The File Allocation Problem --
A Queueing Network Optimization Approach

M. M. Srinivasan
Department of Industrial & Operations Engineering
The University of Michigan
Ann Arbor, Michigan 48109-2117

K. Kant
Department of Computer Science
The Pennsylvania State University
University Park, Pennsylvania 16802

Technical Report 85-19

April 1986
Revised: August 1986

The File Allocation Problem -

A Queueing Network Optimization Approach

M. M. Srinivasan
Department of Industrial & Operations Engineering
The University of Michigan
Ann Arbor, Michigan 48109-2117

K.Kant
Department of Computer Science
The Pennsylvania State University
University Park, Pennsylvania 16802

April 1986
Revised: August 1986

The File Allocation Problem :

A Queueing Network Optimization Approach

M. M. Srinivasan
Department of Industrial & Operations Engineering
The University of Michigan
Ann Arbor, Michigan 48109-2117

K.Kant
Department of Computer Science
The Pennsylvania State University
University Park, Pennsylvania 16802

Abstract:

A set of customers use a connected network of computer installations, each accessing the network from a particular node. These customers share information contained in a set of data files. A typical customer's need is characterized by a request requiring a subset of these files being accessed in a Markovian sequence. The cycle time for this customer is the total time taken on the average to complete his request sequence. The objective is to locate a single copy of each of the files in such a way that a weighted sum of these response times is minimized.

The problem is modelled as a Closed Queueing Network optimization problem. Models are developed for both single and multiple chain cases. An incremental analysis approach is used to solve the single chain case. For the multiple chain case, it is shown how this model approximates to a set partitioning problem under certain conditions. Efficient heuristics are developed to solve this partitioning problem. With certain simplifying assumptions, the associated communication problem is then included in the model.

1. Introduction

The File Assignment Problem (FAP) has generally been recognized as crucial to the design of a good distributed information system. It has hence received a considerable amount of attention in the literature since the time it was first investigated by Chu [5]. The FAP entails allocating a set of F distinct files among a set of M computer installations (nodes). The allocation is to be made so as to optimize some objective function.

The FAP arises in many contexts, though the "file" need not be a file in the conventional sense. For example, consider the problem of assigning tasks to processors in multiprocessor systems. With the introduction of large parallel machines (e.g. hypercubes, trees, butterflies, etc.) and advances in programming to take advantage of such machines, the problem of task allocation (and, possibly, reallocation or migration) is becoming more important. The tasks involved cooperate in order to solve the problem. Thus one could view the subproblems to be solved as transactions that are served by the processes residing on various processors. In this sense, the tasks are like "files" accessed by the control flowing through the system. Since communication in such machines is relatively expensive and tasks are typically computation intensive, a good allocation can make a substantial difference in performance. As another example, consider an expert system implemented on a parallel machine. Apart from the problems of parallel execution of logic programs, which is a topic of current research, the problem of partitioning and allocating the rule base and factual data base are also worth considering. The motivation, again, is that communication is expensive and globally shared databases may be too much of a bottleneck to be viable. Thus, in the allocation problem, the database fragments become "files" and the inference tasks operating on them become the requests. It is remarked that conventional FAP is also very meaningful for both

local and wide area networks since partitioning of files is important both from reliability and performance perspectives.

Work on the file allocation problem has proceeded in two directions. One direction is towards formulating the optimization problem as one which minimizes some cost function typically consisting of file storage costs, and/or communication costs [2,3,5,7,10,13,14,17,18]. The solution here is usually based on solving a constrained 0-1 integer programming problem with either a linear or a non-linear objective function. The other direction is based on formulating the problem as one which optimizes the performance (e.g. the overall response time/throughput) of the system. Here, the problem is modelled as one of optimizing a queueing network of single servers [1,4,5,8,9,24,25]. The queueing networks considered here are assumed to have the product form (PF) property [22].

Both approaches have their advantages and limitations. A major limitation in the first approach is that delays due to queueing are either ignored or not realistically modelled. In the second approach, costs of communication and storage are typically not considered.

In this paper we take the second approach and model the FAP as a queueing network optimization problem, with the objective of minimizing mean overall response times to customer requests.

1.1 The FAP modeled as a queueing network optimization problem:

To see how the FAP gives rise to a queueing network optimization problem, consider a network of service facilities (nodes) used by a number of customers. For example, the nodes could be the computers in an Apollo ring, and the customers could be the processes running on each such computer. (This is an example of the FAP in the conventional sense). These customers share information contained in a set of distinct files. Suppose that a single copy of

each file is to be allocated among the nodes in the network. In the context of the Apollo ring network again, there could be a number of file servers which could store the files. (These file servers operate in a demand paged environment and retrieve one or more pages of information for transmission across the network to the processes making the requests.) Since it is likely that several customers could, at the same time, require information from one or more files stored at a node, some of the requests have to queue up for service.

We may consider a customer request as accessing one or more of these files in some sequence. The result of a request typically generates some information which is to be transmitted back to the customer originating the request. There can also be some information being transmitted between files in the request sequence. If these files are located on different nodes one must also consider transmission times and possible queueing delays at the communication servers. Let the mean time taken to complete a typical request sequence be called the "cycle time". The cycle time depends on the allocation and thus we want to find an allocation that gives the minimum cycle time. For tractability, it is usually assumed that the queueing network models of the system satisfy the PF property.

1.2 Previous work on performance models of the FAP:

One of the earliest works on performance models was the paper by Chen [1], where the problem was posed as an open network optimization problem with a single customer chain (i.e. a single request type.) The files, however, were allowed to be split among the nodes. Closed single chain network models which again allowed non-integral assignment of files were considered by Trivedi et al. [24,25], and Geist and Trivedi [9]. In many cases, however, integral assignments of files are much more realistic. A model by Foster et al. [8] considered a single chain closed queueing network optimization problem where

only integral assignments were allowed. However, their solution method uses a complex two-stage iterative approach, alternately solving a non-linear program in one stage and an integer program in the other. It appears unlikely that the solution scheme could effectively be applied to a reasonably sized problem.

The above methods generally restrict analysis to a central server network, where only one node makes all the demands. None of the above models considered communication times/delays. A model by Bryant and Agre [1] considers a more general closed model with multiple chains (i.e. multiple request types). It also incorporates communication delays. The solution method outlined is a heuristic which, starting with an initial arbitrary allocation, considers moving a file to all other nodes in order to obtain the best location for it. For each hypothesized move, the resulting queueing network is solved for the cycle times for each customer chain using the approximate algorithm of Schweitzer [23]. This scheme is iteratively repeated with each file till no further improvement is noticed. While this model is the most comprehensive performance model of the FAP among those reviewed above, the solution method is a simple enumeration technique which, further, need not find the optimal solution.

The objective of this paper, is to develop heuristics that can substantially cut down the complexity of determining a near optimal integral allocation. We also assume that the models are closed, i.e. a fixed number of statistically equivalent requests circulate endlessly in the network. While closed models are more difficult to deal with than open models, they are more realistic.

