A Dual Priority MVA Model for a Large Distributed System: LOCUS *

Joseph Betser, Mario Gerla, and Gerald Popek

Computer Science Department

University of California, Los Angeles

Los Angeles, CA 90024

May 1984

* This research has been supported by the Advanced Research Projects Agency, under contract DSS-MDA-903-82-C-0189.

A Dual Priority MVA Model for a Large Distributed System: LOCUS *

A twofold extension to an MVA based analytical model is presented. A dual priority approximation is developed for a mixed queueing network. Improved fit to experimental results, obtained from our distributed system, LOCUS, is observed. Issues of file replication in large (up to 100 sites) distributed systems are examined, with respect to performance measures. Important design guidelines are obtained. Finally, the computational requirements of our model are assessed. Performance prediction of yet larger systems appears quite feasible.

1. Introduction

The analytic issues of distributed systems are rich and complex. Moreover, as the builders of such systems are becoming increasingly ambitious, even larger systems are being conceived. This upscaling in the system dimensions leads to an explosion in the combinatorial space of design issues involved. It is becoming more apparent than ever that a thorough understanding of the theoretical foundation of these systems should be sought, in order to generate cost-effective designs.

A DUAL PRIORITY MVA MODEL FOR A LARGE DISTRIBUTED SYSTEM: LOCUS

Joseph Betser
Mario Gerla
Gerald Popek

1984
Report No. CSD-840036

This paper addresses one such design issue, namely the problem of supporting replicated files* in the context of system scaling. Replicated files increase the availability and reliability of data in distributed systems. However, this feature has a cost in terms of system resources. We study these resource demands, and provide a general analytical model, which is then applied to a particular real system for validation and further performance prediction.

1.1 The LOCUS System

LOCUS, the test system used for experimental comparison with the analytic model predictions, is a distributed operating system which supports transparent access to data through a network wide system, permits automatic replication of storage, supports transparent distributed process execution, such as nested transactions, and is upward compatible with Unix. Partitioned operation of subnets and their dynamic merge are also supported [POPE81], [WALK83b]. LOCUS is functioning as the distributed operating system of the UCLA Center of Experimental Computer Science (CECS) network. It has been operating at UCLA since 1981, and is currently run on 17 VAX 11/750s, connected by a 10 Mbit/sec Ethernet, as described in Fig 1.

1.2 High Priority Servers

In LOCUS, application program access to remote and local resources is achieved using the same system semantics. This is the notion of *Network Transparency*. If performance in operations across the network is comparable to intra-site behavior, then *performance transparency* [POPE81] has also been achieved. In order to achieve this goal, LOCUS uses high priority servers. These *kernel* servers are permanently resident in system space, execute in background, and most importantly, run at high priority. Such servers include the *Network Message* server, the *Topology Change* server, and the *Update Propagation* server. The Update Propagation servers are responsible for maintaining replicated files in a con-

* By replicated files we mean identical copies of the same file, stored by disjoint sites.

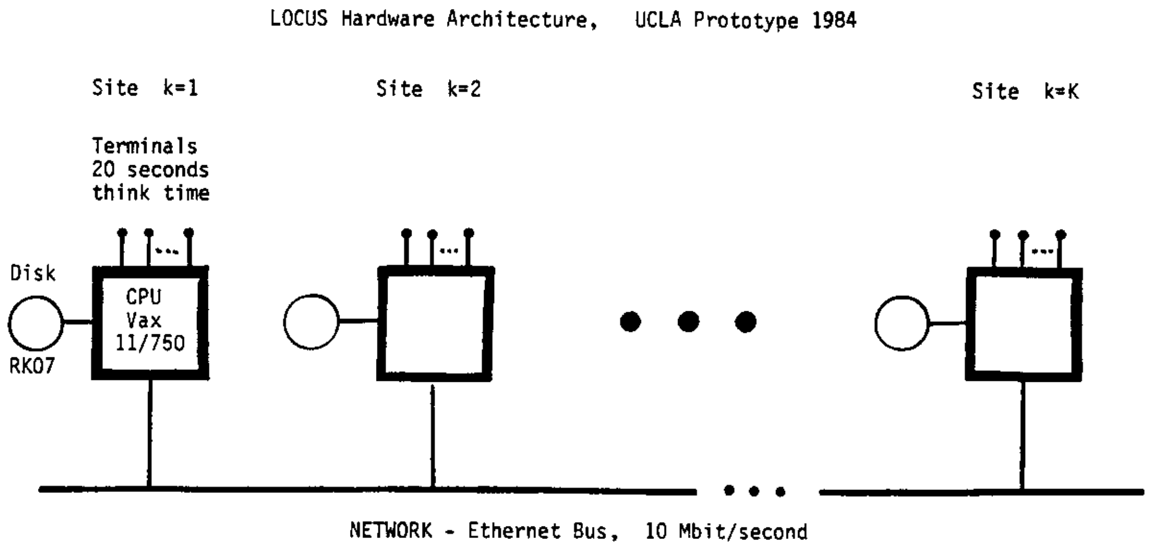

LOCUS Hardware Architecture, UCLA Prototype 1984
Site k=1
Site k=2
Site k=K
Terminals
20 seconds
think time
Disk
CPU
Vax
11/750
RK07
NETWORK - Ethernet Bus, 10 Mbit/second

Fig 1 *The LOCUS System Hardware Architecture*

sistent state, by propagating current modifications to all copies of the replicated file*. This means that whenever there is work for any of these servers, other jobs will be *preempted*, and the site will serve the high priority job immediately. The preempted jobs will *resume* execution when resources become available.

While this approach guarantees high performance for these background activities, system resources are being consumed, and foreground activity is slowed down in the presence of this high priority activity. The increased load on the system is of concern, as it can cause congestion in extreme cases. In fact, the *degree of replication* is an important aspect of system scaling, which we shall address, since significant Update Propagation activity directly affects system performance.

* The high priority scheduling of the Update Propagation servers insures that any momentary inconsistencies will be kept to a short, transient period [EDWA82].

1.3 Synopsis of the Paper

The next sections of this paper first provide a summary of previous work, and then proceed to a presentation of the model derivation, with a description of its applicability to LOCUS. A comparison of our results to previous work follows, and then extensions to larger systems and their performance analysis are discussed. Lastly, implications of the results are considered, and directions for further accomplishment are suggested.

