Performance Analysis of a Rate-Control Throttle where Tokens and Jobs Queue

Arthur W. Berger, *Member, IEEE*

Abstract—**A rate-control throttle is used for overload control in distributed switching systems and computer and communication networks. Typical implementations of the throttle have a token bank where an arriving job is blocked and rejected if the bank is empty of tokens. This paper examines an expanded implementation where an arriving job queues in a finite buffer when the token bank is empty. We show that the steady-state throughput and blocking of jobs depends on the capacity of the job buffer and the capacity of the token bank only via the sum of the two capacities, and not on their individual values. Thus, the job buffer per se is not needed to enhance the robustness of the throughput of the throttle to unknown exogenous job arrival rates. However, a job buffer, along with a token bank, with adjustable buffer capacities does have the potential to shape the departure process and to adapt between a delay control and a work-rejection control.**

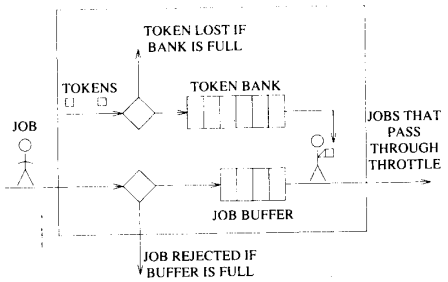

Fig. 1. Diagram of a rate-control throttle.

I. Introduction

A RATE-control throttle is an input regulation technique for overload control. It has been used in distributed switching systems [1], [2], and is applicable to general computer and communication networks. The throttle is typically implemented with a token bank where an arriving job is blocked and rejected if the bank is empty of tokens. This paper considers an expanded throttle where jobs may queue in a finite buffer when the token bank is empty. The operation of the throttle is illustrated in Fig. 1. The throttle contains two finite buffers, one for tokens and one for jobs. The buffer capacities are typically, though not necessarily, constant during the operation of the throttle. The tokens arrive deterministically and evenly spaced from an infinite source. The arrival rate of tokens is the control variable of the throttle. Tokens that arrive to a full bank are blocked and lost. If the bank contains a token when a job arrives to the throttle, then the job is allowed to pass through, and the bank is decremented by one token. If the bank does not contain a token when a job arrives, then the job queues in the job buffer if the buffer is not full. If the job arrives to a full buffer, then the job is rejected. Note that the rate-control throttle differs from sliding window flow control in that there is no constraint on the number of outstanding tokens; rather, tokens are used once and do not circulate back to the token bank.

The rate-control throttle has been used to regulate the requests of users to initiate a call or session. In this application, only the call-setup request is controlled, and a given throttle receives requests from many users. Moreover, during normal nonoverload conditions, the throttle is not turned on, and arriving jobs are not affected. When a monitor detects an overload, then the throttle is activated and remains on until the monitor determines the abeyance of the overload. As this paper investigates qualities of the throttle that are independent of the downstream system, the mechanism to detect the overload is not considered. (Of course, the design of the overall control would include a monitor.) Doshi and Heffes study the overall control for a star topology network with a monitor of the occupancy of the processor at the bottleneck, central node, and throttles at the peripheral nodes [1]. They compare a rate-control throttle containing a token bank but no job buffers versus a sliding window flow control. For a similar network configuration, Kumar describes a monitor that uses stochastic approximation to update the control settings [2].

Recently, the rate-control throttle has been suggested to regulate the packet flow for the duration of a call or session in broadband integrated services digital networks (B-ISDN's). In this application, a throttle is associated with each user and is active for the duration of the session. The bank capacity and token arrival rate are determined at call setup and are such that, with high probability, the throttle will not affect the user's traffic if it remains within agreed limits. Eckberg *et al.* use a leaky bucket (which is almost isomorphic to a token bank), for a throughput-burstiness filter for asynchronous transfer mode (ATM) cells of a B-ISDN [3]. Cells that arrive to an empty token bank are not blocked but rather are marked, are allowed through, and may be discarded if a subsequent node is congested. In [4], Sidi *et al.* also use a token bank for B-ISDN where the cells that arrive to an empty bank are not marked or blocked but rather are delayed in a job buffer, as is done in the present note. For Poisson job arrivals, they determine the Laplace–Stieltjes transform of the distribution of the cell waiting times and interdeparture times.