2. A model of the FAP:

In general, there may be several types of requests for file accesses; therefore, the problem is modelled, as a closed queueing network optimization

problem with multiple chains, where each chain represents a request type and each station corresponds to a node in the system. Let there be M nodes in the problem and R chains. A total of F different files are to be allocated among these nodes. Each customer starts at a designated node, accesses a subset of F files according to a Markovian sequence, and returns to the originating node. Processing each file in the sequence demands a certain amount of work from the system. In particular, processing file f for customer type r takes $\tau_{r,f}$ operations. Each node m executes at a certain rate S_m, which is the number of operations executed per second. The amount of time demanded by customer r from file f, which we call the mean file service time demand on file f by the r^{th} chain, if this file is placed in node m, is then $\tau_{r,f}/S_m$.

For modelling convenience, a customer making requests from node m is assumed to access a dummy file Z_m as the last file in his sequence. There are M such files, one for each node and we shall term these files "sentinel files". Apart from performing a policing duty, these files are also convenient to model any local computation that is performed when the customer's request returns to the node from which it originated.

Let $x_{m,f}$ be a 0-1 variable which is set to 1 if file f is allocated to node m and is set to 0 otherwise.

Every allocation $X = \{x_{m,f}\}$ generates a mean service time demand on the nodes for each type of customer. This is the sum of the various mean file service time demands by customer type r for the files placed at node m, viz.

$$L_{m,r}(X) = \sum_{f=1}^{F} x_{m,f} \cdot \tau_{r,f} / S_m. \qquad (2.1)$$

Let $L_r(X)$ represent the total service demand of the r^{th} customer chain from the network. Then,

$$L_r(X) = \sum_{m=1}^{M} L_{m,r}(X). \qquad (2.2)$$

Let $W_r(N,X)$ denote the cycle time of chains of customers when network population vector is N and allocation vector is X.

The optimization problem can now be written as:

$$\text{P1:} \quad \text{Min} \sum_{r=1}^{R} \beta_r \cdot W_r(N,X) \qquad (2.3)$$

$$\text{s.t.} \sum_{m=1}^{M} x_{m,f} = 1; \quad f=1,2,....,F \ \{\text{single copy only}\}$$

$$x_{m,f} \in \{0,1\} \quad \{\text{integral assignments only}\}$$

Here, $\beta_1,..,\beta_R$ are specified weights.

3. The FAP for a single customer chain:

In this section we consider the special case where there is only one request type, i.e. a single chain in the model. This special case is important owing to the fact that the cycle time in such a network is a monotonic function of the nodal service times and population. We will drop the chain subscript from all quantities in this case.

For the single chain case, the FAP is :

$$\text{P2: Minimize} \quad W(N,X)$$

$$\text{subject to} \quad \sum_{m=1}^{M} x_{m,f} = 1 \ ; \ f = 1,...,F,$$

$$x_{m,f} \in \{0,1\} \ ; \ f = 1,...,F \ ; \ m = 1,...,M.$$

Assuming negligible communication delays, the Mean Value Analysis (MVA) algorithm [20] for single server nodes yields:

$$W(N,X) = \sum_{m=1}^{M} L_m(X) + \sum_{m=1}^{M} L_m(X) \cdot Q_m(N-1,X), \qquad (3.1)$$

where $L_m(X)$ is defined by equation (2.1), and $Q_m(N-1)$ is the mean queue length that would form at node m with N-1 customers in the system.

The cycle time W(N,X) is a complex, non-linear function of the allocation variables and can be computed with O(MN) operations by known iterative algorithms [22]. It is a convex function of the allocation variables [24], and hence the non-linear integer program, P1 could be solved using a branch and bound procedure for small problems. As the number of variables increase, straightforward application of branch and bound algorithms could require a lot of computational effort, and this motivates consideration of some approximate solution techniques.

A heuristic interchange search technique is proposed, and this operates as follows: it starts with an allocation which attempts to balance the loads among the nodes as evenly as possible. This initial assignment is made by assigning each file, in turn, to a node such that the loads are maintained as uniformly distributed as possible among the nodes after each assignment. Once this initial allocation, X_0, is made, the cycle time for this allocation is evaluated. The heuristic then proceeds as follows: each possible combination of node-pairs are considered in turn. The files present in a node-pair are then examined to determine whether moving a file from its present node to the other (a SHIFT operation), or a pairwise exchange of files between nodes (a SWAP operation) would reduce the cycle time. With F files and M nodes, F(M-1) possible SHIFTs and at most F(F-1) possible SWAPs are examined. Assuming F > M, thus $O(F^2)$ such possibilities are examined. The cycle time which would result from each such operation is evaluated, and the lowest of such times, $W(X_1)$, corresponding to an allocation X_1, is compared with $W(X_0)$. If $W(X_1) > W(X_0)$, the search terminates

with X_0 as the best allocation found; otherwise the allocation X_1 is chosen, and this completes one iteration of the search. The next iteration now tries to find an assignment X_2 which has a cycle time less than that of assignment X_1. This process continues till no further decrease in cycle time is possible. Obviously the iterations must terminate, but results obtained need not be optimal.

For each potential SHIFT or SWAP operation considered, the exact evaluation of cycle time takes O(MN) operations as noted earlier. This means that each iteration of the search could take $O(MNF^2)$ operations. We now describe a heuristic for the FAP which obtains good estimates of the cycle time for each candidate allocation with much less computation, thereby improving the efficiency of the search.

3.1 Incremental Analyses:

This approach evaluates the cycle time exactly once at the start of each iteration. Each candidate allocation in the search process is now, however, evaluated with O(1) computations. This is based on incremental analysis which is described below:

Suppose that at the start of some iteration k, the allocation X_{k-1} is known. Then the loads generated at the nodes, L_m, m=1,...,M, and the resulting cycle time $W(X_{k-1})$, can be determined for this allocation. The heuristic then examines each file to determine the effect of a reassignment due to either a SHIFT operation or a SWAP operation. If this reassignment was done, it would change the loads at nodes i and j by some amounts ΔL_i and ΔL_j, leaving the loads at all other nodes unchanged. The resulting cycle time for this reassignment X_k' is then related to $W(X_{k-1})$ by the Taylor series:

$$W(X_k') = W(X_{k-1}) + \left(\Delta L_i \frac{\delta}{\delta L_i} + \Delta L_j \frac{\delta}{\delta L_j}\right) W(X_{k-1})$$

$$+ \frac{1}{2!}\left(\Delta L_i \frac{\delta}{\delta L_i} + \Delta L_j \frac{\delta}{\delta L_j}\right)^2 W(X_{k-1}) + \ldots \qquad (3.2)$$

and, for reasonably small ΔL_i, ΔL_j, we can drop the higher order terms.

For notational convenience, we shall write W to mean $W(X_{k-1})$. For the allocation X_{k-1}, let $Q_n(N)$ denote the mean queue length at node n, n = 1,..,M, at population N, and let λ_N be the corresponding throughput of the network. Let

$$U_i(N) = \lambda_N \cdot L_i, \qquad (3.3)$$

and

$$\Delta Q_i(N) = Q_i(N) - Q_i(N-1). \qquad (3.4)$$

Lemma 3.1:

$$\frac{\delta}{\delta L_i} W = \frac{N}{L_i \lambda_N} \Delta Q_i(N). \qquad (3.5)$$

A proof of Lemma 3.1 is given in Appendix A.