2. Pertinent Work

An initial model of the LOCUS distributed system was presented by [GOLD83]. The model was a central server, product form network, and the solution technique was based on Mean Value Analysis (MVA) [REIS80]. Some work was done to represent background processes, but no attempt to incorporate priorities in the model was made. Significant discrepancies with respect to experimentally observed results were reported.

Recent related work in the general area of priority queueing networks was done by [CHAN82] and [BRYA83]. Both algorithms are MVA based. The approach by Chandy and Lakshmi is based on calculating waiting times for the different job classes, and subsequently determining the remaining performance measures. Bryant et al utilize an iterative process to compute server utilizations, monitoring them to establish convergence. They are basing their work on the approximation that the arrival theorem [SEVC79], [LAVE79] is reasonably accurate for (non product form) priority disciplines. Bryant et al deal with closed networks only. Neither [CHAN82] nor [BRYA83] use a real environment to validate their models. Simulation and exact Global Balance techniques for relatively small models are used.

In this work, care was taken to represent the priority queuing disciplines at the system sites. A Hi-Low dual priority discipline was modeled, in which the *foreground activity (low priority)* performance measures are affected by the *background activity (high priority)*. Our solution technique, unlike [CHAN82] or [BRYA83], is based on reducing the model into a system of nonlinear equations, solving for the chain throughputs, and then for the rest of the performance measures. This approach is based on [GOLD83].

3. Representing Priority Jobs within LOCUS

We model the LOCUS system by the queueing network described in Figure 2 below.

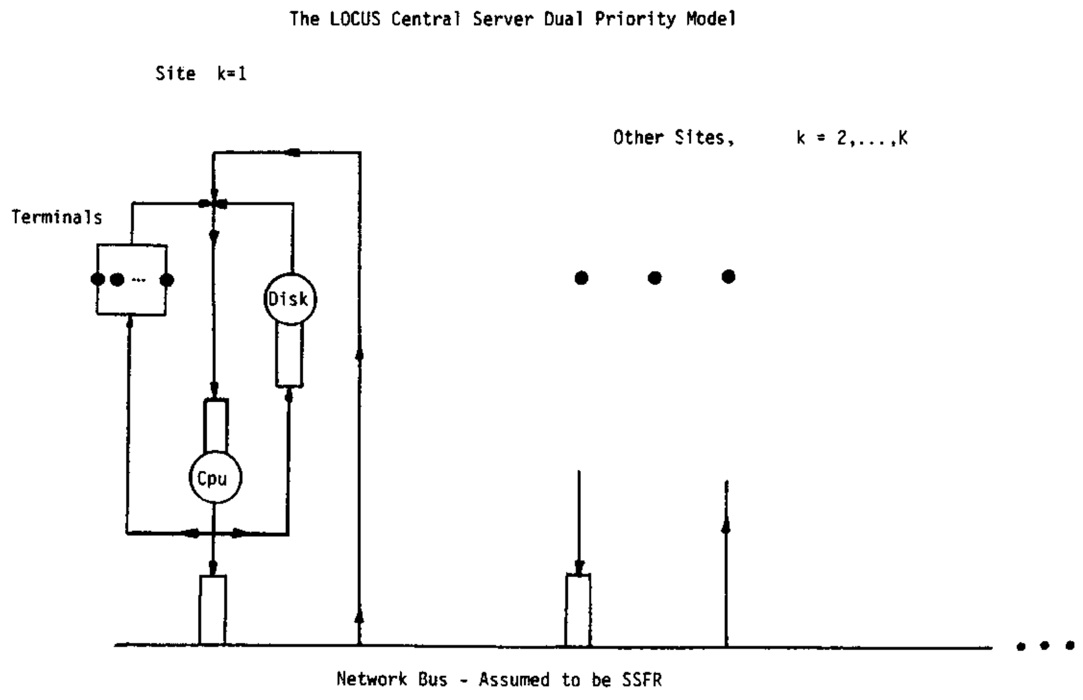

The LOCUS Central Server Dual Priority Model
Site k=1
Other Sites, k = 2,...,K
Terminals
Disk
Cpu
Network Bus - Assumed to be SSFR

Fig 2 *The LOCUS Priority Queueing Network Model*

Our goal in the model derivation is to achieve expressions for the throughputs at the designated service centers. The final step of our derivation will be a system of nonlinear equations which is solved numerically.

3.1 Governing Assumptions:

-- The network contains two types of servers:

- Infinite Servers (IS - terminals)
- Dual Priority Single Server Fixed Rate (SSFR - cpu, disks, network)

-- Two job priorities are present in the network:

- Low priority - foreground jobs - closed chains
- High priority - background jobs - open chains

3.2 Definitions and Notation:

K	- total # of chains (# of sites), indexed by k.
$N_1, N_2, \ldots, N_K$	- population vector.
J	- total # of service centers, indexed by j.
	$1,2,\ldots,J'$ IS centers
	$J'+1,\ldots,J$ SSFR
$d(k)$	- designated server in chain k (typically a user's terminal).

for type k customer at service center j:

T_{kj}	- mean service time of customer k at center j.
θ_{kj}	- # of visits to service center j between visits to d(k)
$a_{kj} \equiv \theta_{kj} T_{kj}$	- mean relative load of customer k at center j.
$\lambda_k(\vec{N})$	- throughput at designated service centers d(k).
$L_{kj}(\vec{N})$	- mean # of customers of type k at center j.
$L_j(\vec{N}) = \sum_k L_{kj}(\vec{N})$	- overall average # of customers at service center j.
$R_{kj}(\vec{N})$	- mean response time of chain k at center j.

3.3 Obtaining the System's Governing Equations

We first state the MVA [REIS80] equations, in which the Schweitzer approximation [SCHW79] has been incorporated [LAVE83]. This yields the response time at the network centers:

$$R_{kj}(\vec{N}) = \begin{cases} T_{kj} & j=1,2,\ldots,J' \\ T_{kj}\left[1 + L_j(\vec{N}) - \dfrac{L_{kj}(\vec{N})}{N_k}\right] & j=J'+1,\ldots,J \end{cases} \tag{1}$$

The top line corresponds to the IS centers, whereas the bottom line corresponds to the SSFR centers.

In deriving a simple model to represent the high priority service of the kernel server jobs, we shall look at a simplified, dual priority service center. Such an SSFR service center is given in figure 3.

