Chapter 7

Models with Multiple Job Classes

7.1. Introduction

Multiple class models, like single class models, provide estimates for performance measures such as utilization, throughput, and response time. The advantages of multiple class models over single class models include:

- Outputs are given in terms of the individual customer classes. For example, in modelling a transaction processing system, response times for each of a number of transaction types could be obtained by including each type as a separate class. With a single class model, only a single estimate for response time representing the average over all transaction types could be obtained.
- For systems in which the jobs being modelled have significantly different behaviors, such as systems with a mixture of CPU and I/O bound jobs, a multiple class model can provide more accurate results. This means that some systems can be modelled adequately only by multiple class models, since the single class assumption that jobs are indistinguishable is unacceptable.

The disadvantages of multiple class models relative to single class models include:

- Since there are multiple customer classes in the model, multiple sets of input parameters (one set per class) are required. The data gathering portion of the modelling process therefore is more tedious.
- Most current measurement tools do not provide sufficient information to determine the input parameters appropriate to each customer class with the same accuracy as can be done for single class models. This not only complicates the process of parameterization, but also means that the potentially greater accuracy of a multiple class model can be offset by inaccurate inputs.
- Multiple class solution techniques are somewhat more difficult to implement, and require more machine resources, than single class techniques.

For the most part, these disadvantages result from inadequate modelling support software, and thus should become less significant as queueing network modelling becomes more widespread. The first two disadvantages can be eliminated by measurement tools that are designed with knowledge of the information required to establish a model. The third disadvantage is significant only if one is developing queueing network modelling software. Commercially available software packages are capable of evaluating multiple class models. Thus, once the model inputs have been obtained, it is no more difficult to deal with a multiple class model than with a single class model.

7.2. Workload Representation

As illustrated in Chapter 4, the inputs of multiple class models largely correspond to those of single class models. The major additional consideration is the specification of scheduling disciplines. Since customers in single class models are indistinguishable, the scheduling disciplines at the various service centers are characterized entirely as being either delay or queueing. However, in multiple class models, customers are distinguishable, and so the choice of scheduling discipline can be important.

There are a large number of scheduling disciplines that can be represented in (separable) multiple class queueing network models. For practical purposes, however, the following disciplines have proven to be sufficient:

- *first-come-first-served (FCFS)* — Under FCFS scheduling, customers are served in the order in which they arrive. Although this is the simplest of scheduling disciplines to implement, it is difficult to model analytically. To do so, it is necessary to impose the restriction that all customer classes have the same service requirement at each visit to the service center in question ($S_{c,k}$). It is possible, however, for different customer classes to require different total numbers of visits to the service center ($V_{c,k}$), thus providing for distinct service demands there ($D_{c,k}$). A FCFS center might be appropriate to represent a disk containing user files for a number of classes. Since the basic operations performed at the device by the various classes are the same, it is reasonable to assume that the average service times across classes are nearly equal. The actual number of file accesses for a customer of each class can be represented in the model by appropriate values of the $V_{c,k}$ for each class c.

- *processor sharing (PS)* — Processor sharing is an idealization of round robin (RR) scheduling. Under RR, control of the processor circulates among all jobs in the queue. Each job receives a *quantum* of service before it must relinquish control to the next job in the queue, rejoining the queue at its tail. Under PS, the length of the quantum is effectively zero, so that control of the processor circulates infinitely rapidly among all jobs. The effect is that jobs are served simultaneously, but each of the n jobs in service receives only $1/n$-th of the full power of the processor. For example, each of three jobs at a processor shared, 3 MIPS (million instructions per second) CPU would receive service at a rate of 1 MIPS. PS often is appropriate to model CPU scheduling in systems where some form of RR scheduling actually is employed.
- *last-come-first-served preemptive-resume (LCFS)* — Under this discipline an arriving job preempts the job in service (if any) and immediately begins service itself. When a job completion occurs, the most recently preempted job resumes service at the point at which it was interrupted. LCFS might be used to model a CPU in a system where the frequency with which high priority system tasks are dispatched is high enough that LCFS is a reasonable approximation.
- *delay* — As in single class models, multiple class delay centers are used to represent devices at which residence time consists entirely of service (there is no queueing delay).

Although the first three disciplines seem quite different, the performance measures obtained from a model will be the same regardless of which is used. In most cases, we therefore distinguish only between *queueing* and *delay* disciplines, without being more specific.

7.3. Case Studies

In this section we present three simple case studies where multiple class separable queueing network models were used to obtain performance projections. The first examines the difference in performance projections provided by single and multiple class models. The second illustrates the principal advantage of multiple class models over single class models, namely the ability to specify inputs and outputs in terms of individual classes. The third demonstrates the successful use of a multiple class model to evaluate a loosely-coupled multiprocessor system.

7.3.1. Contrast with Single Class Models

In this case study we will construct single class and multiple class models of a hypothetical system, and will use these models to project the effects on response times of a CPU upgrade. Our purpose is to illustrate the qualitative differences between the projections that can be obtained from single and multiple class models.

The hypothetical system has two resources, a CPU and a disk. There are two workload components, one batch and the other interactive. Measurements provide the following information (times are in seconds):

$B_{batch,CPU} = 600$	$B_{interactive,CPU} = 47.6$
$B_{batch,Disk} = 54$	$B_{interactive,Disk} = 428.4$
$C_{batch} = 600$	$C_{interactive} = 476$
$N_{batch} = 10$	$N_{interactive} = 25$
$Z_{batch} = 0$	$Z_{interactive} = 30$

To construct a single class model of this system, we define a single "average" customer class, in essence by imagining that the measurement data did not distinguish on the basis of workload type. Our model will have two service centers (*CPU* and *Disk*) and a single, terminal class. This class will have 35 customers with think times of 13.271 seconds ($\frac{476}{600+476} \times 30$). Service demands will be .602 seconds at the CPU ($\frac{600+47.6}{1076}$) and .448 seconds at the disk ($\frac{54+428.4}{1076}$).

The multiple class model will have two service centers and two classes: a batch class of 10 customers, and a terminal class of 25 customers with think times of 30 seconds. Batch service demands will be 1.0 and .09 seconds at the CPU and disk, respectively. Interactive service demands will be .10 and .90 seconds at the CPU and disk, respectively.

