Prioritized Dispersal: a Scheme for Selective Exploitation of Redundancy in Distributed Systems

Yitzhak Birk and Noam Bloch
birk@ee, bloch@tx.technion.ac.il
+972 4 829-4637, 4748
Electrical Engineering Department
Technion – Israel Institute of Technology
Haifa 32000, ISRAEL

Abstract

In distributed redundant-resource systems such as communication networks with multiple paths between nodes, there is a choice in allocating resources to tasks; this can be used for fault-tolerance, but also in order to improve performance. The allocation, however, is complicated by the fact that system state is not known and changes dynamically. Exploiting redundancy by partitioning a message into several submessages and sending them along with several "redundant" submessages along different paths, as is done by dispersal schemes, increases the load, reduces capacity and even increases delay at permissible heavy loads. We present and analyze novel "prioritized dispersal" schemes, whereby "redundant" submessages receive lower priority than the "original" submessages, and show their performance to substantially exceed that of non-prioritized schemes. This extends the beneficial applicability of selective exploitation of redundancy, whose benefits for centralized systems have been established, to distributed systems.

1. Introduction

Redundancy entails the provision of more resources than the required minimum, thereby permitting the provision of the required services by using a sufficient fraction of those resources. In the context of communication networks, those resources may be paths in the network as well as channel time (reflecting the amount of data transmitted over the channel).

For the purpose of this paper, we define *data redundancy* as the creation of $m + r$ symbols from m information symbols such that the original symbols can be reconstructed from any m of the $m + r$ symbols. Data redundancy had originally been used to detect and/or tolerate errors caused by noise in communication channels and in storage media. It has subsequently been further employed to tolerate "higher-level" faults, such as faulty communication nodes [1] and faulty disk drives in disk-array-based storage systems [2]. It has also been proposed to use redundancy in order to improve performance even in the absence of failures: In [3] and [1], the use of redundancy for dispersal routing was shown to sharply reduce delays. With redundant-dispersal routing, each message is divided into submessages, redundant submessages are then constructed and all submessages are transmitted through a maze of paths in the network. The tolerable ratio of lost or delayed submessages with this scheme is much larger since a certain fraction of lost or delayed submessages can be reconstructed.

In the context of data-layout and disk-access scheduling in high-performance video servers, it has been shown that it is useful to distinguish between direct access to the original data and its reconstruction from redundant data [4],[5]. Moreover, it was shown that the exploitation of existing redundancy may be more costly than the redundancy itself (cost of RAM buffers and disk storage, respectively). This gave rise to the notion of *selective exploitation of redundancy*, reflecting the fact that it is not always wise or beneficial to exploit redundancy even if it exists. In this paper, we carry this idea over to the domain of distributed systems, using communication networks as an example.

In communication networks, the redundancy itself includes the creation of multiple paths as well as the ability of the source node to generate and transmit redundant information. Selective exploitation could refer to the decision regarding the fraction of redundant information that should be transmitted. The bulk of the cost of exploitation, however, is incurred within the network in the form of extra load which results in higher packet delays and/or loss probabilities due to buffer overflow. Moreover, unlike storage

0-8186-8135-7/97 $10.00

systems, in which system state can be known to a central controller, the inherently distributed nature of high-speed communication networks and their rapidly-changing state prevent source nodes from making intelligent decisions regarding the degree of redundancy, which limits the effectiveness of redundant-dispersal schemes.

Our focus in this paper is on ways of permitting some adaptation of the degree of redundancy exploitation within the network based on its dynamic state. Specifically, we propose *"prioritized-dispersal"* schemes, whereby low priorities are assigned to the redundant submessages, as a novel improvement over conventional dispersal schemes: 1) losses are more uniformly distributed among messages, and 2) the overload caused by the redundant submessages is mitigated. We note in passing that prioritized-dispersal schemes can also be applied in other contexts, either alone or in conjunction with other schemes, but this is beyond the scope of the current paper.

The remainder of the paper is organized as follows. In section 2, we introduce a family of prioritized-dispersal schemes. In section 3, we provide an approximate queuing model. Section 4 presents derivations of the blocking probabilities for various prioritized-dispersal schemes with finite and infinite queues under the approximate queuing model. In section 5 we sketch similar derivations for delay, section 6 presents numerical results and a comparison among schemes, and section 7 offers concluding remarks.

2. Prioritized-dispersal schemes

Prioritized dispersal, like conventional dispersal, may entail the replication of a message and transmission of the replicas along different paths; alternatively, a message is partitioned into several submessages, several redundant submessages are constructed such that any sufficiently large subset of submessages suffices for reconstruction of the original message, and all the submessages are transmitted along different paths. The novelty of prioritized dispersal is that "redundant" submessages (or replicas) are assigned lower priorities than the original ones. In view of the large communication overhead brought about by replication, we focus on partitioning into submessages. For the performance measures discussed in this paper, it makes little or no difference which submessages are declared to be the "redundant" ones. We therefore conveniently speak of m submessages that jointly comprise the original message, accompanied by r redundant submessages. Reception of any m submessages suffices for reconstruction.

