# LOAD BALANCING IN DISTRIBUTED SYSTEMS WITH MULTIPLE CLASSES AND SITE CONSTRAINTS

Edmundo Silva
Mario Gerla

1984
Report No. CSD-840033

# LOAD BALANCING IN DISTRIBUTED SYSTEMS WITH MULTIPLE CLASSES AND SITE CONSTRAINTS

Edmundo de Souza e Silva

Mario Gerla

UCLA Computer Science Department

Los Angeles, CA 90024

April 1984

This research was supported in part by DARPA contract MDA 903-82- C-0064 and in part by UC-MICRO contract 442515-19900.

# LOAD BALANCING IN DISTRIBUTED SYSTEMS WITH MULTIPLE CLASSES AND SITE CONSTRAINTS

## Abstract

In this paper we consider a distributed computer system formed by heterogeneous computer sites connected by a communication network and supporting multiple job classes. Jobs arriving at a site may be processed locally or sent to a remote site for execution. Restrictions may apply in the remote dispatcher, so that a class of jobs may be allowed to run only in a subset of all sites in the network. In our study, we assume that the communication network as well as each of the computer sites can be modeled as product form queueing networks. Our goal is to find the optimum assignment of jobs to sites so that a weighted sum of response times over classes is minimized. We propose an efficient algorithm for finding the optimum load balancing strategy. The algorithm is based on a downhill procedure to search for the global minimum. We show that the steepest descent direction (a key step in the downhill procedure) can be easily calculated using the Mean Value Analysis Algorithm. A numerical example is presented.

## 1. Introduction.

The past few years have witnessed an increasing number of distributed computer system implementations based on local area networks. In these systems a number of resources (CPU's, file servers, disks, etc.) are shared among jobs originating at different computer sites. An example of successful implementation is the LOCUS operating system [POPE81]. Under the LOCUS system, processes generated by users at one site in the network are allowed to run on other sites.

In general, in a distributed system environment it is desirable to equalize the usage of resources (balance the load) in order to reduce the response time of jobs and improve the utilization of the resources.

The load balancing problem in a distributed resource system is not new and can take different forms for different problems. For example, in a long haul packet computer network (where the resources are the channels), load balancing becomes the routing problem. The goal there is to find optimum paths on which to distribute the packets, so that some well defined performance criterion (overall delay, for instance) is optimized.

In a distributed computer system, the load balancing problem may be formulated as the problem of distributing the execution of processes throughout the network in order to minimize overall user response time. This load balancing problem is in general more complex than the routing problem for the following reasons:

(a) **Optimal selection** of the execution site and **optimal routing** from the originating site to the execution site must be simultaneously accomplished (although the routing problem may become trivial if a bus or ring network is used).

(b) **Multiple job classes** must be considered (e.g., interactive, batch, etc.) with drastically different resource usage characteristics. In particular, some classes behave as "closed" classes, in that their population remains constant during the life of the system (e.g., the number of batch jobs in a multiprogrammed system). Other classes are best modeled as "open", in that the number of users in the class fluctuates statistically due to random inputs (e.g., database inquires originating from a very large terminal population).

(c) Some classes may be restricted to run on a subset of sites.

Most load balancing problems can be formulated as non-linear, multicommodity flow problems. Downhill search techniques can be efficiently used for their solution. If the system includes only open classes and does not pose site restrictions, then any of the methods available for routing optimization in computer networks can be successfully used. In particular, the Flow Deviation method, a downhill search method specially designed for open queueing networks may be employed [FRAT73], [KLEI76]. If the distributed system has a special structure (namely, there is only one class of customers and the local network can be modeled by a single queue) the optimal equilibrium point may be obtained directly by solving (numerically) a set of non-linear equations, rather than using an iterative procedure such as Flow Deviation. An elegant "direct" solution method for this special structure was presented in [TANT84]. Unfortunately, the direct solution does not easily extend to multiple classes, site constraints, and general interconnecting networks.

If the distributed system under consideration includes only closed chains (i.e., closed classes), then a recent extension of the Flow Deviation method due to Kobayashi and Gerla [KOBA83] can be efficiently used. One must bear in mind, however, that the routing problem with multiple closed chains typically leads to local minima. Thus, a further search for the minimum of the local minima is required.

In this paper we consider the more general situation of multiple mixed classes of customers. The open classes may correspond to interactive jobs submitted at terminals or work stations. These jobs may be permitted to execute remotely. At issue is the optimal selection of the remote site, and the optimal path to it. Restrictions may apply in the remote dispatcher. The closed classes may correspond to batch jobs running locally on a computer site or to interactive users running local application programs. The computer site is itself modeled as a network of queues (central server model). Our goal is to minimize a weighted sum of delays (over open and closed chains). Note that only the open chains need to be rerouted for load balancing. Routing is fixed within the closed chains. It can be shown that this guarantees the existence of only one local minimum, which is also the global minimum.

The main contribution of this paper is to provide an efficient algorithm for the optimum solution of a very general class of load balancing problems. The algorithm can be viewed as an extension of the approach in [KOBA83].

In the following, in section 2 we describe in detail the problem to be solved. In section 3 we present the model. Section 4 contains the development of the solution and section 5 gives the proposed algorithm. In section 6 we present some examples. Section 7 concludes the paper.

## 2. Problem Description.

We consider a distributed system as shown in figure 1. In this system each site is composed of a number of resources (CPU's, disks, etc), which are used by processes that run in that site. Sites may not be identical, i.e., they may have different resource configurations and resources with different capacities.

Each site is connected to a large number of terminals. These terminals generate jobs that may have different processing requirements. For instance, some jobs may request more CPU time than others. Furthermore, a class of jobs may be restricted to run only in a subset of the sites in the network. This restriction may arise from security considerations, or from the fact that a site may not possess all the resources required by the class.

We assume that a job may run until completion in its "local" site (which is defined to be the site connected to the terminal that generated the job), or may be transferred via the communication network to a "foreign" site. Once transferred to a foreign site, the job runs until completion and no further transfers are allowed.

This type of configuration may arise in a database application such as an airline reservation system where the requests of hundreds of terminals connected to different sites may also be processed at foreign sites. In addition, some or all of these sites may also be processing local application programs. We assume that these latter programs cannot execute in a foreign site. We refer to this type of programs as the "background load" of a site.

