PUC

Series: Monografias em Ciência da Computação

Nº 5/81

A MULTICLASS QUEUEING NETWORK MODEL OF COMPUTER SYSTEMS WITH VARIABLE DEGREE OF MULTIPROGRAMMING

by

Daniel A. Menascé
Virgilio A. F. Almeida

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO
RUA MARQUÊS DE SÃO VICENTE, 225 – CEP-22453
RIO DE JANEIRO – BRASIL

Series: Monografias em Ciência da Computação

Nº 5/81

Series Editor: Marco A. Casanova July, 1981

A MULTICLASS QUEUEING NETWORK MODEL OF COMPUTER SYSTEMS WITH VARIABLE DEGREE OF MULTIPROGRAMMING*

by

Daniel A. Menascé
Virgilio A. F. Almeida**

* Partially supported by FINEP.

** Centrais Elétricas de Minas Gerais, Belo Horizonte, Minas Gerais, Brasil.

A MULTICLASS QUEUEING NETWORK MODEL OF COMPUTER SYSTEMS WITH VARIABLE DEGREE OF MULTIPROGRAMMING

Daniel A. Menascé*
Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro,
22453 Rio de Janeiro, Brazil

Virgilio A.F. Almeida
Centrais Elétricas de Minas Gerais
Belo Horizonte, Brazil

Abstract

Current closed queueing network models suffer from a severe limitation since they require that the number of jobs in the system be fixed. However, in actual systems, the number of jobs multiprogrammed in each job class is not fixed but varies dinamically depending on the nature of the class and on the amount of memory available for each class. This paper presents the solution of a multiclass queueing network model in which the number of jobs of a given class is not fixed but varies within a given range.

* Partially supported by Financiadora de Estudos e Projetos (FINEP), Brazil.

1. INTRODUCTION

Queueing network models of computer systems have gained considerable attention from the performance analyst in the last few years. The first analytic results in queueing networks are due to Jackson [1 and 2] and Gordon and Newell [3]. In 1971, Buzen [4] introduced a convenient model of multiprogrammed systems - the central server model - and presented [4 and 5] efficient computational algorithms for closed queueing networks. Baskett, Chandy, Muntz and Palacios [6] extended the theory to allow multiple classes, different queueing disciplines and non - exponential serves.

In 1976, Buzen [7 and 8] introduced operational analysis and explained why some of the classic results in queueing theory hold even when the assumptions upon which they were based are not verified. Later, Denning and Buzen [9 and 10] applied operational analysis to the study of queueing networks. Bouhana [11] gave an operational treatment of centralized queueing networks - a generalization of the central server model. Roode [12] extended these results to treat multiclass queueing networks.

All closed queueing networks models suffer from a severe limitation since they require that the number of jobs in the system be fixed. It turns out that in actual systems, the number of jobs multiprogrammed in each job class is not fixed but varies dinamically depending on the nature of the class and on the amount of memory available for each class. In this paper we present the solution of a multiclass queueing network model in which the number of jobs of a given class i $(1 \leq i \leq R)$ is not fixed, but is allowed to vary from zero up to N_i. Let a configuration be a tuple $(n_1,\ldots,n_R)$ such that n_i is the current number of class i jobs being multiprogrammed. We present here an algorithm to obtain performance measures (e.g. throughput, response time) for each job class, averaged over all configurations observed during an observation period.

Section 2 presents the computer system model considered here and characterizes the types of woarkloads analyzed. Section

3 presents the corresponding queueing network model. Section 4 shows how to obtain the normalization constant for the solution of the network and section 5 contains the derivation of several performance measures based on the normalization constant. In section 6, the problem of a variable degree of multiprogramming is solved. Finally, section 7 establishes a parallel between the operational and stochastic approaches to the model introduced in this paper.

2. COMPUTER SYSTEM MODEL AND WORKLOAD CHARACTERIZATION

Our model of a computer system consists of a central subsystem which contains the CPU and I/O devices.

A collection of terminals outside the central subsystem originate timesharing and transaction processing jobs which are executed by the central subsystem along with batch jobs. (see figure 1)

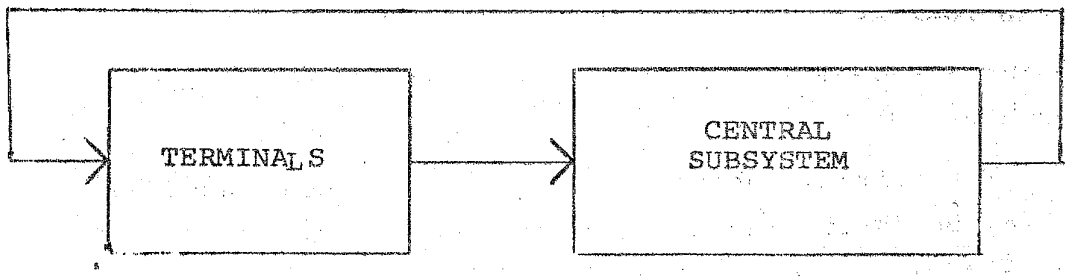

Figure 1 - Computer System Model

Let us describe below the three types of workloads considered in our model, namely batch, timesharing and transaction processing.

The batch type of class is characterized by the average degree of multiprogramming of that class. We assume that a continuous backlog situation exists for batch jobs. In other words, a batch that terminates is immediately replaced by an identical job. For this type of class the main performance

measures are the throughput and the turn-around time.

The timesharing class is described by the number of terminals, by the average think time and by the maximum level of multiprogramming for this class. Think time is defined as the interval between a response to a terminal and the submission of a new command (job) by the same terminal. The main performance measures in this case are response time and throughput.

Finally, the transaction processing class is characterized by the average transaction arrival rate to the central subsystem and by the maximum level of multiprogramming. The main performance measure in this case is response time.

The existence of these three types of workloads implies that a multiclass queueing network is needed. A job class characterizes a common behavior and resource usage statistics observed for a collection of jobs. We consider the existence of R classes numbered from 1 to R. Each class may be of one of the three types described above, i.e. batch, timesharing and transaction processing. However other types of classes may be imagined. The methods describes in this paper work as well for any type of class, provided one can derive the fraction of time that the system is in a given state, in terms of the throughput for that particular class.

The central subsystem is assumed to be a centralized network, as introduced in [11]. Figure 2 illustrates such a type of network. As it can be seen

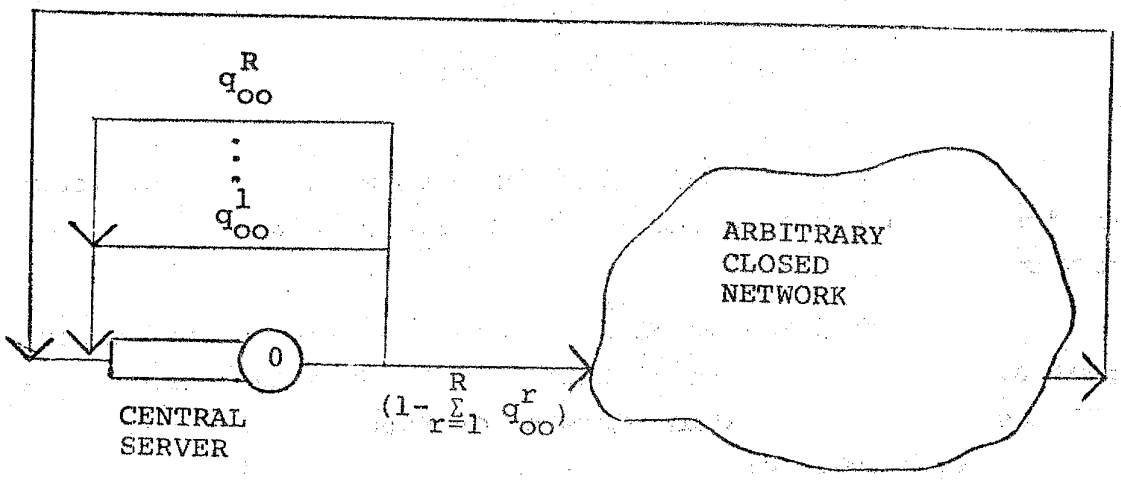

Figure 2 - Centralized Network

a centralized network is a generalization of the central server model [4] in which aside from having a central server (possessing a loop), the rest of the network can have an arbitrary topology.

The loop around the central server represents a conceptual exit of a job from the system, i.e. a fraction q_{oo}^r, for $1 \le r \le R$, of all class r jobs that leave the central server are said to leave the system an are immediately replaced by a new job of the same class. The conceptual exit is an extremely interesting modeling device since it allows us to consider arrivals and departures of jobs in a closed network. Section 3 explores in more detail the advantages of considering a centralized network.

3. QUEUEING NETWORK MODEL

This section presents the quantities which describe the queueing network model used to model computer systems. Consider a queueing network consisting of M devices (e.g. processor , i/o devices). Jobs in the system may belong to any one of R classes. Jobs are not allowed to change classes.