In designing a prioritized-dispersal scheme, one must make several choices: 1) the degree of fragmentation and redundancy (m, r), 2) the priority assigned to each submessage, 3) the order of processing of submessages with equal priorities, and 4) preemption of low-priority submessages by arriving higher-priority ones. These choices span the family of prioritized-dispersal schemes. In the remainder of this section, we shed some light on the considerations involved in making these choices, focusing on the last two. (The degree of fragmentation is parameterized and two priority levels are used throughout: high priority for m submessages and low priority for the remaining r).

The two performance measures studied in this paper are the blocking probability of a **message**, defined as the probability that at least $r+1$ of its submessages are lost due to full queues, and its delay, defined as the time until at least m of its submessages reach the destination. The two are studied for the case of finite and infinite queues, respectively.

Queuing discipline

Even when the arrival rate of the original submessages doesn't exceed the service rate, applying redundancy may cause overload. When the system is overloaded, the number of redundant (and thus lower-priority) submessages increases without bound and at least a fraction of them never get served. Moreover, with a "First Come—First Served" (FCFS) queuing discipline, the expected delay for all low-priority submessages is infinite in such a case. The key goal of the queuing policy is thus the maximization of the (temporal) relevance of the redundant submessages that are served.

The queuing discipline may be based on an order-changing mechanism such as a "Last-Come—First-Served" (LCFS) policy, or on rejection. Rejection mechanisms entail pushing out (discarding) old customers when the buffer is full or timing out (discarding) customers whose waiting time exceeds a predefined threshold. Non-prioritized schemes for overload control were presented by Doshi and Heffes in [6]. They considered both FCFS and LCFS schemes with customer-rejection mechanisms corresponding to pushing out or timing out older customers in the queue. Their comparison among schemes was based on throughput–average-delay characteristics and on the probability distribution functions of delay. They concluded that LCFS service, with or without a timeout mechanism, yields the best performance.

Unfortunately, the LCFS service discipline has many drawbacks, including "social injustice" [7] and, in some cases, the effort needed to reorder customers. To cope with LCFS drawbacks, a compromise was suggested in [8],[9], whereby an arriving customer joins the queue at a probabilistically-chosen queue position, and service is in queue order. The distinction made in prioritized dispersal between redundant and non-redundant submessages enables us to use another kind of compromise: we use LCFS only for low-priority (L-P) submessages, which are the ones that incur the long delays, and high-priority (H-P) submessages are served in FCFS order.

The use of LCFS for systems with tandem queues, e.g. multi-hop paths in a communication network, may give rise to strange, "oscillatory" phenomena. For example, a submessage that overtakes an earlier one in one queue may be overtaken by it in the next one. In such cases, it may be more appropriate to employ a "last generated—first served" queuing discipline for L-P submessages.

Priority discipline

The priority discipline determines whether the service of a low-priority job is preempted by the arrival of a high-priority job. If the preempted service of the L-P job is subsequently resumed from the the point of interruption, the priority discipline is called *preemptive resume*; if it is started from the beginning, the discipline is called *preemptive-repeat*. If there is no preemption, it is *non-preemptive*.

The priority discipline presents a tradeoff: preemption is a more "precise" execution of the queuing policy; however, with a preemptive-resume discipline, the partially-served customers are served after the newly arriving customers which, stochastically, require more service. Consequently, the delay and blocking probability are higher ([10], section 3.9). This is even more severe with a preemptive-repeat discipline, since work is actually lost. It should nonetheless be noted that the drawbacks of preemption do not apply in the case of exponentially-distributed service times, since the newly arriving customers and the partially served customers require, stochastically, the same amount of service.

In the following sections, we develop an approximate queuing model which is then used for performance evaluation of prioritized-dispersal schemes and for a comparison among them as well as between them and non-prioritized schemes.

3. Approximate queuing model

We model the set of paths connecting a given source to a given destination in a redundant-dispersal routing network as a system of $m+r$ parallel queues, one per path. (See Figure 1.) *Task* or *dispersed task* refer to the whole message. Submessages are represented by *subtasks* to be served by a queue. The $m + r$ subtasks jointly making up a given task are assigned randomly to (all) the $m + r$ queues. The queues are assumed to be i.i.d., so it suffices to analyze a single queue.

The order of service in the queues is FCFS for the high priority subtasks and LCFS for the low-priority ones. We consider both preemptive-resume and non-preemptive policies. When using a preemptive-resume policy, it is also applied among low priority subtasks. This is done mainly for simplicity of analysis. However, it may be beneficial with exponentially distributed service times, since the residual service time of a subtask has the same distribution as the service time of the newly arrived low-priority subtasks.

