A Distributed Overload Control Algorithm for Delay-Bounded Call Setup

R. Radhakrishna Pillai, *Member, IEEE*

***Abstract*—As communication networks provide newer services, signaling is becoming more and more compute intensive compared to present day networks. It is known that under overload conditions, the call throughput (goodput) and the network revenue drops to zero even when transport resources are available. A distributed overload control algorithm for delay-bounded call setup is proposed in this paper. The end-to-end delay bound is budgeted among the switching nodes involved in call setup, and these nodes apply a local overload control with a deterministic delay threshold and drop call requests experiencing higher delays. This algorithm does not depend on feedback on network conditions and makes use of only parameters that can be instrumented locally by the switching node. Using an $M/M/1$ queueing model with first-in-first-out (FIFO) service discipline at a switching node, two optimized control schemes are considered for local overload control and compared their performance through analysis: one with arrival rate limit and the other with buffer size limit. Though both the schemes minimize the unproductive call processing at heavy load, the latter is found to yield higher call throughput and lower average call setup delays compared to the former. Also, the buffer size required for the scheme with buffer size limit is typically small and call throughput close to the server capacity can be achieved during overload. The performance of the distributed overload control algorithm in a network is evaluated through simulation experiments, using the scheme with buffer size limit for the local overload control. It shows that the average end-to-end delay could be much less than the end-to-end delay bound, providing room for overprovisioning of the delay bounds. The tradeoff between the number of nodes, call throughput, and average end-to-end delay needs to be considered while deciding the route budgeting the end-to-end delay bound among different nodes along the route. These performance results are expected to serve as lower bounds to more sophisticated local call rejection mechanisms such as push-out or time-out along with a last-in-first-out (LIFO) service discipline.**

***Index Terms*—Call overload control, communication networks, distributed algorithm, signaling performance.**

NOMENCLATURE

λ	Average call arrival rate.
$1/\mu$	Average call processing time.
K	Number of buffers in the queue.
D	Total delay in a queue (queueing delay), including the call processing time.
γ	Call/signaling throughput of the queue (carried load).
q_o	Call overflow probability at the queue due to buffer overflow.
q_l	Call loss probability at the queue due to excessive delay.
q_b	Call blocking probability due to insufficient transport resources.
ρ	$= \lambda/\mu$. Traffic intensity of the queue.
d	Threshold (bound) on total delay at a switching node.

Manuscript received March 5, 1999; revised October 11, 1999 and March 24, 2001; recommended by IEEE/ACM TRANSACTIONS ON NETWORKING Editor R. Rom. A preliminary version of this paper appeared in the proceedings of the *Applied Telecommunication Symposium* (part of the 1998 Advanced Simulation Technologies Conference ASTC'98), Boston, MA, April 5–9, 1998.

The author is with the Kent Ridge Digital Labs, Singapore 119613 (e-mail: pillai@ieee.org or pillai@krdl.org.sg).

Publisher Item Identifier S 1063-6692(01)10552-2.

I. INTRODUCTION

SIGNALING is an essential component of connection-oriented networks such as the public switched telecommunication networks (PSTNs) including integrated services digital networks (ISDNs), asynchronous transfer mode (ATM) networks, and mobile communication networks. Connectionless networks such as local area networks (LANs) and the Internet are introducing signaling protocols such as the resource reservation protocol (RSVP) to set up end-to-end flow or connection [20]. As communication networks provide newer services, signaling becomes more compute intensive. For example, the requirement to provide quality-of-service (QoS) guarantees to accepted calls demands for call-level admission control and reservation of resources on a link-by-link basis. In the case of mobile communication networks, the requirement to provide seamless communication to mobile users necessitates handoff and location management which in turn increase the call processing. The experiments reported in [3] show that like the transport capacity, the ability of a network to admit calls is also limited by the processing capacity of the signaling system. In [10], it has been shown that the processing capacity and the admission control functions can affect call setup delay and accepted call throughput (goodput) significantly. In order to reduce the signaling or call processing load, sometimes network resource provisioning mechanisms such as virtual path connections (VPCs) in ATM networks [2] or differentiated services in the Internet [7], are used to provision the transport resources for a group of connections together. However, it can result in significant wastage of network capacity [19]. Though there are several studies including standardization [5] on call admission control (CAC) to manage the transport resources in a network, much needs to be done on the management of call processing resources including overload control. The impact of managing the call processing resources on the Internet performance also needs to be understood because of the recent interest in the signaling and control for end-to-end QoS. The priority with which signaling messages are processed depends on the signaling event such as call setup, call release, handoff, etc. The signaling messages are processed by the

switching nodes in the case of connection oriented networks and by routers in the case of Internet. However, these signaling messages may be processed only at the access networks and be transported through a DiffServ-based core network in a transparent manner, without any processing. This paper focuses on processing the call setup messages in a fixed switched network. However, some of the results are equally applicable to mobile switch/router-based networks and to other signaling events as well.

Typical call-level QoS measures include call blocking probability and call setup delay. The call processing involves processing of a number of messages and, hence, call setup delay is mainly decided by the call processing resources of the network. However, a call could be blocked due to unavailability of either call processing resources or transport resources or both and hence the call blocking probability is decided by both these factors. Both the blocking probability and the call setup delay are affected by the call arrival statistics. Congestion or overload in the network can give rise to unwanted effects such as long delays and lost calls due to customer impatience and time-outs of signaling protocols. Such delayed or failed service will lead to displeased customers. The situation is worsened by the fact that incidents of this kind caused by a temporary, light overload may cause repeated attempts which will increase loads further, and results in complete service disruption. Maximizing successful signaling sessions is key to maximizing network revenue.

There are several papers in the literature dealing with performance of a single server queue with impatient customers under various service disciplines. The delay distributions and the throughput of "good" customers under last-in-first-out (LIFO) and first-in-first-out (FIFO) schemes with customer rejection mechanisms corresponding to pushing out or timing out old customers in an $M/M/1$ queue with each customer turning "bad" at a random time after its arrival, are compared in [9]. Reference [21] studies the optimality of queueing policies for nonpreemptive queues with impatient customers and shows that LIFO is an optimal service order when the deadlines are *i.i.d.* random variables with a concave distribution function. It also proves that when customer waiting times are unknown, an optimal policy for an $M/M/1$ queue becomes the LIFO-PO (push-out) policy, with a fixed buffer used as rejection threshold. Reference [8] shows that for a FIFO service discipline, to minimize the probability of losing messages with deadlines, an admission policy rejecting messages before link assignment is optimal when the load exceeds a critical value. In a queueing system having customer dead lines that are *i.i.d* random variables with concave cumulative distribution functions, it is shown in [12] that LIFO gives the highest probability of success, and FIFO the lowest, over the class of all work-conserving nonpreemptive service disciplines that are independent of service time and deadline. A closed loop approach to network overload control is adopted in [14], [18] where delays of completed signaling sessions are used to predict the delay and to take the decision to accept or drop new session requests. The use of input buffer limits for congestion control of store-and-forward networks is investigated in [13]. It attempts to control the network input rate by differentiating between input and transit traffic at each node and imposing a limit on the fraction of buffers in a node buffer pool that input traffic can occupy.