We now introduce the concept of high priority background jobs. These jobs are *spawned* by the foreground jobs, and belong to an open chain. These high priority preemptive jobs do not "see" any low priority jobs in front of them, since they are serviced immediately. Furthermore, they run only at SSFR centers, so that their response time R_{0j} is obtained directly from the M/M/1 model.

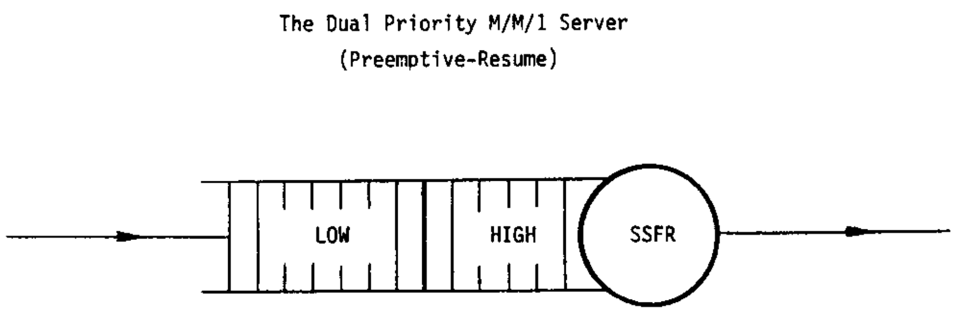


Fig 3 *The Dual Priority SSFR Service Center*

$$R_{0j} = \frac{U_{0j}}{1-U_{0j}} T_{0j} \tag{2}$$

where U_{0j} is the utilization due to the high priority jobs, and T_{0j} is the service time for jobs of this class. The index "$0j$" will denote the high priority job class in the rest of this paper. The response time of the high priority jobs will thus be simply given by (2) above, as per M/M/1. The resulting response time $R_{kj}(\vec{N})$ of the low priority jobs is increased, and will include the following three terms:

first term -	time to serve all low priority customers ahead in the queue.
second term -	service time of all high priority jobs, entering the system before the low priority customer has completed service.
third term -	service time of all high priority customers present in system upon arrival of the low priority customer.

and explicitly

$$R_{kj}(\vec{N}) = [\ 1 + L_j^{k-}(\vec{N})\ ]\ T_j + R_{kj}(\vec{N})\lambda_{0j}T_{0j} + R_{0j} \qquad k=1,2,...,K \tag{3}$$

where

$L_j^{k-}(\vec{N}) = L_j(\vec{N}) - \frac{L_{kj}(\vec{N})}{N_k}$ - is the mean queue length of low priority jobs at center j, as seen by an arriving low priority job of class k.

λ_{0j}	- throughput of high priority jobs.
R_{0j}	- total system time of the high priority job.

Note that at this point we have made the following simplifying assumption regarding SSFR centers.

$$T_{kj} = T_j \qquad k=1,2,...,K \tag{4}$$

that is, all foreground jobs have a uniform service time for the various chains. We now substitute (2) in (3) to obtain:

$$R_{kj}(\vec{N}) = [1 + L_j^{k-}(\vec{N})]\,T_j + R_{kj}(\vec{N})U_{0j} + \frac{U_{0j}}{1-U_{0j}}T_{0j}$$

and lastly

$$R_{kj}(\vec{N}) = \frac{1 + L_j(\vec{N}) - \dfrac{L_{kj}(\vec{N})}{N_k}}{1-U_{0j}}T_j + \frac{U_{0j}}{(1-U_{0j})^2}T_{0j} \qquad k=1,2,...,K \tag{5}$$

3.4 Obtaining the Final Model

Substituting the above (5) in the MVA approximation (1), and using the proper notation, we can express $R_{kj}(\vec{N})$ as follows:

$$R_{kj}(\vec{N}) = \frac{T_j}{1-U_{0j}}\left[1 + L_j(\vec{N}) - \frac{L_{kj}(\vec{N})}{N_k}\right] + \frac{U_{0j}T_{0j}}{(1-U_{0j})^2} \qquad k=1,2,...,K \tag{6}$$

We make one further assumption regarding the SSFR centers,

$$T_{0j} = \beta_j T_j \tag{7}$$

where the β_j are constants for a specific system configuration and server jobs.

Substituting (7) into (5):

$$R_{kj}(\vec{N}) = \frac{T_j}{1-U_{0j}}\left[1 + \frac{\beta_j U_{0j}}{1-U_{0j}} + L_j(\vec{N}) - \frac{L_{kj}(\vec{N})}{N_k}\right] \qquad k=1,2,...,K \tag{8}$$

Next, we wish to express U_{0j} as a function of the closed chain throughputs $\lambda_k(\vec{N})$. Following the same approach as in [HEID81] and [GOLD83], spawned background processes are modeled as open chains, since they leave the network eventually. Thus, the utilization due to the spawned jobs U_{0j} can be expressed by:

$$U_{0j} = \sum_{k=1}^{K} a_{kj}^{BG} \lambda_k(\vec{N}) \tag{9}$$

where a_{kj}^{BG} are the relative loads due to the background traffic (which are known).

After a sequence of algebraic manipulations (which are reported in the appendix) we arrive at the following set of equations:

$$N_k = \lambda_k(\vec{N}) \left\{ \sum_{j=1}^{J'} a_{kj} + \sum_{j=J'+1}^{J} \frac{a_{kj}^{+} \left[1 + \frac{\beta_j U_{0j}}{1 - U_{0j}} \right] \cdot}{\left(1 + \frac{a_{kj}^{+} \lambda_k(\vec{N})}{N_k} \right) \left[1 - \alpha_j^{+}(\vec{N}) \right]} \right\} \qquad k=1,2,...,K \tag{10}$$

where

$$a_{kj}^{+} = \left(\frac{1}{1 - U_{0j}}\right) a_{kj}$$

and

$$\alpha_j^{+}(\vec{N}) = \sum_{k=1}^{K} \left\{ \frac{a_{kj}^{+}\lambda_k(\vec{N})}{1 + \frac{a_{kj}^{+}\lambda_k(\vec{N})}{N_k}} \right\}$$

This is a system of K non linear equations in the K unknowns $\lambda_k(\vec{N})$, $k=1,2,...,K$. The solution method entails choosing an initial solution vector $\lambda_k(\vec{N})$ for the chain throughputs. From there on, the solution is refined via successive iterations.

