N=3 are given in Figure 5.3. Let T, U, Q, and W be the throughput, utilization, mean queue length, and mean wait time, respectively, for queue 1. The following recursive algorithm will determine P(0) (idle time), T, U, and W. The boundary to be used will be the state 0. Other statistics can be gathered in a similar manner.

Algorithm 5.1

1. Initialization

   $P(0)=1$
   $P(1)=\lambda/\mu P(0)=\lambda/\mu$
   $S=P(0)+P(1)=1+\lambda/\mu$
   $U=P(1)=\lambda/\mu$
   $Q=P(1)=\lambda/\mu$

2. Iteration. For n=2, N do step 2.

   $P(n)=((\mu+\lambda)P(n-1)-\lambda P(n-2))/\mu$
   $S=S+P(n)$
   $U=U+P(n)$
   $Q=Q+nP(n)$

3. Determination of statistics

   $P(0)=1/S$
   $U=U/S$
   $T=\mu U$
   $Q+Q/S$
   $W+Q/T$ (Little's Rule [L3])

Note: After having obtained the probability P(0) of the boundary state 0, it is possible to obtain all other state probabilities by running back through the algorithm.

5.3 A CPU-I/O Overlap Processing Model

We now describe a CPU-I/O overlap processing model. We give algorithms for exact two queue and approximate central server network

solutions.

5.3.1 Model Description

The assumption in the two queue model of section 5.2 is that a customer may occupy only one queue at a time. Let queue 1 be the CPU and queue 2 and I/O device. Then, a customer can be thought of as possessing a sequential processing structure. This is illustrated graphically in Figure 5.4.

Let us depart from this structure and allow a program the capability of overlapping CPU and I/O processing. In particular, let a program have two CPU phases, CPU_1 and CPU_2. During CPU_1, a job will occupy only the CPU. At the end of this phase, the job will initiate an I/O request and the second CPU phase, CPU_2. At this time the job will occupy positions in both the CPU and I/O queues. Upon completion of both I/O and CPU_2, the job reverts to CPU_1. This cycle then repeats itself. This structure is illustrated graphically in Figure 5.5. The arc connecting the two edges out of the CPU_1 node denotes a fork and join subgraph.

For the sake of simplicity, let the distributions for the CPU phases and the I/O be exponentially distributed. We can construct a Markovian model describing the overlap process. The diagram for one job is found in Figure 5.6. A job may be in one of four states: CPU_1, CPU_2, I/O, or an overlap (CPU_2-I/O) state.

The above model suffers from a flaw. It assumes that all I/O activity is initiated in parallel with the second phase of the CPU. A more realistic model should allow a mixture of sequential and

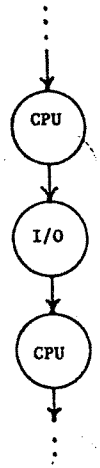


Figure 5.4

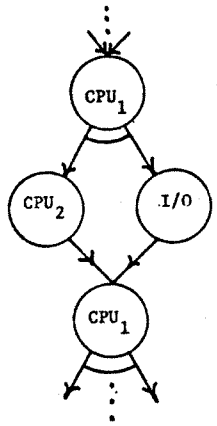


Figure 5.5

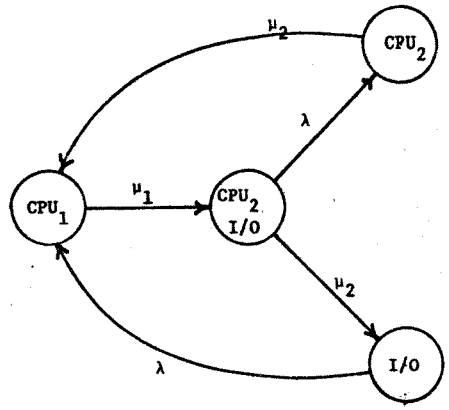


Figure 5.6

overlap processing. Let p be the probability that a job, upon completion of CPU_1, will require overlapped processing of CPU_2 and I/O. Then $\bar{p}=1-p$ is the probability that a job, upon completion of CPU_1, will initiate I/O and relinquish the CPU. This is illustrated graphically in Figure 5.7. The Markov diagram is found in Figure 5.8.