A distributed overload control algorithm for delay-bounded call setup is proposed in this paper. The end-to-end delay bound is budgeted among the switching nodes involved in call setup, and these nodes apply a local overload control with a deterministic delay threshold and drop call requests experiencing higher delays. This algorithm does not depend on feedback on network conditions and makes use of only parameters that can be instrumented locally by the switching node. Using an $M/M/1$ queueing model with FIFO service discipline at a switching node, two optimized control schemes are considered for local overload control and compared their performance through analysis: one with arrival rate limit, considered in [8] and the other one with buffer size limit. An unoptimized version of the latter scheme has been considered in [9] as FIFO-blocking (FIFO-BL). Though both the schemes minimize the unproductive call processing at heavy load, the latter is found to yield higher call throughput and lower average call setup delays compared to the former. Also, the buffer size required for the scheme with buffer size limit is typically small and call throughput close to the server capacity can be achieved during overload. The performance of the distributed overload control algorithm in a network is evaluated through simulation experiments, using the scheme with buffer size limit for the local overload control. It shows that the average end-to-end delay could be much less than the end-to-end delay bound, providing room for overprovisioning of the delay bounds. The tradeoff between the number of nodes, call throughput, and average end-to-end delay needs to be considered while deciding the route budgeting the end-to-end delay bound among different nodes along the route. These performance results are expected to serve as lower bounds to more sophisticated local call rejection mechanisms such as push-out or time-out along with a LIFO service discipline.

Section II describes the algorithm for overload control. Local overload control mechanisms at a switching node are analyzed and compared in Section III. Experimental results from a network with several nodes, are discussed in Section IV, and conclusions are drawn in Section V.

II. Distributed Overload Control Algorithm

Assume that the switching nodes involved use a signaling protocol that sets up the call sequentially, i.e., hop-by-hop. An example is the ATM Forum private network-to-network interface (PNNI) signaling protocol [5]. The basic idea behind the algorithm is budgeting the end-to-end call setup delay among the switching nodes and applying an appropriate local queueing control at each node with a delay threshold. Every switching node queues up call setup requests. If the queueing delay exceeds the threshold, these requests will be dropped. Otherwise, these requests will be allowed to proceed to the next node for processing. When a call setup request proceeds to the next node, it carries information on the number of nodes already traversed and the accumulated setup delay. The local queueing control schemes that maximize signaling throughput under delay threshold are examined in detail in Section III.

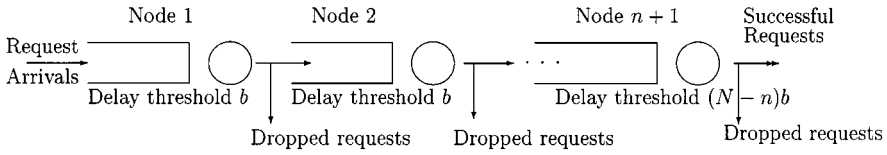

Fig. 1. Illustration of the algorithm using distributed queues.

A possible realization of the algorithm to bound the end-to-end call setup delay in a network with arbitrary number of switching nodes, is described below. For end-to-end call setup, the setup request has to be processed at all nodes involved in the route. Let N be the maximum number of nodes involved in call setup and B be the required upper bound on the end-to-end call setup delay. Assuming that this end-to-end delay bound is budgeted equally among the nodes, the delay threshold at a node is $b = B/N$. Depending on the destination of calls, classify the call setup requests processed at a switching node into two categories—terminating requests and transit requests. Terminating requests at a node correspond to calls destined to subscribers connected or attached to that node. Transit requests arriving at a node correspond to calls destined to subscribers connected or attached to other nodes. Depending on the position of a node in the network, it may process only transit requests (for example, core network switching nodes) or both transit and terminating requests (for example, access network switching nodes). In the case of a transit request, the switching node may not know how many more switching nodes are involved in the call setup. Therefore, it is better to have the same threshold on the queueing delay for all transit requests, irrespective of the origin/destination of the call. In signaling protocols such as UNI/PNNI or RSVP, it is difficult to predict in advance the exact number of switching nodes or routers the call setup request will traverse. However, it is easier to capture the number of switching nodes where the setup message has been processed and the accumulated delay. This information could be useful in the processing of terminating requests.

A call setup request is processed as a transit request at all nodes except at the egress node, by applying an overload control with delay threshold b, independent of the source of the call. Thus, the delay of a transit request at a switching node, when successful, is bounded by b. When a call request arrives at the egress switching node, it is treated as a terminating request. From the call setup message, the egress node knows the entire route the call setup has taken. A terminating request is queued in one of the N queues, depending on the number of nodes participated in the call setup. These queues apply overload control independently with delay thresholds $b, 2b, \ldots,$ and Nb respectively. For example, when a terminating request that has already been processed by n preceding nodes arrives, that request is placed in a queue that applies overload control with delay threshold $(N-n)b$ because the accumulated delays at all the preceding switching nodes together is bounded by nb (Fig. 1). If the delay in this queue exceeds the threshold, the call setup is not allowed to complete. Thus, the end-to-end call setup delay, when successful, will be upper bounded by $nb + (N-n)b = Nb = B$. When the call arrival process to a switching node is Poisson and the processing time is exponentially distributed, the overload control schemes analyzed in Section III can be used to bound the delay.

This method does not make use of feedback or network condition and does not require any network-wide global measurements and co-ordination between queues at different nodes. All the parameters used for queueing control can be obtained locally. As shown by the experimental results in Section IV, this can result in overprovisioning of call processing resources. However, the information on the number of switching nodes participated in the call setup and the accumulated delay contained in the call setup request can be used to further improve the utilization of call processing resources. When there are different classes of signaling sessions with different delay requirements, the requests have to be queued differently and appropriate delay threshold have to be applied for each class. The processing capacity available at a switching node could be partitioned between processing of terminating and transit requests and among terminating requests traversed through different number of switching nodes. This should take fairness for various source–destination pairs into account. These issues are not addressed and are beyond the scope of this paper.

III. Local Overload Control With a Delay Threshold

To realize the end-to-end call setup delay bound described in Section II, it is essential that each node implements an appropriate queueing control with a delay threshold. This section examines two such queueing control strategies.

It is assumed that the call arrival process is Poisson with rate λ and the call processing time is exponentially distributed with mean $1/\mu$. Poisson model is widely used to model the call arrival process in current telephone networks and it is believed that it will be adequate for modeling the call-level behavior in other networks as well [15].