Proposition 3.1:

$$\frac{\delta^2}{\delta L_i^2} W = t_1 + t_2, \qquad (3.6)$$

where

$$t_1 \approx \frac{N}{U_i(N)}\left[\frac{(Q_i(N))^2}{L_i U_i(N)}\left(\frac{1 - U_i(N)}{1-U_i(N)(N-1)/N}\right) - \frac{(Q_i(N-1))^2}{L_i U_i(N-1)}\left(\frac{1 - U_i(N)}{1-U_i(N-1)(N-2)/(N-1)}\right)\right], \qquad (3.6a)$$

and

$$t_2 = \frac{N}{U_i(N)}\left[\frac{(Q_i(N))}{L_i}\left(1 - U_i(N)\right) \Delta Q_i(N)\right]. \qquad (3.6b)$$

The derivation of this approximate expression is given in Appendix B. Using arguments similar to that used for Proposition 3.1, it is possible to obtain an approximate expression for the partial derivative of W with respect to L_i and L_j and this expression is given by Proposition 3.2. The details of the derivation are omitted.

Proposition 3.2:

$$\frac{\delta^2}{\delta L_i \delta L_j} W \approx t_3 + t_4, \tag{3.7}$$

where

$$t_3 = - \frac{N}{\lambda_N L_i L_j} \left[\frac{Q_i(N)\Delta Q_j(N)}{1-U_i(N)(N-1)/N} - \frac{Q_i(N-1)\Delta Q_j(N-1)}{1-U_i(N-1)(N-2)/(N-1)} \right], \tag{3.7a}$$

and

$$t_4 = \frac{N}{\lambda_N L_i L_j} \Delta Q_i(N)\, \Delta Q_j(N). \tag{3.7b}$$

3.1.2 Experimental results:

Table 3.1 tabulates the results of using the incremental approach on some randomly generated problems. Results presented are for some typical problems, indicating the improvement in cycle times resulting from the search over that obtained by the initial load balancing allocation. For illustrating the effectiveness of the heuristic, the values of the cycle time obtained for the initial allocation were scaled to a 100.00 with the cycle time for the best allocation found by the heuristic being correspondingly scaled. For test cases where the number of nodes, files and customer types were reasonably small, the optimum allocation was evaluated by an exhaustive search. For these cases, the cycle times corresponding to the optimal allocation, appropriately scaled, is also indicated in the Table. In general, the improvements ranged from 0 % to 43

%, with the most significant improvements occuring when the network populations were small. The tests were carried out using a VAX 11/750 machine running the VMS operating system. Table 3.1 indicates the time taken by the heuristic and the exhaustive search to obtain these allocations.

In general, in the absence of communication delays, and when the customer populations are large, the FAP for the single chain case generally reduces to a problem of balancing the loads among the nodes as evenly as possible. This follows from the fact that in the case of an isolated M/M/1 station, the response time is a convex function of load with monotonically increasing derivative.

Table 3.1 : Result of some test problems for single chain case.

4. FAP for multiple chain networks:

Let $w_{mr}(\bar{N},X)$ denote the mean residence time of a customer belonging to chain r, at station m when the population vector is $\bar{N}$ and allocation vector is X. Then, under the PF assumption, the MVA algorithm gives:

$$w_{mr}(\bar{N},X) = L_{mr}(X)\left(1 + \sum_{j=1}^{R} q_{mj}(\bar{N}-e_r,X)\right),$$

where $q_{mj}(\bar{N}-e_r,X)$ is the queue length of chain j customers at station m when one chain r customer is removed from the network. In order to use this equation, we assume that:

$$q_{mj}(\bar{N}-e_r,X) = \frac{L_{mj}(X)}{L_j(X)} \quad (N_j-\delta_{jr})$$

where $\delta_{jr} \triangleq 1$, if j=r, else = 0.

The above equation makes the assumption that the mean queue lengths are proportional to the loads at the nodes. This then gives the approximation to the

mean residence time as $\tilde{w}_{mr}(\bar{N},X)$, where

$$\tilde{w}_{mr}(\bar{N},X) = L_{mr}(X)\left(1 - \frac{L_{mr}(X)}{L_r(X)} + \sum_{j=1}^{R} \frac{L_{mj}(X)N_j}{L_j(X)}\right).$$

The approximation for the cycle time for chain r, $\tilde{W}_r(\bar{N},X)$, is then given by

$$\tilde{W}_r(\bar{N},X) = \sum_{m=1}^{M} \tilde{w}_{mr}(\bar{N},X). \qquad (4.1)$$

As it was not possible to provide error bounds for the above approximation, its performance was studied experimentally. For this, all possible allocations on eight different networks, were considered. Since the number of possible allocations grows very fast with the number of nodes and files, the experiments were limited to problems with 4 nodes and 8 files (including the sentinel files), 10 customers chains and total network population of up to 30 customers. About 2000 different network configurations were thus generated for testing.

The error in estimating the cycle times for each chain is expressed as the ratio of the difference between the exact values and the approximate values, to the exact values. The errors were less than 5% whenever the allocations gave a network that was reasonably balanced with regard to time demands. The maximum error found was about 38% for some chain, on an unbalanced allocation. Although such errors of 38% on an experimental result are surely unacceptable, the errors in the approximation for network configurations which gave lower objective function values, namely the allocation which gave a more balanced network, were within 5% as mentioned above. Hence this approximation is expected to perform better for a problem such as the FAP. It must be noted that these errors are based on total network populations of up to 30. It is expected that larger network populations could increase the errors. For larger populations, an approximation based on a load balancing heuristic gives good results.

4.1 The simplified model for multiple chains:

Now consider the case where all processors operate at the same speed S. Assuming negligible communication times and delays, the total load on chain r, $L_r(X)$ is given, using equations (2.1) and (2.2) as

$$L_r(X) = \sum_{m=1}^{M} \sum_{f=1}^{F} \frac{x_{m,f}\,\tau_{r,f}}{S}$$

$$= \frac{1}{S} \sum_{f=1}^{F} \tau_{r,f}.$$

Hence, $L_r(X)$ is independent of the allocation, namely, $L_r(X) = L_r$.

From (4.1), the estimate, $\tilde{W}_r(\bar{N},X)$, of the cycle time for the r^{th} customer is :

$$\tilde{W}_r(\bar{N},X) = \sum_{m=1}^{M} L_{m,r}(X)\cdot\left(1 + \sum_{j=1}^{R} N_j \cdot L_{m,j}(X)/L_j - L_{m,r}(X)/L_r\right). \qquad (4.2)$$

Therefore,

$$\sum_{r=1}^{R} \beta_r\, \tilde{W}_r(\bar{N},X) = \sum_{r=1}^{R} \beta_r \sum_{m=1}^{M} L_{m,r}(X)\cdot\left(1 + \sum_{j=1}^{R} \frac{N_j \cdot L_{m,j}(X)}{L_j} - \frac{L_{m,r}(X)}{L_r}\right), \qquad (4.3)$$

and this can be rewritten, after some elementary algebra, as

$$= \sum_{r=1}^{R} \beta_r \cdot L_r + \sum_{m=1}^{M} \sum_{f=1}^{F} \sum_{g=1}^{F} d_{f,g} \cdot x_{m,f} \cdot x_{m,g},$$

where,

$$d_{f,g} = \sum_{r=1}^{R} \frac{\beta_r \cdot \tau_{r,f}}{S^2} \left(\sum_{j=1}^{R} N_j \tau_{j,g}/L_j - \tau_{r,g}/L_r \right) \qquad (4.4)$$