5.3.2 The Two Queue Algorithm

We have developed a model for CPU-I/O overlap within a single program. The algorithm that follows is for the exact solution of a one customer class, multiprogrammed, two queue network where each program may exhibit overlap characteristics. Both queues are assumed to have FCFS disciplines. Let the mean service times for CPU_1, CPU_2, and I/O be $1/\mu_1$, $1/\mu_2$, and $1/\lambda$, respectively. Let the state of the network be represented by the ordered 3-tuple (i,n,m) where there are n customers in the CPU (queue 1), n=1,...,N, and the customer being served is in the i^{th} phase of CPU processing. This customer has an outstanding I/O request if m=1; otherwise m=0. There are N-n+m I/O (queue 2) requests. For notational convenience, let i=1 if n=0. Let P(i,n,m) be the probability of state (i,n,m). Figure 5.9 gives the state transition diagram for N=3.

Algorithm 5.2 determines the value P(1,0,0). The determination of network statistics are not included. They can be determined in a manner similar to their determination in Algorithm 5.1.

Algorithm 5.2 Determination of P(1,0,0)
(assume N=1)

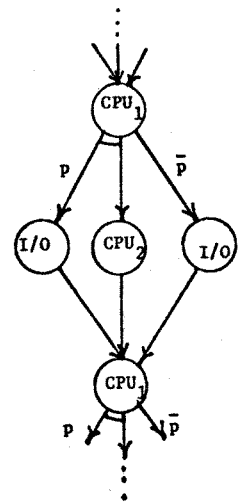


Figure 5.7

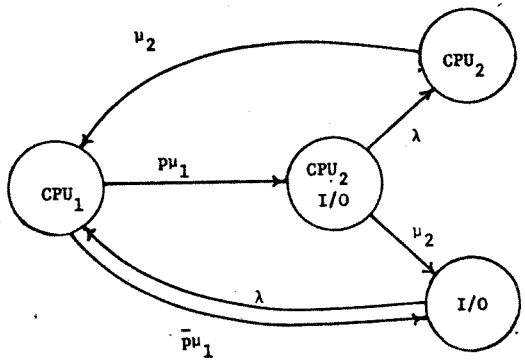


Figure 5.8

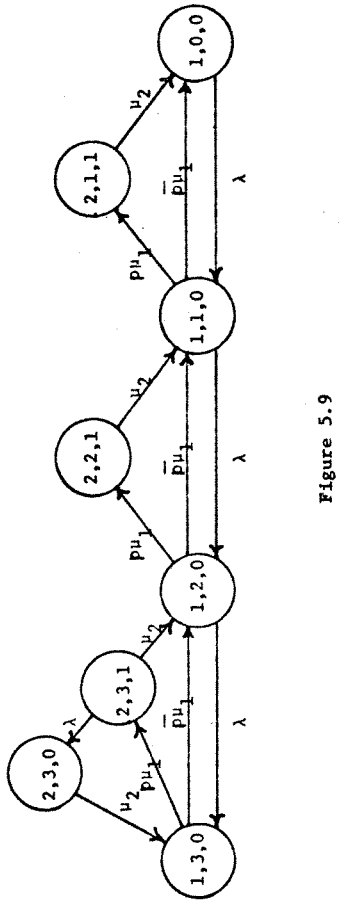


Figure 5.9

1. Initialization

$P(1,0,0)=1$

$P(2,1,1)=(\lambda/(\mu_2+\bar{p}\ (\lambda+\mu_2)/\ p\ ))P(1,0,0)$

This relation is obtained from the balance equations for states (1,0,0) and (2,1,1) as follows:

$$\mu_2 P(2,1,1)+\bar{p}\mu_1 P(1,1,0)=\lambda P(1,0,0) \quad (37)$$

$$p\mu_1 P(1,1,0)=(\lambda+\mu_2)P(2,1,1). \quad (38)$$

Dividing each side of (38) by $p\mu_1$ and substituting into (37) yields

$$\mu_2 P(2,1,1)+\frac{\bar{p}(\lambda+\mu_2)}{p}P(2,1,1)= P(1,0,0).$$

Collecting terms and dividing by $\mu_2+\frac{\bar{p}(\lambda+\mu_2)}{p}$ yields $P(2,1,1)=\frac{\lambda}{\mu_2+\bar{p}(\lambda+\mu_2)/p}\ P(1,0,0)$.