Let,

n_{ir} = number of class r jobs at device i.

$n_i = \sum_{r=1}^{R} n_{ir}$ = number of jobs at device i.

$N_r = \sum_{i=1}^{M} n_{ir}$ = total number of class r jobs

$N = \sum_{r=1}^{R} N_r$ = total number of jobs in the system.

We assume that all devices in the network have a load independent behavior, i.e. the time a job takes to be serviced at the device does not depend on the size of the queue at the device. Let us now define the basic operational quantities of

interest. These quantities represent data collected during an observation period of T seconds.

A_{ir} : number of class r job arrivals at device i

C_{ir}^{j} : number of times that a class r job requested service at device j immediately after being serviced at device i. C_{ir}^{o} is the number of class r jobs that leave the network immediately after leaving device i.

C_{ir} : number of class r job completions at device i. C_{or} is the number of class r exits from the network. $C_{ir} = \sum_{j=1}^{M} C_{ir}^{j}$ and $C_{or} = \sum_{j=1}^{M} C_{jr}^{o}$.

X_{ir} : throughput of class r jobs at device i. $X_{ir} = C_{ir}/T$. X_{or} is the class r system throughput

B_{ir} : total busy time of device i for class r jobs.

S_{ir} : average service time of class r jobs at device i. $S_{ir} = B_{ir}/C_{ir}$.

V_{ir} : average number of completions of class r jobs at device i per class r job system exit. $V_{ir} = C_{ir}/C_{or}$. V_{ir} is also called average number of class r job visits to device i, or device i visit ratio.

Y_{ir} : average usage of device i by class r jobs . $Y_{ir} = V_{ir} S_{ir}$. This is the total amount of service time accumulated by a job in all visits to device i averaged over all class r jobs.

U_{ir} : utilization of device i by class r jobs. $U_{ir} = B_{ir}/T = X_{ir} S_{ir}$.

3.1 - Network Solution

A network state $\underline{n}$ is defined as $\underline{n} = (\underline{n}_1, \underline{n}_2, \ldots, \underline{n}_M)$

where $\underline{n}_i$ is the state of device i given by

$$\underline{n}_i = (n_{i1}, n_{i2}, \ldots, n_{iR})$$

In order for $\underline{n}$ to be a feasible network state it is necessary that

$$\sum_{i=1}^{M} \sum_{r=1}^{R} n_{ir} = N$$

Let $T(\underline{n})$ be the total time that the network was in state $\underline{n}$ during the observation period. The proportion of time that the system is in state $\underline{n}$ is given by

$$p(\underline{n}) = \frac{T(\underline{n})}{T} \qquad (1)$$

and $\sum_{\underline{n}} p(\underline{n}) = 1$ where the summation is over all possible states $\underline{n}$.

The solution to the network was shown by Roode in [12] to be

$$p(\underline{n}) = \frac{1}{G} \prod_{i=1}^{M} \prod_{r=1}^{R} F_{ir}(n_{ir}) \qquad (2)$$

where G is a normalization constant calculated so that $\sum_{\underline{n}} p(\underline{n}) = 1$, and

$$F_{ir}(n_{ir}) = \begin{cases} 1 & \text{if } n_{ir} = 0 \\ X_{ir}\, S_{ir}(n_{ir}) F_{ir}(n_{ir}-1) & \text{if } n_{ir} > 0 \end{cases} \qquad (3)$$

Since we are only considering load independent devices, we have that

$$F_{ir}(n_{ir}) = (X_{ir}\, S_{ir})^{n_{ir}} \quad \text{for } n_{ir} \geq 0 \qquad (4)$$

The solution given in (2) assumes that job flow is balanced, i.e. arrivals equal departures at each device. Therefore the troughputs X_{ir} must be a solution of

$$X_{ir} = \sum_{j=1}^{M} X_{jr} q_{ir}^{j} \quad \text{for } 1 \leq i \leq M \qquad (5)$$

where q_{ir}^{j} is the fraction of class r jobs that move to

device i after leaving device j. The q_{ir}^{j} are called routing frequencies. As is true of any eigenvector equation, the values of the X_{ir} $(1 \leq i \leq M)$ are not uniquely determined. Any solution of (5) may be used in (3). In particular, since

$$V_{ir} \triangleq \frac{C_{ir}}{C_{or}} = \frac{C_{ir}/T}{C_{or}/T} = \frac{X_{ir}}{X_{or}} \qquad (6)$$

we may use (6) in (5), yielding

$$V_{ir} = \sum_{j=1}^{M} V_{jr}\, q_{ir}^{j} \qquad (7)$$

Equation (7) says that the visit ratios are also a solution of (5). In the case of a multiclass centralized network, it can be easily, shown, following an argument similar to the one in [11], that the visit ratios may be uniquely determined since the class r visit ratio for the central server is $1/q_{or}^{o}$. Therefore, if we use V_{ir} in place of X_{ir} in (4) we get

$$F_{ir}(n_{ir}) = Y_{ir}^{n_{ir}} \quad \text{if } n_{ir} > o \text{ and } 1 \text{ if } n_{ir} = o \qquad (8)$$

Notice, the important fact that in the case of a centralized network, the network solution may be obtained directly in terms of the average resource usage Y_{ir} which are far more simpler to measure than the routing frequencies that would be necessary to solve (5).

The network solution can now be expressed as

$$p(\underline{n}) = \frac{1}{G} \prod_{l=1}^{M} Y_{il}^{n_{il}}\, Y_{i2}^{n_{i2}} \cdots Y_{iR}^{n_{iR}}$$

It should be emphasized that no assumption regarding service time distribution was made in [12] in order to obtain expression (2). However the following operational assumptions, which are defined in [10], must hold for expression (2) to be true:

Flow Balance: arrivals equal departures at each system device or network state.

One-Step Behavior: the only observable state changes result from single jobs either entering or leaving the system, or moving from one device into another.

Device Homogeneity: the output rate of a device is independent of the system state and may be depend only on the queue size at the device.

Routing Homogeneity: the routing frequencies between devices independ on the system state.

4. CALCULATION OF THE NORMALIZATION CONSTANT

Computationally efficient algorithms for calculating the normalization constant in closed queueing networks were first presented by Buzen in [4 and 5]. Balbo et al [13] give solution techniques for the case of multi-class models in which jobs may change class membership. Williams and Bhandiwad [14] suggested a generating function approach to the analysis of multi-class queueing networks in which jobs are not allowed to change class membership. They only give expressions for the case of two classes only.

In this section we use a generating function approach, in a similar way as in [14], to derive an expression for the normalization constant for the case of R job classes.

The normalization constant G is defined as

$$G = \sum_{\underline{n} \in S(N,M,R)} \prod_{i=1}^{M} Y_{i1}^{n_{i1}} Y_{i2}^{n_{i2}} \cdots Y_{iR}^{n_{iR}} \qquad (10)$$

where

$$S(N,M,R) = \{\underline{n} \triangleq (n_{11},n_{12},\ldots,n_{1R},n_{21},\ldots,n_{2R},\ldots,n_{MR}) \mid \sum_{i=1}^{M} n_{ir} = N_r \;\&\; \sum_{r=1}^{R} N_r = N \;\&\; n_{ir} \geq 0 \quad \forall i,r\}$$

Let us define the generating function of device i as

$$f_i(T_1,T_2,\ldots,T_R) = \sum_{j=0}^{\infty} (Y_{i1}\,T_1 + Y_{i2}\,T_2 + \ldots + Y_{iR}\,T_R)^j \qquad (11)$$

which yields

$$f_i(T_1,T_2,\ldots,T_R) = \frac{1}{1-(Y_{i1}\,T_1 + \ldots + Y_{iR}\,T_R)} \qquad (12)$$

Let us define the network generating function as

$$g(T_1,\ldots,T_R) = \prod_{i=1}^{M} f_i(T_1,\ldots,T_R) \qquad (13)$$

It can be easily seen that G is the coefficient of term $T_1^{N_1} T_2^{N_2} \ldots T_R^{N_R}$ in $g(T_1,\ldots,T_R)$. Let $G(K_1,\ldots,K_R)$ be the coefficient of the term $T_1^{K_1} T_2^{K_2} \ldots T_R^{K_R}$ in $g(T_1,\ldots,T_R)$.