Table 7.1 shows the outputs for the single class and multiple class models, for the base system and for an upgraded system in which the CPU speed is increased by a factor of five. The single and multiple class models agree well for the base system. They differ considerably for the system with the CPU upgrade, however, even when the projections of the single class model are compared to the "overall" projections of the multiple class model. For example, the multiple class model shows an overall throughput of 5.26 for the system with the upgraded CPU, compared with 2.11 for the single class model. Further, while the single class model projects a 60% improvement in average response time, the multiple class model projects an 80% improvement for batch jobs, but a 200% *degradation* for interactive users.

These differences can be accounted for by the nature of the workload. In the single class model, each "average" job requires a significant

	single class	
	overall	
	base	upgrade
X	1.64	2.11
R	8.07	3.32
U_{CPU}	.985	.254
Q_{CPU}	10.70	.34
U_{Disk}	.733	.946
Q_{Disk}	2.58	6.63

	multiple class					
	overall		batch		interactive	
	base	upgrade	base	upgrade	base	upgrade
X	1.66	5.26	.93	4.64	.74	.62
R	7.52	3.16	10.79	2.16	3.40	10.57
U_{CPU}	1.000	.943	.926	.928	.074	.015
Q_{CPU}	10.57	5.28	9.72	5.20	.85	.08
U_{Disk}	.752	.979	.084	.418	.668	.561
Q_{Disk}	2.37	11.02	.28	4.80	2.09	6.22

Table 7.1 — Single and Multiple Class Results

amount of disk processing, and so the speedup of the CPU has a limited effect due to the performance constraint imposed by this secondary bottleneck. In the multiple class model, the batch class is heavily CPU bound, while the interactive class is heavily I/O bound. Thus, increasing the speed of the CPU greatly increases the batch throughput but is of little direct benefit to the interactive class. Further, because of the increased batch throughput, the interactive class suffers increased competition from the batch class at the disk center, and thus experiences a performance degradation.

In summary, this example illustrates two important points regarding the use of queueing network models:

- A model can project effects that intuition might not recognize. In this case, we have the counter-intuitive result that performance can degrade with a CPU upgrade.
- Single class models of systems with significantly heterogeneous workloads may give misleading results, both because the performance projections for the "average" job may be inaccurate, and because it is not possible to obtain projections for specific classes from the average results.

7.3.2. Modelling Workload Growth

The system studied here was a Digital Equipment Corporation PDP-10 running a special-purpose software package layered on the TOPS-10 operating system. The objective of the study was to project response times as the number of online users increased and as the number of users that simultaneously could be memory resident was altered. Although benchmarking using a remote terminal emulator (RTE) was possible, a queueing network modelling approach was chosen. This decision was motivated by the fact that projections for a large number of system configurations were required, and timely results with rough (say, 30%) accuracy were more desirable than the more accurate but considerably more time consuming results possible using benchmarking.

The system workload was divided into three components, primarily on the basis of similarity of resource usage. The first component consisted of users running jobs, the second of users executing system utility functions (such as printing or plotting), and the third of users editing. All classes were represented as terminal workloads. Service demands for these three classes were obtained by monitoring an RTE experiment involving a representative (although synthetic) jobstream. This base model was validated by comparing model outputs with measurements taken during the RTE experiment. Agreement was good, so the study proceeded to the projection phase.

	class	model		actual	
		R	U_{CPU}	R	U_{CPU}
Benchmark 1 50 users	running jobs	9.97		10.91	
	utility	122.8		99.27	
	editing	63.4		77.8	
	total		63.4		77.8
Benchmark 2 70 users	running jobs	9.2		11.7	
	utility	63.6		70.3	
	editing	1.83		2.03	
	total		97.5		100.0

Table 7.2 – Performance Projections

To assess the impact of workload growth on response times, the workload intensities of the three classes were increased to reflect various larger user populations. The model then was evaluated to obtain performance projections. For several specific user populations, additional RTE experiments were conducted to assess the accuracy of the model. Table 7.2 compares the model results with those obtained during RTE experiments for two user populations. The accuracy is reasonably good, despite the

extremely simple model used. (Response time improves as the user population increases because of an increase in main memory size that was represented in the model and implemented in the actual configuration. This additional memory resulted in reduced swapping. Techniques for modelling swapping are presented in Chapter 9.)

7.3.3. A Multiprocessing System

The configuration under consideration consisted of two Cyber 173 systems with private memories and disk subsystems, plus a set of shared disks supporting a Federated File System (FFS). The Cyber systems were used both to process local workloads and to process FFS requests from remote sites. The purpose of the study was to assess the impact of an expected growth in the batch components of the systems' workloads. Figure 7.1 shows the model that was employed.

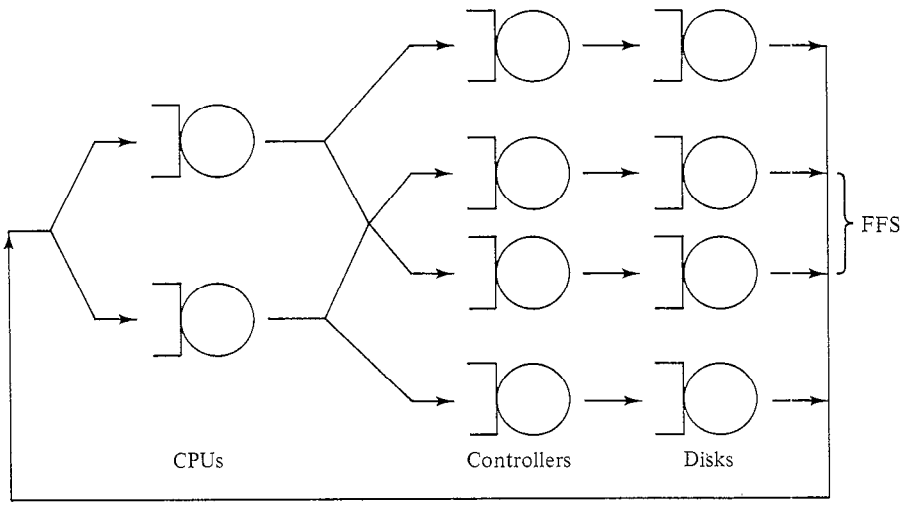


Figure 7.1 – The Multiprocessing System Model

Measurements obtained from software monitors were used to parameterize the model. Service demands were calculated for five workload components: system A interactive, system A batch, system B interactive, system B batch, and FFS accesses by remote systems. All workload components initially were represented using transaction classes, with the FFS arrivals split evenly between systems A and B. An attempt at validating this model showed reasonably accurate throughputs and utilizations, but poor estimates for queue lengths and response times. It was observed that the model projected that on average thirteen jobs would be active simultaneously in each system. However, it was known that