$P(1,1,0)=(\lambda+\mu_2)/(p\mu_1)P(2,1,1)$

$S=P(1,0,0)+P(1,1,0)+P(2,1,1)$

2. Iteration. For n=2,...,N, do step 2.

$$P(2,n,1)=\frac{(\lambda+\mu_2)P(1,n-1,0)-\lambda P(1,n-2,0)+\lambda\bar{p}/pP(2,n-1,0)}{\mu_2+\bar{p}(\lambda+\mu_2)/p}$$

This relation is obtained from the balance equations for state (1,n-1,0) and (2,n,1) as follows:

$$\mu_2 P(2,n,1)+\bar{p}\mu_1 P(1,n,0)+\lambda P(1,n-2,0) = (\lambda+\mu_2)P(1,n-1,0) \quad (39)$$

$$p\mu_1 P(1,n,0)+\lambda P(2,n-1,1)=(\lambda+\mu_2)P(2,n,1). \quad (40)$$

Dividing each side of (40) by $p\mu_1$, rearranging terms, and substituting into (39) yields

$$\mu_2 P(2,n,1)+\bar{p}(\lambda+\mu_2)/pP(2,n,1)-\lambda\bar{p}/pP(2,n-1,1)$$
$$+\lambda P(1,n-2,0)=(\lambda+\mu_2)P(1,n-1,0).$$

Rearranging and collecting terms results in

$$P(2,n,1)=$$
$$\frac{(\lambda+\mu_2)P(1,n-1,0)-\lambda P(1,n-2,0)+\bar{p}/pP(2,n-1,0).}{\mu_2+\bar{p}(\lambda+\mu_2)/p}$$

$P(1,n,0)=(P(2,n,1)(\lambda+\mu_2)-P(2,n-1,1))/(p\mu_1)$

$S=S+P(2,n,1)+P(1,n,0)$

3. Determination of $P(1,0,0)$.

$P(2,N,0)=P(2,N,1)\lambda/\mu_2$

$S=S+P(2,N,0)$

$P(1,0,0)=1/S$

This algorithm has been programmed for the case where λ is a function of the length of the I/O queue.

5.3.3 Central Server Network Algorithm

We now look at the more general problem of central server networks with CPU-I/O overlap. We assume that all customers are identical, the queueing disciplines are FCFS, I/O times are exponential, and CPU times are exponential for each of the two phases. The extension to models with several classes of customers appears to be difficult and is not treated in this dissertation. Our algorithm is based on Norton's Theorem [C2]. Even though this class of networks does not satisfy local balance, except when p=0, we shall apply Norton's Theorem to obtain a simpler two queue network. This resulting network is solved using Algorithm 5.2 yielding solutions approximate

to those of the original network. The algorithm is as follows:

Algorithm 5.3

1. Obtain composite I/O service rates.

Consider the given model. Construct a new model without the CPU and with no overlap. Since the I/O devices have exponential service times, the model satisfies local balance and has a product form solution. Determine the queue dependent composite I/O service rates by analyzing this sorted CPU model.

2. Solve the two queue, CPU-composite I/O CPU-I/O overlap model under the assumption that the composite I/O has a FCFS discipline set $\lambda(n)=R(n)$ as determined in 1.

The CPU parameters of this model are set to those for the original model. The composite I/O parameters have been determined in step 1. Algorithm 5.2 can now be used to solve this model for U, T, Q, and W.

5.3.4 Example

Consider a two I/O model where the I/O's have means of 2 and 4. The CPU phase means are each 1. There are two customers overlapping all I/O and accessing each I/O with equal probability. Let us trace through algorithm 5.3.

Step 1. The composite I/O service rates (from section 4.5) when there are n customers, n=1,2, in the composite I/O queue are 1/3 and 3/7, respectively.

Step 2. We now have a two queue model. The balance equations for the

resulting states are solved using Algorithm 5.2 to obtain U=.655, T=.327, Q=.95, and W=2.9.