With infinite buffers, call requests that are not processed immediately are queued up for processing. At the end of processing, they will either be admitted or be blocked based on the availability of transport resources. Since the call arrival process is Poisson and the call processing time is exponentially distributed, the queueing system is modeled as an $M/M/1$ queueing system. The $M/M/1$ queueing model has been used before in the literature to model the behavior of signaling processors [15]. The average queueing delay including the processing time is

$$E[D] = \frac{1}{\mu - \lambda}$$

where $1/\mu$ is the average call processing time. The queueing delay in the queue D is exponentially distributed with mean $1/(\mu - \lambda)$. A call request is lost if its queueing delay exceeds

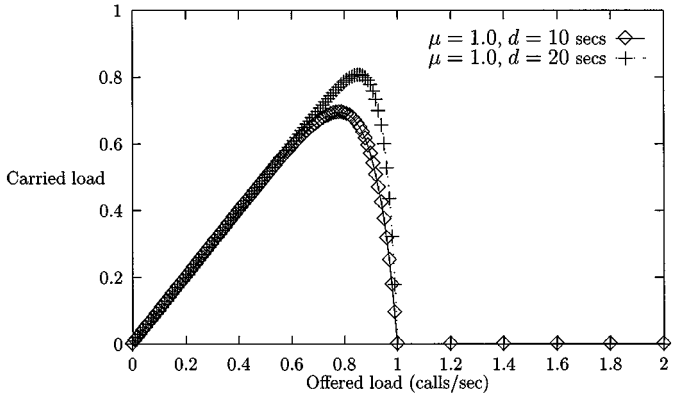

Fig. 2. Carried load of the queueing system.

the threshold d. Whether a call request is lost or not, the server serves it, i.e., the call request always consumes the processing resources. Therefore, the probability that a call request is lost due to excessive delay, q_l, is given by

$$q_l = \text{Prob}\{D > d\} = e^{-\mu(1-\rho)d}.$$

Therefore, the carried load of the queue or call throughput which is defined as the rate at which call requests are processed with delay less than or equal to d, is given by

$$\gamma = \lambda(1 - q_l) = \lambda\left[1 - e^{-\mu(1-\rho)d}\right] \tag{1}$$

and it is plotted in Fig. 2. As shown in the figure, call throughput drops to zero when λ approaches μ. This is true for all finite values of d. This indicates that all call requests are lost when λ approaches μ even when the transport resources are available. This behavior closely agrees with the experimental observations reported in [3].

This phenomenon is due to the failure of the call processing system in meeting the call setup delay requirement. Not only do the failed call requests fail to generate an income, but the fact that they consume processing resources while in the network, means that they actively contribute to delaying other call requests and thus further increase the number of unsuccessful call processing. To increase the number of successful call requests, the call processing resources should be spent on useful work. This is achieved by designing suitable overload control mechanisms.

A. Optimized Overload Control Based on Arrival Rate (Static)

Fig. 2 indicates that for given values of μ and d, there exists a value of λ (say, $\lambda_{\max}$) that maximizes the call throughput of the queue [8]. Therefore, the simplest overload control is to limit the arrival rate to the queue to $\lambda_{\max}$. The value of λ that maximizes the call throughput of the queue γ satisfies

$$\mu d = \lambda_{\max} d + \log(1 + \lambda_{\max} d). \tag{2}$$

This is a transcendental equation and can be numerically solved using the Newton–Raphson method [11]. The maximum value of call throughput $\gamma_{\max}$ is

$$\gamma_{\max} = \lambda_{\max}\left[1 - e^{-(\mu-\lambda_{\max})d}\right].$$

TABLE I
OPTIMAL PERFORMANCE OF THE CALL PROCESSOR FOR VARIOUS VALUES OF DELAY THRESHOLD ($\mu = 1.0$)

d	λ_{max}	γ_{max}	E[D]
1	0.557	0.199	2.26
2	0.604	0.330	2.53
3	0.642	0.423	2.79
4	0.673	0.491	3.06
5	0.699	0.544	3.33
10	0.782	0.693	4.59
20	0.855	0.808	6.91
30	0.889	0.857	9.03
40	0.909	0.885	11.05
50	0.923	0.903	12.98
100	0.954	0.944	21.89

With this overload control, when the call arrival rate is λ, the arrival rate to queue is $\min\{\lambda, \lambda_{\max}\}$. The admitted calls to the queue need not form a Poisson process. Excess calls may be dropped in several ways. If the excess calls are randomly dropped, then the admitted calls also form a Poisson process [6]. In this case, the average delay of an admitted call is less than or equal to $1/(\mu - \lambda_{\max})$ when $\lambda \leq \lambda_{\max}$ and it remains at $1/(\mu - \lambda_{\max})$ when $\lambda > \lambda_{\max}$. Table I shows the computed values of λ that maximizes γ for different values of d, along with $\gamma_{\max}$ and average delay for all processed calls. The average delay of call requests contributed to $\gamma_{\max}$ could be less than this value. For smaller values of d, the call loss is higher and hence the call throughput is correspondingly lower. When $\lambda \leq \lambda_{\max}$, the call throughput is given by (1) and it remains at $\gamma_{\max}$ for $\lambda > \lambda_{\max}$.

Overload Control With a Threshold on Average Queueing Delay: The delay threshold used for overload control in Section III-A is based on the absolute value of the queueing delay. An overload control based on the average value of the queueing delay has been considered in [2]. In this scheme, the arrival rate to the queue is limited to a threshold $\lambda' < \mu$ so as to have an upper bound on the average value of the queueing delay. When the call arrival rate is λ, the arrival rate to the queue is $\min\{\lambda, \lambda'\}$. If the excess calls are randomly dropped, then the admitted calls to the queue form a Poisson process [6]. In this case, the average queueing delay is upper bounded by $1/(\mu - \lambda')$. If d' is the upper bound on the average delay, then

$$\frac{1}{\mu - \lambda'} \leq d'$$

or

$$\lambda' \leq \mu - \frac{1}{d'}.$$

Comparing these two schemes, for the same value of d and d' (called *delay threshold*), the $\lambda_{\max}$ is different from λ', as shown in Table II. For small values of *delay threshold*, λ' is smaller than $\lambda_{\max}$ whereas for higher values of *delay threshold*, λ' is larger than $\lambda_{\max}$. This suggests that if one uses the scheme proposed in [2], the average delay threshold chosen must be much less than the absolute delay threshold of a call request. This could result

TABLE II
COMPARISON OF THE ARRIVAL RATE BOUNDS AND MAXIMUM CARRIED LOADS FOR OVERLOAD CONTROL SCHEMES WITH BOUNDED ABSOLUTE DELAY AND BOUNDED AVERAGE DELAY ($\mu = 1.0$)

delay threshold	Absolute *delay threshold*		Average *delay threshold*	
	λ_{max}	γ_{max}	λ'	γ'
1	0.557	0.199	0.00	0.000
2	0.604	0.330	0.50	0.316
3	0.642	0.423	0.67	0.421
4	0.673	0.491	0.75	0.474
5	0.699	0.544	0.80	0.506
10	0.782	0.693	0.90	0.569
20	0.855	0.808	0.95	0.601
30	0.889	0.857	0.97	0.576
40	0.909	0.885	0.98	0.540
50	0.923	0.903	0.98	0.619
100	0.954	0.944	0.99	0.626

in selecting the value of λ' different from $\lambda_{\max}$ and a consequent decrease in call throughput.

B. Optimized Overload Control Based on Buffer Size Limit (Dynamic)

This scheme makes use of a queue of finite buffer size K and applies a very simple overload control. An arriving call setup request is rejected if there are already K calls waiting to be processed in the queue, including the one being processed (buffer overflow).

Since the arrival process is Poisson and the call processing time is exponentially distributed, the call processor can be modeled as an $M/M/1/K$ queueing system. In this system, a call request could be lost due to one of the following two reasons:

- buffer overflow;
- the queueing delay in the call processor exceeding the bound.