system limitations permitted a maximum of five memory resident jobs. Because of this, the batch and interactive workload components of each system were converted to classes of batch type in the model, with workload intensities corresponding to the measured multiprogramming levels. This change resulted in nearly identical throughputs and utilizations and improved device residence time estimates, and so was adopted as the "validated" model. (This study points out the possible danger in using simple models with transaction classes when studying systems that have memory constraints. A more satisfactory model for memory constrained systems is presented in Chapter 9.)

The increase in the batch workloads was represented by increasing the workload intensities of the corresponding classes in the model, with all other parameters remaining unchanged. These were adjusted so that the estimated model throughput of batch jobs matched the anticipated offered workload. Response time estimates from this model were obtained as indications of the ability of the systems to handle the increased workload. It was projected that the systems would be able to handle the maximum expected batch volumes and still provide adequate interactive and FFS response times.

7.4. Solution Techniques

The solution techniques for multiple class models yield values for performance measures such as utilization, throughput, response time, and queue length, for each individual customer class. These techniques are natural extensions of the single class solution techniques. As in the single class case, the details of the solution technique depend on the types of the workloads (open or closed). This dictates the organization of our discussion.

7.4.1. Open Model Solution Technique

Let C be the number of classes in the model. Each class c is an open class with arrival rate λ_c. We denote the vector of arrival rates by $\vec{\lambda} \equiv (\lambda_1, \lambda_2, \ldots, \lambda_C)$. Because the throughputs of the classes in open models are part of the input specification, the solution technique for these models is quite simple. We list below the formulae to calculate performance measures of interest.

- *processing capacity*

  A system is said to have sufficient capacity to process a given offered load $\vec{\lambda}$ if it is capable of doing so when subjected to the workload over

a long period of time. For multiple class models, sufficient capacity exists if the following inequality is satisfied:

$$\max_k \left\{ \sum_{c=1}^{C} \lambda_c D_{c,k} \right\} < 1$$

This simply ensures that no service center is saturated as a result of the combined loads of all the classes. In the derivations that follow, we will assume that this inequality is satisfied.

- *throughput*

  By the forced flow law the throughput of class c at center k as a function of $\vec{\lambda}$ is:

$$X_{c,k}(\vec{\lambda}) = \lambda_c V_{c,k}$$

- *utilization*

  From the utilization law:

$$U_{c,k}(\vec{\lambda}) = X_{c,k}(\vec{\lambda}) S_{c,k} = \lambda_c D_{c,k}$$

- *residence time*

  As with single class models, residence time is given by:

$$R_{c,k}(\vec{\lambda}) = \begin{cases} D_{c,k} & \text{(delay centers)} \\ D_{c,k}\left[1 + A_{c,k}(\vec{\lambda})\right] & \text{(queueing centers)} \end{cases}$$

  where $A_{c,k}(\vec{\lambda})$ is the average number of customers seen at center k by an arriving class c customer. The intuition behind this formula is similar to that for single class models. For delay centers, a job's residence time consists entirely of its service demand there, $V_{c,k}S_{c,k}$. The explanation of the formula for queueing centers depends on the scheduling discipline used. For FCFS centers, the residence time is simply the sum of an arriving job's own service time, $V_{c,k}S_{c,k}$, and the service times of the jobs already present at the arrival instant, $V_{c,k}\left[A_{c,k}(\vec{\lambda})S_{c,k}\right]$, since at FCFS centers all classes must have the same service time at each visit. For PS centers, the residence time is the basic service requirement, $V_{c,k}S_{c,k}$, "inflated" by a factor representing the service rate degradation due to other jobs competing in the same queue, $1+A_{c,k}(\vec{\lambda})$. For LCFS centers the equation has no simple intuitive explanation, but nonetheless is valid.

An implication of the assumptions made in constructing separable networks is that the queue length seen on average by an arriving customer must be equal to the time averaged queue length. Thus, for queueing centers:

$$R_{c,k}(\vec{\lambda}) = D_{c,k}\left[1+Q_k(\vec{\lambda})\right]$$

where $Q_k(\vec{\lambda})$ is the time averaged queue length at center k (the sum over all classes). Applying Little's law:

$$R_{c,k}(\vec{\lambda}) = D_{c,k}\left[1+\sum_{j=1}^{C}\lambda_j R_{j,k}(\vec{\lambda})\right]$$

Notice now that the right hand side of the above equation depends on the particular class c only for the basic service demand $D_{c,k}$. Thus, $\dfrac{R_{c,k}(\vec{\lambda})}{R_{j,k}(\vec{\lambda})}$ must equal $\dfrac{D_{c,k}}{D_{j,k}}$, giving $R_{j,k}(\vec{\lambda}) = \dfrac{D_{j,k}}{D_{c,k}} R_{c,k}(\vec{\lambda})$. Substituting into the equation above and re-writing, we have:

$$R_{c,k}(\vec{\lambda}) = \frac{D_{c,k}}{1-\sum_{j=1}^{C} U_{j,k}(\vec{\lambda})} \qquad \text{(queueing centers)}$$

- *queue length*

  Applying Little's law to the residence time equation above, the queue length of class c at center k, $Q_{c,k}(\vec{\lambda})$, is:

$$Q_{c,k}(\vec{\lambda}) = \lambda_c R_{c,k}(\vec{\lambda})$$

$$= \begin{cases} U_{c,k}(\vec{\lambda}) & \text{(delay centers)} \\ \dfrac{U_{c,k}(\vec{\lambda})}{1-\sum_{j=1}^{C} U_{j,k}(\vec{\lambda})} & \text{(queueing centers)} \end{cases}$$

