packets until the router's queue length reaches 95 (Fig. 6); but when under high workload of $\lambda = 180$ (reqs/sec), the optimal policy begins dropping packets when router's queue length reaches only 27 (Fig. 8). In this setup, when the router's queue length reaches 82, all incoming traffic are dropped under the optimal policy, as compared to the value of 106 for the case when $\lambda = 60$ (reqs/sec).

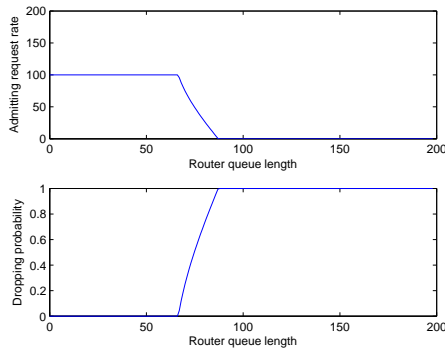


Fig. 7. Optimal admitting traffic rate and dropping probability v.s. router's queue length, when $\lambda = 100$ (reqs/sec).

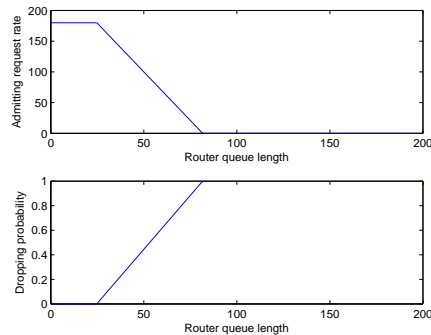


Fig. 8. Optimal admitting traffic rate and dropping probability v.s. router's queue length, when $\lambda = 180$ (reqs/sec).

VI. CONCLUSIONS AND FUTURE WORK

The rapid development and pervasive deployment of information technology (IT) has created a need to enforce service and resource management policies automatically. In this paper we have described how queueing control techniques can be effectively applied to computing system's performance control and management. We use optimal dropping strategies design for Internet routers as a working example. We formulate the problem of designing the optimal dropping strategy as an optimal queueing control problem. We derive the optimal controller designs 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 queueing systems, we believe that queueing control approach has its own advantages over classical control-theoretic approaches in these systems and deserve being further investigated. We hope this work can help to bring new ideas and tools in feedback control of computing systems. Our ongoing work includes integrating the TCP dynamics and router dynamics as queueing networks and design corresponding controllers.

REFERENCES

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

2000.

[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] J. A. Stankovic, C. Lu, S. H. Son, and G. Tao, "The case for feedback control real-time scheduling," in *11th EuroMicro Conference on Real-Time Systems*, 1999.

[5] J. L. Hellerstein, Y. Diao, S. Parekh, and D. M. Tilbury, *Feedback Control of Computing Systems*. Wiley-IEEE Press, 2004.

[6] L. Kleinrock, *Queueing Systems, Vol. 2, Applications*. John Wiley, 1976.

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

[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*, 2001.

[11] 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.

[12] A. Kamra, V. Misra, and E. Nahum, "Yaksha: A controller for managing the performance of 3-tiered websites," 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] 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.

[15] S. Abdelwahed, N. Kandasamy, and S. Neema, "Self-optimization in computer systems via online control: Application to power management," in *IEEE Conference on Autonomic Computing (ICAC)*, 2004, pp. 54-61.

[16] 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.

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

[18] 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, 1987.

[19] 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.

[20] Y. Wang, T. Kelly, and S. Lafortune, "Discrete control for safe execution of it automation workflows," in *Eurosys 2007*, 2007.

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

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

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

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

[25] C. V. Hollot, V. Misra, D. F. Towsley, and W. Gong, "A control theoretic analysis of red," in *IEEE Infocom*, 2001, pp. 1510-1519.

[26] 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.

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