To make the numerical solution more efficient, the equation set (10) is rewritten as:

$$\begin{aligned} N_1 - F_1(\lambda_1) &= 0 \\ N_2 - F_2(\lambda_2) &= 0 \\ &\vdots \\ N_K - F_K(\lambda_K) &= 0 \end{aligned} \tag{11}$$

An efficient non-linear solution package MINPACK was employed in the solution [HEIB82]. It uses minimization techniques to drive the right hand side of equations (11) to zero.

We note that once the throughputs $\lambda_k(\vec{N})$ are obtained, the system response time is immediately obtainable by applying Little's result. This will generate the response time at the designated service centers, i. e. the terminals where the users are sitting.

4. Computational Results

The value of a new model is judged by the significance of the computational results obtained by its use. In this section the model will be applied to different configurations of the LOCUS system. The system performance will be studied, as different sets of parameters will be examined. Previous results will be compared to the figures obtained with this model, and a comparison to the experimental results will be made.

Application of this model to the analysis of systems much larger than the current LOCUS system will follow. Systems consisting of up to 100 sites, with an increasing degree of file replication, will be examined. A study of performance parameters, as a function of the system size, will be presented, and reliability tradeoffs with respect to performance will be indicated. The demands of our model on the computational resources, as the system grows in size, will be studied.

4.1 Obtaining the Model Parameters

A representative system configuration had to be chosen for evaluation. One fact to keep in mind while doing this, is our intention to compare the results to those of previous models, such as [GOLD83]. Accordingly, a decision was made to model 50-page file reads, on which extensive results have been reported.

Next, the a_{kj} must be defined in order to solve for the $\lambda_k(\vec{N})$. Obtaining the values a_{kj} involved *measurements* on the system. Recall that the a_{kj} represent the mean relative load of customer from chain k at service center j. Given below are the SSFR mean service times* for a single page read:

- cpu:	
- local read:	5.25 ms (T_l)
- remote read:	9.06 ms locally (T_{rl}), and 10.43 ms remotely (T_{rr})
- page processing:	9.00 ms (T_p)
- disks:	32.72 ms (T_d)
- network:	0.94 ms (T_n)
- terminals:	400.00 ms (T_t) (think time, 20000/50)

These values are the raw inputs used to evaluate the a_{kj} for a specific page read system loading configuration. This is done by selecting a typical cycle, such as the time between consecutive user commands on a terminal, and obtaining the products of the visit ratios θ_{kj} and the mean service times T_{kj}, which define the a_{kj} (section 3, definitions). similarly, the relative background loads a_{kj}^{BG} are evaluated. These will be used through the iterations to obtain U_{0j} as per equation (9). These service times are used to compute β as well, as we shall see shortly. Note that symmetrically loaded networks are modeled, so that the remote loads are equally distributed among the rest of the sites.

4.2 Comparing Previous Results to the Priority Enhanced Model

We first wish to compare our results to those of a previous background activity model by [GOLD83]. The main thrust in that work was to provide an initial model for the LOCUS system. Background processes were assumed to be spawned by foreground jobs [HEID81], but no priority queueing was

* All measurements reported here were made on UCLA 4.1bsd LOCUS prototype during early 1984.

assumed.

In our work, not only has the above spawning assumption been made, but the model was also extended to represent the fact that *Update Propagation,* and other system servers that execute in the background, run at high priority. This would obviously cause the foreground activity to be further delayed, due to the fact that the background server activity preempts the foreground jobs, and grabs the computation resources for as long as it needs. This is also true for the *Network Message Server* activity, since it too, runs at high priority, to help achieve *performance transparency.*

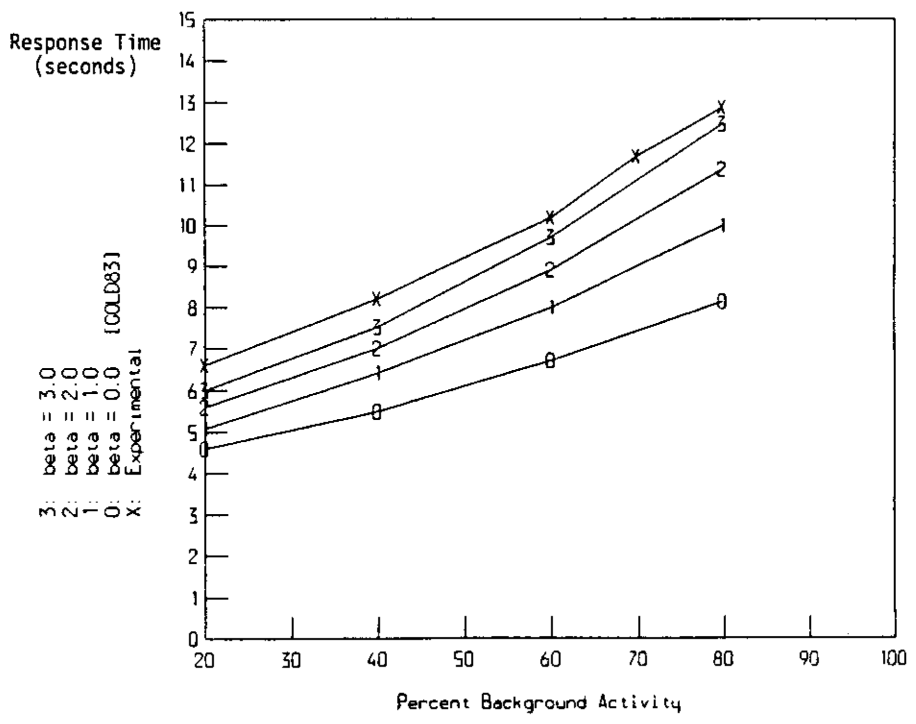
Thus, a test configuration similar to the one modeled in [GOLD83] was constructed, and the dual priority model was applied to it. This gave us an initial feel as to the improvements obtained with our model. The model consisted of 4 sites, each having 10 users. The mean think time of a user at a terminal was 20 seconds, and each user issued a 50-page read command per think period, in addition to a mean cpu processing time per page of 9.0 milliseconds. Of the 50-page read commands, 80% were local and 20% were remote. Figure 4 is a revisited analysis of the system presented in Fig 6 of [GOLD83]*. Note that in the previous model, there has been a substantial discrepancy between the experimental points, and the previous analytical curve (β=0). While the system presented response time values of 5-12 seconds within the plotted range, the analytical prediction was on the optimistic side, predicting 4-8 seconds for the same range. In fact, the discrepancy increased to over 55%, as the background activity increased. Goldberg in his paper suggests that the shortcomings of the model are resulting from both lack of priority representation, and lack of buffer contention** modeling.