Some results on the $M/M/1/K$ queueing system including the delay distribution are shown in the Appendix. Though such a scheme for queueing control has been considered in [9] as FIFO-blocking, its behavior with buffer size variation has not been explored in detail. In steady state, the probability that there are i, $0 \le i \le K$, calls in the system, is given by

$$p_i = \rho^i p_0$$

where $\rho = \lambda/\mu$ and $p_0 = (1-\rho)/(1-\rho^{K+1})$. The average number of calls in the system is given by

$$E[N] = \frac{\rho}{1-\rho} - \frac{(K+1)\rho^{K+1}}{1-\rho^{K+1}}, \qquad \text{for } \lambda \neq \mu$$

and

$$E[N] = \frac{K}{2}, \qquad \text{for } \lambda = \mu.$$

Probability of buffer overflow (q_o) is given by p_K. For given values of λ and μ, q_o decreases with increase in value of K.

The average delay is

$$E[D] = \frac{E[N]}{\lambda(1-p_K)}.$$

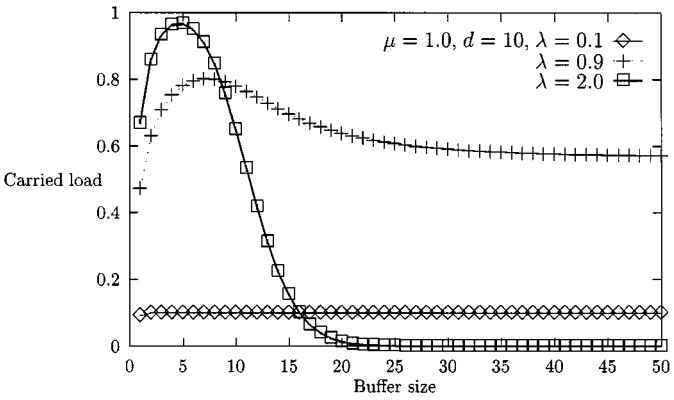

Fig. 3. Variation of the signaling system carried load with the buffer size.

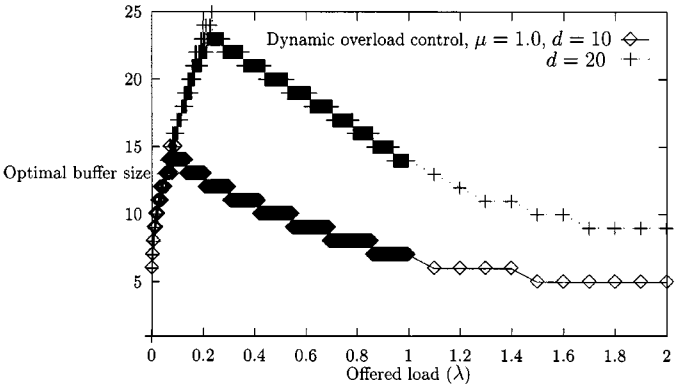

Fig. 4. Optimal buffer size as a function of arrival rate.

The probability of call dropping due to excessive queueing delay is given by

$$q_l = \text{Prob}\{D > d\} = \frac{(1-\rho)e^{-\mu d}}{(1-\rho^K)} \sum_{i=0}^{K-1} \rho^i \sum_{j=0}^{i} \frac{(\mu d)^j}{j!}.$$

1) Call Throughput: The call throughput γ is given by the formula $\gamma =$ Offered load $\times$ (1−Prob. of buffer overflow) $\times$ (1−Prob. of call dropping), i.e., $\gamma = \lambda(1-q_o)(1-q_l)$.

$$\gamma = \frac{\lambda(1-\rho)}{(1-\rho^{K+1})} \sum_{i=0}^{K-1} \rho^i \left[1 - e^{-\mu d} \sum_{j=0}^{i} \frac{(\mu d)^j}{j!}\right]. \quad (3)$$

When $K \to \infty$, the system behavior is similar to that studied in Section III-A.

Call throughput γ is plotted in Fig. 3, as a function of K for various values of λ, d, and μ, the call throughput of the queue initially increases with K and then starts decreasing. When K is small, the effect of buffer overflow dominates that of call dropping due to excessive delay. However, for large values of K, the effect of call dropping due to excessive delay dominates that due to buffer overflow. This is more significant for higher values of call arrival rate.

Using (3), it is easy to compute the optimal buffer size K_{opt} that maximizes the call throughput. Fig. 4 shows the variation of the optimal buffer size with the arrival rate. The optimal buffer size initially increases with increasing arrival rate and thereafter

TABLE III
PERFORMANCE OF DYNAMIC OVERLOAD CONTROL SCHEME FOR DIFFERENT VALUES OF DELAY THRESHOLD ($\mu = 1.0$)

λ	delay threshold $d = 10$			delay threshold $d = 50$			delay threshold $d = 90$		
	K_{opt}	γ_{max}	E[D]	K_{opt}	γ_{max}	E[D]	K_{opt}	γ_{max}	E[D]
0.10	14	0.10	1.11	16	0.1	1.11	16	0.1	1.11
0.20	13	0.20	1.25	23	0.2	1.25	23	0.2	1.25
0.30	12	0.30	1.43	31	0.3	1.43	31	0.3	1.43
0.40	11	0.40	1.67	40	0.4	1.67	40	0.4	1.67
0.50	10	0.50	1.99	49	0.5	2.00	53	0.5	2.00
0.60	9	0.59	2.41	48	0.6	2.5	71	0.6	2.5
0.70	8	0.67	2.84	46	0.7	3.33	85	0.7	3.33
0.80	8	0.75	3.39	44	0.8	5.00	82	0.8	5.00
0.90	7	0.80	3.58	41	0.9	9.45	78	0.9	9.98
1.00	7	0.84	4.00	37	0.97	19.00	70	0.99	35.5
2.00	5	0.97	4.16	20	1.00	19.0	35	1.0	34.0
3.00	4	0.98	3.55	16	1.00	15.5	29	1.00	28.5
4.00	4	0.99	3.68	15	1.00	14.67	26	1.00	25.67
5.00	3	0.99	2.77	14	1.00	13.75	23	1.00	22.75
6.00	3	0.99	2.81	13	1.00	12.8	20	1.00	19.8
7.00	3	1.00	2.84	12	1.00	11.83	18	1.00	17.83
8.00	3	1.00	2.86	12	1.00	11.86	17	1.00	16.86
9.00	3	1.00	2.88	12	1.00	11.88	18	1.00	17.88
10.00	3	1.00	2.89	11	1.00	10.89	26	1.00	25.89

TABLE IV
PERFORMANCE COMPARISON OF STATIC AND DYNAMIC CONTROL SCHEMES FOR VARIOUS VALUES OF DELAY THRESHOLD ($\mu = 1.0$)

d	Static overload control			Dynamic overload control with $\lambda = \lambda_{max}$		
	λ_{max}	γ_{max}	$E[D]$	γ	$E[D]$	K_{opt}
1	0.557	0.199	2.26	0.233	1.36	2
2	0.604	0.330	2.53	0.375	1.38	2
3	0.642	0.423	2.79	0.471	1.71	3
4	0.673	0.491	3.06	0.539	2.03	4
5	0.699	0.544	3.33	0.592	2.07	4
10	0.782	0.693	4.59	0.735	3.29	8
20	0.855	0.808	6.91	0.838	5.48	16
30	0.889	0.857	9.03	0.880	7.51	24
40	0.909	0.885	11.05	0.904	9.43	32
50	0.923	0.903	12.98	0.919	11.29	40
100	0.954	0.944	21.89	0.953	20.20	84