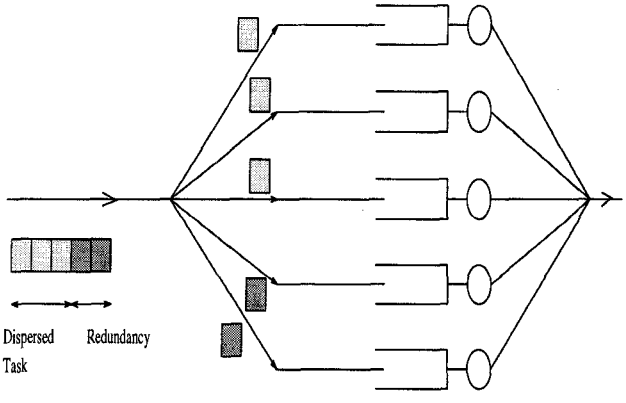

Figure 1. Approximate queuing model for a $(3, 2)$ dispersal system. Each task is dispersed into 3 subtasks and 2 redundant subtasks are constructed. The 5 subtasks are randomly allocated to 5 independent queues with equally distributed service time.

The arrival process of H-P subtasks ("original" submessages in a dispersal-routing system) to any single queue is Poisson with rate λ_h, creating a load of $\rho_h \equiv \frac{\lambda_h}{\mu}$, where $\frac{1}{\mu}$ is the mean service time of a subtask. The arrival process of L-P subtasks (representing redundant submessages) is Poisson with rate λ_l, and $\rho_l \equiv \frac{\lambda_l}{\mu}$. Also, $\frac{\lambda_h}{\lambda_l} = \frac{m}{r}$. The aggregate offered load is $\rho \equiv \frac{\lambda}{\mu}$, where $\lambda \equiv \lambda_h + \lambda_l$. We also assume that the arrival processes are independent.

The service time for each (low- or high-priority) subtask is assumed to be independent from subtask to subtask. Some of the analysis that will be presented holds for generally-distributed service time (with arbitrary Laplace transform $B^*(s)$). The rest of it assumes that the service time of subtasks is exponentially distributed (i.e., task service time distribution is Erlangian). Completion of the extension to generally distributed service time is a topic for future research.

Queuing Model Vs. Real Systems

The $m+r$ submessages constituting a given message are transmitted through $m+r$ different paths. Consequently, the arrival processes to those paths are correlated, and so are the queues representing them. However, if different subsets of those paths are shared among many different (source, destination) pairs, only a small fraction of the traffic through different queues is correlated. Consequently, we conjecture that performance with the independence assumption closely approximates the performance of dispersal routing systems in which each path is shared among many (source, destina-

tion) pairs. This conjecture is verified by simulations.

Figure 2 compares calculated results (based on the approximate model) with simulation results for blocking probability with a preemptive-resume discipline. It shows that when ten (source, destination) pairs share each buffer, calculations are quite accurate, much more so than in the case of only two pairs sharing each buffer. It also shows that the independence assumption yields optimistic results. Similar comparisons for the non-preemptive discipline as well as ones for the probability of delay exceeding a given threshold lead to the same conclusions, but the relevant figures are omitted for brevity. In the remaining figures, we use calculated results.

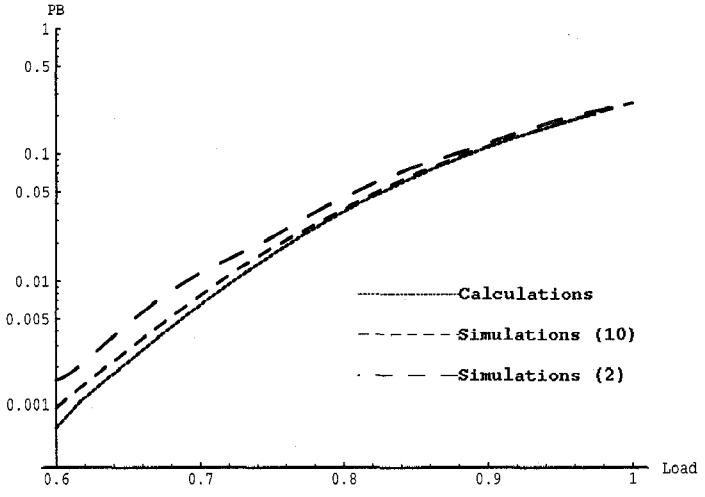

Figure 2. Blocking probability with a prioritized-dispersal system — simulations Vs. analytical results. The simulation plots correspond to sharing of each path by two and ten different (source, destination) pairs. Queue capacity is 10 subtasks.