Our goal is to balance the load in the computer network, so that the overall delay is minimized. As a byproduct, the model will permit us to investigate several performance issues. For instance, we will be able to study the influence of the background load at a site on the assignment of database requests to sites, the effect of the speed of the communication network, etc.

## 3. The Model.

We model a distributed system as a collection of "central server models", one for each site, plus a queueing model for the communication network. This representation was first proposed by Goldberg et al [GOLD83] to model the LOCUS distributed system.

Local application programs, or background load, are modeled by a closed chain for each site, as indicated in figure 2. (In this figure we assume that these local jobs are interactive.) We assume that the remaining jobs are generated by a large number of terminals, and so can be modeled as open chains with Poisson arrivals and total throughput $\Lambda^{o}$. Upon arrival at site $i$ a request $r_i$ can be either executed locally, or immediately transferred to one of the other sites in the system that belong to the set of sites where $r_i$ can be executed.

We assume that the service demands at each queue in the network are exponentially distributed. Each queue is either processor sharing (PS), or first-come-first-serve (FCFS) with fixed capacity, or "infinite server" (IS) (at FCFS queues it is assumed that different classes of jobs have the same service demands). Furthermore, the decision of transferring a job to another site is independent of the state of the network. In summary we have a product form mixed queueing network model.

We define the following notation, where superscripts $c$ and $o$ represent closed and open chain respectively (when the superscript is omitted in the table below, the parameter may refer to either an open or closed chain).

| | | |
|---|---|---|
| $K^c$ | = | total number of closed chains representing background jobs at computer sites. |
| $K^o$ | = | total number of open chains. |
| $J$ | = | total number of service centers in the network (including the communication network). |
| $N_k$ | = | population of the closed chain at site $k$. |
| $\vec{N}$ | = | population vector = $(N_1,\ldots\ldots,N_{K^c})$. |
| $\lambda^c_k(\vec{N})$ | = | mean throughput of the closed chain at site $k$ for population $\vec{N}$. |
| $\lambda^o_v$ | = | mean throughput of open chain $v$. |
| $\theta_{kj}$ | = | visit ratio of a chain $k$ to service center $j$. |
| $x_{kj}$ | = | mean service request of a chain $k$ job at center $j$. |
| $\mu_{kj}$ | = | $1/x_{kj}$. |
| $a_{kj}$ | = | relative utilization = $\theta_{kj}x_{kj}$. |
| $\rho^o_j$ | = | total utilization of open chain jobs at center $j$. |
| $\lambda^o_{vj}$ | = | throughput (or flow) of open chain $v$ at center $j$ (= $\lambda^o_v\theta^o_{vj}$). |
| $\Lambda^o$ | = | total open throughput = $\sum\limits_v \lambda^o_v$. |
| $L_{kj}(\vec{N})$ | = | mean number of chain $k$ jobs at center $j$ with population $(\vec{N})$. |
| $L^c_j(\vec{N})$ | = | total mean number of closed chain jobs at center $j$. |
| $L^o_j(\vec{N})$ | = | total mean number of open chain jobs at center $j$. |
| $\vec{e}_k$ | = | unit multidimensional vector in the direction $k$. |

We define the overall delay $D(\vec{N})$ as the following weighted sum of all chain delays:

$$D(\vec{N}) = \frac{w^o \sum\limits_{v=1}^{K^o} \lambda^o_v D^o_v(\vec{N}) + w^c \sum\limits_{k=1}^{K^c} \lambda^{Nc}_k(\vec{N}) D^c_k(\vec{N})}{w^o \sum\limits_{v=1}^{K^o} \lambda^o_v + w^c \sum\limits_{k=1}^{K^c} \lambda^{Nc}_k(\vec{N})} \tag{3.1}$$

where: (1) $D^o_v(\vec{N})$ and $D^c_k(\vec{N})$ represent the response time of open chain $v$ and closed chain $k$, respectively; (2) $\lambda^{Nc}_k(\vec{N})$ represents the "nominal" throughput of a closed chain $k$ obtained when all open chain throughputs are set to zero; (3) $w^o$ and $w^c$ are arbitrary constant weighting factors for open and closed chains, respectively. Note that if the nominal throughput for a closed chain is replaced by the actual throughput, and the weights are set to 1, the expression above gives the total average delay. However, using total average delay as the objective function leads to unfairness to closed chains, since an increase in open chain utilization at a site causes an increase in local chain delay, but also a decrease in local chain throughput (the product remains constant by Little's result). This implies that at heavy load the impact of closed chain delays on the objective function becomes negligible with respect to the open chain delays. For this reason, we chose to assign a fixed coefficient, the nominal throughput, to each closed chain. In addition, we introduced the weighting factors $\{w\}$ to reflect the relative importance of open and closed chains in the model. Without loss of generality we assume $w^o = w^c = 1$ throughout the rest of the paper.

From Little's result:

$$D_v^o(\vec{N}) = \frac{\sum_{j=1}^{J} L_{vj}(\vec{N})}{\lambda_v^o} \qquad \text{and} \qquad D_k^c(\vec{N}) = \frac{N_k - \lambda_k^c(\vec{N}) x_{k_z}^c}{\lambda_k^c(\vec{N})} \tag{3.2}$$

where $x_{k_z}^c$ is the average think time of an interactive closed chain $k$ job at the terminal, and $\lambda_k^c(\vec{N}) x_{k_z}^c$ gives the average number of closed chain k jobs at the terminals (for batch jobs, $x_{k_z}^c = 0$).

Substituting (3.2) into (3.1),

$$D(\vec{N}) = \frac{\sum_{j=1}^{J} L_j^o(\vec{N}) + \sum_{k=1}^{K^c} \lambda(\vec{N})_k^{Nc}(N_k/\lambda_k^c - x_{k_z}^c)}{\Lambda^o + \sum_{k=1}^{K^c} \lambda_k^{Nc}(\vec{N})} \tag{3.3}$$

Our goal is to minimize the non-linear function $D(\vec{N})$ by optimally distributing the open chain traffic among the various sites. This is equivalent to optimizing $D(\vec{N})$ with respect to the open chain flows $\{\lambda_{vj}^o\}, \forall v, j$.