The first term in the objective function is now a constant and is removed from the objective function. The problem P1 is now reformulated as:

$$P1': \quad \text{Min} \sum_{m=1}^{M} \sum_{f=1}^{F} \sum_{g=1}^{F} d_{f,g} \cdot x_{m,f} \cdot x_{m,g}$$

$$\text{s.t.} \sum_{m=1}^{M} x_{m,f} = 1; \quad f=1,2,\ldots,F$$

$$x_{m,f} \quad \varepsilon \quad \{0,1\}; \quad f = 1,..,F; \; m = 1,..,M$$

with $d_{f,g}$ as defined by equation (4.4)

4.1.1 The set partitioning problem:

Problem P1' can now be interpreted as a set partitioning problem as follows: We have a weighted, undirected complete graph G. The vertices in this graph are the files labelled 1 through F. The edges connecting two vertices f and g represent the queueing delays that would be induced if files f and g were placed on the same node. Let $w(f,g) = (d_{f,g} + d_{g,f})$ represent the weight on the edge connecting vertex (file) f with vertex g in this graph. We can partition this graph into M vertex disjoint cliques. The problem is then to do the partitioning such that the sum of the weights on the edges in all the cliques is minimized.

This is still a hard problem. It falls into the class of NP-complete problems and is essentially in the same form as the k-min cluster problem discussed by Sahni and Gonzalez [21]. As a result, we look for a heuristic solution technique. Some heuristics have been proposed for the maximization version for this type of a partitioning problem (Lin and Kernighan [15], Sahni and Gonzalez [21]). The maximization version is to partition the set of F vertices into M disjoint sets (nodes) such that the weights on the edges joining vertices in different sets (nodes) is maximized. (Obviously, this is equivalent to the

problem of minimizing the sum of the weights on the arcs within the subsets.) Sahni and Gonzalez show that the maximization version of the problem, termed as the k-max cut problem, has an ε-approximation algorithm. They give a one-step heuristic algorithm for this version which obtains a bound on the closeness of their heuristic solution with respect to the optimal value. If EW^* represents the optimal value for this version of the problem, and if EW is the value found by their heuristic, then the error bound obtained by them is

$$\frac{|\ EW^* - EW\ |}{EW^*} \leq \frac{1}{M}.$$

Hence, for the maximization version, the value returned by the heuristic quickly approaches the optimal as the number of partitions required increases.

The heuristic proposed by Sahni and Gonzalez [21] does guarantee a good bound on the maximization version and hence it is very likely that it performs well when the minimization version is considered. We use it to obtain an initial allocation and then try to improve it by using another heuristic based on the scheme detailed by Lin and Kernighan [15]. The heuristic proposed by Lin and Kernighan is a version of an interchange search method and is similar, in method of operation, to the search technique used in section 3 for the single chain case. Adapting it to our model, it would operate as follows: starting with an allocation, it examines each pair of nodes and tries to obtain a better allocation by exchanging one or more files in one of these nodes with a corresponding number of files in the other node. The exchange is carried out if the sum of the Internal Weights in these two nodes decreases as a result of the exchange. Each such exchange is called a SWAP. Our algorithm, which we call LOCSEARCH, is essentially the same as this. In addition though, for every pair of nodes considered, we also try to look for a favorable repositioning, or SHIFT, of the files currently at these nodes, from one node to another, before looking for a SWAP. Also we have to worry about keeping the sentinel files

always positioned at the same node.

The time complexity of the algorithm is given here: Calculation of arc weights takes time $O(R^2 F^2)$. The initial allocation has time complexity of $O(F^3M)$. The algorithm for scanning all pairs of nodes and analyzing each pair of files in these nodes for a potential change in the location of the files, takes $O(F^2)$ time. It may be necessary to perform SWAP or SHIFT several times to get close to optimal. Hence this algorithm involves a one time cost of $O(R^2 \cdot F^2 + F^3 \cdot M)$. The cost per iteration is then $O(F^2)$. In contrast, the interchange search algorithm proposed by Bryant and Agre [1], which we shall call INTSEARCH, has time complexity $O(F \cdot M \cdot R^2)$ per iteration.

4.2. Experimental results :

The LOCSEARCH algorithm was tested for its effectiveness in finding good solutions. To determine how close the best allocation found by LOCSEARCH was relative to the optimal allocation, it was compared to the optimal allocation found by exhaustive search. A number of randomly generated examples were tested. The tests were carried out on a VAX 11/750 machine running the VMS operating system. It may be noted that with M nodes and F files, the number of allocations to be considered in an exhaustive search is N^F, and hence exhaustive search becomes prohibitively expensive in terms of computational effort as the number of nodes and files increase. For example, a reasonably small problem with 4 nodes, 6 files and 4 customer chains, required almost half and hour of CPU time to obtain the optimal solution through an exhaustive search. Hence the performance of the heuristic in obtaining allocations close to the optimal could perforce only be tested out for problems of a reasonably small size in the number of nodes and number of files. For larger problems, the performance of the heuristic was tested only by comparing the improvement

obtained by the technique over that found by a simple load balancing heuristic. Some of these results are tabulated in Table 4.1(a).

The LOCSEARCH heuristic is also compared with the INTSEARCH heuristic of Bryant and Agre [1], as shown in Table 4.1(a). The INTSEARCH heuristic used the same starting allocation here, as used by LOCSEARCH for purposes of comparison of the algorithms. The time taken by the LOCSEARCH and INTSEARCH algorithms in obtaining the final allocation is reported, as also the time taken for the exhaustive search, where applicable. The network populations were restricted to be between 1 to 5 customers per chain.

In the table, the Min values reported represent the cycle time values for the network based on the final allocations found by the two heuristics and by the exhaustive search technique. These values have been normalized relative to the value of the cycle time for the initial allocation found by the load balancing heuristic. The value of the allocation found by the load balancing heuristic was, for convenience, scaled to a 100.00. Thus, an entry of 85.7 in a column for "Min found", for example, would represent an improvement in cycle time of 14.3% over the allocation found by the load balancing heuristic.

The table shows that the load balancing heuristic performs well in some cases too, although certainly, the partitioning algorithm almost always improves on the initial solution found thus. The table also shows that the initial assignment obtained by BALANCE is better than the final value returned LOCSEARCH in one of the cases. This is due to the approximation to the cycle time used by the LOCSEARCH algorithm in evaluating allocations.

Table 4.1(a) : Same speed at all nodes; Initial allocation: Load Balance

Now, the effectiveness of the algorithm of Sahni and Gonzalez [21] in obtaining a good initial allocation was tested. This was done by repeating the LOCSEARCH algorithm over all the problems generated earlier, except that the starting allocation was now provided by the algorithm of Sahni and Gonzalez, instead of by the load balancing heuristic. Table 4.1(b) reports the results of some of these experiments. These results are for the same test problems as reported in Table 4.1(a). In this table, the cycle time values returned by the heuristic of Sahni and Gonzalez and by LOCSEARCH have both been normalized relative to the cycle time found by the load balancing heuristic. It can be seen from the table that the algorithm of Sahni and Gonzalez is much more effective in providing a good allocation compared to that found by the load balancing heuristic for the problems of smaller size. In fact, in two cases, it does provide the optimal allocation. However, the load balancing heuristic appears more effective in providing initial allocations for larger problems.