As mentioned earlier, since all queues are statistically identical, it suffices to analyze a single queue. In the following sections, we analyze a single priority queue with FCFS service for H-P jobs and LCFS service for L-P jobs. In section 4, we derive the blocking probability for a limited-capacity priority M/M/1/K queue. In section 5, we sketch the derivation of the delay distribution for an unlimited-capacity priority M/M/1 queue, which appears in [11]. Derivation of the above results for a generally-distributed service time and the delay distribution for limited-capacity queues is a topic for future research.

4. Blocking probability for priority M/M/1/K queue

Let P_{B_h}, P_{B_l} and P_B be the blocking probabilities of a high priority subtask, a low priority subtask and a dispersed task, respectively. With the independence assumption, the $m+r$ queues accessed by the $m+r$ subtasks of a dispersed task are i.i.d.; Consequently, a dispersed task's blocking probability is equal to the m^{th} out of $m+r$ order statistics from a parent population equal to the distribution of a single queue. Accordingly,

$$P_B = 1 - \sum_{i_m=(m-r)^+}^{m} \left(\binom{m}{i_m} (1-P_{B_h})^{i_m} (P_{B_h})^{m-i_m} \times \sum_{i_r=m-i_m}^{r} \binom{r}{i_r} (1-P_{B_l})^{i_r} (P_{B_l})^{r-i_r} \right). \quad (1)$$

Let $\tilde{P}_B$ be the overall blocking probability of a subtask:

$$\tilde{P}_B = \frac{\lambda_h}{\lambda} P_{B_h} + \frac{\lambda_l}{\lambda} P_{B_l}. \quad (2)$$

With a non-preemptive discipline, the aggregate number of customers in the queue is distributed identically to that of an ordinary M/M/1/K queue with an arrival rate $\lambda_h + \lambda_l$. With an exponentially distributed service time, this also holds for a preemptive-resume discipline, since the residual service time of a preempted L-P customer has the same distribution as the service time of an unserved customer. Consequently [12],

$$\tilde{P}_B = \frac{1-\rho}{1-\rho^{K+1}} \rho^K. \quad (3)$$

In the following subsections, we derive P_{B_l} and P_{B_h} for an exponentially distributed service time. Substituting them in (1) yields the blocking probability P_B. In section 4.1 we analyze a queue with a preemptive-resume discipline, and in section 4.2 we analyze the non-preemptive case.

4.1. Blocking probabilities with a preemptive-resume discipline

The blocking probability of the high priority subtasks is exactly the same as for a non-priority system to which only high priority subtasks arrive. Consequently,

$$P_{B_h} = \frac{1-\rho_h}{1-\rho_h^{K+1}} \cdot \rho_h^K \quad (4)$$

By (2), (3), (4), the blocking probability of the low priority subtasks is

$$P_{B_l} = \frac{\lambda}{\lambda_l} \frac{1-\rho}{1-\rho^{K+1}} \rho^K - \frac{\lambda_h}{\lambda_l} \frac{1-\rho_h}{1-\rho_h^{K+1}} \cdot \rho_h^K. \quad (5)$$

4.2. Blocking probabilities with a non-preemptive discipline

Van Doremalen [13] presented a recursion and an explicit formula for the calculation of the blocking probabilities for this system. He showed that

$$P_{B_l} = \sum_{i=0}^{K} \frac{1-\rho}{1-\rho^{K+1}} \cdot \rho^i \cdot P_{B_l}(i), \tag{6}$$

where

$$P_{B_l}(i) = \frac{(\frac{\mu}{\lambda_h})^i - 1}{(\frac{\mu}{\lambda_h})^K - 1}. \tag{7}$$

P_{B_h} can now be calculated using (2), (3) and (6):

$$P_{B_h} = \frac{\lambda}{\lambda_h} \cdot \frac{1-\rho}{1-\rho^{K+1}} \cdot \rho^K - \frac{\lambda_l}{\lambda_h} \cdot \sum_{i=0}^{K} \frac{1-\rho}{1-\rho^{K+1}} \cdot \rho^i \cdot P_{B_l}(i). \tag{8}$$

5. Delay distribution for a priority M/M/1 queue

Let $S_h(t)$ and $S_l(t)$ be the sojourn time probability distributions of high- and low-priority subtasks, respectively. Let $S(t)$ be the delay distribution of a dispersed task. Finally, let $S_h^*(s)$, $S_l^*(s)$, and $S^*(s)$ be the respective Laplace transform. With the independence assumption, the $m+r$ queues accessed by the $m+r$ subtasks of a dispersed task are i.i.d. Consequently, the sojourn times of each high- and low-priority subtask are distributed according to $S_h(t)$ and $S_l(t)$, respectively, independent among subtasks. Therefore, the dispersed task delay is equal to the m^{th} out of $m+r$ order statistics from a parent population equal to the distribution of a single queue. Accordingly,

$$S(t) \equiv Pr(sojourn\ time \leq t) = \sum_{i_m=(m-r)^+}^{m} \left(\binom{m}{i_m} (S_h(t))^{i_m} (1-S_h(t))^{m-i_m} \times \sum_{i_r=m-i_m}^{r} \binom{r}{i_r} (S_l(t))^{i_r} (1-S_h(t))^{r-i_r} \right). \tag{9}$$