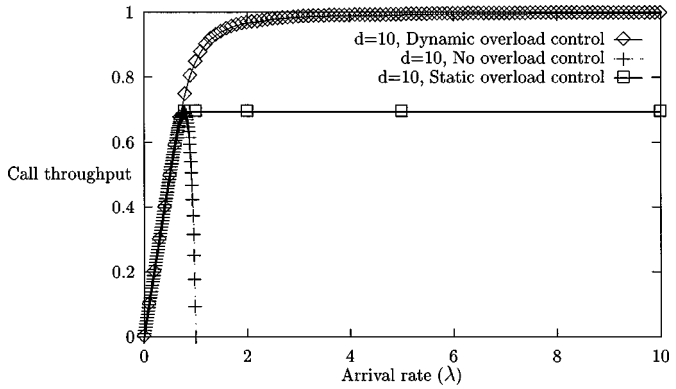

Fig. 5. Comparison of call throughputs under various overload control schemes, for the same delay threshold

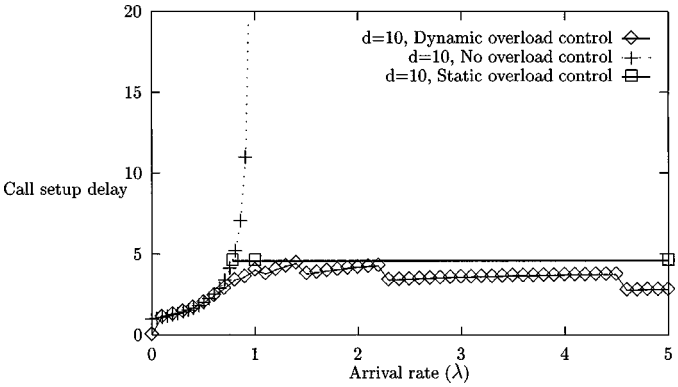

Fig. 6. Comparison of average call setup delays under various overload schemes, for the same delay threshold.

decreases with increasing arrival rate. Table III shows the computed values of optimal buffer size, call throughput, and the average delay for various values of arrival rate. As shown by the table, it is possible to achieve call throughput very close to the maximum server capacity μ, without increasing the delay. This improvement in performance is achieved my minimizing the unproductive processing of call setup requests. Table IV compares the performance of static and dynamic overload control schemes for the same value of arrival rate $\lambda_{\max}$. This indicates that the dynamic overload control performs better than the static overload control. Figs. 5 and 6 summarize the performance of various overload control schemes. The dynamic overload control significantly outperforms the static overload control at higher values of offered load. For example, when $d = 10$, with static overload control, the offered load is limited to 0.782 and the corresponding values of maximum call throughput and the average delay are 0.693 and 4.59, respectively. With dynamic overload control, there is no limit on the offered load and call throughput close to the processor capacity can be achieved with much lower values of average delay, as shown in Table III. At higher values of offered load, higher values of call throughput is achieved with

lower values of average delay. As shown in Fig. 6, the average delay under dynamic overload control takes a sawtooth profile under overload. Also, the variation of this delay with arrival rate is found to be large for larger values of delay threshold d. Under overload, the buffer requirement is found to increase linearly with the delay threshold.

2) Performance Under Overload: When the offered load to the queue is much higher compared to its capacity, i.e., $\lambda \gg \mu$, (3) reduces to

$$\gamma = \mu \left[e^{-\mu d} \sum_{j=K}^{\infty} \frac{(\mu d)^j}{j!} \right].$$

This shows that, under overload, the call throughput can be improved by using smaller values of buffer size.

3) Implementation of Overload Control: Implementation of the static overload control is easier than that of the dynamic overload control. The dynamic overload control can be implemented as follows: measure the offered load λ and use this value to compute the optimal buffer size K_{opt} that maximizes the value of γ given by (3). Reject an incoming call if there are K_{opt} calls in the queue, including the one under processing. In the analysis, it is assumed that λ does not vary with time. In practice, one may make a quasistatic assumption of arrival process, i.e., λ does not vary during a given interval and use the same value of K_{opt} throughout this interval. If the interval is small, then the control will be more effective but the number of computations required will be large. If the interval is large, the control will be less effective but the number of computations required will be small. In any case, the numerical examples quoted indicate that the optimal buffer size is not large for reasonable values of d and hence the number of computations required is also not large.

The value of the delay threshold has to be set by the network manager, based on the number of switching nodes involved in a call path, customer patience, signaling time-out values, etc. When there are different classes of signaling sessions having different delay requirements, different queues have to be maintained with different thresholds.

4) Further Improvements to Dynamic Overload Control: In the analysis of the overload control schemes, it is assumed that the call requests in the queue are processed on a FIFO basis. These performance results are expected to serve as lower bounds to more sophisticated local call rejection mechanisms such as push-out or time-out along with a LIFO service discipline, as considered in [9], [21]. For example, when the request is ready for processing, it is possible to decide whether the waiting time of request has already crossed the delay threshold. If it has crossed the threshold, the request can be dropped without being processed. Even if the waiting time of request has not crossed the threshold, an intelligent prediction algorithm can predict whether the total delay is likely to cross the threshold had the request been processed. Accordingly, a decision can be taken whether to drop the call or not. Other ideas for further optimization include priority treatment based on the number of switching nodes traversed by the request or the total delay accumulated at the preceding switching nodes, etc. However, the impact of these service disciplines on the call setup performance is not addressed and is beyond the scope of this paper.

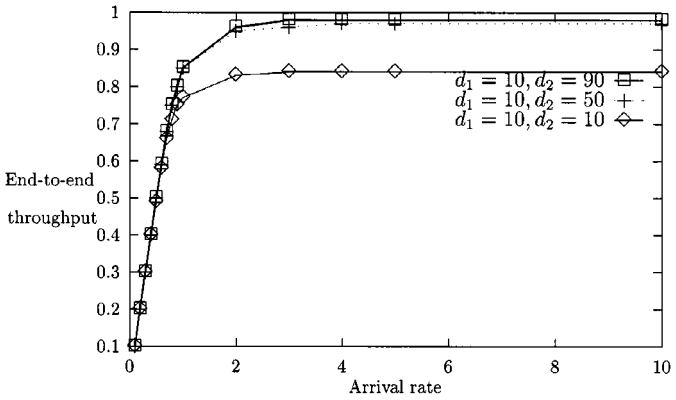

Fig. 7. Comparison of end-to-end call throughput in a two-node network, under various combinations of delay bounds.

5) Call Admission Probability: A call will be admitted to the network if and only if:

- it does not overflow due to unavailability of buffers;
- it is not dropped due to excessive delay; and
- it is not blocked due to unavailability of transport resources.

Therefore, when these three events are statistically independent, the call admission probability of the network is $(1-q_o) \times (1-q_l) \times (1-q_b)$.