## 4. The Solution Approach.

We define our load balancing problem as follows:

**Given:**

- The service demands of jobs from all chains at all service centers in the network.
- The visit ratios of the closed chain jobs (background jobs) of a site at each service center at that site, as well as the visit ratios of open chain jobs at centers of a site.
- The number of background jobs at each site.

- The throughput rate of each open chain $v$.
- The set of sites where open chain $v$ jobs can execute.
- The "local" site of each open chain job.

**Minimize**: The overall delay $D(\vec{N})$.

**With respect to:** The open chain flows at each service center $\{\lambda_{vj}^{o}\}$.

The solution method we use is a downhill technique based on the "flow deviation" method [FRAT73], [KLEI76]. The development we present below parallels the approach used by Kobayashi and Gerla [KOBA83] to find the optimum routing in a closed queueing network. In this paper we restrict the development to a single closed chain per site as indicated in section 3. However, the approach can be directly extended to multiple closed chains at each site.

To compute the steepest descent direction for the downhill technique we need to compute the (partial) derivatives of the overall delay function with respect to each open chain flow at each service center.

First, we should observe that a mixed product form queueing network with PS, FCFS fixed rate service centers and IS service centers can be reduced to an equivalent closed queueing network where the service requests $x_{kj}^{'}$ of closed chain $k$ jobs at center $j$ (PS or FCFS) is given by the following expression [LAVE83]:

$$x_{kj}^{'} = \frac{x_{kj}}{1 - \rho_j^o} \qquad (4.1)$$

Second, we note that in our model, closed chain jobs (background load) at a site do not interfere with closed chain jobs from another site. Therefore, we can decompose our model into $K^c$ independent mixed network models each with one closed chain, plus the queueing model of the communication network.

Finally, we recall that in a mixed network the open chain population at center $i$, $L_i^o$, can be expressed in terms of the closed chain by [LAVE83]:

$$L_i^o(N_k) = \begin{cases} \dfrac{\rho_i^o}{1-\rho_i^o}[1 + L_i^c(N_k)] & i \in \text{site } k \text{ or communication network, } i \neq IS \text{ centers.} \\ \rho_i^o & i \in \text{site } k \text{ or communication network, } i = IS \text{ centers.} \end{cases} \tag{4.2}$$

where the vector notation was dropped due to the reduction of the problem to $K^c + 1$ independent single chain mixed networks.

Therefore, equation (3.3) can be rewritten as:

$$D(\vec{N}) = \frac{\displaystyle\sum_{\substack{i=1\\ i\neq IS}}^{J} \frac{\rho_i^o}{1-\rho_i^o}[1 + L_i^c(N_k)\,] + \sum_{\substack{i=1\\ i=IS}}^{J} \rho_i^o + \sum_{k=1}^{K^c} \lambda_k^{Nc}(N_k)\,[N_k/\lambda_k^c(N_k) - x_{ko}]}{\Lambda^o + \displaystyle\sum_{k=1}^{K^c} \lambda_k^{Nc}(N_k)} \tag{4.3}$$

The independent variables in our optimization problem are the $\{\lambda_{vj}^o\}$. Recalling that $\rho_j^o = \sum_v \lambda_{vj}^o x_{vj}^o$, we note that we can also use the $\{\rho_j^o\}$ as independent variables. In the following, for convenience, we take the derivative of $D(\vec{N})$ with respect to the total utilization of open chain jobs at center $j$ ($\rho_j^o$). In this computation we exploit the fact that: (1) the performance measures of site (or communication network) $u$ are not altered when we vary the open chain throughput at a center $j$ in site (or communication network) $k$, $k \neq u$; and, (2) the total open throughput remains constant as do the nominal throughputs of the closed chains. For convenience of notation, we label the centers belonging to the same site (or communication network) as center $j$ as $l,\ldots,L$.

$$\frac{\partial D(\vec{N})}{\partial \rho_j^o} = \qquad (j \in \text{site } k \text{ or communication network})$$

$$\begin{cases} \dfrac{\dfrac{1 + L_j^c(N_k)}{(1 - \rho_j^o)^2} + \displaystyle\sum_{\substack{l = l \\ l \neq IS}}^{L} \dfrac{\rho_l^o}{1 - \rho_l^o} \dfrac{\partial L_l^c(N_k)}{\partial \rho_j^o} - \delta(j) \dfrac{\lambda_k^{Nc}(N_k) N_k}{(\lambda_k^c(N_k))^2} \dfrac{\partial \lambda_k^c(N_k)}{\partial \rho_j^o}}{\Lambda^o + \Lambda^{Nc}(\vec{N})} & j \neq \text{IS} \\ \dfrac{1}{\Lambda^o + \Lambda^{Nc}(\vec{N})} & j = \text{IS} \end{cases} \tag{4.4}$$

where

$$\Lambda^{Nc}(\vec{N}) = \sum_{k = 1}^{K} \lambda_k^{Nc}(N_k)$$

and

$$\delta(j) = \begin{cases} 1 & j \notin \text{communication network} \\ 0 & \text{otherwise} \end{cases}$$

It remains to find the derivatives of the closed chain throughputs and queue lengths in relation to the utilization of open chain jobs at center $j$.

To this end, we first introduce the normalization constant for the corresponding closed network with service requests given by (4.1). Therefore, for site $k$ [BASK75]:

$$G_k^c(N_k) = \sum_{|\vec{n}_k| = N_k} \prod_{\substack{l = l \\ l \neq IS}}^{L} \left( \frac{a_{kl}^c}{1 - \rho_l^o} \right)^{n_l} \times I_k(\vec{n}_k) \tag{4.5}$$

where

$$I_k(\vec{n}_k) = \prod_{\substack{l = l \\ l = IS}}^{L} \frac{\left(a_{kl}^c\right)^{n_l}}{n_l!} \qquad \vec{n}_k = (n_l, \ldots, n_L), \quad |\vec{n}_k| = n_l + \cdots + n_L$$

and the equilibrium probability of the state $\vec{n}_k$ is:

$$P_k^c(\vec{n}_k) = \frac{1}{G_k^c(N_k)} \prod_{\substack{i=1 \\ i \neq IS}}^{L} \left( \frac{a_{ki}^c}{1-\rho_i^o} \right)^{n_i} \times I_k(\vec{n}_k) \tag{4.6}$$