For H-P subtasks in preemptive-resume systems, the delay distribution is the same as in a non-priority M/G/1 queue with arrival rate λ_h. For H-P subtasks in non-preemptive systems, we quote the results of Daigle [14], section 4.6.1. Daigle derived the Laplace transform of the sojourn time of a tagged H-P subtask as the probability generating function for the number of HP subtasks left in the system at its departure, evaluated at the point $z = 1 - \frac{s}{\lambda}$. Validity of this technique is based on the fact that the H-P subtasks in the system at the end of a sojourn time are the H-P subtasks that arrived (according to a poisson process) during that sojourn time. Such a technique cannot be used for L-P subtasks which are served in LCFS order.

Preemptive-resume systems as seen by the L-P subtasks are modeled as an LCFS system with exceptional first service and service interruptions at rate λ. The service times of all (high and low priority) subtasks that arrived during the sojourn time of a tagged L-P subtask are included in it's sojourn time. Finally, for L-P subtasks with an N-P discipline, we quote the results of Doshi and Lipper [15]. A detailed derivation of the delay distributions is omitted for brevity. The interested reader is referred to [11].

5.1. Delay distribution for priority M/M/1 – preemptive-resume

5.1.1 Delay distribution of high-priority subtasks

$$S_h(t) = 1 - e^{-\mu(1-\rho_h)t} \tag{10}$$

5.1.2 Delay distribution of low priority subtasks

$$S_l^*(s) = B_f^*(s + \lambda - \lambda Y_0^*(s)), \tag{11}$$

Where,

$$Y_0^*(s) = \frac{1}{2\lambda} \left(\mu + \lambda + s - \left((\mu + \lambda + s)^2 - 4\lambda\mu \right)^{0.5} \right), \tag{12}$$

and

$$B_f^*(s) = \frac{s(1-\rho_h)}{s - \lambda_h + \lambda_h \frac{\mu}{s+\mu}} \cdot \frac{\mu}{s+\mu}. \tag{13}$$

$S_l(t)$ is obtained by numerically inverting $S_l^*(s)$.

5.2. Delay distribution for priority M/M/1 – non-preemptive

5.2.1 Delay distribution of high-priority subtasks

For H-P subtasks in a non-preemptive system, we quote the results of [14], section 4.6.1. The sojourn time is given by

$$S_h(t) = \frac{\lambda_l\ (\lambda_h - \mu)}{e^{\mu t}\, \lambda_h\, \mu\ (\rho_h - 1)} + \frac{e^{(\lambda_h - \mu)\, t}\ (\lambda_h + \lambda_l - \lambda_h\, \rho)}{\lambda_h\ (\rho_h - 1)} - \frac{\lambda_l + \mu - \mu\, \rho}{-\mu + \mu\, \rho_h}. \tag{14}$$

5.2.2 Delay distribution of low-priority subtasks

For L-P subtasks in a non-preemptive system, we quote the results of [15]. The sojourn time LST (Laplace-Stieltjes Transform) can be calculated as follows:

$$S_l^*(s) = B^*(s) \cdot J^*(s + \lambda - \lambda G^*(s)), \qquad (15)$$

where

$$J^*(s) = \begin{cases} \frac{s(1-\rho)+(1-B^*(s))\lambda}{s-\lambda_h-\lambda_h B^*(s)} & \rho < 1 \\ \frac{(1-\rho_h)(1-B^*(s))}{-s\mu-\rho_h+\rho_h B^*(s)} & \rho > 1 \end{cases} \qquad (16)$$

and

$$G^*(s) = B^*(s + \lambda - \lambda G^*(s)) \qquad (17)$$

$S_l(t)$ is obtained by substituting $\frac{\mu}{\mu+s}$ for $B^*(s)$ and numerically inverting $S_l^*(s)$.

6. Blocking probability and delay distribution in dispersal systems – numerical results and comparison

In order to evaluate the prioritized-dispersal schemes, we numerically compare the performance of all variants of dispersal systems. These include redundant dispersal systems with and without priority, non-redundant dispersal systems and non-dispersal systems. The service time of a subtask is assumed to be exponentially distributed in all cases. The performance measures are blocking probability (of the task) with a given buffer capacity and the probability of exceeding a given (overall task) delay. The former are derived for a buffer capacity of 10 subtasks, and the latter are derived for a delay threshold of 5 times the mean service time of an entire task. Results are shown for various "net" loads, defined as the load of non-redundant subtasks. (For detailed derivations pertaining to all schemes, see [11].)