IV. Experimental Results

Simulation experiments were carried out to understand the performance of the overload control algorithm in a network with multiple switching nodes, as shown in Fig. 1. The arrival process to Node 1 is Poisson with rate λ. The call processing time at each node is independent and exponentially distributed with rate $\mu = 1.0$. The dynamic overload control scheme discussed in Section III is applied at each node. The delay threshold at Node i is denoted as d_i.

For a given arrival rate and delay threshold, the optimal buffer size K_{opt} that maximizes the call throughput of Node 1 (γ_1) is computed using (3). The arrival process to Node 2 is approximated as a Poisson process with rate γ_1. For a given delay threshold at Node 2, the optimal buffer size is computed using (3). The call throughput of the second and the remaining nodes are found through simulation. Again, by approximating the arrival process to these queues as Poisson processes with rate equal to the call throughput of the preceding queue, the optimal buffer sizes are computed and used in the simulation.

Two scenarios were considered for detailed study.

Scenario 1: A two-node network with unequal delay budgeting where delay threshold of the first node is always 10 and delay threshold of the second node takes values of 10, 20, …, 90. This scenario gives rise to end-to-end delay bounds of 20, 30, …, 100.

Scenario 2: A network of two to ten nodes with every node having an equal delay threshold of 10. This scenario also gives rise to end-to-end delay bounds of 20, 30, …, 100, depending on the actual number of nodes involved.

Under Scenario 1, the end-to-end call throughput is shown in Fig. 7. The call throughput of Node 1 is as shown in Table III.

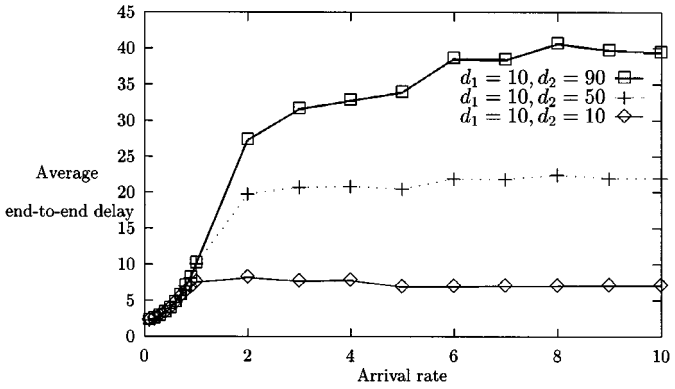

Fig. 8. Comparison of average end-to-end call setup delays in a two-node network, under various combinations of delay bounds.

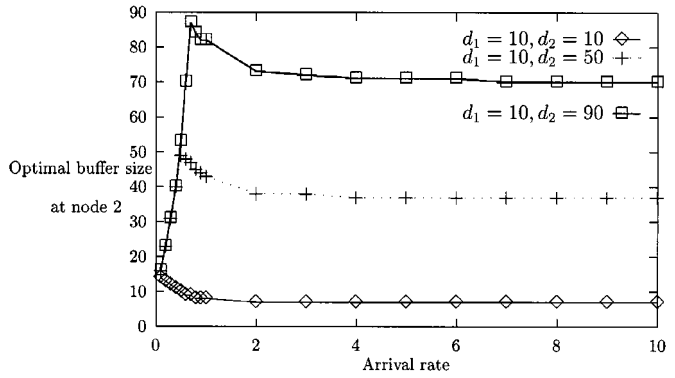

Fig. 9. Comparison of optimal buffer size at the second node in a two-node network, under various combinations of delay bounds.

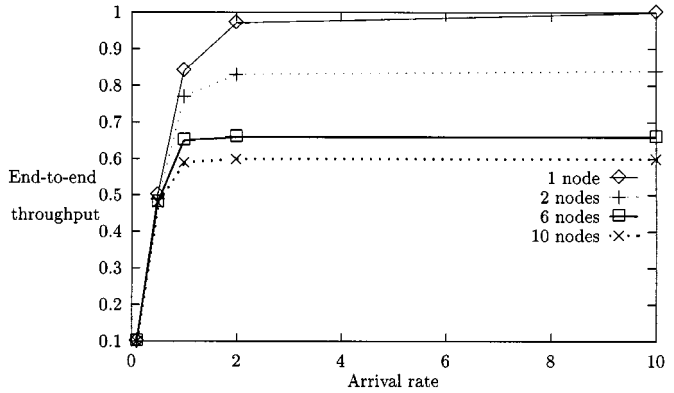

Fig. 10. Variation of end-to-end call throughput with different number of nodes in tandem, with equal budgeting of delay bounds among the nodes.

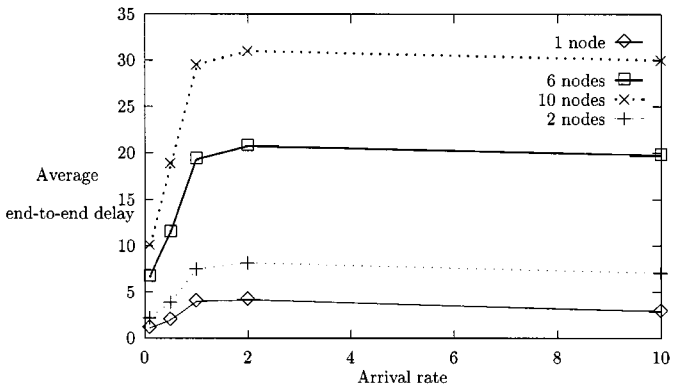

Fig. 11. Variation of average end-to-end setup delay with different number of nodes in tandem, with equal budgeting of delay bounds among the nodes.

The excess offered load gets controlled at this node either through buffer overflow or through call dropping. As a result, the offered load to the second node is bounded by the call processing rate of the first node. It is observed that when $\lambda \leq 0.6$, Node 1 acts as $M/M/1/\infty$ queue without any control and the call throughput is approximately equal to the arrival rate. When $0.6 < \lambda \leq 2.0$, the overload control comes into effect and the call throughput becomes less than the arrival rate. It is found that the behavior of Node 2 closely approximates that of an $M/M/1/K$ queue with dynamic overload control.

Under Scenario 1, the average end-to-end setup delay is shown in Fig. 8. This is the end-to-end setup delay of all calls processed at Node 2. The end-to-end delay of calls contributed to the end-to-end call throughput will be less than this value. The average end-to-end delay is found to be less than or equal to 40% of the end-to-end delay bound, in all the cases observed. This gives room for overprovisioning of the delay bound, in actual practice. The variability of this delay is found to be significant when λ is close to 1. As in the case of a single node analyzed in Section III, a sawtooth behavior of this delay is observed in the overload region.

Under Scenario 1, the optimal buffer size required for Node 2 is shown in Fig. 9. As the value of delay threshold increases, the optimal buffer size also increases. Among the values shown, the highest value of buffer size computed is 87, which corresponds to $\lambda = 0.7$ and $d_2 = 90$.