Now, taking the derivative of (4.5) with respect to $\rho_j^o$, gives:

$$\frac{\partial G_k^c(N_k)}{\partial \rho_j^o} = \frac{1}{1-\rho_j^o} \sum_{|\vec{n}_k| = N_k} n_j \prod_{\substack{i=1 \\ i \neq IS}}^{L} \left( \frac{a_{ki}^c}{1-\rho_i^o} \right)^{n_i} \times I_k(\vec{n}_k) \qquad j \neq IS \tag{4.7}$$

$$= \frac{G_k^c(N_k)}{1-\rho_j^o} \sum_{|\vec{n}_k| = N_k} n_j P_k^c(\vec{n}_k) \tag{4.8}$$

The summation on the right hand side of (4.8) is by definition the average queue length of closed chain jobs at service center $j$, therefore:

$$\frac{\partial G_k^c(N_k)}{\partial \rho_j^o} = \frac{G_k^c(N_k)}{1-\rho_j^o} L_j^c(N_k) \tag{4.9}$$

Equation (4.9) is similar to one obtained by Kobayashi and Gerla [KOBA83].

In order to proceed with the development we need to take the second derivative of (4.5) with respect to $\rho_j^o$,

$$\frac{\partial^2 G_k^c(N_k)}{\partial (\rho_j^o)^2} = \frac{G_k^c(N_k)}{(1-\rho_j^o)^2} \sum_{|\vec{n}_k| = N_k} n_j P_k^c(\vec{n}_k) + \frac{G_k^c(N_k)}{(1-\rho_j^o)^2} \sum_{|\vec{n}_k| = N_k} n_j^2 P_k^c(\vec{n}_k)$$

$$= \frac{G_k^c(N_k)}{(1-\rho_j^o)^2} \left( L_j^c(N_k) + S_j^c(N_k) \right) \tag{4.10}$$

where $S_j^c(N_k)$ is the second moment of queue lengths of the closed chain jobs at center $j$ of site $k$.

Similarly, taking the derivative of (4.7) with respect to $\rho_t^o$, $t \neq j$, $t \neq IS$:

$$\frac{\partial^2 G_k^c(N_k)}{\partial\rho_j^o \partial\rho_t^o} = \frac{G_k^c(N_k)}{(1-\rho_j^o)(1-\rho_t^o)} \sum_{|\vec{n}_k| = N_k} n_k n_t P_k^c(\vec{n}_k)$$

$$= \frac{G_k^c(N_k)}{(1-\rho_j^o)(1-\rho_t^o)} J_{jt}^c(N_k) \tag{4.11}$$

where $J_{jt}^c(N_k)$ is the joint moment of queue lengths of the closed chain jobs at center $j$ and $t$.

The second derivative of $G_k^c(N_k)$ with respect to $\rho_j^o$ and the derivative of $\partial G_k^c(N_k)/\partial\rho_j^o$ with respect to $\rho_t^o$ can also be obtained by taking the derivative of equation (4.9) with respect to $\rho_j^o$ and $\rho_t^o$, respectively. We obtain:

$$\frac{\partial^2 G_k^c(N_k)}{\partial(\rho_j^o)^2} = \frac{1}{(1-\rho_j^o)^2} G_k^c(N_k) L_j^c(N_k) + \frac{1}{1-\rho_j^o}\frac{\partial G_k^c(N_k)}{\partial\rho_j^o} L_j^c(N_k) + \frac{1}{1-\rho_j^o} G_k^c(N_k) \frac{\partial}{\partial\rho_j^o} L_j^c(N_k)$$

$$= \frac{G_k^c(N_k)}{(1-\rho_j^o)^2}\left(L_j^c(N_k) + \left(L_j^c(N_k)\right)^2 + (1-\rho_j^o)\frac{\partial}{\partial\rho_j^o} L_j^c(N_k)\right) \tag{4.12}$$

$$\frac{\partial^2 G_k^c(N_k)}{\partial\rho_j^o \partial\rho_t^o} = \frac{1}{1-\rho_j^o}\frac{\partial G_k^c(N_k)}{\partial\rho_t^o} L_j^c(N_k) + \frac{1}{1-\rho_j^o} G_k^c(N_k)\frac{\partial}{\partial\rho_t^o} L_j^c(N_k)$$

$$= \frac{G_k^c(N_k)}{(1-\rho_j^o)(1-\rho_t^o)}\left(L_t^c(N_k) L_j^c(N_k) + (1-\rho_t^o)\frac{\partial}{\partial\rho_t^o} L_j^c(N_k)\right) \tag{4.13}$$

Now, equating (4.10) with (4.12) we obtain:

$$L_j^c(N_k) + S_j^c(N_k) = L_j^c(N_k) + \left(L_j^c(N_k)\right)^2 + (1-\rho_j^o)\frac{\partial}{\partial\rho_j^o} L_j^c(N_k)$$

$$V_j^c(N_k) = (1-\rho_j^o)\frac{\partial}{\partial\rho_j^o} L_j^c(N_k) \tag{4.14}$$

where $V_j^c(N_k)$ is the variance of the queue length of closed queue jobs at center $j$ of site $k$.

Similarly, equating (4.11) with (4.13) we obtain:

$$J_{ji}^{c}(N_k) = L_i^c(N_k)L_j^c(N_k) + (1 - \rho_i^o)\frac{\partial}{\partial \rho_i^o}L_j^c(N_k)$$

$$V_{ji}^{c}(N_k) = (1 - \rho_i^o)\frac{\partial}{\partial \rho_i^o}L_j^c(N_k) \qquad (4.15)$$

where $V_{ji}^{c}(N_k)$ is defined as the covariance of queue lengths of closed queue jobs at centers $j$ and $i$ of site $k$.

We note that a similar development can be used to obtain higher moments of queue lengths as a function of the derivative of lower moments with respect to an input parameter of the system. Expressions similar to (4.14) can also be obtained for a network with multiple closed chains.