Figure 3 shows the improvement achieved by the prioritized-dispersal scheme. It compares blocking probabilities and those of the delay exceeding a given threshold as a function of load for the following systems: $(1,0)$ non-dispersal; $(4,0)$ non-redundant dispersal; $(4,1)$ redundant dispersal without a priority mechanism, and redundant dispersal with preemptive-resume and non-preemptive disciplines ($(4,1)-$PR, $(4,1)-$NP). The results show that, across a wide range of loads, the prioritized-dispersal systems have the best performance. With $\rho = 0.7$, for example, the probability for the delay to exceed 5 is 3.5 times smaller with $(4,1)-$PR than with $(4,0)$, and 20 times smaller than with $(4,1)$. Blocking probability when $\rho = 0.7$ is 5.7 times smaller with $(4,1)-$PR than with $(4,0)$ and 2.6 times smaller than with $(4,1)$.

One may notice that performance with a $(1,0)$ non-dispersed system is very poor relative to performance with

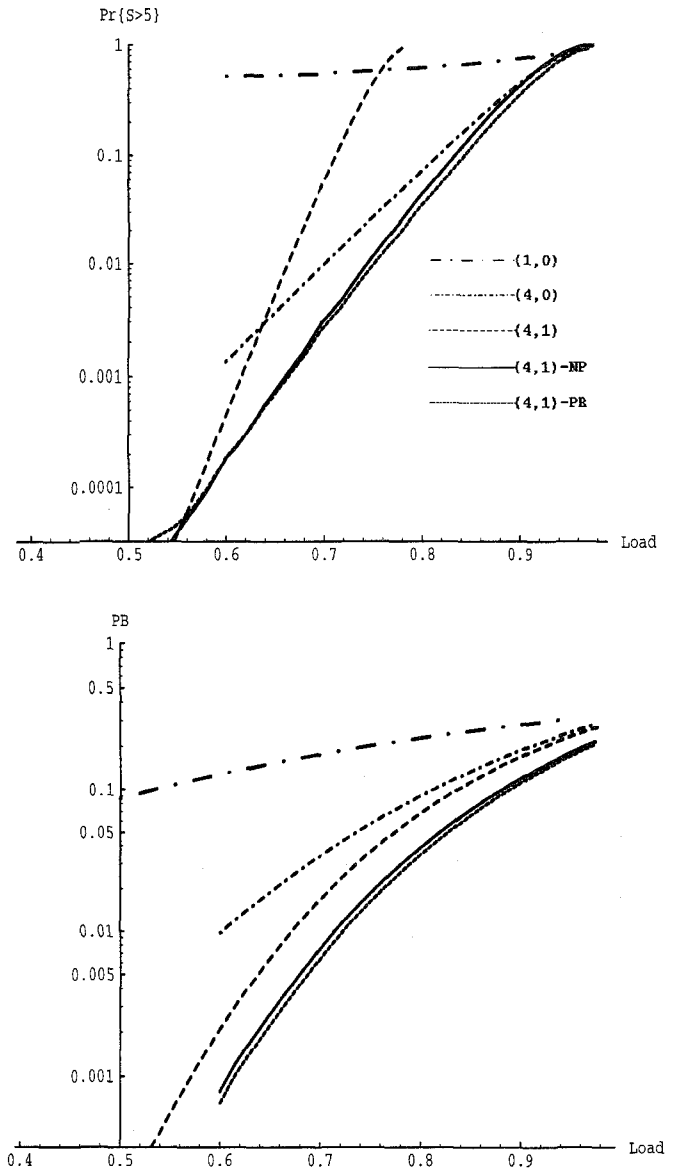

Figure 3. The probability that a dispersed task's delay exceeds 5 full-task service times (top) and the blocking probability with K=10 (bottom) for the following systems: $(1,0)$ non-dispersal; $(4,0)$ non-redundant dispersal; $(4,1)$ non-prioritized redundant dispersal; $(4,1)-$PR redundant-dispersal with a preemptive-resume priority discipline; and $(4,1)-$NP redundant-dispersal with a non-preemptive priority discipline.

dispersed systems. This is due to the fact that with non-dispersed systems, service time of a task is m times larger and each task occupies m buffer locations upon arrival. (For fairness of comparison, the m buffers are assumed to be freed one by one as the serice of the non-dispersed task progresses.)