- *system response time*

  The response time for a class c customer, $R_c(\vec{\lambda})$, is the sum of its residence times at all devices:

$$R_c(\vec{\lambda}) = \sum_{k=1}^{K} R_{c,k}(\vec{\lambda})$$

- *average number in system*

  The average number of class c customers in system can be calculated using Little's law, or by summing the class c queue lengths at all centers:

$$Q_c(\vec{\lambda}) = \lambda_c R_c(\vec{\lambda}) = \sum_{k=1}^{K} Q_{c,k}(\vec{\lambda})$$

These formulae are summarized as Algorithm 7.1.

processing capacity : $\max_k \left\{ \sum_{c=1}^{C} \lambda_c D_{c,k} \right\} < 1$

throughput : $X_c(\vec{\lambda}) = \lambda_c$

utilization : $U_{c,k}(\vec{\lambda}) = \lambda_c D_{c,k}$

residence time :

$$R_{c,k}(\vec{\lambda}) = \begin{cases} D_{c,k} & \text{(delay)} \\ \dfrac{D_{c,k}}{1 - \sum_{j=1}^{C} U_{j,k}(\vec{\lambda})} & \text{(queueing)} \end{cases}$$

queue length : $Q_{c,k}(\vec{\lambda}) = \lambda_c R_{c,k}(\vec{\lambda})$

$$= \begin{cases} U_{c,k}(\vec{\lambda}) & \text{(delay)} \\ \dfrac{U_{c,k}(\vec{\lambda})}{1 - \sum_{j=1}^{C} U_{j,k}(\vec{\lambda})} & \text{(queueing)} \end{cases}$$

system response time : $R_c(\vec{\lambda}) = \sum_{k=1}^{K} R_{c,k}(\vec{\lambda})$

average number in system : $Q_c(\vec{\lambda}) = \lambda_c R_c(\vec{\lambda}) = \sum_{k=1}^{K} Q_{c,k}(\vec{\lambda})$

Algorithm 7.1 – Open Model Solution Technique

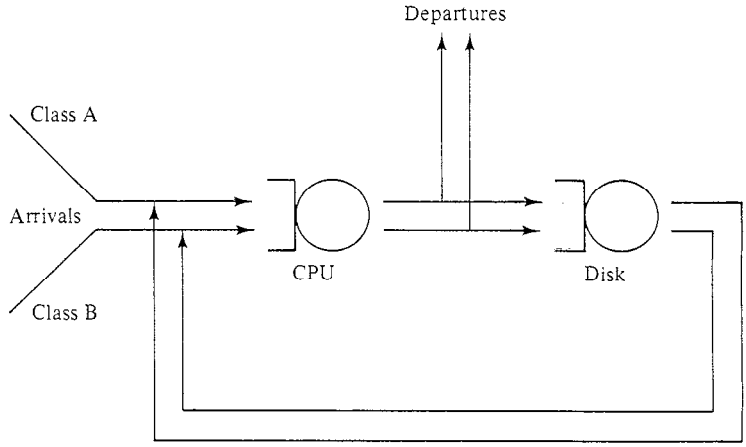
Open Model Example

Figure 7.2 shows a simple open model with two customer classes and two service centers, and illustrates the calculation of various performance measures. (All times are in seconds.)

Model Inputs:

$$V_{A,CPU} = 10 \quad V_{A,Disk} = 9 \quad V_{B,CPU} = 5 \quad V_{B,Disk} = 4$$
$$S_{A,CPU} = 1/10 \quad S_{A,Disk} = 1/3 \quad S_{B,CPU} = 2/5 \quad S_{B,Disk} = 1$$
$$D_{A,CPU} = 1 \quad D_{A,Disk} = 3 \quad D_{B,CPU} = 2 \quad D_{B,Disk} = 4$$
$$\lambda_A = 3/19 \text{ jobs/sec.} \qquad \lambda_B = 2/19 \text{ jobs/sec.}$$

Model Structure:



Selected Model Outputs:

$$X_{A,CPU}(\vec{\lambda}) = \lambda_A V_{A,CPU} = \frac{3}{19} \times 10 = 1.58 \text{ jobs/sec.}$$

$$U_{A,CPU}(\vec{\lambda}) = \lambda_A D_{A,CPU} = \frac{3}{19} \times 1 = .158$$

$$R_{A,CPU}(\vec{\lambda}) = \frac{D_{A,CPU}}{1 - \sum_{j=A}^{B} U_{j,CPU}(\vec{\lambda})} = \frac{1}{12/19} = 1.58 \text{ secs.}$$

$$Q_{A,CPU}(\vec{\lambda}) = \frac{U_{A,CPU}(\vec{\lambda})}{1 - \sum_{j=A}^{B} U_{j,CPU}(\vec{\lambda})} = \frac{3/19}{1 - (\frac{3}{19} + \frac{4}{19})} = .25 \text{ jobs}$$

$$R_A(\vec{\lambda}) = R_{A,CPU}(\vec{\lambda}) + R_{A,Disk}(\vec{\lambda}) = \frac{19}{12} + \frac{57}{2} = 30.08 \text{ secs.}$$

Figure 7.2 – Open Model Example

7.4.2. Closed Model Solution Techniques

A closed, multiple class model consists of C classes, each of which has a fixed population. We denote the workload intensity by $\vec{N} \equiv (N_1, \ldots, N_C)$, where N_c is the class c population size. Because the throughputs of closed classes are not provided as inputs, obtaining solutions for closed models is somewhat more complicated than for open models. The solution technique used is an extension of the single class mean value analysis (MVA) algorithm. Like its single class counterpart, multiple class MVA relies on three key equations:

- *For each class, Little's law applied to the queueing network as a whole*

$$X_c(\vec{N}) = \frac{N_c}{Z_c + \sum_{k=1}^{K} R_{c,k}(\vec{N})} \tag{7.1}$$

- *For each class, Little's law applied to the service centers individually*

$$Q_{c,k}(\vec{N}) = X_c(\vec{N})\, R_{c,k}(\vec{N}) \tag{7.2}$$

It also is useful to consider the total queue length at center k:

$$Q_k(\vec{N}) = \sum_{c=1}^{C} Q_{c,k}(\vec{N})$$

- *For each class, the service center residence time equations*

$$R_{c,k}(\vec{N}) = \begin{cases} D_{c,k} & \text{(delay centers)} \\ D_{c,k}\left[1 + A_{c,k}(\vec{N})\right] & \text{(queueing centers)} \end{cases} \tag{7.3}$$

where $A_{c,k}(\vec{N})$ is the arrival instant queue length at center k seen by an arriving class c customer.