In order to find the derivatives of the closed chain throughput of site $k$ ($\lambda_k^c(N_k)$) with respect to the open chain utilization at center $j$ in $k$ ($\rho_j^o$), we express $\lambda_k^c(N_k)$ in terms of the normalization constant, [LAVE83]:

$$\lambda_k^c(N_k) = \frac{G_k^c(N_k - 1)}{G_k^c(N_k)} \qquad (4.16)$$

Now, taking the derivative of (4.16) with respect to $\rho_j^o$ and using (4.9),

$$\frac{\partial \lambda_k^c(N_k)}{\partial \rho_j^o} = \frac{\partial G_k^c(N_k - 1) / \partial \rho_j^o}{G_k^c(N_k)} - \frac{G_k^c(N_k - 1)}{\left(G_k^c(N_k)\right)^2}\frac{\partial G_k^c(N_k)}{\partial \rho_j^o}$$

$$= \frac{G_k^c(N_k - 1)}{G_k^c(N_k)} \times \frac{L_j^c(N_k - 1)}{1 - \rho_j^o} - \frac{G_k^c(N_k - 1)}{G_k^c(N_k)} \times \frac{L_j^c(N_k)}{1 - \rho_j^o}$$

$$= \frac{\lambda_k^c(N_k)}{1 - \rho_j^o}\left(L_j^c(N_k - 1) - L_j^c(N_k)\right) \qquad (4.17)$$

where service center $j$ is assumed to be in the same site of closed chain $k$.

Finally, substituting (4.14), (4.15) and (4.17) into (4.4) and noting that $\partial D(\vec{N}) / \partial \lambda_{vj}^{o} = (1/\mu_{vj}^{o}) \partial D(\vec{N}) / \partial \rho_j^{o}$, we obtain:

$$\frac{\partial D(\vec{N})}{\partial \lambda_{vj}^{o}} =$$

$$\begin{cases} \dfrac{\dfrac{1 + L_j^c(N_k)}{1 - \rho_j^o} + \dfrac{\rho_j^o V_j(N_k)}{1 - \rho_j^o} + \sum\limits_{\substack{l=1 \\ l \neq j,\, l \neq IS}}^{L} \dfrac{\rho_l^o V_{lj}(N_k)}{1 - \rho_l^o} + \delta(j) \dfrac{\lambda_k^{Nc}(N_k)}{\lambda_k^c(N_k)} N_k [L_j^c(N_k) - L_j^c(N_k - 1)]}{\mu_{vj}^o (1 - \rho_j^o)(\Lambda^o + \Lambda^{Nc}(\vec{N}))} & j \neq IS \\ \dfrac{1}{\mu_{vj}^o (\Lambda^o + \Lambda^{Nc}(\vec{N}))} & j = IS \end{cases} \quad (4.18)$$

Equation (4.18) is given in terms of mean values of throughputs and queue lengths, variance and covariance of queue lengths. In the appendix we show that this equation can be easily obtained using mean value analysis (MVA) recursion.

## 5. The Algorithm.

The key to the flow deviation method is to associate a length or weight for an open chain job to each queue, given by:

$$l_{vj} \triangleq \frac{\partial D(\vec{N})}{\partial \lambda_{vj}^{o}} \qquad j = 1, \ldots, J \qquad (4.19)$$

However, due to the particular characteristics of our problem, we can further simplify the solution by noting that, once a job is assigned to a site, its behavior is preestablished by the routing probabilities given as input parameters. In other words, the "route" of jobs within a site is fixed. This observation allow us to define a "site" weight, as:

$$lsit_{kv} \triangleq \sum_{j \in k} \theta_{vj}^{o} l_{vj} \qquad (4.20)$$

where the index $k$ represents the site and index $v$, a particular open chain. In the same way, the weight of the communication network ($lnet_v$) can be defined.

The weight given by (4.20) represents the linear rate at which $D(\vec{N})$ increases with an infinitesimal increase of the flow of open chain $v$ at site $k$. We also define $lsit_{kv} \triangleq \infty$ if jobs of chain $v$ are not allowed to visit site $k$.

We outline a flow deviation algorithm for load balancing (FDLB) in a distributed computer network of the type described in section 3. The algorithm is similar to the ones described in [KLEI76] and [KOBA83].

We define the assignment matrix $\vec{A}$ where element $A_{kv}$ gives the percentage of jobs of type $v$ assigned to site $k$. Similarly we define the network vector $\vec{F}$ where element $F_v$ gives the percentage of jobs of type $v$ assigned to a foreign site, i.e., $F_v = \sum_{k \neq \text{home site of } v} A_{kv}$

We assume that we have an initial feasible assignment, i.e., initial values for $\vec{A}$ so that $\rho_j^o < 1$ for all service centers in the network which are not IS centers.

FDLB algorithm (*):

Step 1: let $n = 0$ and let $\vec{A}^{(0)}$ be an initial feasible assignment.

Step 2: Compute weights $l_{vj}$ using MVA and equations (A.3) and (A.4). Compute the weights of each site ($lsit_{kv}$) and the weight of the network ($lnet_v$) using equation (4.20).

Step 3: For each class of open chain jobs, find the cheapest way to execute the job, i.e., find the matrix $\vec{A}^*$ so that element $A^*_{kv} = 1$ if $lsit_{kv} + net_{kv} < lsit_{tv} + net_{tv}$ for all $t \neq k$, where

$$net_{kv} \triangleq \begin{cases} 0 & \text{if } k \text{ is the local site for a job of class } v \\ lnet_v & \text{otherwise} \end{cases}$$

Find $\vec{F}$ as defined above.

Step 4: Compute the incremental delay $b^{(n)}$ and $b^*$ for the assignment $\vec{A}^{(n)}$ and cheapest assignment $\vec{A}^*$,

(*) The description of the algorithm follows similar descriptions given in [KLEI76] and [KOBA83]. However, in the implementation, we deviate the flow for one chain at a time, since it was observed that this equivalent approach tends to reduce the overall number of iterations.

respectively:

$$b^{(n)} = \sum_v \left[ \sum_k lsit_{kv} \times A_{kv}^{(n)} + lnet_v \times F_v^{(n)} \right]$$

$$b^* = \sum_v \left[ \sum_k lsit_{kv} \times A_{kv}^* + lnet_v \times F_v^* \right]$$

Step 5: Stopping rule:

if $|b^{(n)} - b^*| < \epsilon$, where $\epsilon > 0$ is a properly chosen tolerance, stop. Otherwise go to Step 6.