Table 4.1(b) : Same speed at all nodes; Initial allocation: Sahni and Gonzalez.

About 100 different problems were thus tested. Based on these test results, we make some observations:

(i) It appears that LOCSEARCH runs faster than INTSEARCH although, on one occasion, it returned with a value worse than the initial allocation value. This is due to the error in the approximation scheme adopted by us for the solution of the queueing network.

(ii) The best allocations found by both LOCSEARCH and INTSEARCH are usually quite close to the optimum, wherever it was possible to evaluate the

optimum through exhaustive search.

(iii) In general, for these cases, LOCSEARCH appears to do much better than INTSEARCH both in terms of speed of execution and in finding better allocations. However, in quite a few cases, the allocations found represented only a marginal improvement over the initial load balancing heuristic. Also, from Table 4.1(a), it can be seen that LOCSEARCH performs significantly faster than the INTSEARCH heuristic as the problem size gets large.

We now consider the use of the above approach on networks where the speeds of the various nodes was allowed to be different. The major difficulty here is that the weights on the arcs between files f and g, viz w(f,g), are no longer constant, but vary for each allocation, since each allocation causes the L_r values to change. The LOCSEARCH algorithm can be modified to account for this. This modified algorithm has a more complex objective function since equation (4.1) is now used in place of equation (4.2) for the objective function, and this results in the presence of a variable term in the denominator of the objective function. The search method is more complex as a result, although it proceeds along similar lines as for the case where all nodes were identical.

The initial allocation is based on balancing the loads at all nodes. We also do not have a partitioning algorithm in the former sense, since the w(f,g) values change with the allocation. Also, internal weights are to be recalculated for each possible change in allocation, viz a time complexity of $O(R^2 \times F^2)$ for each change in allocation. This makes the algorithm less effective in terms of execution speed. However, comparison with the performance of INTSEARCH for these cases indicates that LOCSEARCH does better on these cases too. About twenty different test problems were randomly generated and the heuristics were tested here. Table 4.1(c) details some of the representative results.

Table 4.1(c): Different speeds at nodes; Initial allocation: Load Balancing

The results in Table 4.1(c) indicate that (i) On the average LOCSEARCH takes less than half the time taken by INTSEARCH, although it is not as effective in finding better allocations than the INTSEARCH heuristic. (ii) Both heuristics find values upto 40% lower than those found by the load balancing heuristic.

5. Modeling communication times/delays:

So far, we have ignored the communication times and their attendant delays. In general, including these effects in the model makes it very difficult to analyze the model except, probably, through brute force enumeration techniques. In this section, two approaches are outlined for modeling the communications problem. Both approaches, however, are approximation techniques for which no error bounds are available, and both assume that the sum of the mean communication times/delays in a cycle, henceforth referred to as the communications component of the cycle time, are a relatively small component of the cycle time; the major component being the sum of the mean residence times at the node.

5.1 A two stage approach to model the communication component:

This approach first obtains a set of promising allocations ignoring the communication delay altogether. The next stage then evaluates each allocation by considering the time demands these allocations place on the communication channels, with their attendant queueing delays. We try to adjust the mean service times at the nodes in such a manner that the mean response times at the nodes obtained with these adjusted service time demands absorb these

communication times and delays. This is an iterative procedure which we outline below. The underlying premise here is that the communications component of the cycle time is not significant enough to affect the allocations found promising in stage 1.

The primary reason for considering a 2-stage approach to the FAP is twofold: (i) to avoid large increase in problem size, since modelling the channel servers as service stations in the model, may increase the number of stations from M to M(M+1)/2; (ii) to avoid assumption of a fixed communication path between each pair of nodes. In the second stage, we use an open model consisting of only communication servers which carry the traffic between various files (whose locations were determined in step 1). More specifically, let $\lambda_{mnr}(\bar{N},X)$ denote the throughput of the communication server sending chain r traffic from node m to node n and let $\lambda_r(\bar{N},X)$ be the throughput of chain r for allocation X. For this chain, let $v_{mr}(X)$ be the visit ratio to node m, and $f_{mnr}(X)$ the fraction of traffic from node m, bound for node n. Then, $\lambda_{mnr}(\bar{N},X) = v_{mr}(X)\cdot f_{mnr}(X)\cdot\lambda_r(\bar{N},X)$. Let σ_i denote the set of files allocated to node i. Let z_{fgr} be the average number of bits transmitted by a chain r request from file f to file g. Then the average time, τ_{mnr}, needed to transmit data from node m to node n for a chain r request is

$$\tau_{mnr} = \sum_{f\varepsilon\sigma_n} \sum_{g\varepsilon\sigma_n} z_{fgr}/\mu_{mn}, \qquad (5.1)$$

where μ_{mn} is the speed of the communication link connecting nodes m and n. Thus we have an R-chain open network with upto M(M-1)/2 stations. It can be solved easily to get the mean response times, $\delta_{mnr}(\bar{N},X)$. Assuming that there is no fixed path between nodes for a message to travel on, we can solve for the optimal routing for these messages using one of several schemes outlined by various authors (Kleinrock [11], Reiser [19], Kobayashi and Gerla [12]). For example, Kleinrock considers each channel server as an M/M/1 queue in his

model, and considers the problem of obtaining a routing of the messages for minimizing the total delay experienced in the network for all the messages circulating in the network.

Let us suppose we have solved the routing problem given an allocation X, and obtained the mean response times $\delta_{nmr}(\bar{N},X)$ arising therefrom for transmitting messages between every pair of nodes n,m for each customer class r. We now try to adjust the mean service time demands, $L'_{mr}(X)$ at the nodes in such a manner that the resulting nodal response times $w'_{mr}(\bar{N},X)$ absorb the communication delays (i.e. the mean response times at the channels) $\delta_{mnr}(\bar{N},X)$. For clarity of the ensuing discussion, we henceforth omit the subscript X. Hence, we first set

$$w'_{mr}(\bar{N}) = w_{mr}(\bar{N}) + \sum_{n=1}^{M} f_{mnr} \cdot \delta_{mnr}(\bar{N}). \tag{5.2}$$

The MVA algorithm gives

$$w'_{mr}(\bar{N}) = L'_{mr}\left(1 + Q'_m(\bar{N}-e_r)\right),$$

and this is approximated as (also see Schweitzer [23]),

$$w'_{mr}(\bar{N}) \approx L'_{mr}\left(1 + \sum_{\substack{j=1 \\ j\neq r}}^{R} q'_{mj}(\bar{N}) + \frac{N_r-1}{N_r} q'_{mr}(\bar{N})\right). \tag{5.3}$$

Once the $w'_{mr}(\bar{N})$ have been estimated using equation (5.2), the new cycle time with these mean response times is calculated, and the throughput is obtained using Little's result [16]. Now the new mean queue lengths at the nodes is obtained at population vector $\bar{N}$. These values are now used in equation (5.3) to obtain the L'_{mr} values.

With these new L'_{mr} values, the closed network can now be resolved for the candidate allocations to determine better values for the throughputs to be used in Stage II, and hence better estimates for w_{mr} for use in equation (5.2) and

so on. If the communications component is small to begin with, such iteration may, however, not be necessary.