Define now

$$g_1(T_1,\ldots,T_R) = f_1(T_1,\ldots,T_R) \qquad (14)$$

and

$$g_i(T_1,\ldots,T_R) = f_i(T_1,\ldots,T_R)\; g_{i-1}(T_1,\ldots,T_R) \quad \text{for } 1<i\leq M \qquad (15)$$

Let $G_i(K_1,\ldots,K_R)$ be the coefficient of $T_1^{K_1} \ldots T_R^{K_R}$ in $g_i(T_1,\ldots,T_R)$. Using (12) in (15) we get

$$g_i(T_1,\ldots,T_R) = g_{i-1}(T_1,\ldots,T_R) + g_i(T_1,\ldots,T_R) \sum_{r=1}^{R} y_{ir}\,T_R \qquad (16)$$

Equating the coefficients of the term $T_1^{K_1} \ldots T_R^{K_R}$ at both sides of expression (16) it follows that

$$G_i(K_1,\ldots,K_R) = G_{i-1}(K_1,\ldots,K_R) + \sum_{r=1}^{R} Y_{ir}\, G_i(K_1,\ldots,K_r-1,\ldots,K_R) \quad (17)$$

where $K_1,\ldots,K_R$ represent the number of jobs from classes $1,\ldots,R$ respectively.

Expression (17), along with the initial conditions given below, provides a computationally efficient way of calculating the normalization constant.

$$G_i(K_1,\ldots,K_R) = 1 \quad \text{if} \quad K_1 = \ldots = K_R = 0 \quad (18)$$

and

$$G_o(K_1,K_2,\ldots,K_R) = \begin{cases} 1 & \text{if } K_1=\ldots=K_R=0 \\ 0 & \text{otherwise} \end{cases} \quad (19)$$

Finally, expressions (17) through (19) allow us to calculate the normalization constant as

$$G = G_M(N_1,\ldots,N_R)$$

5. PERFORMANCE MEASURES

One of the nice properties of the normalization constant is that several useful performance measures can be derived from it.

Let

$P_i(n_{i1},\ldots,n_{iR})$: proportion of time that $n_{i1},\ldots,n_{iR}$ jobs of classes 1 through R, respectively, are at device i.

$U_i(N_1,\ldots,N_R)$: utilization of device i, when the system contains $N_1,\ldots,N_R$ jobs of classes 1 through R respectively.

$U_{ir}(N_1,\ldots,N_R)$: utilization of device i by class r jobs, when there are $N_1,\ldots,N_R$ jobs of classes 1 through R, respectively, in the system.

$X_o(N_1,\ldots,N_R)$: system throughput when the system contains $N_1,\ldots, N_R$ jobs of classes 1 through R respectively.

$X_{or}(N_1,\ldots, N_R)$: throughput of class r jobs when the system contains $N_1,\ldots, N_R$ jobs of classes 1 through R respectively.

$Q_{ir}(N_1,\ldots,N_R)$: average number of class r jobs present at device i when there are $N_1,\ldots N_R$ jobs of classes 1 through R, respectively, in the system.

5.1 - Marginal Proportion of Time

The proportion of time that $\ell_{i1},\ldots,\ell_{iR}$ jobs of classes 1 through R are present at device i is given by

$$p_i(\ell_{i1},\ldots,\ell_{iR}) = \sum_{\underline{n}\in S_i^1(N,M,R)} p(\underline{n}) \qquad (20)$$

where

$$S_i^1(N,M,R) = \{\underline{n}=(n_{11},n_{12},\ldots,n_{1R},n_{21},\ldots,n_{2R},\ldots,n_{MR}) \mid \sum_{j=1}^{M} n_{jr}=N_r \;\&\; \sum_{r=1}^{R} N_r = N \;\&\; n_{jr}\geq 0 \quad \text{for } j\neq i \;\&\; n_{ir}=\ell_{ir} \quad \forall_{j,r}\}$$

The summation of equation (18) can be calculated using generating functions. In this case the generating function of device i, f_i, can only have a term in $T_1^{\ell_{i1}}\ldots T_R^{\ell_{iR}}$.

If we consider Leibniz formula [15]

$$(a+b+c+\ldots+\ell)^n = \sum_{\alpha+\beta+\ldots+\lambda=n} \frac{n!}{(\alpha!\beta!\ldots\lambda!)} a^{\alpha}b^{\beta}\ldots \ell^{\lambda}$$

one can see that the generating function for device i must be equal to

$$f_i^*(T_r,\ldots, T_R) = \frac{(\ell_{il}+\ldots+\ell_{iR})!}{\ell_{il}!\ldots\ell_{iR}!} (Y_{il}\ T_l)^{\ell_{il}} \ldots (Y_{iR}\ T_R)^{\ell_{iR}} \tag{21}$$

Now, the generating function to calculate $p_i(\ell_{il},\ldots,\ell_{iR})$ is given by

$$h(T_1,\ldots, T_R) = \frac{g(T_1,\ldots, T_R)}{f_i(T_1,\ldots, T_R)} \times f_i^*(T_1,\ldots, T_R) \tag{22}$$

The summation in (18) can now be easily obtained by dividing the coefficient of $T_1^{N_1} \ldots T_R^{N_R}$ in $h(T_1,\ldots, T_R)$ by $G_M(N_1,\ldots, N_R)$. Using (20), (21) and (22) we get

$$p_i(\ell_{il},\ldots,\ell_{iR}) = \frac{(\ell_{il}+\ldots+\ell_{iR})!}{\ell_{il}!\ldots\ell_{iR}!} Y_{il}^{\ell_{il}} \ldots Y_{iR}^{\ell_{iR}} \times$$

$$\times \frac{[G_M(N_1-\ell_{il},\ldots,N_R-\ell_{iR}) - \sum_{r=1}^{R} Y_{ir}\ G_M(N_1-\ell_{il},\ldots,N_r-\ell_{ir}-1,\ldots,N_R-\ell_{iR})]}{G_M(N_1,\ldots, N_R)} \tag{23}$$

5.2 - Utilization

The utilization of device i is given by the expression below,

$$U_i(N_1,\ldots, N_R) = 1 - p_i(n_{il},\ldots, n_{iR}) \tag{24}$$

where $n_{il} = \ldots = n_{iR} = 0$.

Using expression (23) in (24) we get

$$U_i(N_1,\ldots,N_R) = \frac{\sum_{r=1}^{R} Y_{ir}\ G_M(N_1,\ldots,N_r-1,\ldots, N_R)}{G_M(N_1,\ldots, N_R)} \tag{25}$$

The utilization of device i by class r jobs is defined as

$$U_{ir}(N_1,\ldots,N_R)=\sum_{n_{i1}=0}^{N_1}\cdots\sum_{n_{ir}=1}^{N_r}\cdots\sum_{n_{iR}=0}^{N_R}\frac{n_{ir}}{n_{i1}+\ldots+n_{iR}}\,p_i(n_{i1},\ldots,n_{iR}) \tag{26}$$

Replacing (23) in (26) and after considerable algebraic manipulation, it follows that

$$U_{ir}(N_1,\ldots,N_R)=\frac{Y_{ir}\,G_M(N_1,\ldots,N_r-1,\ldots,N_R)}{G_M(N_1,\ldots,N_R)} \tag{27}$$

5.3 - Throughput

From (6) we know that $V_{ir} = X_{ir}/X_{or}$ But since $X_{ir} = U_{ir}/S_{ir}$ it follows that

$$X_{or} = \frac{X_{ir}}{V_{ir}} = \frac{U_{ir}}{V_{ir}\,S_{ir}} = \frac{U_{ir}}{Y_{ir}} \tag{28}$$

Using (27) in (28) we get

$$X_{or}(N_1,\ldots,N_R) = \frac{G_M(N_1,\ldots,N_r-1,\ldots,N_R)}{G_M(N_1,\ldots,N_R)} \tag{29}$$

The total system throughput, X_o, is the sum of the system throughputs for all classes. Hence,

$$X_o = \frac{\sum_{r=1}^{R} G_M(N_1,\ldots,N_r-1,\ldots,N_R)}{G_M(N_1,\ldots,N_M)} \tag{30}$$

5.4 - Average Queue Length

The average queue length at device i, or equivalently the average number of jobs at device i (including those getting service from the device) is given by the expression bellow,

$$Q_{ir}(N_1,\ldots,N_R) = \sum_{n_{i1}=0}^{N_1}\cdots\sum_{n_{ir}=0}^{N_r}\cdots\sum_{n_{iR}=0}^{N_R} n_{ir}\,p_i(n_{i1},\ldots,n_{ir},\ldots,n_{iR}) \tag{31}$$

where $p_i(n_{i1},\ldots,n_{ir},\ldots,n_{iR})$ is calculated from the expression

in (23). After considerable manipulation of expression (31) it follows that

$$Q_{ir}(N_1,\ldots,N_R) = \frac{1}{G_M(N_1,\ldots,N_R)} \sum_{n_1=0}^{N_1} \cdots \sum_{n_r=1}^{N_r} \cdots \sum_{n_R=0}^{N_R} \frac{(n_1+\ldots+n_R)!}{n_1!\ldots n_R!} \times$$

$$\times \frac{n_r}{n_1+\ldots+n_R} \times Y_{i1}^{n_1}\ldots Y_{iR}^{n_R}\, G_M(N_1-n_1,\ldots,N_R-n_R) \qquad (32)$$

6. Varying the Degree of Multiprogramming per Job Class

The central server model [4], which has been extensively used as a model of multiprogrammed computer systems, has a severe limitation in the fact that the total number of jobs in the system is fixed. This limitation extends to all closed queueing network models. It turns out that in actual systems the number of jobs multiprogrammed in each job class is not fixed but varies dinamically depending on the class nature and mainly on the amount of memory available for each class during the period of observation.