The contribution of the present note is to analyze the throughput and blocking of jobs for a general Markovian arrival process. Of particular interest is the robustness of the throughput to unknown job arrivals rates (and, more generally, to unknown job arrival processes), given constant parameter values of the throttle. In the B-ISDN application, these parameters would indeed be constant for the duration of the call. For the regulation

Manuscript received April 20, 1990; revised September 13, 1990. This paper was presented at INFOCOM, San Francisco, CA, June 1990.

The author is with AT&T Bell Laboratories, Holmdel, NJ 07733.

IEEE Log Number 9040752.

of call setups, the token arrival rate is typically updated periodically, and the arrival rate of jobs may change markedly within an update interval. For example, the users may reattempt if a previous request is blocked, and the total arrival rate seen by the throttle can increase significantly. Moreover, sometimes the number of possible values for the control setting is constrained and the designer must choose these values with care. The more robust the throttle, the less the designer need be concerned with the arrival rate of jobs, and the more the designer can tune for changes in the desired departure rate.

The following analysis assumes a stationary arrival process of jobs and thus is an approximation to the nonstationary arrival process of call setups with reattempts in the previous example. However, for sustained overloads of possibly unknown magnitude, the superposition process of first attempts and retries may approach a steady state and could be modeled reasonably as a stationary process.

II. Model of Rate-Control Throttle

Consider a constant control setting where the arrival rate of tokens is r. To present the results under general conditions, assume the token arrival process is renewal with interarrival times having a general distribution $F(\cdot)$ with finite mean r^{-1}. Later, we consider a special case where the token interarrival times are evenly spaced; $F(\cdot)$ has a jump of size 1 at r^{-1}. Assume the job arrival process is a Markovian arrival process (MAP) and is independent of the token arrival process. The MAP is introduced by Lucantoni *et al.* in [5] and is a class of semi-Markov processes that models a broad range of arrival processes, yet is analytically and numerically tractable. The generality of the MAP may prove useful for modeling the bursty packet arrival process in ATM networks; a special case of the MAP, the Markov-modulated Poisson process, has been useful for modeling stationary, packetized, voice, and data traffic [6]. Also, the MAP could model the presumably simplier arrival process of call-setup requests, which is frequently assumed to be a Poisson process. Although, since customers typically reattempt when their setup request is blocked, the resulting superposition of first attempts and reattempts is a complicated process [7]. The MAP could be used to partially characterize the correlations of the interarrival times of the stationary version of this superposition process and would be an improvement over the classical approximation of inflating the rate of a Poisson process.

The key quality of the MAP for the present note is that its future evolution is independent of the past, given its current state. The following section is an abbreviated description of the MAP. For more details, see [5].

A. Markovian Arrival Process

The MAP is a Markov process with a finite number of transient states m and one absorbing state. The epoch of absorption constitutes an arrival of the modeled job arrival process. Upon absorption, the MAP immediately restarts in a transient state, and the cycle repeats.

More formally, and using the notation in [5], if the MAP is in transient state i, $i \in 1, \cdots, m$, it will remain in i for a length of time that is exponentially distributed with parameter λ_i. At the end of this period, the process may move to another transient state or may be absorbed and then instantaneously restart in some transient state. Let q_{ij} be the probability the process goes to transient state j, given that is has been in transient state i. Let p_{ij} be the probability the process is absorbed and then restarts in state j, given that it has been in transient state i. Note that:

$$\sum_{j=1, j\neq i}^{m} q_{ij} + \sum_{j=1}^{m} p_{ij} = 1, \qquad 1 \le i \le m.$$

Let the matrix $P(n, t)$ denote the counting function whose ijth entry, $P_{ij}(n, t)$, is the probability the number of job arrivals over the interval $(0, t]$ is n and the state of the MAP at time t is j, given that the state of the MAP at time 0 is i. The z-transform of $P(n, t)$ with respect to n, $P^*(z, t) = \sum_{n=0}^{\infty} P(n, t) z^n$, is equal to:

$$P^*(z, t) = e^{(C+zD)t}, \qquad |z| \le 1, t \ge 0,$$

where C is an $m \times m$ matrix whose ijth entry equals $\lambda_i q_{ij}$ for $i \neq j$ and equals $-\lambda_i$ for $i = j$, and where D is an $m \times m$ matrix whose ijth entry equals $\lambda_i p_{ij}$.