We note that performance measures can be computed using the above equations once the $A_{c,k}(\vec{N})$ are known.

As with single class models, there are two approaches to the evaluation of closed models, exact and approximate. (We emphasize again that the word "exact" refers to how the solution relates to the model, not how the solution of the model relates to the system being modelled.) As with the single class MVA algorithms, the two methods differ in how the arrival instant queue lengths are computed.

7.4.2.1. Exact Solution Technique

To obtain an exact solution of a closed model, one must compute the values of the $A_{c,k}(\vec{N})$ exactly. Given these values, equations (7.1)-(7.3) can be applied to compute the full solution of the model. The key to the exact MVA solution technique is the multiple class generalization of the relationship used in the single class case:

$$A_{c,k}(\vec{N}) = Q_k(\overrightarrow{N-1_c}) \tag{7.4}$$

where $\overrightarrow{N-1_c}$ is population $\vec{N}$ with one class c customer removed. Intuitively, the queue length seen upon arrival to a center is equal to the time averaged queue length at the center with the arriving customer removed from the network.

Beginning from the trivial solution of the network with the empty population $\vec{0}$ ($Q_k(\vec{0}) = 0$ for all centers k), equation (7.4) can be used, along with equations (7.1)-(7.3), to construct iteratively the solutions for increasing populations, culminating in performance measures for the population of interest, $\vec{N}$. Note that in general the solution for each population $\vec{n}$ requires as input C solutions, one for each population $\overrightarrow{n-1_c}$, $c = 1, \ldots, C$. Figure 7.3 illustrates this by showing the precedence relations of the solutions required to evaluate a network with 3 class A customers and 2 class B customers: the solution of the empty network is required to compute solutions with populations consisting of a single customer, (1A,0B) and (0A,1B), which then can be used to compute solutions for populations with two customers, etc. As a result of these complex dependencies, the time and space requirements of the multiple class algorithm are significantly greater than those of the single class algorithm. They are proportional to:

$$\text{time:} \quad CK \prod_{c=1}^{C} (N_c+1) \qquad \text{arithmetic operations}$$

$$\text{space:} \quad K \prod_{\substack{c=1 \\ c \neq c_{max}}}^{C} (N_c+1) \qquad \text{storage locations}$$

where c_{max} is the index of the class with the largest population. A significant implication of these time and space requirements is that it can be impractical to compute the exact solution of networks with more than a few customer classes. For example, the solution of a network with 10 centers and 5 classes of 10 customers each requires more than 8,000,000 arithmetic operations and 145,000 storage locations. (In contrast, a single class model with 10 centers and 50 customers requires roughly 500 arithmetic operations and 10 storage locations.) This is the motivation for the approximate solution technique to be described in the next section.

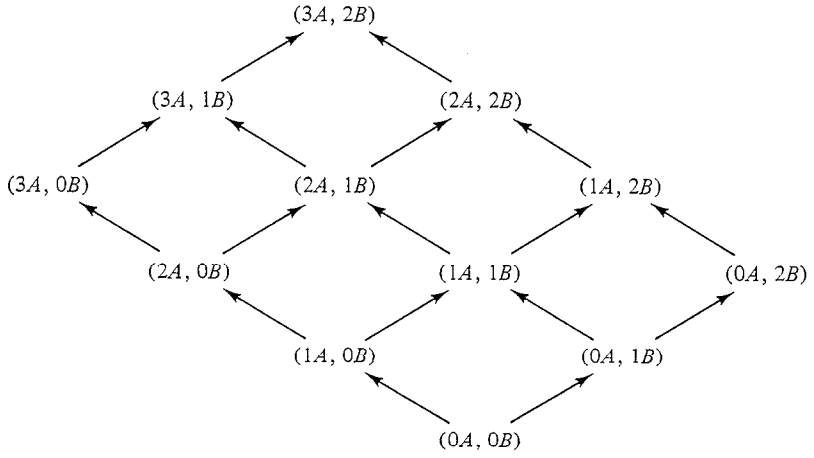


Figure 7.3 – Precedence of Intermediate Solutions

for $k \leftarrow 1$ to K do $Q_k(\vec{0}) \leftarrow 0$

for $n \leftarrow 1$ to $\sum_{c=1}^{C} N_c$ do

for each feasible population $\vec{n} \equiv (n_1, \ldots, n_C)$ with n total customers do

begin

for $c \leftarrow 1$ to C do

for $k \leftarrow 1$ to K do

$$R_{c,k} \leftarrow \begin{cases} D_{c,k} & \text{(delay)} \\ D_{c,k}\left[1 + Q_k(\overrightarrow{n-1_c})\right] & \text{(queueing)} \end{cases}$$

for $c \leftarrow 1$ to C do $X_c \leftarrow \dfrac{n_c}{Z_c + \sum_{k=1}^{K} R_{c,k}}$

for $k \leftarrow 1$ to K do $Q_k(\vec{n}) \leftarrow \sum_{c=1}^{C} X_c R_{c,k}$

end

Algorithm 7.2 – Exact MVA Solution Technique (Closed Models)

The exact MVA solution technique appears as Algorithm 7.2. When this algorithm terminates, the values of $R_{c,k}$, X_c, and Q_k (all for population $\vec{N}$) are available immediately. Other model outputs are obtained by using Little's law. Here is a summary:

class c system throughput:	X_c
class c system response time:	$N_c/X_c - Z_c$
average number of class c in system:	$N_c - X_c Z_c$
class c throughput at device k:	$X_c V_{c,k}$
class c utilization of device k:	$X_c D_{c,k}$
class c queue length at device k:	$X_c R_{c,k}$
class c residence time at device k:	$R_{c,k}$

Closed Model Example (Exact Solution)

Table 7.3 shows the computation required by the MVA solution of a closed model corresponding to the open model of Figure 7.2. The open classes have been replaced by batch classes, each with one customer. Other parameter values are the same.

	population vectors			
	(0A,0B)	(1A,0B)	(0A,1B)	(1A,1B)
$R_{A,CPU}$	-	1	-	4/3
$R_{A,Disk}$	-	3	-	5
$R_{B,CPU}$	-	-	2	5/2
$R_{B,Disk}$	-	-	4	7
X_A	-	1/4	-	3/19
X_B	-	-	1/6	2/19
$Q_{A,CPU}$	0	1/4	-	4/19
$Q_{A,Disk}$	0	3/4	-	15/19
$Q_{B,CPU}$	0	-	1/3	5/19
$Q_{B,Disk}$	0	-	2/3	14/19

Table 7.3 – Exact MVA Computation

7.4.2.2. Approximate Solution Technique

Because the exact solution technique can require excessive time and space for large numbers of classes, the approximate solution technique often is the only one that can be used in practice. Moreover, since the approximate technique is quite accurate, it is useful as a general technique, even for networks that could be solved exactly.