Examining the plots in Fig 4, one notes that increasing values of the β parameter, lift up the analytically predicted curve. The evaluation of β for these experiments is done by using equation (7).

* Note that only the line corresponding to β=0 was given in [GOLD83], while the lines for other values of β have been obtained with the dual priority model.

** Buffer contention has to do with page storage in main memory. There is a finite number of such buffers available at each site. When the system is waiting for buffers to be freed, additional delays are incurred.

RESPONSE TIME VS. PERCENT BACKGROUND ACTIVITY



EXPERIMENTAL AND ANALYTICAL RESULTS

Fig 4 *Foreground Response Time as a Function of Background Activity*

$$\beta_j^{cpu} = \frac{T_{0j}}{T_j} = \frac{T_{rl} + T_{rr}}{T_l + T_p} = \frac{10.43 + 9.06}{5.25 + 9.00} = 1.37$$

for the disk access we have identical service times for local and remote accesses

$$\beta_j^{disk} = \frac{32.72}{32.72} = 1.00$$

Since the disk and the cpu are the dominating elements in the network (as we shall see in section 4.3.2 below), we obtain a global β as an average of the above β_j, i.e. β=1.18. Such values of β result in an approach to within 20-25 % of the actual experimental points. The remaining discrepancy is due to still unmodeled phenomena, such as buffer contention. Nevertheless, a significant improvement in our prediction capability is demonstrated.

4.3 Update Propagation Analysis - Tackling Larger Systems

So far, improvement of the new model with respect to previous models has been observed. Application of the model to Update Propagation activity is the next step. Below, the dual priority model is used to analyze larger systems, and scaling effects are studied. The system is scaled by increasing the number of identical sites, such as in figure 1. Furthermore, full file replication throughout the system is assumed, so that maximum background activity will be generated. That means that each replicated file has a copy at every site. Symmetrical loading as in section 4.2 above, is exercised on the system, and systems having from 2 to 100 sites are analyzed.

The mean service times per single propagated page are as follows:

cpus:	16.14 ms locally (T_{ul}), and 10.72 ms remotely (T_{ur})
disks:	32.71 ms locally, and 32.71 ms remotely (one access each) (T_d)
network:	0.94 ms (T_n)
terminals:	0.00 ms (background activity)

These are the raw inputs used to obtain the a_{kj} matrix elements representing the Update Propagation loads on the various service centers. Users at sites were simulated to be modifying replicated files. That was the only foreground activity in the system. This activity required a cpu processing of 9.00 ms (T_p) per data page. At the end of each think period (20 secs), the user committed modifications to a replicated file, which in turn, invoked Update Propagation in the background, to all other sites. The modification size consisted of 50-page updates, to be propagated to all sites. The evaluation of β for this system configuration is as follows:

$$\beta_j^{cpu} = \frac{T_{0j}}{T_j} = 0.5\,\frac{(T_{ul} + T_{ur})}{T_p} = 0.5\,\frac{(16.14 + 10.72)}{9.0} = 1.49$$

β_j^{disk} will be unity in this case as well, since identical disk service times are involved. Thus the global β here will be averaged to be 1.24.

4.3.1 Response Time Affected by Increasing Degree of Replication

The effect of the replication factor* on the response time is examined in this section. The results are plotted in figure 5 below.

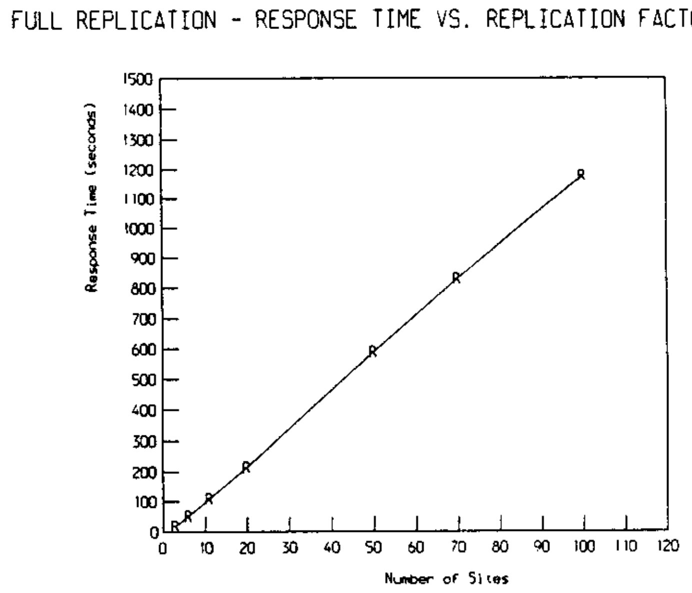


Fig 5 *Large Systems - Response Time vs. Replication Factor*

The increase in response time with increasing replication is nearly linear, which is explained by the following global argument. Update traffic grows as K^2 (with the replication factor K), while the number of servers grows as K, such that work per server increases linearly with K. This increase in response time is the performance penalty we pay for increased reliability and availability.

We note that the response time is becoming larger than a minute for replication factor of more than 10. This observation means that user files which are updated frequently, should not have a large replication factor**. It is important to realize that the load exerted on our system in this configuration is immense. We assumed that at each site, one of the 10 users modifies 50 pages of a file (thus generating a

* *Replication factor* and *degree of replication* are equivalent terms.

** System files, which are updated infrequently, do not have this problem, and in fact, are replicated throughout the system.

50-page update propagation to all other sites) every 2 seconds*. We did so since we were interested in identifying system bottlenecks under heavy load.

4.3.2 System Bottleneck Identification

In order to obtain better insight into the system's saturation behavior, we evaluated the server utilization as a function of the replication factor. Figure 6 below shows the total, foreground, and background utilizations of the SSFR centers in our system. Clearly, system saturation is disk dominated, as the disks demonstrate a utilization larger than 90% for any replication factor**. The cpus are at around 60% throughout, and the network utilizations increase from a fraction of a percent for small degrees of replication, to a noticeable 69% for K=100.