As an example, when C and D are scalars and are equal to $-\lambda$ and λ, respectively, then the MAP simplifies to the Poisson process. The MAP becomes a renewal process with interevent times distributed as a two-branch hyperexponential if C and D are 2×2 matrices equal to:

$$C = \begin{bmatrix} -\lambda_1 & 0 \\ 0 & -\lambda_2 \end{bmatrix}, \qquad D = \begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} \cdot [\alpha_1 \; \alpha_2\},$$

$\alpha + \alpha_2 = 1$. The MAP becomes a Markov-modulated Poisson process, if with probability 1 the process restarts in the same state that it absorbed from, i.e., if the nonzero (positive) entries of D are along the diagonal.

III. Analysis of Rate-Control Throttle

Define the notation:

λ = the mean arrival rate of jobs.

λ' = the throughput of jobs that pass through the throttle, i.e., jobs that are not rejected.

By definition, in steady state λ' equals $\lambda \times [1 - \text{Prob(job is blocked)}]$. Moreover, since each job that passes through the throttle requires a token, and tokens either depart with a job or are lost from a full token bank, λ' also equals the rate that tokens depart with jobs, which equals $r \times [1 - \text{Prob(token is blocked)}]$. Thus, for token bank and job buffer capacities that are finite,

$$\begin{aligned} \lambda' &= \lambda \times [1 - \text{Prob(job is blocked)}] \\ &= r \times [1 - \text{Prob(token is blocked)}]. \end{aligned} \tag{1}$$

Using an embedded Markov renewal process, we can determine the Prob(token is blocked). Then, from (1), the throughput and blocking of jobs is known trivially.

A. State Evolution Equation

Define the notation:

T_n = arrival epoch of the nth token. $T_0 = 0$.

$X(t)$ = the number of tokens in the token bank at time t. It is convenient to view $X(t)$ as left continuous. For example, $X(T_n)$ = number of tokens in the token bank just before the arrival of the nth token. $X(T_n^+)$ = number of tokens in the token bank just after the arrival of the nth token. Let X_n denote $X(T_n)$.

$Y(t)$ = the number of jobs in the job buffer at time t. $Y_n = Y(T_n)$.

$J(t)$ = the state of the MAP at time t. $J_n = J(T_n)$. $J(t) \in \{1, \cdots, m\}$.

V_{n+1} = the number of job arrivals over the interval $(T_n, T_{n+1}]$.

C_J = capacity of the job buffer. C_J is finite.

C_T = capacity of the token bank. C_T is finite.

The operation of the throttle implies that the token bank and the job buffer do not simultaneously contain entities. Thus, we can define:

$$Z(t) = \begin{cases} X(t) & \text{if } Y(t) = 0 \\ -Y(t) & \text{if } X(t) = 0 \end{cases} \qquad Z_n = Z(T_n).$$

Note that, given $J(t)$, the future stochastic behavior of the MAP is independent of the past and, thus, $\{Z(t), J(t)\}$ is a semi-regenerative process, where the T_n's are regeneration points and the sequence $\{Z_n, J_n, T_n\}$ is an embedded Markov renewal process [8].

The state-evolution equation of Z_n follows directly from the definition of the throttle. In particular, suppose $Z_n \geq 0$, then at time T_n there exist Z_n tokens in the token bank. A token then arrives, and the number of tokens increase by 1 if there is room in the token bank. Thus, at T_n^+ the number of tokens in the bank is $\min(Z_n + 1, C_T)$. Then, over the interval $(T_n, T_{n+1}]$, V_{n+1} jobs arrive. With each job arrival, the number of tokens $Z(t)$ is decremented by 1. Enough jobs may arrive so that all of the tokens are used, and jobs begin to queue in the job buffer, in which case $Z(t)$ becomes negative. However, the queue of jobs is limited by the capacity of job buffer C_J. Thus, $Z(t)$ can become no more negative then $-C_J$. In summary, at T_{n+1}, after the V_{n+1} job arrivals,

$$Z_{n+1} = \max\left(\min(Z_n + 1, C_T) - V_{n+1}, -C_J\right), \quad C_T \geq 0, \quad C_J \geq 0. \tag{2}$$

If $Z_n < 0$, then an abbreviated version of the above argument pertains.