If there were several candidate allocations to begin with, the best one can be selected on the basis of the cycle time of the network after accounting for the communication times.

5.2 Incorporating communication times in stage 1:

This approach includes a measure of the communications component of the cycle time during the first stage itself. The major assumption here is that the communications traffic is small compared to the mean residence times at the nodes, with the result that there is very little queueing at the channels; hence the channels can be modelled as delay servers. In addition, the assumption is made that all nodes are fully interconnected and operate at the same speed S_c, which is the number of bits of information transferred per second over a channel. In this case, the cycle time for the r^{th} chain will include the mean time to communicate all the required information generated during a cycle. Let $C_r(X)$ be the sum of the mean communication time demands made on all the channels by the r^{th} chain for a given allocation X. By the delay server assumption, this quantity is independent of $\bar{N}$.

The cycle time for chain r customers is then given by the MVA theorem as

$$W_r(\bar{N},X) = \sum_{m=1}^{M} w_{mr}(N,X) + C_r(X)$$

$$= \sum_{m=1}^{M} L_{mr}(X) + \sum_{m=1}^{M} L_{mr}(X)Q_m(\bar{N}-e_r,X) + C_r(X) \qquad (5.4)$$

As before, the assumption is made that the steady state distribution of chain r customers at a node is determined by the ratio of the mean time demands they make on that node to their total time demand over all nodes. The queue lengths at the channels are assumed very small. The mean queue length at node m,

$Q_m(\bar{N}-e_r,X)$, is hence approximated (as in section 4) by

$$Q_m(\bar{N}-e_r,X) = \sum_{j=1}^{R} q_{m,j}(N-e_r,X)$$

$$\approx \sum_{j=1}^{R} \frac{L_{mj}(X)}{L_j(X)} \cdot N_j - \frac{L_{mr}(X)}{L_r(X)}.$$

As before, when all nodes operate at the same speed, $L_j(X) = L_j$, independent of X. Letting $\tilde{W}_r(\bar{N},X)$ denote the approximation to the cycle time,

$$\tilde{W}_r(\bar{N},X) = L_r + C_r(X) + \sum_{m=1}^{M} L_{mr}(X)\left(\sum_{j=1}^{R} \frac{L_{mj}(X)\cdot N_j}{L_j} - \frac{L_{mr}(X)}{L_r}\right). \tag{5.5}$$

For simplicity of notation, unless otherwise specified, the indices m,n range over 1,...,M. Similarly, the indices f,g range over 1,...,F.

Now the access of file f by the customer from chain r produce, on the average, z_{fgr} bits of information for transmission to file g. If files f and g are not co-resident, the time taken to transmit this information is $S_c \cdot z_{fgr}$. Hence,

$$C_r(X) = \sum_{\substack{m,n \\ m\neq n}} \sum_{f,g} S_c \cdot z_{fgr} \cdot x_{mf} \cdot x_{ng}.$$

Interchanging summations and noting that $\sum_n x_{ng} = 1$,

$$C_r(X) = S_c \sum_{f,g} z_{fgr} \sum_{m} x_{mf}(1 - x_{mg}).$$

Let $\sum_{f,g} z_{fgr} = k_r$. Hence

$$C_r(X) = S_c \cdot k_r - S_c \sum_{m} \sum_{f,g} z_{fgr} \cdot x_{mf} \cdot x_{mg} \tag{5.6}$$

The approximation to the objective function hence gives:

$$\sum_{r=1}^{R} \beta_r \tilde{W}_r(\bar{N},X) = \sum_{r=1}^{R} (\beta_r L_r + S_c \cdot k_r) + \sum_{r=1}^{R} \beta_r \sum_{m=1}^{M} L_{mr}(X) \left(\sum_{j=1}^{R} \frac{L_{mj}(X)}{L_j} N_j - \frac{L_{mr}(X)}{L_r} \right)$$

$$- S_c \sum_{r=1}^{R} \beta_r \sum_{m=1}^{M} \sum_{f=1}^{F} \sum_{g=1}^{F} z_{fgr}\, x_{mf}\, x_{mg}.$$

Noting that the first summation is independent of the allocation, and after some straightforward algebra, the objective function can be rewritten as

$$\sum_{r=1}^{R} \beta_r \tilde{W}_r(\bar{N},X) = \sum_{m=1}^{M} \sum_{f=1}^{F} \sum_{g=1}^{F} d_{fg} \cdot x_{mf} \cdot x_{mg} - \sum_{m=1}^{M} \sum_{f=1}^{F} \sum_{g=1}^{F} x_{mf}\, x_{mg} \sum_{r=1}^{R} S_c\, \beta_r\, z_{rfg},$$

where d_{fg} is defined by equation (4.4).

Hence the problem here is to

Minimize

$$\sum_{m=1}^{M} \sum_{f=1}^{F} \sum_{f=1}^{F} e_{fg} \cdot x_{mf} \cdot x_{mg}, \tag{5.7}$$

subject to $\sum_{m=1}^{M} x_{mf} = 1 ; \quad f=1,\ldots,F;$

with

$$x_{mf} \quad \varepsilon \quad \{0,1\} ; \quad f=1,\ldots,F; \; m=1,\ldots,M,$$

where

$$e_{fg} = d_{fg} - \sum S_c \cdot \beta_r \cdot z_{rfg}. \tag{5.8}$$

It can be observed that this problem is again interpretable as a graph partitioning problem.

6. Conclusion:

The FAP modelled as a queueing network optimization problem is a complex problem. Usually this problem is addressed as a special case of a queueing network optimization problem with a single class of customers, without considering communication delays. In many cases, the problem has been posed as one for which continuous valued solutions are adequate. The objective of this paper was to extend the means by which this complex problem may be addressed and where integer solutions were necessary.

This paper considered approximate heuristic solutions to the FAP which enables the problem to be modelled with many customer chains, and which found integer allocations. A single chain version of the model was first addressed,

and this was analyzed using an incremental analytic approach. The multiple chain version of the problem was transformed, by an approximation, into a graph partitioning problem which was then analyzed with the aid of some existing algorithmic techniques which were adapted for this case. Finally, some means of approximately modeling communication times and delays have been proposed.

REFERENCES

1. Bryant, M., and Agre, J.R, "A queueing Network approach to the module allocation problem in distributed systems", Performance Evaluation Review, v 10, No 3, Fall 1981, pp. 191-204.

2. Casey, R.G., "Allocation of copies in an information network", Proceedings of the AFIPS Spring Joint Computer Conference, 1972, v 40, AFIPS Press, Montvale, N.J., 1972, pp. 617-625.

3. Ceri, S., Martella, G., and Pelagatti, G., "Optimal file allocation in a computer network: a solution method based on the knapsack problem", Computer Networks, 6, 1982, pp. 345-357.

4. Chen, P.P.-S. "Optimal file allocation in multilevel storage systems", Proceedings of the AFIPS National Computer Conference, 1973, v 42, AFIPS press, Arlington, Va, pp. 277-282.

5. Chu, W.W., "Optimal file allocation in a computer network", IEEE Transactions on Computers, vol C18, No 10, Oct 1969, pp. 885-889.

6. Dowdy, L.W., and Foster, D.V., "Comparative models of the file assignment problem", Computing Surveys, v 14, no 2, Jun 1982, pp. 287-313.