Examining the foreground and background plots, we note that as K increases, so does the background traffic, to the point that it dominates the SSFR utilizations. The foreground utilizations drop to a mere fraction of a percent for systems having close to 100 sites with full replication. This is the reason for the long foreground response times, observed in Figure 5 above.

It is interesting to notice that there is no "explosion" of the system in the classical sense, as the utilizations never reach unity. The response time increases, so that the users do less and less foreground work (are blocked), as the interval between consecutive user commands is increased.

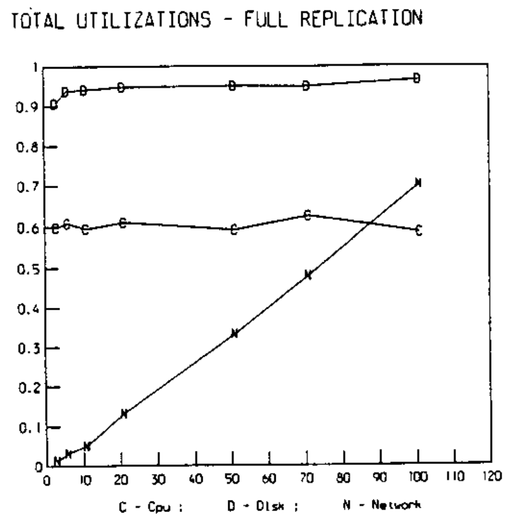
4.3.3 Computational Resource Requirements

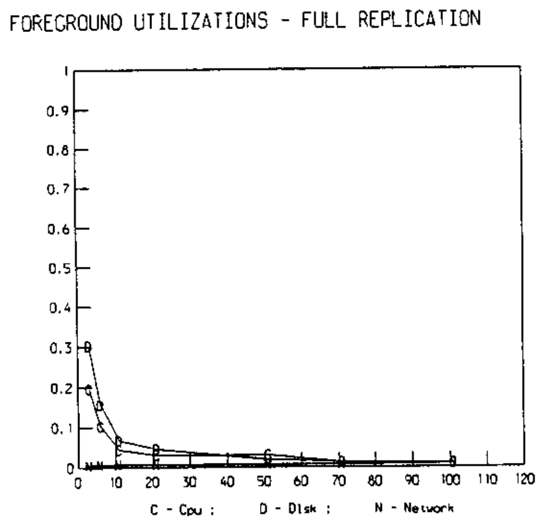
It is of interest to be able to predict system performance of large distributed systems. However, the increasing size of these systems poses rising computational demands on our computational resources***. Let us study this issue.

* Our experience indicates that file modifications involve a few pages at most, per user command.

** The RK07s will be replaced within a few months with higher performance disk drives.

*** All runs of the dual priority model were executed on a VAX 11/780 with a floating point accelerator.





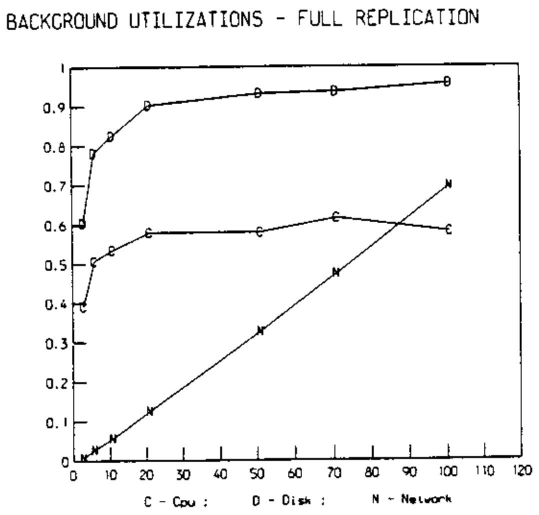
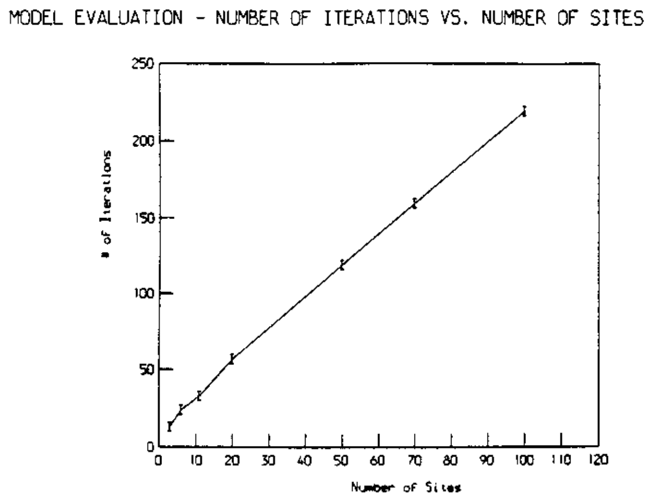


Fig 6 *SSFR (Cpu, Disk, Network) Utilizations vs. Replication Factor*

The major computation effort is in the solution of the non linear system. Not only does the *time* to execute a single iteration grow with the number of sites K, but so does the total *number* of iterations it takes to converge to a solution. Figure 7 below expresses these relationships. A nearly linear increase in the number of iterations, with the growth of the system, is observed. Expressing the total computation time with respect to system size for the dual priority model reveals a polynomial behavior, more precisely,



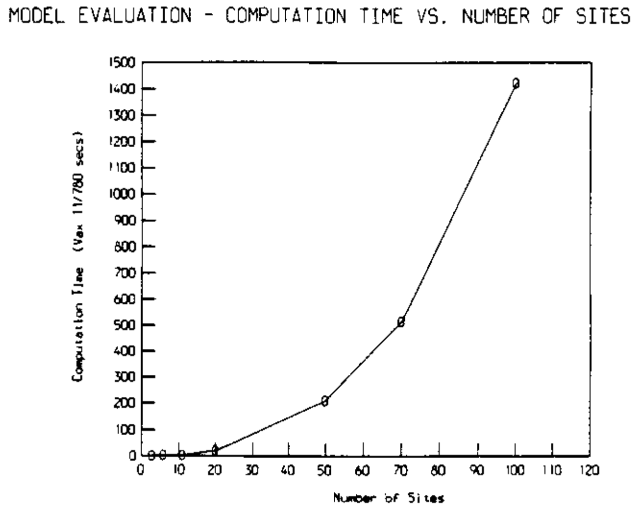


Fig 7 *Large Systems - Number of Iterations and Total Computation Time vs. Number of Sites*

an almost quadratic growth. This is an encouraging result, as *exact* Mean Value Analysis would require $O\,[\;JK\prod_k (N_k + 1\;)\;] >> O[K^2\;]$, *without priority representation.*

5. Discussion and Future Work

The need for priority queueing in distributed systems was motivated. Network performance transparency, and replicated file consistency and availability were presented as the primary beneficiaries of a priority queueing policy. A simple dual priority approximation was derived, based on a product form MVA approximation. The system was assumed to be of a central server type, with SSFR priority centers, as well as IS centers. An M/M/1 priority scheme was assumed for the SSFR centers. The model was incorporated into an MVA scheme with the Schweitzer approximation, and a nonlinear equation system was solved to obtain the performance metric prediction.