Note that if $C_J = 0$ and $C_T > 0$, then there is no buffer for jobs, which is the standard case for the rate-control throttle. Likewise, if $C_T = 0$ and $C_J > 0$, then jobs but not tokens queue. In this case, one can view the throttle as a gate on the job queue that opens only for an instant (the instant of a token arrival) and allows one of the queued jobs to pass through, if any are present.

B. Semi-Markov Kernel

Define:

$A(k, t)$ equals an $m \times m$ matrix whose ijth element equals the probability the token interarrival time, $T_{n+1} - T_n$, is less than t, and k jobs arrive during (T_n, T_{n+1}), and the MAP is in state j at time T_{n+1}, given that the MAP is in state i at time T_n.

$B(k, t)$ equals $\Sigma_{n=k}^{\infty} A(n, t)$.

Conditioning on the token interarrival time, $A(k, t)$ can be expressed as:

$$A(k, t) = \int_0^t P(k, s)dF(s)$$

where $P(k, s)$ is the counting function of the MAP defined in Section II-A. Ordering the states (Z_n, J_n) as $(-C_J, 1), (-C_J, 2), \cdots, (-C_J, m), (-C_J + 1, 1), \cdots$, the semi-Markov kernel, denoted $Q(t)$, is the following block martix, where i ranges from $-C_J$ to $C_T - 1$.

	$-C_J$	$-C_J + 1$	$-C_J + 2$	$\cdots$ $i+1$ $\cdots$	C_T
$-C_J$	$B(1, t)$	$A(0, t)$	0	0	0
$-C_J + 1$	$B(2, t)$	$A(1, t)$	$A(0, t)$	0	0
$-C_J + 2$	$B(3, t)$	$A(2, t)$	$A(1, t)$	0	0
$\cdots$					
i	$B(C_J + 1 + i, t)$	$A(C_J + i, t)$	$A(C_J + i - 1, t)$	$A(0, t)$	0
$\cdots$					
$C_T - 1$	$B(C_J + C_T, t)$	$A(C_J + C_T - 1, t)$	$A(C_J + C_T - 2, t)$	$A(C_T - 1 - i, t)$	$A(0, t)$
C_T	$B(C_J + C_T, t)$	$A(C_J + C_T - 1, t)$	$A(C_J + C_t - 2, t)$	$A(C_T - 1 - i, t)$	$A(0, t)$

C. Equivalence to GI/MAP/1/K System

Lemma 1: The rate-control throttle with jobs arriving as a MAP is equivalent to a GI/MAP/1/K queueing system where the server operates in the following nonstandard manner:

> upon a service completion that leaves the system empty, the server continues to operate; the MAP continues to transit among the m states, possibly being absorbed and restarting.

Comment: The nonstandard feature of the server corresponds in the throttle to the jobs continuing to arrive, irrespective of the token arrival process. Furthermore, the absorption and restarting of the MAP of the server while the system is empty corresponds in the throttle to the arrival of a job that is blocked and rejected at a full job buffer. As is standard, an arrival to the GI/MAP/1/K system that finds the system empty immmediately enters service. Its service time is the time until the next absorption of the MAP. This time has the same distribution as the time between absorptions of the MAP, given common starting states of the MAP. Loosely speaking, the throttle is a GI/MAP/1/K queue with the state variable of the throttle, Z_n, shifted up by C_J.

Proof: The candidate GI/MAP/1/K queue has an interarrival time distribution G equal to the token interarrival time distribution F, has the parameters of the service process equal to those of the job arrival process to the throttle, and has a system capacity K equal to $C_J + C_T$. Let:

$U(t)$ = the number of entities in the GI/MAP/1/K system at time t. View $U(t)$ as left continuous. Thus, let $U(T_n)$ = the number of entities in the system just prior to the nth arrival.

$L(t)$ = the state of the MAP of the server of the GI/MAP/1/K system at time t.