5.4 An I/O-I/O Overlap Processing Model

5.4.1 Model Description

We consider a system in which a job may initiate two consecutive I/O requests before relinquishing the CPU. This is illustrated graphically in Figure 5.10. There are two CPU phases subscripted 1 and 2. At the end of CPU_1, the job initiates one I/O request and enters the second CPU phase. At the end of CPU_2, the job initiates a second I/O request and relinquishes the CPU. Upon completion of both I/O requests, the job reenters CPU_1 and repeats the cycle. This is illustrated by the fork and join subgraph in Figure 5.10. We assume a job will initiate parallel I/O requests with a probability p. The other branch in Figure 5.10 represents the case when the job decides to not overlap two I/O requests, but to do them in sequence. This case occurs with probability $\bar{p}$=1-p. If the times are exponential, a Markov model can be used. This process is illustrated as a Markov state transition diagram for one job in (Figure 5.11). The states include phases 1 and 2 of the CPU (CPU_1, CPU_2), an I/O phase (I/O), and the states in which the second CPU phase is overlapped with one I/O request (CPU_2-I/O), and in which the two I/O requests are overlapped (2I/O). The mean times for CPU_1, CPU_2 and I/O will be $1/\mu_1$, $1/\mu_2$, and $1/\lambda$, respectively.

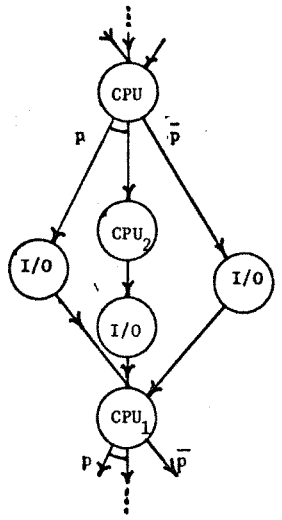


Figure 5.10

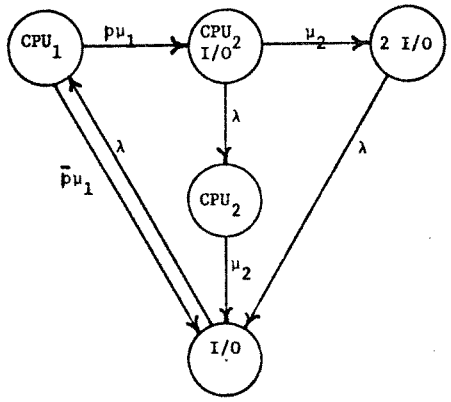


Figure 5.11

5.4.2 The Two Queue Model

Consider a two queue network as in Figure 5.2, with N identical customers. The CPU has a FCFS discipline and the I/O is processor shared. We may define a state in this system to be (i,j,n,k) where n=0,1,...,N is the number of customers in the CPU, i=0,1,...,2N is the number of I/O requests, j=0,1,...,N is the number of customers each with two I/O requests, and k=1,2 is the CPU phase for the customer being serviced in the CPU. For notational convenience k=1 when n=0. Let P(i,j,n,k) be the probability of state (i,j,n,k). The state transition diagram for N=2 can be found in Figure 5.12.

Algorithm 5.4 Determination of P(N+j,j,0,1),j=0,1,...,N

See Appendix E.

5.4.3 Central Server Network Algorithm

Algorithm 5.4 allows us to solve for a restricted set of two queue systems with I/O-I/O overlap capabilities. We can use it as a basis for an algorithm which will give accurate, though not exact, solutions for arbitrary central server systems. We assume identical customers, I/O devices with exponential times and FCFS disciplines, and an overlap behavior as described in section 5.4.1. As in Algorithm 5.3, we apply Norton's Theorem to obtain a simpler two queue system. This resulting system will be solved by Algorithm 5.4.