Step 6: Find the value $\alpha$ in the range $0 \leq \alpha \leq 1$ such that the assignment $\vec{A}' = (1 - \alpha)\vec{A}^{(n)} + \alpha\vec{A}^*$ minimizes $D(\vec{N})$.

Let $\vec{A}^{(n+1)} = \vec{A}'$.

Step 7: Let $n = n + 1$ and go to Step 2.

This algorithm finds the global minimum for $D(\vec{N})$ due to the convexity of $D(\vec{N})$ with respect to the open chain flows and the convexity of the space of the feasible flows. The convexity of $D(\vec{N})$ can be deduced from the fact that the function $D(\vec{N})$ in (3.3) is a sum of convex functions over the open chain flows. An intuitive (but not rigorous) explanation for that can be given as follows: we observe that equation (3.3) can be decomposed into a weighted sum of the number of open chain jobs in a site plus the response time of the closed chain jobs in that site. We can verify that both the number of open chain jobs in a site and the response time of closed chain jobs in that site are increasing without bound (and with a non-decreasing rate) as a function of the open chain flow at that site. Thus, the delay of each site is a convex increasing function over the open chain flows, and the weighted sum of these delays is also convex.

Convergence is guaranteed by the fact that each iteration provides an improvement in the objective function. It is worthwhile to note that if the background load is reduced to zero (no closed chain jobs) this algorithm is identical to the flow deviation algorithm reported in [KLEI76], adapted for our load balance problem.

The amount of computation required by each iteration of the FDLB algorithm is only $O(\#sites \times \#open\ chains)$ for Step 3, since the computation of the "cheapest path" is trivial, and $O(\sum_{site\ i}(\#\ centers\ of\ site\ i)\times(\#\ background\ jobs\ of\ site\ i) + com.\ net.)$ for the MVA algorithm, since the model consists of $K^c$ independent single closed chains plus the model of the communication network. Note that in Step 6 the MVA algorithm is repeated several times, and the MVA computation becomes the dominant term.

## 6. Examples.

In this section we apply the FDLB algorithm to a distributed computer system consisting of 3 sites linked by a slotted ring, as illustrated in figure 3. For the slotted ring, we used the closed chain model proposed by Bux [BUX81].

There are 2 classes of open chain jobs $o1$ and $o3$ arriving at sites 1 and 3, respectively. Class $o1$ can request service from any of the 3 sites, but class $o3$ can only request service from sites 2 and/or 3. Sites 1 and 2 have background load, modeled as closed chains $c1$ and $c2$ respectively. Table 1 shows the visit ratios and service requirements for each class of job at each service center in the network, as well as other input parameters.

As an illustration, let us consider the behavior of jobs $o1$ and $c1$ at site number 1. A job of class $o1$ spends an average of 30 msec at the CPU before issuing an I/O request. The service time of each I/O device is 50 msec. On the average, a job of class $o1$ visits the CPU 5 times and the I/O devices 4 times before completion. On the other hand, a background job of class $c1$ spends an average of 90 msec at the CPU and 50 msec at each I/O device, and it visits the CPU and I/O devices 10 times before a visit to the terminals.

Figure 4 shows the percentage of foreign jobs processed at site 2 (after balancing the load) when the number of background jobs at site 2 ($N_2$) varies from 0 to 20. When $N_2 = 0$, site 2 processes 80% of the jobs of class $o1$ and 36% of the jobs of class $o3$. When $N_2 = 20$, site 2 processes only 37% and 4% of the jobs of class $o1$ and $o3$, respectively. It is interesting to mention that when jobs of class $o3$ are allowed to be processed at site 1, and load balancing is applied, 22% of these jobs are processed at site 1 and 15% at site 2 (when $N_2 = 0$). Furthermore, all jobs of class $o1$ are sent to site 2.

Figure 5 shows the effect of load balancing on overall delay as a function of the number of background jobs at site 2. The effect is illustrated by plotting the relative difference between the delays when load balancing is not used and when it is used (*). For instance, when $N_2 = 0$ and load balancing is used the average overall delay is .95 sec in contrast with 9.2 sec when load balancing is not used (the relative difference is 870% in this case).

## 7. Conclusions.

We have presented an efficient method for balancing the load in a distributed system with heterogeneous computer sites and multiple classes of customers. Jobs of a particular class may be restricted to run only on a subset of the sites in the network. Each site, as well as the communication network, are represented as product form queueing networks.

We took into consideration the effect of the background load at each site, modeled as closed chain jobs. The effect of the communication delay incurred when a job requests service at a foreign site was also considered.

Our approach is based on a downhill search technique known as the Flow Deviation method. We have defined weights for each site in the network which allow the computation of the assignment which will reduce the overall delay $D(\vec{N})$. The parameters necessary for the computation of these weights can be easily obtained using the MVA algorithm.

(*) We define relative difference of the delays as: (Delay without load balancing - Delay with load balancing) / Delay with load balancing.

The definition of the delay function presented in equation (3.1) is suitable for studying the behavior of the network under different minimization requirements. In particular, by setting $w^o = 0$ ($w^c = 0$) we can study the effect of load balancing when the object is to minimize the delay of the open (closed) chain jobs only.

The method developed in this paper is not restricted to the function defined in (3.1), and can be used to minimize other objective functions as well. Furthermore, more complex communication network structures can be considered. In particular, the algorithm can be extended to account for jobs that request service in more than one site before completion. Another application being investigated is the routing of packets through gateways of interconnected packet switching networks.

**Appendix.**

We show that equation (4.18) can be easily solved recursively. Heffes [HEFF82] has already observed that moments of queue lengths can be obtained recursively in terms of lower moments. To demonstrate that, he used the recursive formula for the marginal probability distribution given in [REIS80]. Below, we show that the recursion for calculating the variances in equation (4.18) can be obtained directly from the MVA equations. We prove the result for single closed chain networks, but the same approach can be used for multiple chain networks.

The well known MVA equation which relates the mean queue lengths of closed chain jobs at a service center as a function of queue lengths of closed chain jobs with one less closed chain job in the network becomes, for mixed networks:

$$L_j(N) = \lambda(N)\frac{a_j}{1-\rho_j^o}[1 + L_j(N-1)] \tag{A.1}$$

where the superscript $c$ was dropped to simplify the notation, and $j$ is a PS or FCFS service center.