We show that the joint stochastic process $\{U(t), L(t)\}$ is equivalent in sample paths to the stochastic process $\{Z(t) + C_J, J(t)\}$. For any sample path of the renewal process with interevent time distribution F, the epochs of token arrivals to the throttle coincide with the epochs of arrivals to the GI/MAP/1/K queue. Likewise, for any sample path of the MAP, then $L(t) = J(t)$. As for $U(t)$ and $Z(t)$, first consider the embedded epochs. $U(T_n)$, denoted U_n, evolves according to the state equation:

$$U_{n+1} = \max\left(\min\left(U_n + 1, K\right) - V_{n+1}, 0\right), \qquad (3)$$

where, as before V_{n+1} equals the number of absorption of the MAP in the interval $(T_n, T_{n+1}]$. Making the variable substitutions $K = C_J + C_T$ and $U_n = Z_n + C_J$, then (3) becomes:

$$Z_{n+1} + C_J = \max\left(\min\left(Z_n + C_J + 1, C_J + C_T\right) - V_{n+1}, 0\right).$$

Subtracting C_J from both sides and bringing C_J inside the $\max(\cdot)$ and the $\min(\cdot)$ yields:

$$Z_{n+1} = \max\left(\min\left(Z_n + 1, C_T\right) - V_{n+1}, -C_J\right),$$

which is identical to the state evolution equation (2). Furthermore, for t between T_n and T_{n+1}, $Z(t)$ and $U(t)$ each evolve by the common MAP. Thus, the two joint stochastic processes $\{U(t), L(t)\}$ and $\{Z(t) + C_J, J(t)\}$ are equivalent in sample paths. □

D. The Blocking of Jobs is a Function of $C_J + C_T$

Theorem 1: For jobs arriving according to a Markovian arrival process (MAP) and tokens arriving according to a renewal process that is independent of the job arrival process, then the probability a job is blocked depends on the capacity of the job buffer, C_J, and the capacity of the token bank, C_T, only via the sum of the two capacities, $C_J + C_T$.

Proof: The proof uses the simple observation that for constant $C_J + C_T$ the semi-Markov renewal kernel $Q(t)$ is invariant. Consider the Markov chain associated with the Markov renewal process $\{Z_n, J_n, T_n\}$ whose transition matrix is the limit $t \to \infty$ of the semi-Markov kernel $Q(t)$. This Markov chain is positive recurrent, aperiodic, and irreducible and hence its limiting distribution equals its stationary distribution, denoted by ν. ν is invariant over C_J and C_T, given that $C_J + C_T$ is constant. The sum of the last m components of ν is the probability a token is blocked, which from (1) determines the throughput of jobs and the probability a job is blocked. Thus, the probability a job is blocked depends on C_J and C_T only via their sum, $C_J + C_T$. □

The key implication of Theorem 1 is that the job buffer can be eliminated without affecting the steady-state throughput and blocking of jobs, as long as the token bank capacity is increased by what had been the capacity of the job buffer. One advantage of eliminating the job buffer is that jobs that pass through the throttle are not delayed. The presence of a job buffer introduces the performance degradation of additional delay to those jobs that are admitted; in contrast, the delay of tokens is not a performance degradation. (The delay of jobs is examined in [9].)

Note that the blocking probability in Theorem 1 is a steady-state concept and is independent of the initial state $\{Z_o, J_o\}$. Thus, Theorem 1 implies that for two rate-control throttles, each with the same value for $C_T + C_J$ and each driven by the same MAP of jobs and renewal process of tokens, the blocking probabilities are the same for the two sytems regardless of the initial state. However, in the special case where the two J_o's are the same and where the difference in the two Z_o's equals difference in token bank capacities, then one can make the stronger statement that for each sample path the epochs of blocking coincide for the two systems.

Theorem 2: For two rate-controls throttles, denoted "a" and "b," with buffer capacities C_J^a, C_T^a and C_J^b, C_T^b, respectively, such that $C_J^a + C_T^a = C_J^b + C_T^b$, then for a common MAP of jobs and renewal process of tokens, the epochs at which jobs are blocked coincide for the two systems, as do the epochs at which tokens are blocked, for all sample paths if the initial states are related as:

$$Z_o^a + C_J^a = Z_o^b + C_J^b, \quad J_o^a = J_o^b \qquad (4a)$$

or, equivalently,

$$Z_o^a - C_T^a = Z_o^b - C_T^b, \quad J_o^a = J_o^b. \qquad (4b)$$

Proof: The proof is based on the observation that the two rate-control throttles are equivalent to the same GI/MAP/1/K queue. From Lemma 1, each rate-control throttle is equivalent in sample paths to a GI/MAP/1/K queueing system with a continuous server. Each of the two queueing systems have the same capacity, $K = C_J^a + C_T^a = C_J^b + C_T^b$ and are driven by the same MAP of jobs and renewal process of tokens. Thus, if the two queueing systems have common intial states, then their sample paths would be the same. Let $U^a(t)$ = the number of entities at time t that are in the queueing system that is equivalent to rate throttle "a" and, likewise, for $U^b(t)$. From the proof of Lemma 1, $U^a(t) = Z^a(t) + C_J^a$ and $U^b(t) = Z^b(t) + C_J^b$. Thus, from (4a), at time $t = 0$, $U^a(0)$ equals $U^b(0)$. Also, the initial states of the MAP's are given to be equal. Thus, the two rate-control throttles are equivalent in samples paths to the same GI/MAP/1/K queueing system. In particular, $U^a(t) = U^b(t)$ and, thus,

$$Z^a(t) + C_J^a = Z^b(t) + C_J^b \quad t \geq 0 \qquad (5a)$$

or, equivalently,

$$Z^a(t) - C_T^a = Z^b(t) - C_T^b \quad t \geq 0. \qquad (5b)$$

Consider the epochs at which tokens are blocked. Tokens arrive at the embedded points, T_n, $n = 0, 1, \cdots$, and are blocked iff the token bank is full. For example, a token is blocked in system "a" at time T_n iff $Z^a(T_n) = C_T^a$ and, likewise, for system "b." Moreover, (5b) implies that $Z^a(T_n) = C_T^a$ iff $Z^b(T_n) = C_T^b$. Thus, the epochs at which tokens are blocked coincide for the two systems. Similarly, a job is blocked in system "a" at time t iff $Z^a(t) = -C_J^a$ and, likewise, for system "b." Equation (5a) implies that $Z^a(t) = -C_J^a$ iff $Z^b(t) = -C_J^b$. Thus, the epochs which jobs are blocked coincide for the two systems. □

IV. Application and Discussion of Analysis

A. Robustness to Job Arrival Process

In [10], the robustness of a rate-control throttle with a token bank and no job buffer is studied. It is shown that in overload and for a fixed control settting, the departure rate from the throttle is surprisingly insensitive to the arrival rates beyond the maximum desired value, if the capacity of the token bank is 10 or more. With a capacity of 10 or more and for Poisson arrivals, then as a practical matter for the application of regulation of

call-setup requests in telecommunication switching systems, one can ignore the arrival rate and adjust r only for changes in the desired job departure rate [10]. From Theorem 1 above, we see that the presence of a job buffer, per se, does not increase this robustness. A token bank alone yields the same robustness if its capacity is augmented by what would have been the capacity of the job buffer. Fig. 2 illustrates this robustness. For hyperexponential job interarrival times, deterministic token interarrival times, and for r held fixed, Fig. 2 plots the normalized departure rate λ'/r, versus the normalized arrival rate of jobs, λ/r, indexed by $C_J + C_T$. The computation of the departure rate requires the computation of the block matrices of $Q(\infty)$, for which an algorithm by Lucantoni and Ramaswami was used [11]. Fig. 2 shows the substantial improvement in robustness as $C_J + C_T$ increases from 1 to 10.

Fig. 3 shows how the robustness is affected by changes in the parameters of the hyperexponential distribution of job interarrival times. (To show the details, the scale differs from that in Fig. 2.) Note that for Poisson job arrivals, the departure rate is close to ideal. As a possibly unexpected result, note that for a common coefficient of variation, cases 2 and 4, the case with higher skewness is closer to the ideal for λ around r, while the case with lower skewness is closer to the ideal for $\lambda >> r$. Informally, one might think that (1) higher skewness is more "stressful," and also that (2) over the class of H_2 distributions, the limiting case of batch Poisson with geometric batch sizes is the most "stressful." However, both informal notions cannot be true simultaneously since the batch Poisson with geometric batch sizes has the *minimum* skewness over the class of H_2 distributions.

An intuitive explanation can be given for the reversal in cases 2 and 4. Consider the scenario where $\lambda >> r$. Here, the job buffer is frequently full, and the token bank is lightly loaded and is frequently empty. Arriving tokens are blocked only when there is an atypically long job interarrival time. Such interarrival times occur more frequently when the distribution has a long tail. This occurs at high skewness for given λ and c^2. Thus, for $\lambda >> r$, higher skewness of job interarrival times causes higher blocking of tokens and a lower throughput of jobs for given c^2.