As shown in section 5, performance measures can be efficiently calculated as a function of the normalization constant for a fixed configuration $(N_1,\ldots, N_R)$ where N_i is the number of class i jobs in the system. The procedure introduced in this section allows one to obtain performance measures for each class when several different configurations exist during the observation period. In other words, we do not require anymore that the number of class i jobs be fixed, but we allow it to vary from 0 up to N_i.

A condensed description of the algorithm for class composition will be given below. Sections 6.1 through 6.4 present the complete an formal specification of the algorithm. Some notation must be introduced first. Let $\underline{C} = (n_1,\ldots,n_R)$ be a configuration, where n_i $(1\leq i\leq R)$ is the number of class i jobs in the system. During the observation period, several configurations $\underline{C}$ may be observed. The subconfiguration $\underline{C}^r$ is defined as $(n_1,\ldots,n_r)$. $\underline{C}^o$ is the null configuration and

$\underline{C}^R = \underline{C}$. Now, let $\underline{C}_+^r$ be defined as the complementary sub-configuration $(n_{r+1},\ldots,n_R)$. Thus, $\underline{C}_+^R$ is the null configuration and $\underline{C}_+^o = \underline{C}$. $\underline{C}$ may also be written as $(\underline{C}^r,\underline{C}_+^r)$. The notation $X_{or}(*)$ will be used to denote the throughput of class r jobs for configuration *. Let p(*) denote the fraction of time that configuration * occurred. The goal of the algorithm is to obtain the throughput of class r $(1\le r\le R)$ jobs averaged over all configurations observed during the observation period. Consider the following Algol-like description of the algorithm.

```
for r:=R step -1 until 1 do
    begin
        for all C^r do
            X_or(C^r) := Σ_{C_+^r} X_or(C^r, C_+^r) p(C_+^r);
        for all C^r = (C^{r-1}, n_r) do
            calculate p_r(n_r | C^{r-1}) in terms of X_or(C^r);
            {these calculations depend on the type of class r
             (e.g. batch, timesharing, transaction)}
    end
{ at this point we have X_01(C^1) and p_1(C^1)}
for r:=2 to R do
begin
    p_r(n_r) := Σ_{C^{r-1}} p_r(n_r | C^{r-1}) p(C^{r-1}) ;
    X_or(n_r) := Σ_{C^{r-1}} X_or(C^{r-1}, n_r) p(C^{r-1})
end
{calculate average throughput per class}
for r:= 1 to R do
    X_or := Σ_{n_r=0}^{N_r} X_or(n_r) p_r(n_r) ;
```

6.1 - Batch Processing Class

The batch processing (BP) class is modeled assuming a continuous backlog situation, i.e. as soon as a job of this class leaves, it is substituted right away by another identical job. Therefore, the number of batch jobs in the central sub-

system (see figure 3) oscillates between two consecutive integers N_L and N_H. The average number of jobs in the system is N.

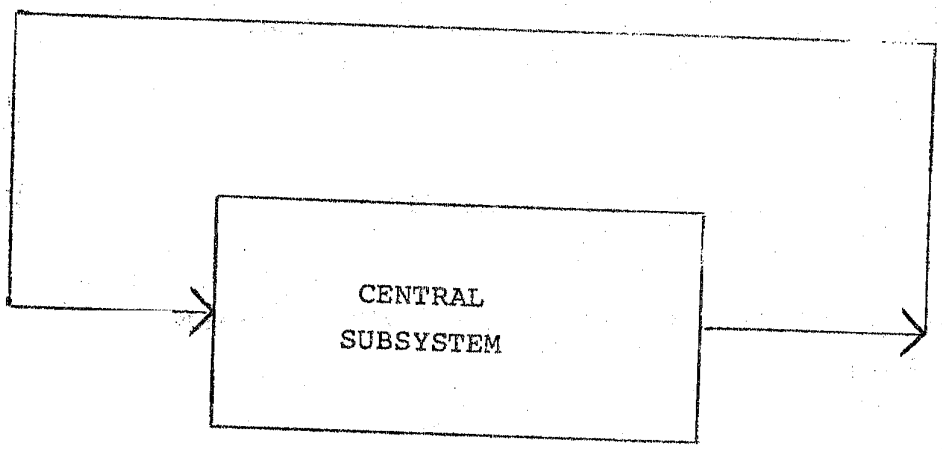

Figure 3 - Batch Processing

Let,

N: average number of batch jobs in the central subsystem.

$X_o(N)$: throughput of the batch class when there are N jobs of this class in the system.

R: average response time of batch jobs.

N_L: greatest integer smaller than N. ($\lfloor N \rfloor$)

N_H: smallest integer greater than N. ($\lceil N \rceil$)

$p(\ell)$: fraction of time that there are ℓ jobs in the system.

$I(\ell)$: amount of time during which there are ℓ jobs in the system.

If N is an integer, the average response time of batch jobs can be obtained from Little's Law [16](1). Thus

$$R = \frac{N}{X_o(N)} \qquad (33)$$

(1) The operational counterpart of Little's Law was proved by Buzen in [7].

Let us now examine the case where N is not an integer.

From the definitions of N, N_L and N_H and from our continuous backlog assumption, it follows that,

$$p(N_L) + p(N_H) = 1 \tag{34}$$

$$\frac{N_L\, I(N_L) + N_H\, I(N_H)}{T} = N \tag{35}$$

If we observe that $p(\ell) = I(\ell)/T$, expression (35) can be rewritten as

$$N_L\, p(N_L) + N_H\, p(N_H) = N \tag{36}$$

From (34) and (36) it follows that

$$p(N_L) = N_H - N \tag{37}$$

$$p(N_H) = N - N_L \tag{38}$$

The average throughput is given by

$$X_o(N) = p(N_L)X(N_L) + p(N_H)X(N_H) \tag{39}$$

and the average response time of batch jobs can be obtained by applying Little's Result.

Hence,

$$R = \frac{N}{X(N)} \tag{40}$$

Since N_L and N_H are integers, the throughputs $X(N_L)$ and $X(N_H)$ can be obtained from equation (29), i.e. through the normalization constant. Notice therefore that formula (39) allows us to obtain the throughput when the number of jobs in the system is not an integer.

6.2 - Timesharing

Figure 4 shows a computer system devoted exclusively to the processing of timesharing jobs.

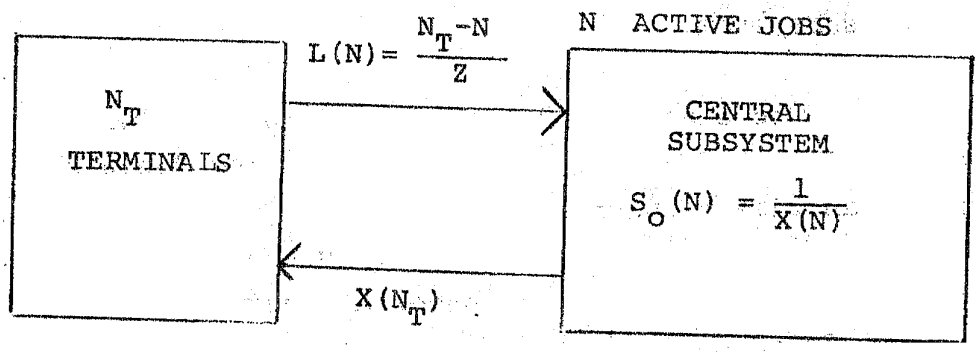

Figure 4 - Timesharing Class

Let;

Z : average think time (i.e., average time between the system response to a user request and the submission of a new request)

N_T: number of terminals

J : maximum level of multiprogramming (i.e. maximum number of active timesharing jobs in the central subsystem)

N : current number of active jobs in the system.

$X_o(N)$: throughput of the timesharing class when there are N jobs in memory.

L(N) : average arrival rate of timesharing jobs when there are N jobs in the central subsystem.

$\bar{N}_M$: average number of jobs in main memory

$\bar{N}_Q$: average number of jobs waiting for memory

$\bar{N}_S$: average number of jobs in the central subsystem. $\bar{N}_S=\bar{N}_Q+\bar{N}_M$.

p(N) : fraction of time that there are N jobs in the central subsystem.