7. Fisher, M.L., and Hochbaum, D.S., "Data base location in computer networks", Journal of the ACM, v 27, no 4, October 1980, pp. 718-735.

8. Foster, D.V., Dowdy, L.W., and Ames, J.E., "File assignment in a computer network", Computer Networks, v 5, Sept 1981, pp. 341-349.

9. Geist, R.M., and Trivedi, K.S., "Optimal design of multilevel storage hierarchies", IEEE Transactions on Computers, C31, no 3, Mar 1982, pp. 249-259.

10. Irani, K.B., and Khabbaz, N.G., "A methodology for the design of communication networks and the distribution of data in distributed supercomputer systems", IEEE transactions on computers, vol C-31, no 5, May 1982, pp. 419-434.

11. Kleinrock, L., Queueing systems, Volume 2: Computer applications, Wiley, New York, 1976.

12. Kobayashi, H. and Gerla, M., "Optimal routing in closed queueing networks", ACM Transactions of Computer Systems, v 1, no 4, Nov 1983, pp. 294-310.

13. Laning, L.J., and Leonard, M.S., "File allocation in a distributed computer communication network", IEEE Transactions on Computers, v C-32, no 3, March 1983, pp. 232-244.

14. Levin, K.D., and Morgan, H.L., "A dynamic optimization model for distributed databases", Operations Research, v 26, no 5, Sept-Oct 1978, pp. 824-835.

15. Lin, S. and Kernighan, B.W., "An efficient heuristic procedure for partitioning graphs", Bell System Technical Journal, Feb 1970, pp. 291-307.

16. Little, J.D.C., "A proof of the queueing formula L = λW", Operations Research, v 9, 1961, pp. 383-387.

17. Mahmoud S. and Riordan, J.S., "Optimal allocation of resources in distributed information networks", ACM Transactions on Database Systems, v 1, no 1, March 1976, pp. 66-78.

18. Morgan, H.L., and Levin, K.D., "Optimal program and data location in computer networks", Communications of the ACM, v 20, no 5, pp. 315-322.

19. Reiser, M., "A queueing network analysis of computer communications with window flow control", IEEE Transactions on Communications, v 27, Aug 1979, pp. 1199-1209.

20. Reiser, M., and Lavenberg, S.S., "Mean value analysis of closed multichain queueing networks", Journal of the ACM, v 27, no 2, April 1980, pp. 313-322.

21. Sahni, S., and Gonzalez, T., "P-complete approximation problems" Journal of the ACM, v 23, no 3, July 1976, pp. 555-565.

22. Sauer, C.H., and Chandy, K.M., Computer systems performance modelling, Prentice-Hall, Englewood Cliffs, N.J., 1981.

23. Schweitzer, P., "Approximate analysis of multiclass closed networks of queues", Proceedings of the International Conference on Stochastic Control and Optimization, Amsterdam, 1979.

24. Trivedi, K.S., Wagner, R.A. and Sigmon, T.M., "Optimal selection of cpu speed, device capacities, and file assignments", Journal of the ACM, v 27, no 3, July 1980, pp. 457-473.

25. Trivedi, K.S., and Sigmon, T.M., "Optimal design of linear storage hierarchies", Journal of the ACM, v 28, no 2, April 1981, pp. 270-288.

Table 3.1

Problem Number	Size of Problem (Nodes, Files, Customers)	Minimum found		Time taken (secs)	
		Heuristic	Exhaustive	Heuristic	Exhaustive
1	2, 5,12	100.00	100.00	0.02	0.23
2	3, 4, 8	95.42	95.42	0.04	0.50
3	2,10, 6	95.74	95.14	0.08	4.46
4	3, 6,10	97.36	97.16	0.05	5.27
5	3, 7, 5	93.20	92.79	0.07	9.41
6	3, 7, 5	97.28	96.40	0.07	10.57
7	4, 6, 8	100.00	100.00	0.03	31.59
8	4, 7, 5	97.01	95.34	0.09	86.41
9	3, 9, 8	95.03	94.79	0.05	125.89
10	3,16, 5	87.28	-	0.24	-
11	4, 9,15	98.09	-	0.21	-
12	3,18,15	96.47	-	0.56	-
13	12,29,15	86.11	-	2.16	-
14	15,50,10	78.92	-	8.16	-
15	18,55, 5	56.79	-	8.02	-
16	18,55,30	92.97	-	8.12	-

Table 3.1 : Result of some test problems for single chain case.

Size of Prob M, F, R	LOCSEARCH Min found	LOCSEARCH Time taken (secs)	INTSEARCH Min found	INTSEARCH Time taken (secs)	EXHAUSTIVE Min found	EXHAUSTIVE Time taken (secs)
3, 4, 3	85.7	0.21	85.7	0.62	85.7	9.3
3, 4, 5	73.6	0.32	73.6	1.06	73.6	29.0
3, 5, 4	97.5	0.34	100.0	0.73	97.2	71.0
3, 6, 4	78.7	0.44	78.7	1.96	78.7	171.0
4, 4, 6	100.3	0.47	100.0	1.64	99.6	339.1
2,10, 5	92.2	0.84	100.0	1.08	82.9	487.7
4, 6, 4	87.2	0.51	89.0	1.62	87.2	1525.0
7,10, 5	98.6	1.60	99.3	16.44	-	-
8,16, 4	97.4	3.40	99.1	35.75	-	-
10,25,15	97.1	32.63	99.1	597.43	-	-

Table 4.1(a) : Same speed at all nodes; Initial allocation: Load Balance

Size of Prob M, F, R	SAHNI & GONZALEZ	LOCSEARCH Min found	EXHAUSTIVE Min found
3, 4, 3	85.73	85.73	85.73
3, 4, 5	73.64	73.64	73.64
3, 5, 4	98.77	97.19	97.19
3, 6, 4	93.31	78.68	78.68
4, 4, 6	101.57	100.31	99.60
2,10, 5	103.30	92.19	82.19
4, 6, 4	95.51	87.20	87.20
10,25,15	100.75	96.78	-
7,10, 5	107.14	98.58	-
8,16, 4	104.66	97.39	-

Table 4.1(b) : Same speed at all nodes; Initial allocation: Sahni and Gonzalez [21]

Size of Prob M, F, R	LOCSEARCH Min found	LOCSEARCH Time (secs)	INTSEARCH Min found	INTSEARCH Time (secs)	EXHAUSTIVE Min found	EXHAUSTIVE Time (secs)
4, 4, 4	65.8	0.94	62.0	1.73	62.0	98.7
4, 4, 6	59.6	1.56	63.1	3.33	59.6	261.9
4, 4, 6	69.9	1.57	74.1	3.39	69.9	269.4
4, 4, 6	100.0	0.57	100.0	2.11	100.0	336.9
2,12, 5	97.8	2.99	96.6	1.82	94.0	1504.9
3, 8, 6	95.0	3.86	97.2	6.21	92.3	5059.7
4, 7, 6	88.6	3.52	89.7	6.90	85.4	23172.6
4, 8, 8	91.3	7.16	91.7	13.70	-	***
7,12,10	83.5	25.50	85.6	74.0	-	-
10,26,15	87.8	361.87	87.6	685.0	-	-
11,29,16	90.3	491.79	90.3	1442.7	-	-

Table 4.1(c): Different speeds at nodes; Initial allocation: Load Balancing

*** : Job was aborted after it took more than 100000 seconds of CPU time.

Appendix A

Proof of Lemma 3.1

Lemma 3.1:

$$\frac{\delta}{\delta L_i} W = \frac{N}{L_i \lambda_N} \Delta Q_i(n).$$

Proof:

Denote the normalizing constant at population N by G_N. From the Throughput law (Sauer and Chandy [22]) and Little's rule [16], we have:

$$W = \frac{G_N}{G_{N-1}}, \tag{A1}$$

where

$$G_N = \sum_{\langle n_m \rangle} \prod_{m=1}^{M} (L_m)^{n_m}, \tag{A2}$$

and

$$\sum_{\langle n_m \rangle} = \{n_1, \ldots, n_m \mid \sum_{m=1}^{M} n_m = N\}$$

denotes the set of all possible arrangements of N customers among the M nodes.