The model was applied to a 4-site system, similar to the one studied by [GOLD83]. The dual priority model improved performance prediction, as increased foreground delays were projected, although further refinement is still justified. The problem of supporting file replication for large distributed systems was addressed, and the performance penalties incurred by high degree of replication were demonstrated. The dominating resource bottlenecks were then identified, and a conceptual design policy was devised. This policy calls for a high degree of replication for seldomly modified system files, and a carefully optimized replication factor for user files, which are subject to intensive modifications. The computational effort involved in obtaining a solution for the model proved to be polynomial with the system size. This indicates the feasibility of analyzing larger distributed systems.

5.1 Future Avenues to Pursue

Many further directions can be taken to extend this research. One may want to address even larger systems, or do reliability and performance cost optimizations in the replication context. An interesting issue is that of modeling the buffer contention in LOCUS, in order to refine the results obtained so far. Loads on the system could be asymmetric and, in fact, one may want to address a large array of static and dynamic load balancing issues. More heterogeneous systems including different processors, workstations and I/O devices, consisting of several networks interconnected via gateway links, are only a few of the rich set of open problems still to be tackled.

5.2 Acknowledgements

We are indebted to many members of the LOCUS research project for having had the opportunity to conduct this research. Special thanks to Bruce Walker and Richard Guy for very efficient system support. Some of the results presented here are due, in part, to interaction with Edmundo de Souza e Silva, and Steven Berson. We would like to thank Edmundo Silva for reviewing the draft of this paper, giving his constructive comments. Much gratitude is extended to Arthur Goldberg, for his numerous suggestions throughout this study. Lastly, we would like to thank Steve Lavenberg for his timely advice.

APPENDIX - Detailed Derivation of the Non-Linear Equation Set

Staring off from equation (8):

$$R_{kj}(\vec{N}) = \frac{T_j}{1-U_{0_j}}\left[1 + \frac{\beta_j U_{0_j}}{1-U_{0_j}} + L_j(\vec{N}) - \frac{L_{kj}(\vec{N})}{N_k}\right] \tag{8}$$

and introducing the short hand notation:

$$A_j \equiv \frac{\beta_j U_{0_j}}{1-U_{0_j}} \tag{a.1}$$

(8) reduces to:

$$R_{kj}(\vec{N}) = \frac{T_j}{1-U_{0j}}\left[1 + A_j + L_j(\vec{N}) - \frac{L_{kj}(\vec{N})}{N_k}\right] \quad \text{(a.2)}$$

Going after mean queue lengths

$$L_{kj}(\vec{N}) = \theta_{kj}\lambda_k(\vec{N})R_{kj}(\vec{N}) = \theta_{kj}\lambda_k(\vec{N})\frac{T_{kj}}{1-U_{0j}}\left[1 + A_j + L_j(\vec{N}) - \frac{L_{kj}(\vec{N})}{N_k}\right] \quad \text{(a.3)}$$

Recalling that $a_{kj} = \theta_{kj}T_{kj} = \theta_{kj}T_j$ we have

$$L_{kj}(\vec{N}) = \frac{a_{kj}\lambda_k(\vec{N})}{1-U_{0j}}\left[1 + A_j + L_j(\vec{N}) - \frac{L_{kj}(\vec{N})}{N_k}\right] \quad \text{(a.4)}$$

Letting $a_{kj}^{+} \equiv \frac{a_{kj}}{1-U_{0j}}$ to obtain:

$$L_{kj}(\vec{N}) = a_{kj}^{+}\lambda_k(\vec{N})\left[1 + A_j + L_j(\vec{N}) - \frac{L_{kj}(\vec{N})}{N_k}\right] \quad \text{(a.5)}$$

Extracting $L_{kj}(\vec{N})$ from both sides we get:

$$L_{kj}(\vec{N}) + a_{kj}^{+}\lambda_k(\vec{N})\frac{L_{kj}(\vec{N})}{N_k} = a_{kj}^{+}\lambda_k(\vec{N})\left[1 + A_j + L_j(\vec{N})\right]$$

or

$$L_{kj}(\vec{N}) = \frac{a_{kj}^{+}\lambda_k(\vec{N})\left[1 + A_j + L_j(\vec{N})\right]}{1 + \frac{a_{kj}^{+}\lambda_k(\vec{N})}{N_k}} \quad \text{(a.6)}$$

Summing over all k=1,2,...,K we obtain

$$L_j(\vec{N}) = \left\{\sum_{k=1}^{K}\left[\frac{a_{kj}^{+}\lambda_k(\vec{N})}{1 + \frac{a_{kj}^{+}\lambda_k(\vec{N})}{N_k}}\right]\right\}\left[1 + A_j + L_j(\vec{N})\right] \quad \text{(a.7)}$$

Denoting

$$\alpha_j^{+}(\vec{N}) \equiv \sum_{k=1}^{K}\left\{\frac{a_{kj}^{+}\lambda_k(\vec{N})}{1 + \frac{a_{kj}^{+}\lambda_k(\vec{N})}{N_k}}\right\} \quad \text{(a.8)}$$

we obtain

$$L_j(\vec{N}) = \alpha_j^{+}(\vec{N})\left[1 + A_j + L_j(\vec{N})\right]$$

or

$$L_j(\vec{N}) = \frac{\alpha_j^{+}(\vec{N})}{1 + \alpha_j^{+}(\vec{N})}\left[1 + A_j\right] \quad \text{(a.9)}$$