The multiple class approximate solution technique is a straightforward extension of the single class approximation. Equations (7.1)-(7.3) are employed, but the arrival instant queue lengths are estimated iteratively. The estimates are obtained based on the time averaged queue lengths at the service centers with the full customer population. Thus, the approximate solution technique does not require that one first compute solutions for all populations between the zero population and the full population, but instead iterates at the full population. An initial guess for time averaged queue lengths is made to start the iteration. The approximating function is applied to this guess, and the resulting approximate arrival instant queue lengths are used in equation (7.3). Applications of equations (7.2) and (7.1) result in new estimates for time averaged queue lengths, which then can be used to begin the next step of the iteration. The iteration continues until successive estimates of time averaged queue lengths are sufficiently close. The approximate solution technique is summarized as Algorithm 7.3.

1. Set $Q_{c,k}(\vec{N}) \leftarrow \frac{N_c}{K}$ for all c,k.
2. Approximate $A_{c,k}(\vec{N})$ by $h_c\left[Q_{1,k}(\vec{N}), \ldots, Q_{C,k}(\vec{N})\right]$, for all c,k. (The choice of h_c is discussed in the text.)
3. Apply equations (7.1)-(7.3) to compute a new set of $Q_{c,k}(\vec{N})$ for all c,k.
4. If the $Q_{c,k}(\vec{N})$ resulting from Step 3 do not agree to within some tolerance (e.g., 0.1%) with those used as inputs in Step 2, return to Step 2 using the new $Q_{c,k}(\vec{N})$.

Algorithm 7.3 – Approximate MVA Technique (Closed Models)

The significant advantage of this method over the exact technique is that it iterates on solutions of the network with the full customer population $\vec{N}$, rather than building up from the solution for the empty network. The approximation therefore requires much less storage than the exact technique, since it maintains the solution of the network for only one population ($\vec{N}$). In particular, the storage requirement is proportional to the product of C and K. The savings in time are harder to quantify because of the iterative nature of the approximate algorithm, although empirically these savings are considerable. The number of operations required per iteration is proportional to the product of C and K. (In other words, the populations of the classes are not a consideration.) Less than two dozen iterations typically are required for convergence to less than a 0.1% change in queue lengths. The accuracy of the technique

typically is within a few percent of the exact solution for throughputs and utilizations, and within 10% for queue lengths and residence times.

As noted, the approximate solution technique is built upon estimates for the arrival instant queue lengths at each device for each class that depend only on information obtained from the solution of the network with the full population. A particular estimate for the function h_c that has been used successfully is:

$$\begin{aligned} A_{c,k}(\vec{N}) &= Q_k(\overrightarrow{N-1_c}) \\ &\approx h_c\left[Q_{1,k}(\vec{N}), \ldots, Q_{C,k}(\vec{N})\right] \\ &\equiv \left[\frac{N_c-1}{N_c} Q_{c,k}(\vec{N})\right] + \sum_{\substack{j=1 \\ j\neq c}}^{C} Q_{j,k}(\vec{N}) \end{aligned} \tag{7.5}$$

Comparing equation (7.5) to the exact formula (7.4), it is evident that the assumption made in the approximation is that the removal of a customer from the network does not affect the placement of customers in other classes, and reduces queue lengths in its own class in proportion to their original size. Equation (7.5) has worked well in practice. More sophisticated estimates also have been used, although these are somewhat more difficult to implement and require more machine resources, in terms of both time and space.

An important benefit of the approximate technique is that non-integer multiprogramming levels easily are incorporated in the model. One simply sets N_c to the (non-integer) multiprogramming level and applies the approximation. No interpolation between separate integer solutions is required.

Closed Model Example (Approximate Solution)

Table 7.4 shows the intermediate and final values for the example given in Section 7.4.2.1 (Table 7.3), calculated using the approximate solution technique. The iteration was halted when the maximum change in all queue length estimates was less than .001.

7.4.3. Mixed Model Solution Technique

Mixed queueing network models are those in which some classes are open and some are closed. Such models may be constructed, for instance, to model a mixed batch and transaction processing system. We denote the workload intensity vector of the entire model by $\vec{I} \equiv (N_1 \text{ or } \lambda_1, N_2 \text{ or } \lambda_2, \ldots, N_C \text{ or } \lambda_C)$. Mixed models are evaluated using Algorithm 7.4.

iteration	class	performance measure			
		$Q_{c,CPU}$	$Q_{c,Disk}$	X_c	R_c
0	A	.500	.500		
	B	.500	.500		
1	A	.250	.750	.167	6.000
	B	.333	.667	.111	9.000
2	A	.211	.790	.158	6.333
	B	.263	.737	.105	9.500
3	A	.195	.805	.154	6.474
	B	.253	.747	.104	9.579
4	A	.193	.807	.154	6.495
	B	.249	.751	.104	9.610
5	A	.192	.808	.154	6.508
	B	.248	.752	.104	9.614
exact solution	A	.211	.789	.158	6.333
	B	.263	.737	.105	9.500

Table 7.4 – Approximate MVA Computation

An important aspect of queueing phenomena is illustrated by Step 2 of Algorithm 7.4. In that step, the performance measures of the closed classes of a mixed model are computed by creating a model that consists only of closed classes; the open classes have been eliminated. The effect of the open classes on closed class performance measures is represented by "inflating" the service demands of the closed classes at all devices. The "inflation factor" used is $1-U_{\{O\},k}$, which is the percentage of time that the processor is not in use by the open classes. In essence, this factor indicates the effective speed of the processor as seen by the closed classes, given that some of its time is allocated to other (in this case open) classes. For example, if a 3 MIPS (million instructions per second) CPU is utilized 33% by transactions constituting an open class in the model, it appears to be a 2 MIPS CPU to the other classes. Dividing all service demands by .67 to create the closed model of Step 2 simply reflects the fact that more processing time is required on the effectively slower processor. This technique of inflating service times, which often is referred to as *load concealment*, will be used repeatedly in later chapters to reduce the complexity of models by eliminating customer classes while still incorporating their effects on performance.