The analysis of the timesharing case will be lone considering the central subsystem as a load dependent server with mean time between service completions equal to $S_o(N) = 1/X(N)$. Notice that X(N) is the throughput of the central subsystem obtained, under a constant load of N jobs, i.e. studying the offline behavior of the central subsystem. This kind of approach is the decomposition principle introduced by Courtois in [17]. This principle allows the analyst to replace a subsystem by a single state dependent server. The service rate of this server is determined by studying the subsystem in isolation. Little error is obtained with this approximation if the rate at which transitions occur within the subsystem is much greater than the rate at which the subsystem interacts with the rest of the system. However, as observed by Denning and Buzen in [9], a theorem proved by Chandy et al [18] asserts that if the product form solution holds for $p(\underline{n})$

then the replacement of the central subsystem by an equivalent server with service rate determined offline yields an exact result.

Let us consider the central subsystem in more detail as in figure 5. If there are N jobs in the central subsystem then min(N,J) jobs are in memory. These jobs may be in one of three different states: ready (R), executing (E) and suspended (S). If there are more than J jobs in the central subsystem, then there will be (N-J) jobs waiting for memory. The central subsystem may now be viewed as being formed of a server and a queue. The throughput of this server is equal to $X_o(N)$ if $N \leq J$ and $X_o(J)$ if $N > J$.

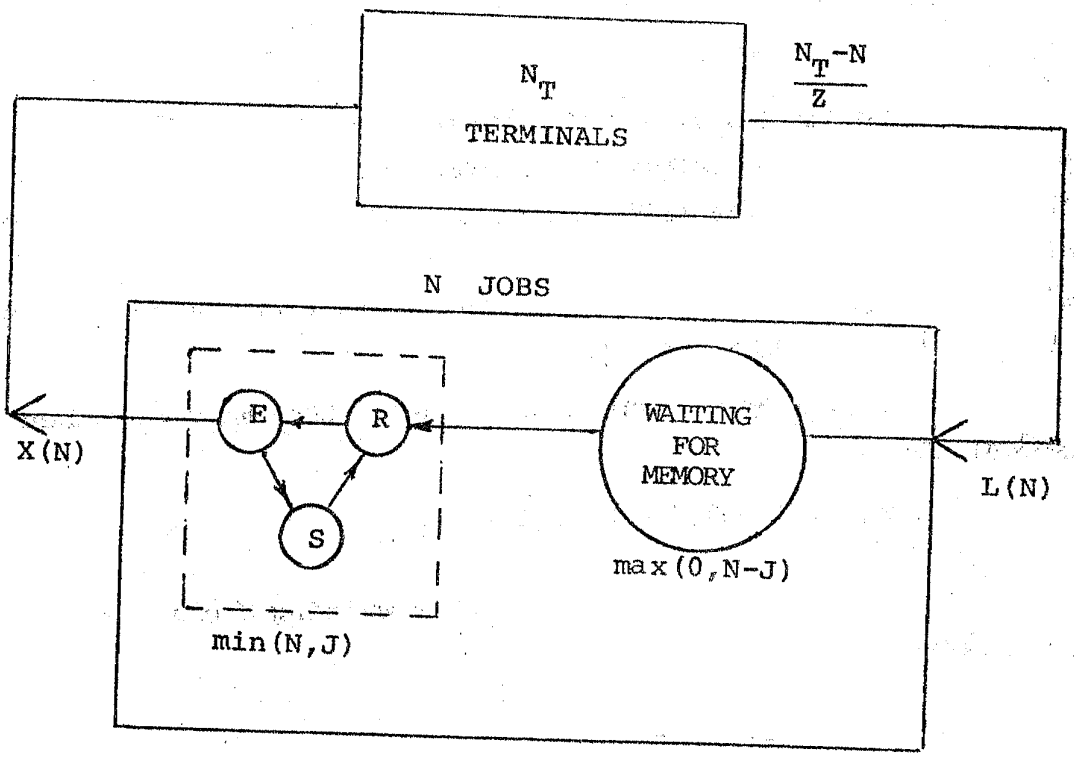

Figure 5. Detail of Central Subsystem

Let us write balance equations for the fraction of time, p(N), that there are N jobs in the central subsystem. If one assumes that the system is in operational equilibrium and if one assumes single arrivals and departures [8] one can

obtain the balance equations given below, following an argument similar to the one presented by Buzen [8] to derive the General Birth-Death Formula.

$$p(N) = \frac{N_T - N + 1}{Z} \cdot \frac{1}{X_o(N)} \cdot p(N-1)$$

$$\text{for} \quad N = 1, \ldots, J \tag{41}$$

$$p(N) = \frac{N_T - N + 1}{Z} \cdot \frac{1}{X_o(J)} \cdot p(N-1)$$

$$\text{for} \quad N = J+1, \ldots, N_T \tag{42}$$

We also have the normalization equation

$$\sum_{i=0}^{N_T} p(i) = 1 \tag{43}$$

Setting $p(0)=1$ we obtain a first set of values of $p(N)$ for $1 \leq N \leq N_T$ recursively from equations (41) and (42). In order to get the true values of $p(N)$ one must divide each of the values obtained by the normalization constant $C = \sum_{i=0}^{N_T} p(i)$, where the $p(i)$'s of the previous summation are the ones obtained in the first place.

The average number of jobs in the memory is then calculated as

$$\bar{N}_M = \sum_{i=1}^{J-1} i p(i) + J \sum_{i=J}^{N_T} p(i) \tag{44}$$

The average number of jobs waiting for memory may be calculated as

$$\bar{N}_Q = \sum_{i=1}^{N_T - J} i \, p_Q(i) \tag{45}$$

where $p_Q(i)$ is the fraction of time that there are i jobs waiting for memory. The values of $p_Q(i)$ are obtained directly in terms of the values of $p(N)$ as follows.

$$p_Q(i) = p(i+J) \quad \text{for} \quad i = 1, \ldots, N_T - J \tag{46}$$

$$p_Q(0) = \sum_{i=0}^{J} p(i) \tag{47}$$

The average number of jobs in the system is now given by

$$\bar{N}_S = \bar{N}_M + \bar{N}_Q \qquad (48)$$

Notice, that while $\bar{N}_S$ could be obtained directly as $\sum_{i=0}^{N_T} i\ p(i)$, the analysis above yields several interesting memory utilization reports which include the average degree of multiprogramming ($\bar{N}_M$) and the distribution of jobs in execution and waiting for memory.

Let $X(N_T)$ be the central subsystem throughput when there are N_T terminals.

From the values of $p(N)$ one can calculate $X(N_T)$ as

$$X(N_T) = \sum_{N=0}^{N_T} X_o(N)\ p(N) \qquad (49)$$

Job flow balance implies that

$$X_o(N) = \frac{N_T - N}{Z} \qquad (50)$$

Replacing (50) in (49) it follows that

$$X(N_T) = \sum_{N=0}^{N_T} \frac{N_T - N}{Z}\ p(N) = \frac{N_T}{Z} - \frac{1}{Z} \sum_{N=0}^{N_T} Np(N) = \frac{N_T - \bar{N}_S}{Z} \qquad (51)$$

Finally, applying Little's law we get the response time, R, of timesharing jobs.

$$R = \frac{\bar{N}_S}{X(N_T)} \qquad (52)$$

6.3 - Transaction Processing Class

Figure 6 presents a computer system subject to a load of transaction type jobs which arrive from terminals at an average arrival rate of L transactions per second.

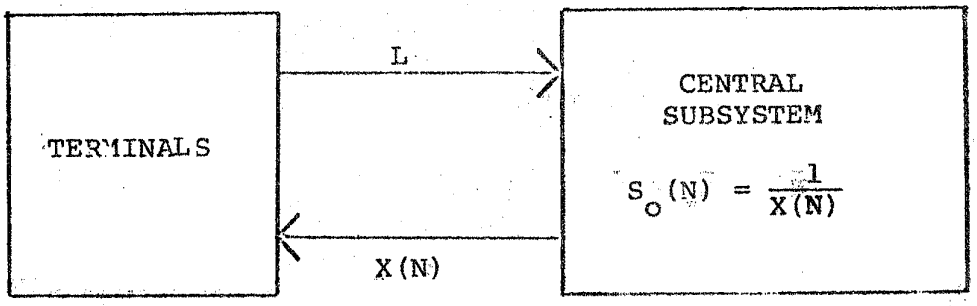

Figure 6 - Transaction Processing Class

Let,

L: average arrival rate of transaction type jobs.

J: maximum level of multiprogramming for the transaction class

$X_o(N)$: throughput of the transaction class when there are N transaction type jobs in main memory.

N: current number of active jobs in the system

$\bar{N}_M$, $\bar{N}_Q$ and $\bar{N}_S$: as defined in section 6.2.

p(N) : fraction of time that there are N jobs in the central subsystem.