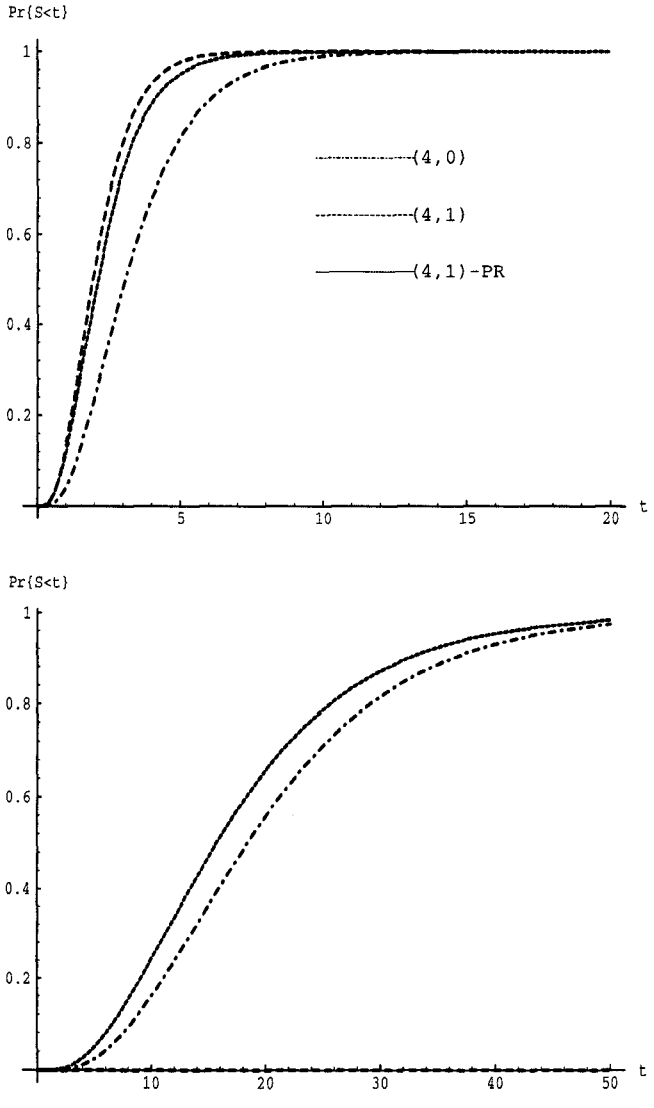

Figure 4. The probability density function of the delay with $\rho = 0.4$ (top) and $\rho = 0.9$ (bottom) for the following systems: $(4,0)$ non-redundant dispersal; $(4,1)$ non-prioritized redundant dispersal; $(4,1)-\text{PR}$ redundant-dispersal with a preemptive-resume priority discipline.

Figure 4 shows the effect of prioritized dispersal on the probability distribution function of task delay at both low load ($\rho = 0.4$) and heavy load ($\rho = 0.9$). At low load, the non-prioritized $(4,1)$ redundant dispersal scheme slightly outperforms the $(4,1)$ prioritized-dispersal scheme, with the non-redundant $(4,0)$ dispersal a distant third. A heavy "net" load ($\rho = 0.9$) results in a load of more than 1.0 with the redundant schemes. With a FCFS policy and no priorities, the implication is infinite delay for all subtasks, resulting in the absence of a plot for the $(4,1)$ scheme. In this case, the $(4,1)-\text{PR}$ is thus far better than its competitors.

Figure 5 demonstrates the importance of the LCFS discipline for L-P subtasks, mainly at moderate loads. The figure depicts the probability for the delay to exceed 5 with a non-preemptive priority discipline, as a function of load. For very heavy load, close to 1, most of the L-P subtasks are lost and they have only a small effect on performance. For low loads, the probability of two or more unserved subtasks waiting in the queue is very small, so the queuing discipline again hardly affects performance. However, at an intermediate load of $\rho = 0.8$, the probability of a task's delay exceeding the threshold of 5 is approximately twice as high if FCFS rather than LCFS is used for L-P subtasks.

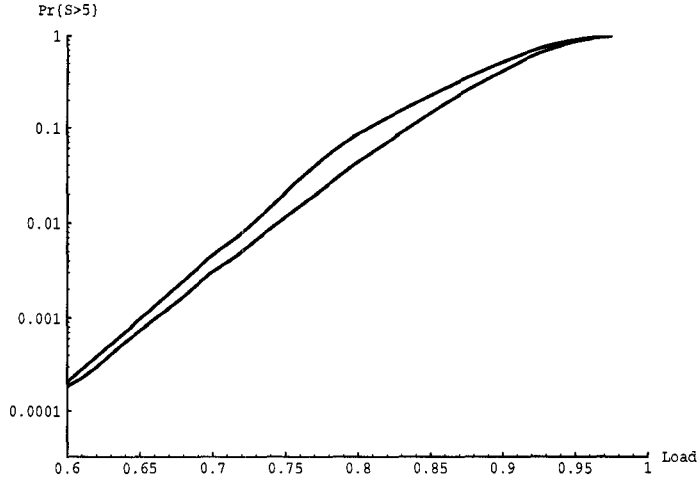

Figure 5. The probability for a dispersed task's delay to exceed 5 full-task service times with a preemptive–resume priority discipline and with LCFS (lower curve) or FCFS (upper curve) queuing disciplines for L-P subtasks.