Under Scenario 2, the end-to-end call throughput is shown in Fig. 10. All nodes have the same delay threshold of 10. The call throughput of Node 1 is as shown in Table III. The excess offered load gets controlled at this node either through buffer overflow or through call dropping. As a result, the offered load to the subsequent nodes are bounded by the service rate of the first node. When $\lambda \leq 0.5$, the end-to-end call throughput is approximately equal to the arrival rate. In a ten-node network, the end-to-end call throughput under overload is approximately 60% of the call processing rate of the first node. In this case also, it is found that the behavior of Nodes 2–10 closely approximates that of $M/M/1/K$ queues with dynamic overload control.

Under Scenario 2, the average end-to-end setup delay is shown in Fig. 11. All nodes have the same delay threshold of 10. This is the end-to-end setup delay of all calls processed at the last node. The end-to-end delay of calls contributed to the end-to-end call throughput will be less than this value. The average delay is found to be less than or equal to 40% of the end-to-end delay bound, in all the cases observed. This gives room for overprovisioning of the delay bound, in actual practice. As in the case of a single node analyzed in Section III, a sawtooth behavior of this delay is observed in the overload region.

Fig. 12 compares the end-to-end throughput performance in Scenarios 1 and 2, for providing the same value of end-to-end delay bound. This could be useful to decide the route when multiple routes are possible to the same destination but with different number of nodes. In the overload region, Scenario 1 gives much higher throughout compared to Scenario 2. This

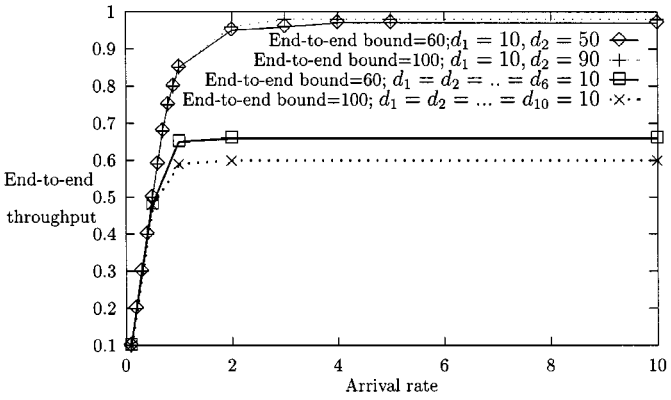

Fig. 12. Comparison of end-to-end throughput under equal and unequal budgeting of end-to-end delay bound.

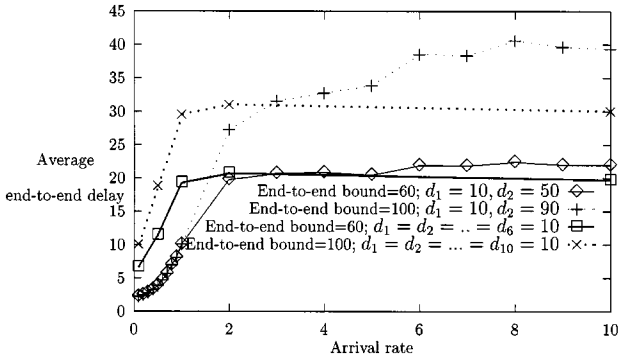

Fig. 13. Comparison of average end-to-end setup delay under equal and unequal budgeting of end-to-end delay bound.

is because of the fewer number of nodes through which the signaling undergoes overload control. In each case, the excess traffic offered gets controlled at Node 1. Each subsequent node reduces the traffic further, depending on the arrival rate and delay threshold. The traffic reduction becomes more with smaller delay threshold. In Scenario 2, since the traffic passes through more number of nodes with smaller delay thresholds compared to Scenario 1, the call throughput is correspondingly lower. Fig. 13 compares the average end-to-end delay in Scenario 1 and 2, for providing the same value of end-to-end delay bound. In the underload region, Scenario 1 gives better delay performance and in the overload region, Scenario 2 gives better delay performance compared Scenario 1. The difference in the delay performance becomes significant for higher values of end-to-end delay bound. In each case, the excess traffic offered gets controlled at Node 1 with average delay as shown in Table III. The average delay at each subsequent node depends on the arrival rate and delay threshold and it increases with the delay threshold. The variation of the average delay with delay bound is found to be nonlinear, especially at higher loads. The average end-to-end delay under Scenario 2 is always additive. The nonlinear behavior of the average delay in Node 2 when its arrival rate is close to 1, is the reason for the worse delay performance of Scenario 1 compared to Scenario 2. Also under Scenario 2, the end-to-end delay is more stable in the overload region, compared to Scenario 1. Table V compares the total buffer requirements at all nodes together, for providing the same value of end-to-end delay bound under the two scenarios. At low load, the total buffer requirements in the two scenarios differs considerably. The difference is significant at higher values of end-to-end delay bound. At higher loads, the buffer requirement for the two scenarios are approximately the same. The total buffer requirement is an indication of the computational complexity of the overload control algorithm.

TABLE V
COMPARISON OF TOTAL BUFFER REQUIREMENTS FOR THE SAME END-TO-END DELAY BOUND UNDER THE TWO SCENARIOS

λ	No. of buffers for end-to-end delay bound = 60		No. of buffers for end-to-end delay bound = 100	
	Scenario 1	Scenario 2	Scenario 1	Scenario 2
0.1	30	84	30	140
0.5	59	60	63	100
1.0	50	49	88	85
2.0	43	45	78	81
10.0	40	43	73	79

Therefore, the tradeoff between the number of nodes, call throughput, and average end-to-end delay needs to be considered while deciding the route budgeting the end-to-end delay bound among different nodes along the route.

V. CONCLUSION

A distributed overload control algorithm for delay-bounded call setup is proposed in this paper. The end-to-end delay bound is budgeted among the switching nodes involved in call setup, and these nodes apply a local overload control with a deterministic delay threshold and drop call requests experiencing higher delays. Using an $M/M/1$ queueing model with FIFO service discipline at a switching node, two optimized control schemes are considered for local overload control and compared their performance through analysis: one with arrival rate limit and the other with buffer size limit. Though both the schemes minimize the unproductive call processing at heavy load, the latter is found to yield higher call throughput and lower average call setup delays compared to the former. Also, the buffer size required for the scheme with buffer size limit is typically small and call throughput close to the server capacity can be achieved during overload. The performance of the distributed overload control algorithm in a network is evaluated through simulation experiments, using the scheme with buffer size limit for the local overload control. It shows that the average end-to-end delay could be much less than the end-to-end delay bound, providing room for overprovisioning of the delay bounds. The tradeoff between the number of nodes, call throughput, and average end-to-end delay needs to be considered while deciding the route budgeting the end-to-end delay bound among different nodes along the route. These performance results are expected to serve as lower bounds to more sophisticated local call rejection mechanisms such as push-out or time-out along with a LIFO service discipline.

Future work involves understanding the impact of using other queueing and service disciplines on the end-to-end performance, different ways of budgeting the end-to-end delay bound among the nodes, the effect of overprovisioning the delay bounds, and extending these results to other call processing events in a switch/router-based fixed and mobile communication network.