Let $\{O\}$ be the set of open classes and $\{C\}$ the set of closed classes.

1. For each center k, obtain its utilization by each open class:

$$U_{c,k}(\vec{I}) = \lambda_c D_{c,k} \qquad c \in \{O\}$$

and its total utilization by all open classes:

$$U_{\{O\},k}(\vec{I}) = \sum_{c \in \{O\}} \lambda_c D_{c,k}$$

This simply is an application of the forced flow law and the utilization law to each open class.

2. Solve the closed model consisting of the K centers and the closed customer classes (but no open classes). The service demand $D^*_{c,k}$ of each class $c \in \{C\}$ at each center k in the closed model is set to:

$$D^*_{c,k} = \frac{D_{c,k}}{1 - U_{\{O\},k}(\vec{I})} \qquad c \in \{C\}$$

where $D_{c,k}$ is the service demand of class c at center k in the original mixed model. The throughputs, queue lengths, and residence times obtained from the solution of this model are the performance measures for the corresponding closed classes in the mixed model. Utilizations can be computed by applying the utilization law to the original set of service demands $D_{c,k}$.

3. Residence times and queue lengths for the open classes can be computed using the performance measures of the closed classes:

$$R_{c,k}(\vec{I}) = \frac{D_{c,k}\left[1 + Q_{\{C\},k}(\vec{I})\right]}{1 - U_{\{O\},k}(\vec{I})} \qquad c \in \{O\}$$

$$Q_{c,k}(\vec{I}) = \lambda_c R_{c,k}(\vec{I}) \qquad c \in \{O\}$$

where $Q_{\{C\},k}(\vec{I})$ is the total queue length of all closed classes at center k obtained from the solution of the closed model in Step 2.

Algorithm 7.4 – Exact MVA Solution Technique (Mixed Models)

Mixed Model Example

Figure 7.4 shows a mixed model with four classes and two centers. Classes A and B are open, while classes C and D are closed. As shown in the figure, the solution of the model is obtained in three steps corresponding to those of Algorithm 7.4.

7.5. Theoretical Foundations

As with single class models, certain assumptions about the behavior of a model are necessary to the mathematical proof that the solution obtained by the MVA procedure gives the exact performance measures for that model. With only one exception, the assumptions required in the multiple class case are straightforward extensions of those required in the single class case:

- *service center flow balance* — The number of arrivals of each class at each center is equal to the number of completions of that class there.
- *one step behavior* — Only a single customer can move (arrive to or depart from a service center) at a time.
- *routing homogeneity* — Given a more detailed view of customer behavior that includes the routing patterns of customers, routing homogeneity is satisfied if the proportion of time that a customer of class c leaving center j proceeds directly to center k depends only on c, j, and k, and is independent of the number of customers or their classes currently at any of the centers, for all c, j, and k.
- *device homogeneity* — This is the one assumption whose extension from the single class case is less than straightforward. In the single class case, we allowed the rate of completions of jobs from a center to vary in an arbitrary manner with the number of jobs at that center (although the rate could not otherwise be dependent on the number or placement of customers within the network). In the multiple class case, we do not allow completely arbitrary variation in completion rate as a function of population. Specifically, let n be the total number of customers at center k, n_c be the number of class c customers there, and $\mu_{c,k}(n,n_c)$ be the completion rate of class c customers at center k with those queue lengths. Device homogeneity is satisfied when:

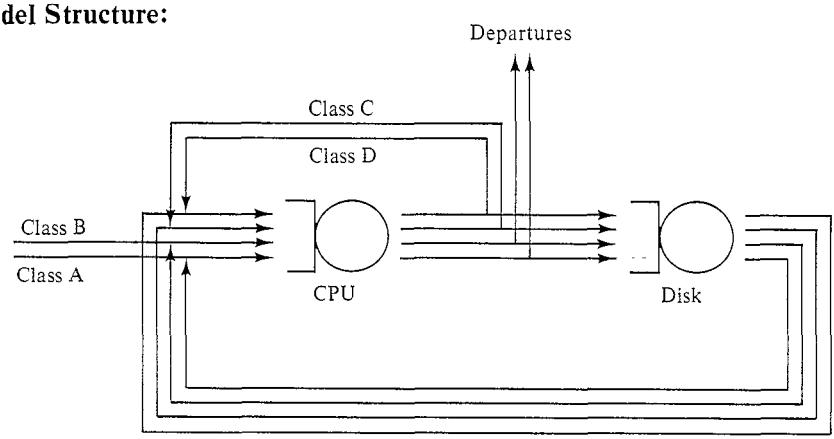
$$\mu_{c,k}(n,n_c) = \frac{n_c}{n}\,\mu_{c,k}(1,1)\;a_k(n)$$

for all c and k, where $a_k(n)$ is a positive constant for fixed k and n. This assumption will be discussed further in Chapter 8.

Model Inputs:

$D_{A,CPU} = 1/4$ $D_{B,CPU} = 1/2$ $D_{C,CPU} = 1/2$ $D_{D,CPU} = 1$
$D_{A,Disk} = 1/6$ $D_{B,Disk} = 1$ $D_{C,Disk} = 1$ $D_{D,Disk} = 4/3$
$\lambda_A = 1$ $\lambda_B = 1/2$ $N_C = 1$ $N_D = 1$

Model Structure:



Evaluation:

1. Compute the total utilization of the devices by the open classes:

$$U_{\{O\},CPU}(\vec{I}) = \lambda_A D_{A,CPU} + \lambda_B D_{B,CPU} = .5$$
$$U_{\{O\},Disk}(\vec{I}) = \lambda_A D_{A,Disk} + \lambda_B D_{B,Disk} = .667$$

2. Solve the closed model obtained by deleting the open classes and inflating the service demands of the closed classes:

$$D^*_{C,CPU} = \frac{.5}{1-.5} = 1 \qquad D^*_{D,CPU} = \frac{1}{1-.5} = 2$$
$$D^*_{C,Disk} = \frac{1}{1-.667} = 3 \qquad D^*_{D,Disk} = \frac{1.333}{1-.667} = 4$$

This model is equivalent to the closed model solved in the example of Section 7.4.2, so the same performance measures will result, e.g., CPU queue lengths are .211 and .789 for classes C and D.