Taking the derivative on both sides of equation (A.1) with respect to $\rho_j^o$ gives:

$$\frac{\partial L_j(N)}{\partial \rho_j^o} = \frac{\partial \lambda(N)}{\partial \rho_j^o}\frac{a_j}{1-\rho_j^o}[1 + L_j(N-1)] + \lambda(N)\frac{a_j}{(1-\rho_j^o)^2}[1 + L_j(N-1)] + \lambda(N)\frac{a_j}{1-\rho_j^o}\frac{\partial L_j(N-1)}{\partial \rho_j^o} \tag{A.2}$$

Now, using (4.14) and (4.17) in (A.2):

$$V_j(N) = \lambda(N)\frac{a_j}{1-\rho_j^o}\Big([1 + L_j(N-1) - L_j(N)][1 + L_j(N-1)] + V_j(N-1)\Big) \tag{A.3}$$

Similarly an expression for the covariance $V_{ij}(N)$ can be obtained:

$$V_{ij}(N) = \lambda(N)\frac{a_i}{1-\rho_i^o}\Big([L_j(N-1) - L_j(N)][1 + L_i(N-1)] + V_{ij}(N-1)\Big) \qquad i \neq j \tag{A.4}$$

where $V_j(0) = V_{ij}(0) \triangleq 0$.

Clearly, equations (A.3) and (A.4) can be solved recursively in $N$.

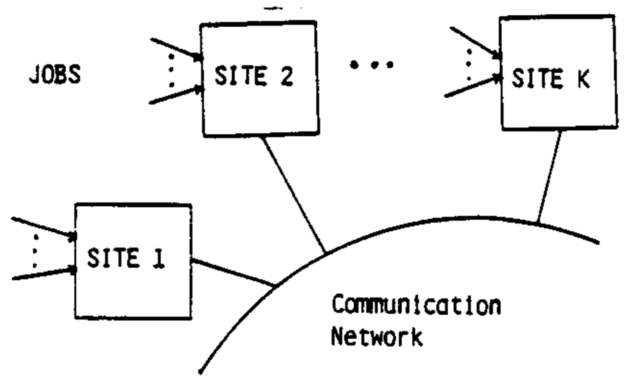

Figure 1. A distributed system.

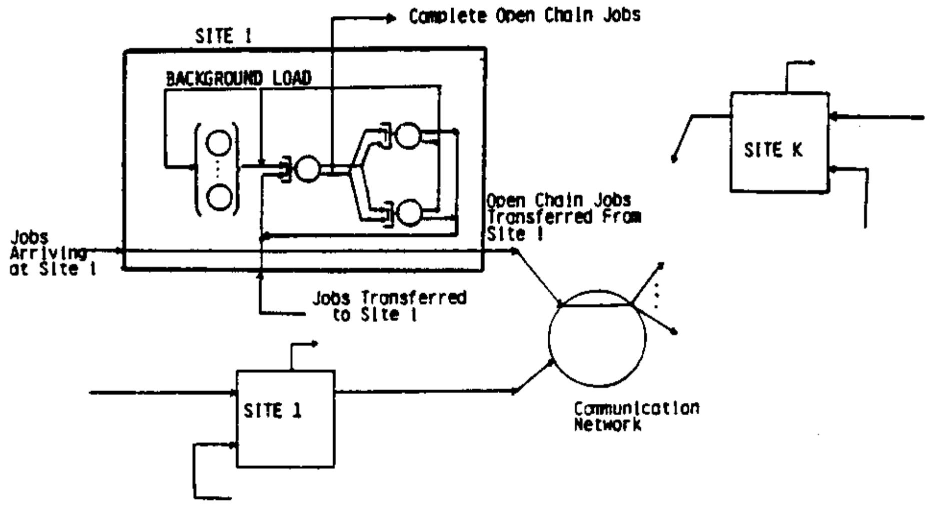

Figure 2. The model.

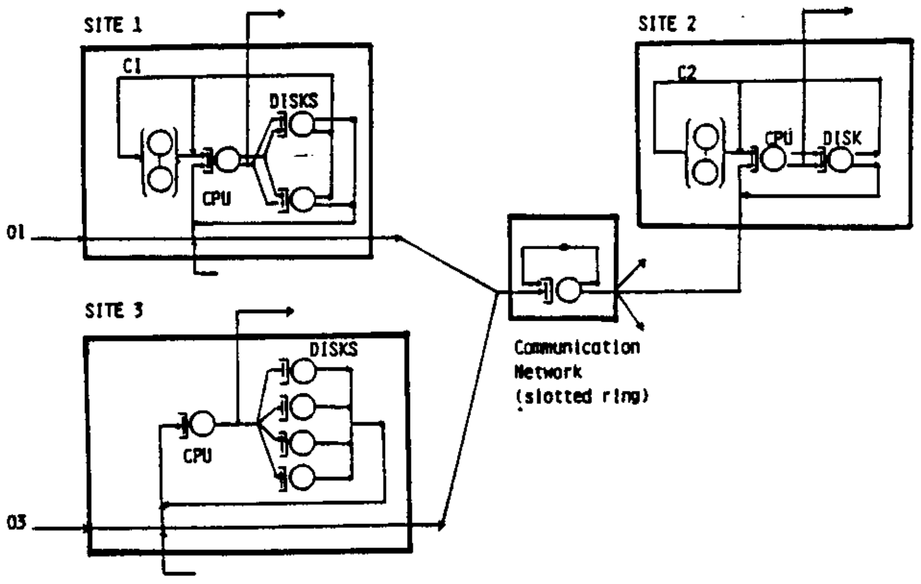

Figure 3. Example. A distributed computer system with 3 sites.