Now consider the opposite scenario of $\lambda < r$. Here, the token bank is frequently full. In contrast, for the job buffer to be full, a number of jobs need to arrive closely spaced. Typically, this would occur if one of the branches of the H_2 distribution has a high arrival rate and if that branch has a nonnegligible probability of occurring. This is the case for low skewness, with the limiting case of one branch with infinite arrival rate: batch Poisson arrivals with geometric batch sizes. Thus, given $\lambda < r$, if the job interarrival times have a low skewness, then the job buffer is more likely to be full at job arrival epochs, and thus an arriving job is more likely to be blocked, and the throughput of jobs will be lower. Thus, for $\lambda < r$, lower skewness of job interarrival times causes a lower throughput of jobs for given c^2.

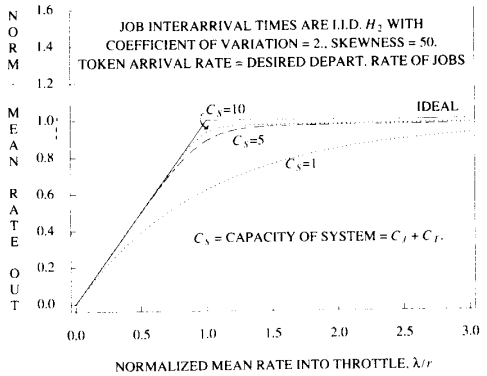

Fig. 2. Normalized mean departure of jobs λ'/r versus arrival rate, given the control settting is fixed: indexed by system capacity.

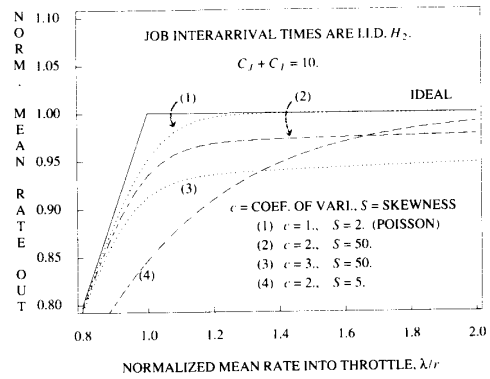

Fig. 3. Normalized mean departure rate of jobs λ'/r versus arrival rate, given the control setting is fixed: indexed by second and third moments of job arrival process.

B. Short-Term Congestions and False Alarms

Given the above, one can ask whether there are any advantages in having a job buffer. One advantage is the reduction of the negative consequences of false alarms by the monitor. This advantage is potentially useful for the regulation of call-setup requests. Suppose that the throttle is turned on only when a monitor detects an overload in the system or network, and otherwise is inactive. Typically, the monitor detecting the overload is not perfect and will occasionally give false alarms. This occurs when the overall arrival rate is nominal, but there has been a statistically random clump of call of call-setup requests that temporarily congests the system. In this case, the system will clear itself without any action being taken. Nevertheless, the monitor detects an overload and the throttle needlessly turns on. The throttle design can take these false alarms into account by initializing the token bank with an appropriate number of tokens. In the case of no job buffer, the designer is faced with a performance tradeoff. By initializing the token bank with a "large" number of tokens, the effects of the false alarm is minimized because the next few job arrivals will get through and, with the next update from the monitor, the downstream system may have cleared and the throttle will turn off. On the other hand, the designer would like to initialize the token bank with a "small" number of tokens so that, during a true overload (not a false alarm), the throttle will activate quickly. By introducing a buffer for the jobs, this performance tradeoff is partially avoided. The designer can initialize the token bank with a small number of tokens so that the throttle will activate quickly for a true overload and, in the case of a false alarm, rather than having jobs needlessly blocked and rejected, they are delayed in a queue. However, during a sustained overload when all jobs cannot be served, it is preferable in some systems, such as telecom-

munications switching systems, not to delay those call setups that are served. A possible adaptive implementation is to have the throttle turn on with a large capacity for the job buffer when the monitor first detects an overload. If the congestion continues, then the throttle reduces the capacity of the job buffer to zero and increases the capacity of the token bank.