APPENDIX
RESULTS ON $M/M/1/K$ QUEUEING SYSTEM

Consider an $M/M/1/K$ queueing system. Let r_i be the probability that there are i customers in the system just before a customer arrival who actually enters the system. Then $r_i \neq p_i$ because i) no arrival enters the system when K customers are there so r_K is zero and ii) $\sum_{i=0}^{K-1} p_i \neq 1$. If it is assumed that $r_i = kp_i$, $i = 0, 1, \ldots, K-1$, then it is easy to show that $r_i = p_i/(1-p_K)$, $i = 0, 1, \ldots, K-1$ [1]

$$\begin{aligned} D(t) &= \text{Prob}\{D \le t\} \\ &= \sum_{i=0}^{K-1} \text{Prob}\{D \le t | N = i\}\text{Prob}\{N = i\} \\ &= \sum_{i=0}^{K-1} \left[\int_0^t \frac{\mu(\mu x)^i}{i!} e^{-\mu x} dx\right] r_i \\ &= 1 - \sum_{i=0}^{K-1} r_i \sum_{k=0}^{i} e^{-\mu t}\frac{(\mu t)^k}{k!} \\ &= 1 - \sum_{i=0}^{K-1} \frac{p_i}{1-p_K} \sum_{j=0}^{i} e^{-\mu t}\frac{(\mu t)^j}{j!} \end{aligned}$$

for some constant K. The delay distribution in an $M/M/1/K$ queueing system is given by

$$\frac{(1-\rho)}{(1-\rho^K)} \sum_{i=0}^{K-1} \frac{\rho^i \mu^{i+1} x^i}{i!} e^{-\mu x}.$$

In steady state, the probability that there are i, $0 \le i \le K$, customers in the system, is given by

$$p_i = \rho^i p_0$$

where $\rho = \lambda/\mu$ and $p_0 = (1-\rho)/(1-\rho^{K+1})$. Since there are never more than K customers in the system, the system reaches steady state for all values of λ and μ. If $\lambda = \mu$, $p_i = 1/(K+1)$ for $0 \le i \le K$. The average number of customers in the system is given by

$$E[N] = \frac{\rho}{1-\rho} - \frac{(K+1)\rho^{K+1}}{1-\rho^{K+1}}, \qquad \text{for } \lambda \neq \mu$$

and

$$E[N] = \frac{K}{2}, \qquad \text{for } \lambda = \mu.$$

Probability of buffer overflow (q_o) is given by p_K. For given values of λ and μ, q_o decreases with increase in value of K.

The average delay is

$$E[D] = \frac{E[N]}{\lambda(1-p_K)}.$$

The probability of customer loss due to excessive delay (q_l) is given by

$$q_l = \text{Prob}\{D > d\} = \frac{(1-\rho)e^{-\mu d}}{(1-\rho^K)} \sum_{i=0}^{K-1} \rho^i \sum_{j=0}^{i} \frac{(\mu d)^j}{j!}.$$

ACKNOWLEDGMENT

The author is grateful to the anonymous reviewers and to Dr. U. Mukherji whose critical comments helped to improve the content of the paper. The author also acknowledges the support from Dr. I. T. M. Chit, Dr. J. Biswas, Prof. A. Lazar, and Dr. R. Viswanathan of ISS.

REFERENCES

[1] A. O. Allen, *Probability, Statistics, and Queueing Theory*. New York: Academic, 1978.
[2] N. G. Aneroussis and A. A. Lazar, "Virtual path control for ATM networks with call-level quality of service guarantees," in *Proc. IEEE INFOCOM*, San Francisco, CA, Mar. 1996, pp. 312–319.
[3] N. G. Aneroussis, A. A. Lazar, and D. E. Pendarakis, "Taming Xunet III," *ACM Comput. Commun. Rev.*, vol. 25, no. 3, pp. 44–65, Oct. 1995.
[4] ATM Forum, "ATM user-network interface specification, version 3.1,", Sept. 1994.
[5] ——, "Private network–network interface specification 1.00,", Mar. 1996.
[6] D. Bertsekas and R. Gallager, *Data Networks*, 2nd ed. Englewood Cliffs, NJ: Prentice Hall, 1992.
[7] S. Blake *et al.*, "An architecture for differentiated services," , IETF RFC 2475, Dec. 1998.
[8] C. G. Cassandras, H. Kallmes, and D. Towsley, "Optimal routing and flow control in networks with real-time traffic," in *Proc. IEEE INFOCOM*, 1989, pp. 784–791.
[9] B. T. Doshi and H. Heffes, "Overload performance of several processor queueing disciplines for the $M/M/1$ queue," *IEEE Trans. Commun.*, vol. 34, pp. 538–546, June 1996.
[10] R.-H. Hwang, J. F. Kurose, and D. Towsley, "On-call processing delay in high speed networks," *IEEE/ACM Trans. Networking*, vol. 3, pp. 628–639, Dec. 1995.
[11] S. S. Kuo, *Computer Applications of Numerical Methods*. Reading, MA: Addison-Wesley, 1972.
[12] M. H. Kallmes, D. Towsley, and C. G. Cassandras, "Optimality of last-in–first-out (LIFO) service discipline in queueing systems with real-time constraints," in *Proc. 28th IEEE Conf. Decision and Control*, 1989, pp. 1073–1074.
[13] S. S. Lam and M. Reiser, "Congestion control of store-and-forward networks by input buffer limits—An analysis," *IEEE Trans. Commun.*, vol. 27, pp. 127–134, Jan. 1979.
[14] A. Lars and A. Arvidsson, "A congestion control algorithm for signaling networks based on a state machine controlled by network delays," in *Proc. Bangkok Int. Teletraffic Seminar*, Bangkok, Thailand, 1995, paper no. 31.
[15] K. Murakami and M. Katoh, "Control architecture for next-generation communication networks based on distributed databases," *IEEE J. Select. Areas Commun.*, vol. 7, pp. 418–423, Apr. 1989.
[16] R. R. Pillai, "Signaling performance and overload control in ATM networks," ISS, Tech. Rep. TR96-222-0, Nov. 1996.
[17] S. M. Ross, *Stochastic Processes*. New York: Wiley, 1983.
[18] P. Stefen and A. Arvidsson, "A profit optimizing strategy for congestion control in signaling networks," in *Proc. Bangkok Int. Teletraffic Seminar*, Bangkok, Thailand, 1995, paper no. 39.
[19] M. Veeraraghavan, M. M. Kshirsagar, and G. L. Choudhury, "Concurrent ATM connection setup reducing need for VP provisioning," in *Proc. IEEE INFOCOM*, CA, Mar. 1996, pp. 303–311.
[20] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zapalla, "RSVP: A new resource ReSerVation Protocol," *IEEE Network*, vol. 7, pp. 8–18, Sept. 1993.
[21] Z.-X. Zhao, S. S. Panwar, and D. Towsley, "Queueing performance with impatient customers," in *Proc. IEEE INFOCOM*, 1991, pp. 400–409.

R. Radhakrishna Pillai (S'89–M'95) received the B.Tech. degree from University of Kerala, Kerala, India, and the M.E. and Ph.D. degrees from the Indian Institute of Science, Bangalore.

He is a Research Staff Member at Kent Ridge Digital Labs, Singapore, where he does research on communication networks. During 1993–1995, he was with Tata Elxsi India, Ltd., Bangalore. His recent work covers ATM networks, multimedia, mobile wireless networks, and policy-based management of Internet QoS.