Similarly to the analysis of the timesharing class done in section 6.2 we are going to consider the central subsystem as a load dependent server with mean time between service completions equal to $S_o(N) = 1/X(N)$. The principle of decomposition is used again in this analysis. Let us consider the central subsystem in more detail, as shown is figure 7. There may be up to J jobs in memory. The remaining jobs must wait for memory. Therefore, the central subsystem is again viewed as a server and a queue. The throughput of this server is $X_o(N)$ if $N \leq J$ and $X_o(J)$ if $N > J$.

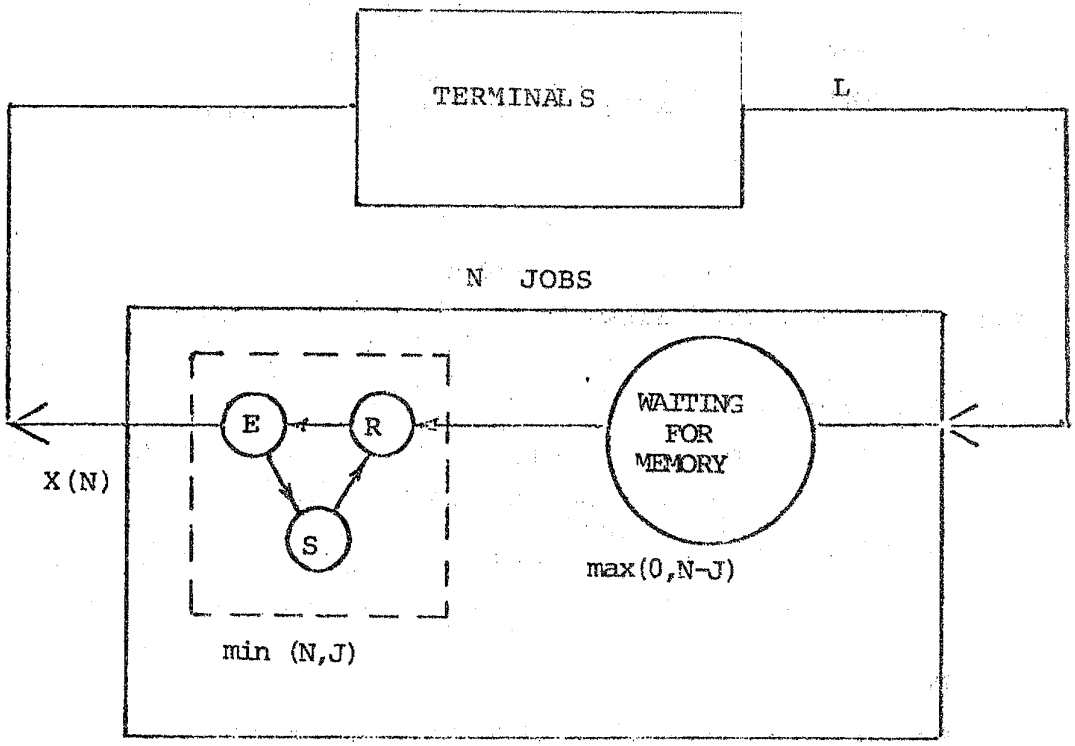

Figure 7 - Detail of Central Subsystem

Assuming that the system is in operational equilibrium and assuming single arrivals and departures one can derive balance equations for p(n) in a way similar to Buzen in [8] . Let N^* be the maximum number of jobs in the central subsystem during the observation period. Then

$$p(n) = \frac{L}{X_o(n)}\, p(n-1) \qquad n=1,\ldots,J \tag{53}$$

$$p(n) = \frac{L}{X_o(J)}\, p(n-1) \qquad n=J,\ldots,N^* \tag{54}$$

The normalization equation is

$$\sum_{n=0}^{J-1} p(n) + \sum_{n=J}^{N^*} p(n) = 1 \tag{55}$$

From (54) we can obtain the value of p(n) as a function of p(J) for $n \geq J$.

$$p(n) = \left(\frac{L}{X_o(J)}\right)^{n-J} p(J) \qquad n=J,\ldots,N^* \tag{56}$$

If N^* is large and $L<X_o(J)$ we can write[1]

$$\sum_{n=J}^{N^*} p(n) = \sum_{n=J}^{N^*} \left(\frac{L}{X_o(J)}\right)^{n-J} p(J) \simeq \frac{p(J)}{1-\frac{L}{X_o(J)}} \qquad (57)$$

We are now ready to find the true values of $p(n)$ for any value of n as follows.

1. Set $p(o)=1$
2. Calculate $p(n)$, for $n=1,\ldots,J$, iteratively using (53).
3. Use the values obtained in step 2 to calculate

$$C = \sum_{n=0}^{J-1} p(n) + \frac{p(J)}{1-\frac{L}{X_o(J)}}$$

4. Divide the values of $p(n)$ obtained in step 2 by C. These are the true values of $p(n)$ for $n=1,\ldots,J$. The true value of $p(o)$ is $1/C$.
5. The true value of $p(n)$ for $n>J$ is obtained from expression (56) where $p(J)$ is the one obtained in step 4.

The average number of jobs in memory, $\bar{N}_M$, is given by

$$\bar{N}_M = \sum_{i=1}^{J-1} ip(i) + J\sum_{i\geq J} p(i) \qquad (58)$$

Using (57) in (58) we obtain

$$\bar{N}_M = \sum_{i=1}^{J-1} ip(i) + \frac{Jp(J)}{1-\frac{L}{X_o(J)}} \qquad (59)$$

The average number of jobs waiting for memory, $\bar{N}_Q$, is obtained by

$$\bar{N}_Q = \sum_{i\geq 1} ip_Q(i) \qquad (60)$$

where $p_Q(i)$ is the fraction of time that there are $\underline{i}$ jobs waiting for memory. Since $p_Q(i) = p(i+J)$ for $i\geq 1$, it follows that

(1) More precisely we require that $\left[\frac{L}{X_o(J)}\right]^{N^*-J+1} << 1$.

$$\bar{N}_Q = \sum_{i\geq 1} ip(i+J) = \sum_{i\geq 1} i \left[\frac{L}{X_o(J)}\right]^i p(J) =$$

$$= p(J) \frac{L}{X(J)} \frac{\partial}{\partial\left(\frac{L}{X_o(J)}\right)} \sum_{i\geq 1} \left[\frac{L}{X_o(J)}\right]^i = p(J) \frac{\frac{L}{X_o(J)}}{\left[1-\frac{L}{X_o(J)}\right]^2} \qquad (61)$$

Notice that the distribution of jobs in memory is easily obtained from $p(n)$ as follows.

$$p_Q(0) = \sum_{i=0}^{J} p(i)$$

$$p_Q(i) = p(i+J) \quad \text{for} \quad i\geq 1$$

The average number of jobs in the central subsystem, $\bar{N}_S$, is given by

$$\bar{N}_S = \bar{N}_M + \bar{N}_Q$$

From Little's law we obtain the average response time of transaction type jobs, as

$$R = \frac{\bar{N}_S}{L} \qquad (62)$$

6.4 - Composition of Job Classes

As mentioned already, queueing network theory literature contains algorithms to analyze queueing networks with multiple classes. However, these results require that the number of jobs in the system be fixed. This section presents an extension to the operational analysis of queueing networks that allows the performance analyst to deal with models which contain multiple classes of jobs and where the number of jobs in each class may vary in a given range, determined by the maximum degree of multiprogramming of each class. Therefore, the number of jobs in the system is not fixed anymore. This is achieved by the class composition algorithm given below.

First we introduce the new notation used in this section.

Then, we present the algorithm followed by a set of explanatory comments about the crucial steps. The reader is advised to follow the algorithm and the comments in parallel.

Let,

$p_r(n_r|n_{r-1},\ldots,n_1)$: fraction of time that there are n_r class r jobs in the central subsystem given that there are $n_{r-1},\ldots,n_1$ jobs in classes r-1,..., 1 respectively.

$p_r(n_r)$: fraction of time that there are n_r class r jobs in the central subsystem, independently of the number of jobs in other classes.

$X^+_{or}(n_1,\ldots,n_r)$: throughput of class r jobs when there are n_1 class 1 jobs,...,n_r class r jobs, independently of how many jobs there are of classes r+1 through R. For r=R, $X^+_{or}(n_1,\ldots,n_R) = X_{or}(n_1,\ldots,n_R)$.

$X_{or}(n_r)$: throughput of class r jobs when there n_r jobs of this class in the central subsystem, independently of the number of jobs in the other classes.

We assume that any class may be of one of the three types defined in section two, namely batch, timesharing and transaction. It should be noted however that there may be any number of classes of a given type.

Consider the graphical representation of classes given in figure 8. This example will be used to elucidate the class composition algorithm. There are three classes of jobs with multiprogramming levels varying between 0 and 1, 0 and 2, and 0 and 3 for classes 1 through 3 respectively. Notice that a path from the root to a leaf of the tree shown in figure 8 represents one of the possible configurations, and the tree represents all possible configurations.