7. Conclusions

In this paper, we presented and analyzed the "prioritized dispersal" family of schemes for resource allocation in redundant-resource systems, focusing on the context of communication networks. Numerical results for Poisson arrivals and exponentially-distributed service times show a

dramatic reduction in both blocking probability and that of exceeding a given delay threshold relative to non-prioritized and non-redundant allocation schemes. The results for generally distributed service time are expected to be similar, but their derivation has yet to be completed.

The distinction made with prioritized dispersal between redundant and non-redundant resources moreover enables us to combine prioritized dispersal with "Join the shortest queue" allocation schemes [16]. With such schemes, system state should be known at request-arrival epochs. When such a scheme is applied to dispersal routing systems, only m submessages are transmitted through the m least loaded paths (out of $m + r$ paths). Recently, it was suggested to apply such schemes to redundant disk arrays [5], [17]. With the combined scheme, high-priority subtasks would be allocated to the least loaded servers while low priority subtasks would be allocated to the other servers. This combined scheme is expected to outperform its constituent schemes, and warrants further exploration.

Another performance improvement for those schemes which was left for future research is achieved by assigning different low priorities to different redundant subtasks of each dispersed task. Such an approach is likely to further improve the distribution of losses among different messages. Analysis of prioritized dispersal with multiple low priorities can be carried out by aggregation of classes to two priority classes.

In summary, prioritized dispersal appears to be an attractive technique for improving performance of distributed, redundant-resource systems, and serves as yet another example of the merit of selective exploitation of redundancy.

Acknowledgments. The authors would like to thank Moshe Sidi for providing the code that was used to compute the inverse Laplace transform. Also, they are grateful to Scientific and Engineering Software Inc. for providing the Workbench simulation package which was used in obtaining the simulation results presented in this paper.

References

[1] M.O. Rabin, "Efficient Dispersal of Information for Security, Load Balancing, and Fault tolerance", *J. ACM*, vol. 36, pp. 335-348, Apr. 1989.

[2] D.A. Patterson, G.A. Gibson and R.H. Katz, "A Case For Redundant Arrays of Inexpensive Disks", *proc. ACM SIGMOD Conf.*, Chicago,IL, pp. 109-116, June 1988.

[3] N.F. Maxemchuk, "Dispersity Routing", *proc. Int. Commun. Conf.*, pp. 41.10-41.13, 1975.

[4] Y. Birk, "Method and apparatus for supplying data streams", U.S. Patent No. 5,592,612, Jan. 1997.

[5] Y. Birk, "Random RAIDs with selective exploitation of redundancy for high performance video servers", *Proc. NOSSDAV'97, St. Louis, MO, May 1997.*

[6] B.T. Doshi, H. Heffes, "Overload performance of several processor queueing Disciplines for the M/M/1 queue", *IEEE Trans. on Commun.*, vol. COM-34, no. 6, pp. 538-546, June 1986.

[7] R.C. Larson, "Perspectives on queues: Social justice and the psychology of queues", *Operations Research*, vol. 35, no. 6, pp. 895-905, Nov.-Dec. 1987.

[8] S.Li.J. Ozekici and F.S. Chou, "Queues with impolite customers", *Queueing Systems* 45, pp. 267-277, 1994.

[9] S.Li.J. Ozekici and F.S. Chou, "Waiting time in M/G/1 queue with impolite arrival disciplines", *prob. Eng. Inf. Sci.* 9, pp. 255-267, 1995.

[10] L. Kleinrock, *Queueing Systems*, vol. 2. New York: Wiley, 1976.

[11] Y. Birk and N. Bloch, "Priority Flooding: A Novel Scheme for Load Balancing in Redundant-Resource Systems ," CC Pub. #191, Electrical Engr. Dept, Technion, May. 1997.

[12] L. Kleinrock, *Queueing Systems*, vol. 1. New York: Wiley, 1976.

[13] J.B.M. Van Doremalen, "A Note on 'Analysis of a finite capacity nonpreemptive queue'," *Computers and Operations Research*, vol. 13, no. 4, pp. 525-526, 1986.

[14] J.N. Daigle, *Queueing Theory for Telecommunications*, Addison-Welsey, 1992.

[15] B.T. Doshi, E.H. Lipper, "The throughput performance of a prioritized LIFO service discipline", *Operations Research Letters*, vol. 3, no. 2, pp. 75-80, June 1984.

[16] F.A. Haight, "Two queues in parallel", *Biometrika* 45, pp. 401-410, 1958.

[17] S. Berson, R.R. Muntz and W.R Wong, "Randomized Data Allocation for Real-time Disk I/O", *proc. IEEE Compcon'96* vol. 11, no. 4, pp. 631-640, May 1996.