Substituting this result in the above (a.6) expression to obtain $L_{kj}(\vec{N})$:

$$L_{kj}(\vec{N}) = \frac{a_{kj}^{+}\lambda_k(\vec{N}) \left[1 + A_j + \frac{\alpha_j^{+}(\vec{N})}{1-\alpha_j^{+}(\vec{N})} (1 + A_j) \right]}{1 + \frac{a_{kj}^{+}\lambda_k(\vec{N})}{N_k}} \qquad \text{(a.10)}$$

and lastly

$$L_{kj}(\vec{N}) = \frac{a_{kj}^{+}\lambda_k(\vec{N}) \left[1 + A_j \right]}{\left(1 + \frac{a_{kj}^{+}\lambda_k(\vec{N})}{N_k} \right) (1 - \alpha_j^{+}(\vec{N}))} \qquad \text{(a.11)}$$

And now, that the necessary expressions for the SSFR queue length are established, it is high time to use Little's result again and obtain a general expression for the number of customers at each chain k:

$$N_k = \lambda_k(\vec{N}) \left\{ \sum_{j=1}^{J'} a_{kj} + \sum_{j=J'+1}^{J} \frac{a_{kj}^{+} \left[1 + \frac{\beta_j U_{0j}}{1 - U_{0j}} \right]}{\left(1 + \frac{a_{kj}^{+} \lambda_k(\vec{N})}{N_k} \right) \left[1 - \alpha_j^{+}(\vec{N}) \right]} \right\} \qquad (10)$$

which is the required equation set (10).

REFERENCES

[BASK 75] Baskett F., Chandy K., Muntz R., and Palacios F., "Open, Closed, and Mixed Networks of Queues with Different Classes of Customers", JACM Vol. 22, April 1975, pp. 248-260.

[BETS 84] Betser J., "Performance Issues of a Large Distributed System: LOCUS", Master's Thesis, Computer Science Department, University of California, Los Angeles, 1984.

[BRYA 83] Bryant R., Krzesinski A., and Teunissen P., "The MVA Pre-Empt Resume Priority Approximation", Proceedings of the 1983 ACM Sigmetrics Conference, August 29-31 1983.

[CHAN 82a] Chandy M., Neuse D., " Linearizer: A Heuristic Algorithm for Queueing Network Models of Computing Systems", CACM February 1982, pp. 126-134.

[CHAN 82b] Chandy K., Lakshmi S., "An Approximation Technique for Queueing Networks with Preemptive Priority Queues", University of Texas at Austin 1982.

[EDWA 82] Edwards D., "Implementation of Replication in LOCUS: A Highly Reliable Distributed Operating System", Master's Thesis, Computer Science Department, University of California, Los Angeles, 1982.

[GOLD 83] Goldberg A., Lavenberg S., Popek G., "A Validated Distributed System Performance Model", PERFORMANCE '83 - The Ninth International Symposium on Computer Performance Modeling, Measurement, and Evaluation. May 25-27, 1983, College park, Maryland.

[GOLD 84] Goldberg A., Personal communication.

[GUY 84] Guy R., "Reconciliation of Replicated Name Spaces", Master's Thesis, Computer Science Department, University of California, Los Angeles, 1984.

[HIEB 82] Hiebert, K., "An Evaluation of Mathematical Software that Solves Systems of Non Linear Equations", ACM Transactions on Mathematical Software, Volume 8, No. 1 March 1982, pp 5-20.

[HEID 81] Heidelberger P. and Trivedi K., "Queueing Network Models for Parallel Processors with Asynchronous Tasks", IBM Research Report, RC 9102 (#39808), 1981.

[JACK 63] Jackson J., "Jobshop like Queueing Systems", Management Science 10, 1963, pp 131-142.

[KLEI 76] Kleinrock L., "Queueing Systems, Volume 2: Computer Applications", John Wiley and Sons, 1976.

[LAVE 79] Lavenberg S., and Reiser, M. "Stationary State Probabilities of Arrival Instants for Closed Queueing Network with Multiple Types of Customers", IBM research report RC 7592, IBM Thomas J. Watson Research Center., Yorktown Heights, N.Y. April 1979, and Journal of Applied Probability, December 1980.

[LAVE 83] Lavenberg S., editor, "Computer Performance Modeling Handbook", Academic Press 1983.

[POPE 81] Popek G., Walker B., Chow J., Edwards D., Kline C., Rudisin G., and Thiel G. "LOCUS: A Network Transparent High Reliability Distributed System", Proceedings of the Eighth Symposium on Operating Systems Principles, Pacific Grove, California, December 1981.

[RAGH 82] Raghavendra C., Gerla M., and Avizienis A. "Reliability Optimization in the Design of Distributed Systems", The Third International Conference on Distributed Computing Systems, Fort Lauderdale, Florida, October 1982.

[RASK 78] Raskin L., "A Performance Evaluation of Multiple Processor Systems", Ph.D Thesis, Carnegie Mellon University, CMU-CS-78-141, August 1978.

[REIS 79] Reiser M., "A Queuing Network Analysis of Computer Communication Networks with Window Flow Control", IEEE Transactions on Communications, COM-27.8 (August 1979), pp 1199-1209.

[REIS 80] Reiser M., and Lavenberg S., "Mean Value Analysis of Closed Multichain Queueing Networks", JACM 27.2, April 1980, pp 313-322.

[SCHW 79] Schweitzer P., "Approximate Analysis of Multiaccess Closed Networks of Queues", International Conference of Stochastic Control and Optimization, Amsterdam, 1979.

[SEVC 79] Sevcik K., and Mitrani I., "The Distribution of Queueing Network States at Input and Output Instants", Proceedings of the 4th International Symposium on Modeling and Performance Analysis of Computer Systems, North Holland, Amsterdam, 1979.

[SURI 83] Suri R., "Robustness of Queueing Network Formulas", JACM 30.3 July 1983, pp 564-594.

[WALK 83a] Walker B., "Issues of Network Transparency and File Replication in Distributed Systems: LOCUS", Ph.D. Dissertation, Computer Science Department, University of California, Los Angeles. 1983.

[WALK 83b] Walker B., Popek G., English R., Kline C., and Thiel G., "The LOCUS Distributed Operating System", Proceedings of the The Ninth Symposium on Operating Systems Principles, Bretton Woods, NH, October 10-13, 1983.

[ZAHO 81] Zahorjan J., and Wong E., "The Solution of Separable Queueing Network Models Using Mean Value Analysis", CACM 1981.