Hence

$$\frac{\delta}{\delta L_i} W = N \frac{\delta}{\delta L_i} \left(\frac{G_N}{G_{N-1}} \right)$$

$$= \frac{N}{G^2_{N-1}} \left(G_{N-1} \frac{\delta}{\delta L_i} G_N - G_N \frac{\delta}{\delta L_i} G_{N-1} \right), \tag{A3}$$

where

$$\frac{\delta}{\delta L_i} G_N = \frac{\delta}{\delta L_i} \left(\sum_{\langle n_m \rangle} \prod_{m=1}^{M} (L_m)^{n_m} \right)$$

$$= \frac{1}{L_i} \sum_{\langle n_m \rangle} n_i \prod_{m=1}^{M} (L_m)^{n_m}$$

$$= \frac{1}{L_i} \sum_{\langle n_m \rangle} n_i \, G_N \, P\{n_1, \ldots, n_m\},$$

with

$$P\{n_1,\ldots,n_m\} = \prod_{m=1}^{M} \frac{(L_m)^{n_m}}{G_N}$$

denotes the probability of having n_m customers at node m, m = 1,...,M.

Hence

$$\frac{\delta}{\delta L_i} G_N = \frac{1}{L_i} G_N Q_i(N),$$

and so we obtain

$$\frac{\delta}{\delta L_i} W = \frac{N}{\lambda_N L_i} \left(\frac{G_N}{G_{N-1}} Q_i(N) - \frac{G_N}{G_{N-1}} Q_i(N-1) \right)$$

$$= \frac{N}{\lambda_N L_i} \left(Q_i(N) - Q_i(N-1)\right).$$

Let

$$U_i(N) = \lambda_N L_i \tag{A4}$$

denote the utilization node i, at customer population N.

Thus,

$$\frac{\delta}{\delta L_i} W = \frac{N}{U_i(N)} \left(Q_i(N) - Q_i(N-1)\right). \tag{A5}$$

The term $\delta W/\delta L_j$ is evaluated in a like manner.

□

Appendix B

Derivation of Proposition 3.1:

Proposition 3.1:

$$\frac{\delta^2}{\delta L_i^2} W = t_1 + t_2, \tag{B1}$$

where

$$t_1 \approx \frac{N}{U_i(N)}\left[\frac{(Q_i(N))^2}{L_i U_i(N)}\left(\frac{1 - U_i(N)}{1-U_i(N)(N-1)/N}\right) - \frac{(Q_i(N-1))^2}{L_i U_i(N-1)}\left(\frac{1 - U_i(N)}{1-U_i(N-1)(N-2)/(N-1)}\right)\right],$$

and

$$t_2 = \frac{N}{U_i(N)}\left[\frac{(Q_i(N))}{L_i}\left(1 - U_i(N)\right)\Delta Q_i(N)\right).$$

We have

$$\frac{\delta^2}{\delta L_i^2} W = \frac{\delta}{\delta L_i}\frac{\delta}{\delta L_i} W$$

$$= \frac{\delta}{\delta L_i}\left[\frac{N}{U_i(N)}\left(Q_i(N) - Q_i(N-1)\right)\right]$$

$$= t_1 + t_2, \tag{B2}$$

where

$$t_1 = \frac{N}{U_i(N)}\frac{\delta}{\delta L_i}\left(Q_i(N) - Q_i(N-1)\right), \tag{B3}$$

and

$$t_2 = \frac{N}{(U_i(N))^2}\left(Q_i(N) - Qi(N-1)\right)\frac{\delta}{\delta L_i} U_i(N) . \tag{B4}$$

Now,

$$\frac{\delta}{\delta L_i} U_i(N) = \frac{\delta}{\delta L_i} L_i \lambda_N$$

$$= \lambda_N + L_i \frac{\delta}{\delta L_i}\lambda_N,$$

and hence, using $\lambda_N = N/W$,

$$\frac{\delta}{\delta L_i} U_i(N) = \lambda_N - \frac{NL_i}{W^2} \frac{\delta}{\delta L_i} W$$

$$= \lambda_N - \lambda_N \left(Q_i(N) - Q_i(N-1)\right),$$

which, on simplification, gives

$$\frac{\delta}{\delta L_i} U_i(N) = \frac{Q_i(N)}{L_i} \left(1 - U_i(N)\right), \tag{B5}$$

and hence, setting $\Delta Q_i(N) = Q_i(N) - Q_i(N-1)$, we get

$$t_2 = \frac{N}{U_i(N)} \left[\frac{(Q_i(N))}{L_i} \left(1 - U_i(N)\right) \Delta Q_i(N) \right].$$

Now, to evaluate the term t_1 given by equation (B3), we use the mean value analysis of Reiser and Lavenberg [19], to write

$$\frac{\delta}{\delta L_i} Q_i(N) = \frac{\delta}{\delta L_i} U_i(N) + \frac{\delta}{\delta L_i} \left(U_i(N) Q_i(N-1)\right)$$

$$= \left(1 + Q_i(N-1)\right) \frac{\delta}{\delta L_i} U_i(N) + U_i(N) \frac{\delta}{\delta L_i} Q_i(N-1)$$

$$= \frac{Q_i(N)}{U_i(N)} \frac{\delta}{\delta L_i} U_i(N) + U_i(N) \frac{\delta}{\delta L_i} Q_i(N-1). \tag{B6}$$

Now, the term $Q_i(N-1)$ is approximated, using the well known Schweitzer heuristic (Schweitzer [23]), as

$$Q_i(N-1) \approx Q_i(N) \cdot (N-1)/N. \tag{B7}$$

Hence, from (B6) and (B7),

$$\frac{\delta}{\delta L_i} Q_i(N) \approx \frac{Q_i(N)}{U_i(N)} \frac{\delta}{\delta L_i} U_i(N) + U_i(N) \cdot \frac{N-1}{N} \frac{\delta}{\delta L_i} Q_i(N),$$

from which we get,

$$\frac{\delta}{\delta L_i} Q_i(N) \approx \left[\frac{Q_i(N)}{U_i(N)} \frac{\delta}{\delta L_i} U_i(N) \right] / \left[1 - \frac{N-1}{N} \cdot U_i(N) \right]. \tag{B8}$$

From equations (B3), (B8), and (B5), we get the desired expression for t_1. □