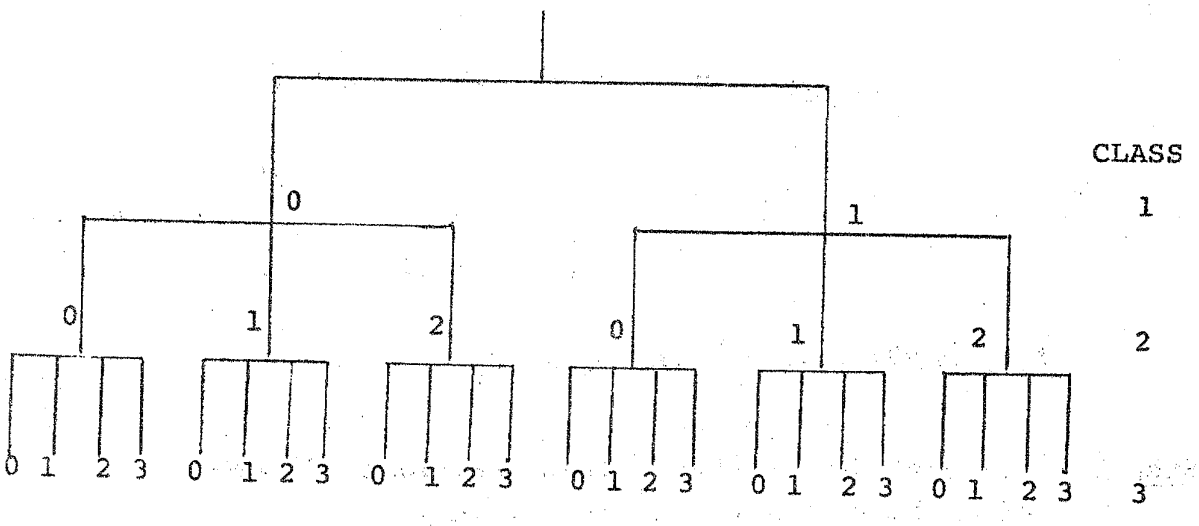

Figure 8 - Graphical Representation of three classes.

Class Composition Algorithm

Step 1 - Set r=R

Step 2 - [calculation of class r throughput]. Determine the class r throughput for every possible configuration $(n_1,\ldots,n_R)$, according to the formula given below and derived in section 5.3, i.e., calculate

$$X_{or}(n_1,\ldots,n_r,\ldots,n_R) = \frac{G_M(n_1,\ldots,n_r-1,\ldots,n_R)}{G_M(n_1,\ldots,n_R)}$$

for $n_1=0,\ldots,N_1$; $n_2=0,\ldots,N_2;\ldots;$ $n_R=0,\ldots.N_R$

Step 3 - [step 4 must be skipped for class R jobs] If r=R then go to step 5.

Step 4 - [calculation of class r throughput independent of higher classes] Determine the class r throughput for every possible configuration $(n_1,\ldots,n_r)$ independently of the number of jobs in classes r+1 through R, i.e., calculate

$$X^{+}_{or}(n_1,\ldots,n_r) = \sum_{n_R=0}^{N_R}\cdots\sum_{n_{r+1}=0}^{N_{r+1}} X_{or}(n_1,\ldots,n_R) \times$$

$$p_R(n_R|n_{R-1},\ldots,n_r,\ldots,n_1) \times$$

$$\ldots\times$$

$$p_{r+1}(n_{r+1}|n_r,\ldots,n_1)$$

for $n_1=0,\ldots,N_1;\ldots;n_r=0,\ldots,N_r$

Step 5 - [calculation of conditional fractions of time]
Using the values of $X^{+}_{or}(n_1,\ldots,n_r)$ obtained in the previous step calculate the probability $p_r(n_r|n_{r-1},\ldots,n_1)$ for $n_1=0,\ldots,N_1;\ldots;n_r=0,\ldots,N_r$. These probabilities are calculated accordingly to the methods described in sections 6.1 through 6.3 for each type of class.

Step 6 - Set $r=r-1$. If $r\geq 1$ then go to step 2.

Step 7 - [Obtaining the unconditional fractions of time and throughputs] Calculate the unconditional fraction of time $p_r(n_r)$ for $r=2,\ldots,R$ and for $n_r=0,\ldots,N_r$, as follows.

$$p_r(n_r) = \sum_{n_1=0}^{N_1}\cdots\sum_{n_{r-1}=0}^{N_{r-1}} p_r(n_r|n_{r-1},\ldots,n_1) \times$$

$$\times\, p_{r-1}(n_{r-1}|n_{r-2},\ldots,n_1)\times\ldots\times p_2(n_2|n_1)\times p_1(n_1)$$

Calculate the unconditional throughput $X_{or}(n_r)$ for $r=2,\ldots,R$ and for $n_r=0,\ldots,N_r$ as follows.

$$X_{or}(n_r) = \sum_{n_1=0}^{N_1}\cdots\sum_{n_{r-1}=0}^{N_{r-1}} X^{+}_{or}(n_r,\ldots,n_1) \times$$

$$p_{r-1}(n_{r-1}|n_{r-2},\ldots,n_1)\times\ldots\times p_2(n_2|n_1)\times p_1(n_1)$$

Step 8 - [Obtaining the average throughput for each class, over all observed configurations]
Calculate

$$X_{or} = \sum_{n_r=0}^{N_r} X_{or}(n_r)p_r(n_r)$$

for $r=1,\ldots,R$.

Comments on the Class Composition Algorithm

Comment on Step 2: In the example of figure 7, one would calculate $X_{or}(0,0,0)$, $X_{or}(1,0,0),\ldots,$ $X_{or}(3,2,1)$.

Comment on Step 4: Let us first derive the expression given in Step 4 for $X^{+}_{or}(n_1,\ldots,n_r)$. Some definitions are in order. Let

$C_{or}(n_1,\ldots,n_R)$: number of class r system completions when there are n_1 class 1 jobs,...,n_R class R jobs.

$C^{+}_{or}(n_1,\ldots,n_r)$: number of class r system completions when there n_1 class 1 jobs,..., n_r class r jobs.

$I(n_1,\ldots,n_R)$: amount of time during which there n_1 class 1 jobs,...,n_R class R jobs.

$I_r(n_1,\ldots,n_r)$: amount of time during which there are n_1 class 1 jobs,...,n_r class r jobs.

From the above definitions we may write the obvious relationships below.

$$X^{+}_{or}(n_1,\ldots,n_r) = \frac{C^{+}_{or}(n_1,\ldots,n_r)}{I_r(n_1,\ldots,n_r)} \tag{63}$$

$$X_{or}(n_1,\ldots,n_R) = \frac{C_{or}(n_1,\ldots,n_R)}{I(n_1,\ldots,n_R)} \tag{64}$$

$$C^{+}_{or}(n_1,\ldots,n_r) = \sum_{n_R=0}^{N_R}\cdots\sum_{n_{r+1}=0}^{N_{r+1}} C_{or}(n_1,\ldots,n_R) \tag{65}$$

Using (64) in (65) it follows that

$$C^{+}_{or}(n_1,\ldots,n_r) = \sum_{n_R=0}^{N_R}\cdots\sum_{n_{r+1}=0}^{N_{r+1}} X_{or}(n_1,\ldots,n_R)I(n_1,\ldots,n_R) \tag{66}$$

Dividing both sides of (66) by $I_r(n_1,\ldots,n_r)$ and using (66) we get

$$X^+_{or}(n_1,\ldots,n_r) = \sum_{n_R=0}^{N_R} \cdots \sum_{n_{r+1}=0}^{N_{r+1}} X_{or}(n_1,\ldots,n_R) \times \times \frac{I(n_1,\ldots,n_R)}{I_r(n_1,\ldots,n_r)} \tag{67}$$

Rewriting the ratio $I(N_1,\ldots,n_R)/I_r(n_1,\ldots,n_r)$ in the form below, we get

$$\frac{I(n_1,\ldots,n_R)}{I_r(n_1,\ldots,n_r)} = \frac{I(n_1,\ldots,n_R)}{I_{R-1}(n_1,\ldots,n_{R-1})} \times \frac{I_{R-1}(n_1,\ldots,n_{R-1})}{I_{R-2}(n_1,\ldots,n_{R-2})} \times\ldots\times \times \frac{I_{r+1}(n_1,\ldots,n_{r+1})}{I_r(n_1,\ldots,n_r)} \tag{68}$$

If we use the fact that

$$p_r(n_r|n_{r-1},\ldots,n_1) = \frac{I_r(n_1,\ldots,n_r)}{I_{r-1}(n_1,\ldots,n_{r-1})}$$

in (68) and (67) we get the expression for $X^+_{or}(n_1,\ldots,n_r)$ given in Step 4.