3. Using the queue lengths of the closed classes, compute the performance measures of the open classes. For example:

$$R_{A,CPU} = \frac{.25\ (1+1.0)}{1-.5} = 1.0 \qquad R_{B,CPU} = \frac{.5\ (1+1.0)}{1-.5} = 2.0$$

Figure 7.4 – Example Mixed Network

- *homogeneous external arrivals* — The rate of arrival of customers of each class is independent of the number and class of the customers currently in the system or the placement of those customers.

While these assumptions are sufficient for the model to be separable (and thus to be efficiently evaluated), the solution techniques that have been presented so far require one additional restriction:

- *service time homogeneity* — The completion rate of class c customers at center k times the ratio of the total number of customers at k to the number of class c customers at k is constant for all fixed c and k (i.e., when $a_k(n) = 1$ for all k,n).

This last assumption ensures that all service centers are *load independent*, which means that the rate of service is independent of the current state of the queue at the device. Somewhat more complicated models can be constructed using *load dependent* service centers, whose service rates depend on their queue lengths. These will be discussed in Chapter 8.

7.6. Summary

In this chapter we have focused on multiple class, separable queueing network models. We are interested in separable networks because they are reasonably accurate models of computer systems and can be solved efficiently; more general models require excessively large amounts of time and space. Exact solutions of separable models with a few customer classes, and accurate approximate solutions of models with many customer classes, can be obtained with modest machine resources.

The major advantage of multiple class models over single class models is also the main drawback. By identifying distinct workload components, output performance measures for each can be given separately. At the same time, input parameter values are required for each individual class. This typically requires considerable additional effort over that for a single class model, as measurement tools often do not provide sufficient information about resource consumption by classes.

While certain restrictive assumptions are required to construct separable models, it often is the case that separable models accurately project the behavior of complex computer systems despite these restrictions. In cases where aspects of a computer system important to its performance cannot be represented directly, variations on simple separable models must be used. These variations are the subject of Part III.

7.7. References

Almost all work with multiple class models has been conducted in the stochastic setting. The work of Baskett et al. [1975], which describes separable models for open, closed, and mixed workloads, is probably the most referenced paper in the area of queueing network models. Chandy et al. [1977] describe the stochastic properties required for a network to be separable.

The case studies in Sections 7.3.1, 7.3.2, and 7.3.3 were reported by Denning and Buzen [1978], Sanguinetti and Billington [1980], and Lindzey and Browne [1979], respectively.

[Baskett et al. 1975]
Forest Baskett, K. Mani Chandy, Richard R. Muntz, and Fernando G. Palacios. Open, Closed, and Mixed Networks of Queues with Different Classes of Customers. *JACM 22,2* (April 1975), 248-260.

[Chandy et al. 1977]
K. Mani Chandy, John H. Howard, Jr., and Donald F. Towsley. Product Form and Local Balance in Queueing Networks. *JACM 24,2* (April 1977), 250-263.

[Denning & Buzen 1978]
Peter J. Denning and Jeffrey P. Buzen. The Operational Analysis of Queueing Network Models. *Computing Surveys 10,3* (September 1978), 225-261. Copyright © 1978 by the Association for Computing Machinery.

[Lindzey & Browne 1979]
G.E. Lindzey, Jr. and J.C. Browne. Response Analysis of a Multi-Function System. *Proc. ACM SIGMETRICS Conference on Simulation, Measurement and Modeling of Computer Systems* (1979), 19-26. Copyright © 1979 by the Association for Computing Machinery.

[Sanguinetti & Billington 1980]
John Sanguinetti and Richard Billington. A Multi-Class Queueing Network Model of an Interactive System. *Proc. CMG XI International Conference* (1980), 50-55.

7.8. Exercises

1. What is the principal advantage of multiple class models over single class models? The principal disadvantage?

2. Evaluate the open model example of Figure 7.2 by hand with the following independent changes:
   a. Both arrival rates halved.
   b. $D_{A,CPU}$ doubled.
3. Extend the solution of the closed network shown in Table 7.3 to the case of two class A and two class B customers. Check your results against those obtained using the multiple class, exact MVA implementation in Chapter 19.
4. Construct an "equivalent" single class model to the model of Figure 7.2. Compare the performance measures of the single class model to the aggregate measures of the multiple class model.
5. In evaluating a model with a one transaction and one batch class, the solution technique involves the removal of the transaction class and the "service time inflation" of the batch class. This procedure yields an exact solution.

   Investigate the use of service time inflation to remove a batch class from a model. Consider a model with two batch classes and five centers. Class A has service demands 1, 2, 2, 2, 2 at the five centers, while class B has service demands 3, 1, 1, 1, 1.
   a. Use the software in Chapter 19 to obtain solutions to the model with populations (2 A , 2 B), (2 A , 8 B), and (2 A , 16 B).
   b. For each population $\vec{N}$, construct an approximate model with respect to class A by removing the class B customers from the model, and inflating the class A service demand at each center k by $1 - U_{B,k}(\vec{N})$. Compare the results for response time and system throughput with those obtained in (a). How do you derive sensible utilizations for class A from this approximate model?
   c. Give an intuitive explanation for the differences observed using the two class model of (a) and the single class approximation of (b).
6. Implement the approximate MVA solution technique (Algorithm 7.3) for models with two closed (batch or terminal) classes.
7. Argue that $O\left(KC\prod_{c=1}^{C}(N_c+1) \right)$ is the correct expression for the time complexity of Algorithm 7.2.
8. Argue that $O\left(KC \right)$ is the correct expression for the time complexity of Algorithm 7.3 (assuming that the number of iterations does not depend upon K or C).