C. Shaping the Output Process

A second use for the job buffer is to shape the output process from the throttle. This has the potential for the application to ATM B-ISDN's where an end device shapes its traffic to stay within the characteristics agreed to with the network when the session was initiated [3]. Note that Theorem 1 above pertains to the first moment of the output process. For given total capacity $C_T + C_J$, the nature of the output process *does* depend on the individual values of C_T and C_J [4]. The potential for shaping the output process is relevant to both stationary and nonstationary job arrival processes, though the latter is more challenging to analyze.

V. Conclusion

This note has examined a rate-control throttle where arriving jobs queue in a finite buffer if the token bank is empty. We have shown that the steady-state throughput and blocking of jobs is invariant to the presence of the job buffer, as long as the token bank capacity is increased by what would have been the capacity of the job buffer. Thus, the job buffer per se is not needed to enhance the robustness of the throughput of the throttle to unknown job arrival rates. Overall, the job buffer is not a useful enhancement to a pure work-rejection throttle. However, the presence of a job buffer along with a token bank and adjustable capacities would enable: (1) a tuning of the output process, which may be useful in applications to B-ISDN; and (2) an overload control that adapts between a pure delay control and a pure work-rejection control.

Acknowledgment

The author would like to thank D. Lucantoni for thoughful and stimulating discussions and to thank the anonymous reviewers for their constructive comments.

References

[1] B. T. Doshi and H. Heffes, "Analysis of overload control schemes for a class of distributed switching machines," in *Proc. 10th Int. Teletraffic Cong.*, Montreal, Canada, 1983, paper 5.2.2.

[2] A. Kumar, "Adaptive load control of the central processor in a distributed system with a star topology," *IEEE Trans. Comput.*, vol. 38, pp. 1502–1512, Nov. 1989.

[3] A. E. Eckberg, D. T. Luan, and D. M. Lucantoni, "Bandwidth management: A congestion control strategy for broadband packet networks—Characterizing the throughput-burstiness filter," *Int. Teletraffic Cong. Specialist Sem.*, Adelaide, Australia, Sept. 1989, paper 4.4.

[4] M. Sidi, W. Z. Liu, I. Cidon, and I. Gopal, "Congestion control through input rate regulation," in *Proc. GLOBECOM'89*, Dallas, TX, Nov. 1989, pp. 1764–1768.

[5] D. M. Lucantoni, K. S. Meier-Hellstern, and M. F. Neuts, "A single server queue with server vacations and a class of non-renewal arrival processes," *Advances Appl. Prob.*, Sept 1990.

[6] H. Heffes and D. M. Lucantoni, "A Markov modulated characterization of packetized voice and data traffic and related statistical multiplexer performance," *IEEE J. Select. Areas Commun.*, vol. 4, pp. 856–868, Sept. 1986.

[7] P. K. Reeser, "Simple approximation for blocking seen by peaked traffic with delayed, correlated reattempts," in *Proc. 12th Int. Teletraffic Cong.*, Torino, Italy, June 1988, paper 3.1B.5.

[8] E. Cinlar, *Introduction to Stochastic Processes*. Englewood Cliffs, NJ: Prentice-Hall, 1975.

[9] A. W. Berger, "Performance analysis of a rate control throttle where tokens and jobs queue," in *Proc. INFOCOM'90*, San Francisco, CA, June 1990, pp. 30–38.

[10] A. W. Berger, "Overload control using rate control throttle: Selecting token bank capacity for robustness to arrival rates," *IEEE Trans. Automat. Contr.*, vol. 36, Feb. 1991.

[11] D. M. Lucantoni and V. Ramaswami, "Efficient algorithms for solving the non-linear matrix equations arising in phase type queues," *Commun. Stat.—Stochast. Models*, vol. 1, pp. 29–51, 1985.

Arthur W. Berger (S'82–M'83) was born in New York City on April 17, 1953. He received the B.S. degree in mathematics from Tufts University, Medford, MA, in 1974, and the M.S. and Ph.D. degrees in applied mathematics from Harvard University, Cambridge, MA, in 1980 and 1983, respectively.

Since 1983, he has been a Member of Technical Staff at AT&T Bell Laboratories, Holmdel, NJ, where he has worked on network planning and the performance analysis of telecommunication switching systems. His research interests are in the control of queueing systems with application to the analysis and design of overload controls and of resource allocation schemes.

Dr. Berger is a member of the IEEE Communications Society and the IEEE Control Systems Society.