| JOB CLASS | SITE | CENTER | TYPE | VISIT RATIOS | SERVICE TIMES (msec) |
|---|---|---|---|---|---|
| *o*1<br>6 jobs/sec | 1 | cpu<br>disk(1 or 2) | PS<br>FCFS | 5<br>2 | 30<br>50 |
| | 2 | cpu<br>disk | PS<br>FCFS | 5<br>4 | 21<br>15 |
| | 3 | cpu<br>disk(1 ... 4) | PS<br>FCFS | 5<br>1 | 30<br>140 |
| | com. net. | | PS | | 10 |
| *o*3<br>3.5 jobs/sec | 1<br>(*) | cpu<br>disk(1 or 2) | PS<br>FCFS | 5<br>2 | 50<br>50 |
| | 2 | cpu<br>disk | PS<br>FCFS | 5<br>4 | 35<br>15 |
| | 3 | cpu<br>disk(1 ... 4) | PS<br>FCFS | 5<br>1 | 50<br>140 |
| | com. net. | | PS | | 10 |
| *c*1<br>5 jobs | 1 | terminal<br>cpu<br>disk(1 or 2) | IS<br>PS<br>FCFS | 1<br>10<br>5 | 4000<br>90<br>50 |
| *c*2<br>X jobs<br>(variable) | 2 | terminal<br>cpu<br>disk(1 or 2) | IS<br>PS<br>FCFS | 1<br>10<br>10 | 3000<br>40<br>15 |

Table 1. Parameters for the example.

(*) To illustrate the case where class *o*3 is not restricted to run at site 1.

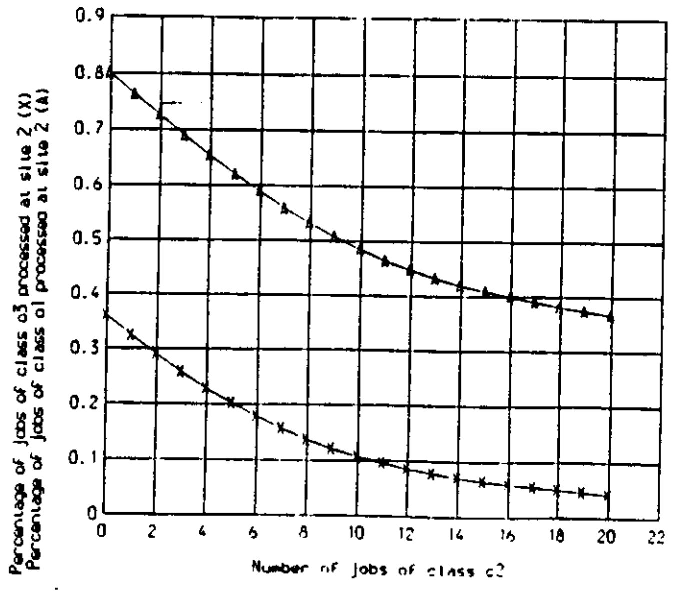

Figure 4. Percentage of foreign jobs processed at site 2.

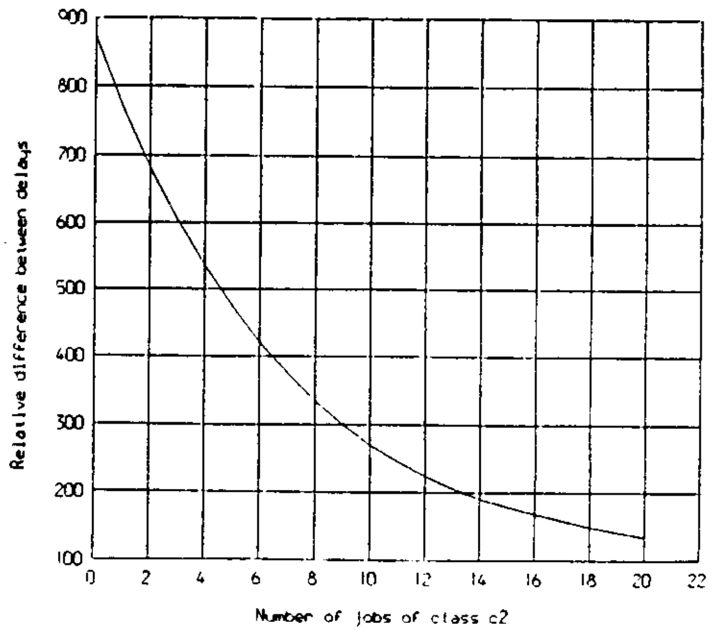

Figure 5. Relative difference (%) of the overall delays.

## References.

[BASK75] F. Baskett, K.M. Chandy, R.R. Muntz and F. Palacios, "Open, Closed and Mixed Networks of Queues with Different Classes of Customers", Journal of the ACM 22, 2, 248-260, April 1975.

[BUX81] W. Bux, "Local-Area Subnetworks: A Performance Comparison", IEEE Transactions on Communications COM 29, no. 10, 1465-1473, 1981.

[FRAT73] L. Fratta, M. Gerla and L. Kleinrock, "The Flow Deviation Method - An Approach to Store-and-Forward Communication Network Design", Networks 3, 1973.

[GOLD83] A. Goldberg, G. Popek and S.S. Lavenberg, "A Validated Distributed System Performance Model", Performance 83, A.K. Agrawala and S.K. Tripathi (Editors), 251-268, 1983.

[HEFF82] H. Heffes, "Moment Formulae for a Class of Mixed Multi-Job-Type Queueing Networks", The Bell System Technical Journal, vol. 61, no. 5, 709-745, May-June 1982.

[KLEI76] L. Kleinrock, "Queueing Systems, Volume II: Computer Applications", Wiley-Interscience, New York, 1976.

[KOBA83] H. Kobayashi and M. Gerla, "Optimal Routing in Closed Queueing Networks", ACM Transactions on Computer Systems, vol. 1, no. 4, 294-310, November 1983.

[LAVE83] S.S. Lavenberg (Editor), "Computer Performance Modeling Handbook", Academic Press, New York, 1983.

[POPE81] G. Popek, B. Walker, J. Chow, D. Edwards, C. Kline, G. Rudisin, G. Thiel, "LOCUS: A Network Transparent, High Reliability Distributed System", Proceedings of the Eighth Symposium on Operating Systems Principles, California, December 1981.

[REIS80] M. Reiser and S.S. Lavenberg, "Mean Value Analysis of Closed Multichain Queueing Networks", Journal of the ACM 27, 313-322, 1980.

[TANT84] A.N. Tantawi and D. Towsley, "Optimal Load Balancing in Distributed Computer Systems", IBM Research Report, RC 10346, January 1984.