In the example of figure 7, for r=2 one would calculate $X^+_{02}(0,0)$, $X^+_{02}(1,0),\ldots,X^+_{02}(2,1)$ using the values of $X_{02}(0,0,0)$, $X_{02}(1,0,0),\ldots,X_{02}(3,2,1)$ obtained in step 2 and the values of $p_3(0|0,0),\ldots,p_3(3|2,1)$ obtained in Step 5 for r=3.

Comment on Step 5 : Assume that class 2 is of type timesharing. Then, the balance equations for $p_2(n_2|n_1)$ would be given by

$$p_2(n_2|n_1) = \frac{N_2-n_2+1}{Z} \times \frac{1}{X^+_{02}(n_2,n_1)} \times p_2(n_2-1|n_1)$$

for $n_2=1,\ldots,J$

and

$$p_2(n_2|n_1) = \frac{N_2-n_2+1}{Z} \times \frac{1}{X^+_{02}(J,n_1)} \times p_2(n_2-1|n_1)$$

for $n_2=J+1,\ldots,N_2$

accordingly to equations (41) and (42).

Comment on Step 7: The expression for $p_r(n_r)$ follows directly from the operational counterpart of the theorem of total probability [19], which we call. theorem of total fraction of time. We state and prove this theorem below.

Theorem of Total Fraction of Time: Let S be a system configuration and let $S_i=1,\ldots,n$ be system configurations such that

i) if the system is in configuration S, it must be in exactly one of configurations S_i

ii) if the system is in configuration S_i it must not be at configuration S_j for $j\neq i$.

iii) the system must be in configuration S_i for an amount of time greater than zero, for every $i=1,\ldots,n$

Then

$$p(S) = \sum_{i=1}^{n} p(S|S_i)p(S_i)$$

where $p(S|S_i)$ is the fraction of time that the system is observed in configuration S given that it is in configuration S_i.

Proof: By definition,

$$p(S|S_i) = \frac{I(SS_i)}{I(S_i)} \qquad (69)$$

where $I(SS_i)$ is the total amount of time during which the system is in configuration S and S_i, and $I(S_i)$ is the total amount of time during which the system is in configuration S_i.

If we observe that

$$I(S) = \sum_{i=1}^{n} I(SS_i) \qquad (70)$$

and if we use (69) in (70) it follows that

$$I(S) = \sum_{i=1} p(S|S_i)\ I(S_i) \qquad (71)$$

Dividing both sides of (71) by T it follows that

$$p(S) = \sum_{i=1}^{n} p(S|S_i)p(S_i)$$

and the theorem is proved.

From the definition of $p(S|S_i)$ it follows that

$$p(S|S_i) = \frac{I(SS_i)/T}{I(S_i)/T} = \frac{p(SS_i)}{p(S_i)} \qquad (72)$$

Now, from the theorem just proved we have that

$$p_r(n_r) = \sum_{n_1=0}^{N_1} \cdots \sum_{n_{r-1}=0}^{N_{r-1}} p_r(n_r|n_{r-1},\ldots,n_1)p(n_{r-1},\ldots,n_1) \qquad (73)$$

From (72) it follows that

$$p(n_{r-1},\ldots,n_1) = p_{r-1}(n_{r-1}|n_{r-2},\ldots,n_1)\times p\ (n_{r-2},\ldots,n_1) =$$

$$= p_{r-1}(n_{r-1}|n_{r-2},\ldots,n_1)\times p_{r-2}(n_{r-2}|n_{r-3},\ldots,n_1)\times p(n_{r-3},\ldots,n_1) =$$

$$= \ldots = p_{r-1}(n_{r-1}|n_{r-2},\ldots,n_1)\times\ldots\times p_2(n_2|n_1)\times p_1(n_1) \qquad (74)$$

Finally, using (74) in (73) we get the expression for $p_r(n_r)$ used in step 7.

In order to derive the expression for $X_{or}(n_r)$ let us first state the theorem below.

Theorem of Total Throughput: Let S and S_i for $i=1,\ldots,n$ be system configurations as defined for the theorem of total fraction of time.

Then,

$$X(S) = \sum_{i=1}^{n} X(S|S_i)p(S_i)$$

where $X(S|S_i)$ is the throughput of the system when it is in configuration S given that it is in configuration S_i, and $X(S)$ is the throughput when the system is in configuration S.

Proof: By definition $X(S) = C(S)/I(S)$ where $C(S)$ is the number of completions when the system is in configuration S. Also, by definition

$$X(S|S_i) = \frac{C(SS_i)}{I(S_i)} \tag{75}$$

If we observe that

$$C(S) = \sum_{i=1}^{n} C(SS_i) \tag{76}$$

and if we use (75) in (76) it follows that

$$C(S) = \sum_{i=1}^{n} X(S|S_i) I(S_i) \tag{77}$$

Dividing both sides of (77) by $I(S)$ the theorem is proved.

The expression for $X_{or}(n_r)$ used in step 7 is obtained as a direct consequence of the theorem of total throughput and of expression (74).

7. OPERATIONAL VERSUS STOCHASTIC APPROACH

The results contained in this paper are phrased in terms of operational analysis. Stochastic counterparts for these results can be obtained provided certain assumptions are made. The term "fraction of time" should be replaced by probability throughout the paper. The results of sections 4 and 5 are valid for stochastic queueing networks which have a product form solution such as the one in (9). The class composition algorithm of section 6 is valid in the stochastic approach.

8. CONCLUSIONS

A multiclass queueing network model of computer systems was presented in this paper. This model extends queueing network theory in the sense that it allows the performance analyst to treat the case where the degree of multiprogramming for each job class is not fixed, but is allowed to vary within a specified range. Although the results contained here are given an operational analysis treatment, it is shown how one could obtain stochastic counterparts of most of the results and algorithms in this paper. The techniques suggested here were used to write a computer program to analyze multiprogrammed computer systems. The results obtained from the program showed a remarkable accuracy when compared with actual measurements.

REFERENCES

1. Jackson, J.R. "Networks of Waiting Lines", Operations Research, 5, 518-521, 1957.

2. Jackson, J.R., "Jobshop-Like Queueing Systems", Management Science, 10, No.1, 131-142, 1963.

3. Gordon, W.J. and G.F. Newell, "Closed Queueing Systems with Exponential Servers", Operations Research, 15, 254-265, 1967.

4. Buzen, J.P., "Queueing Network Models of Multiprogramming", Ph.D. Thesis, Div. of Engineering and Applied Science, Harvard Univ., Cambridge, Mass., May 1971 (NTIS AD 731 575, Aug. 1971).

5. Buzen, J.P., "Computational Algorithms for Closed Queueing Networks with Exponential Servers", CACM, 16, 527-531, 1973.

6. Baskett, F., K.M. Chandy, R.R. Muntz, and F. Palacios - Gomez, "Open, Closed, and Mixed Networks of Queues with Different Classes of Customers", JACM, 22, 248-260, 1975.

7. Buzen, J.P., "Fundamental Operational Laws of Computer System Performance", Acta Informatica, 7, 2, 167-182 , 1976.

8. Buzen, J.P., "Operational Analysis : the key to the New Generation of Performance Prediction Tools", Proceedings of the IEEE COMPCON, IEEE, New York, 1976.

9. Denning, P.J., and J.P. Buzen, "Operational Analysis of Queueing Networks", Measuring, Modelling and Evaluating Computer Systems, eds. H. Beilher and E. Gelenbe, North-Holland Publishing Company, 151-172, 1977.

10. Denning, P.J. and J.P. Buzen, "The Operational Analysis of Queueing Network Models", ACM Computing Surveys, 10,3, 225-261, 1978.

11. Bouhana, J., "Operational Aspects of Centralized Queueing Networks", Ph.D. Thesis, Computer Science Dept., Univ. of Wisconsin, Madison, January 1978.

12. Roode, J.D., "Multiclass Operational Analysis of Queueing Networks Models", Performance of Computer Systems, North Holland Publishing Company, 339-352, 1979.

13. Balbo, G., S. C. Bruell, and H.D. Schwetman, "Customer Classes and Closed Network Models - A Solution Technique", Information Processing 1977, North Holland Publishing Company, 1977.

14. Williams, A.C. and R.A. Bhandiwad, "A Generating Function Approach to Queueing Network Analysis of Multiprogrammed Computers", Networks, 6, 1-22, 1976.

15. Knuth, D., "The Art of Computer Programming", Vol. I., Addison Wesley, 1979.

16. Little, J.D.C., "A Proof of the Queueing Formula $L=\lambda W$" , Operations Research, 9, 383-387, 961.

17. Courtois, P.J., "Decomposability, Instabilities and Saturation in Multiprogrammed Systems", CACM, 18, 7, 371-377, 1975.

18. Chandy, K.M., U.Herzog, and L. Woo, "Parametric Analysis of

Queueing Networks", IBM Journal of Research and Development, 1, 36-42, January 1975.

19. Ash, Robert B., Basic Probability Theory, John Wiley & Sons, New York, 1970.