Algorithm 5.5

1. Obtain the composite I/O service rates.

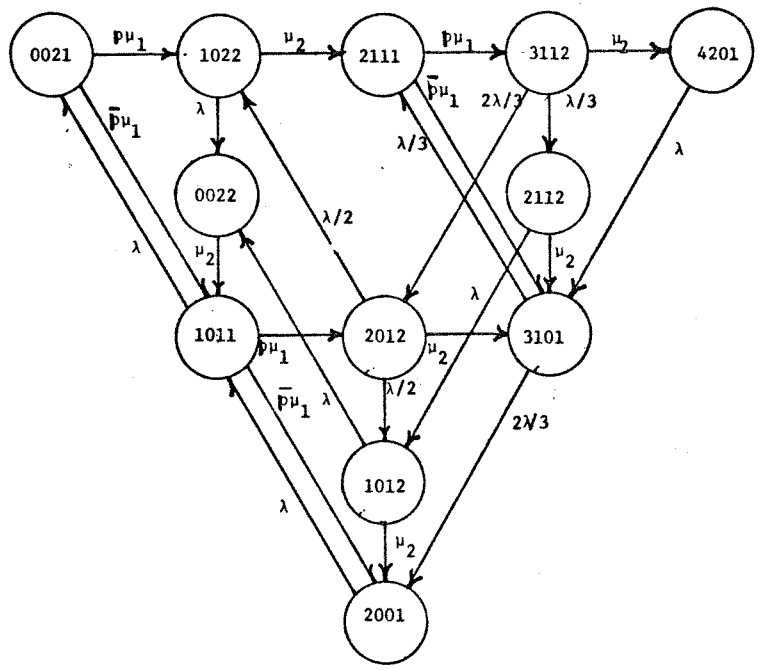


Figure 5.12

2. Solve the resulting two queue system with Algorithm 5.4.

Due to the assumption of a FCFS I/O discipline, by Algorithm 5.4 Algorithm 5.5 is not exact for a two queue network.

5.5 Validation by Simulation

We have written a simulator, using the simulation language ASPOL [C7], which simulates arbitrary central server networks exhibiting either CPU-I/O or I/O-I/O processing overlap characteristics. For validation of the approximation models, the following hypothesis is posed.

Hypothesis: For a given z, the differences in the measures of CPU utilization, mean CPU queue length, and mean CPU wait time between the simulator and the approximation models are:

1) less than z for the CPU utilization,
2) less than z times the number of customers in the network for the mean CPU queue length, and
3) less than z times the cycle time for the mean CPU wait time.

z may be thought of as the model tolerance. For a given z it is possible to determine the level of confidence with which the hypothesis can be held true. In [C3], a tolerance of .05 is considered go be good, and a tolerance of .10 adequate. For our validation purposes, we set z to .05. Results from Smith [S3], and Lavenberg [L1], on the statistics of Markovian processes, allow us to use the t-statistic [M1] for determining the confidence levels.

We ran 36 CPU-I/O overlap models to test the hypothesis. In general these models are fairly well balanced, though several are strongly CPU bound or I/O bound. Most models are constructed to show strong overlap patterns. This is done by setting p to one and Mean1/Mean2 (μ_2/μ_1) to 0. The model descriptions are found in Table 5.1 and the results in Table 5.2. The "f" under the branching probability headings, refers to the state dependent routing strategy whereby jobs go to the queue with the shortest queue length. Under the Confidence heading, 90 refers to a confidence level of greater than 90%, 1 refers to a level of less than 1%, and 50 refers to an uncertain level due to a slow convergence. The hypothesis, with a tolerance of .05, fits all the models with a confidence level greater than 90%. A non-overlap model [S1], for the same hypothesis, fits only 16 of the 36 models with a confidence level greater than 90%.

Also for validation purposes, 60 I/O-I/O models were run. Again, most are well balanced, but some are strongly CPU or I/O bound. Most are constructed to show strong overlap behavior. A large number of two queue models are run to check the validity of using the PS discipline in place of the FCFS discipline for the I/O queue. The last three models include the state dependent routing strategy whereby jobs go to the queue with the shortest queue length. The model descriptions are found in Table 5.3 and the results in Table 5.4. The hypothesis, with a tolerance of .05, fits all but one model with a confidence greater than 90%. With a tolerance of .06, the hypothesis fits the deviant model. Because there is more potential parallelism

Table 5.1 - CPU-I/O Overlap Model Descriptions

Model	No. Cust.	CPU			I/O 1(5)		I/O 2(6)		I/O 3(7)		I/O 4(8)	
		Mean1	Mean2	Prob. Overlap	Prob	Mean	Prob	Mean	Prob	Mean	Prob	Mean
1	2	0.	1.000	1.000	.250	2.500	.250	2.500	.250	2.500	.250	2.500
2	4	0.	1.000	1.000	.250	2.500	.250	2.500	.250	2.500	.250	2.500
3	8	0.	1.000	1.000	.250	2.500	.250	2.500	.250	2.500	.250	2.500
4	2	0.	1.000	1.000	.250	1.800	.250	1.800	.250	1.800	.250	1.800
5	4	0.	1.000	1.000	.250	1.800	.250	1.800	.250	1.800	.250	1.800
6	8	0.	1.000	1.000	.250	1.800	.250	1.800	.250	1.800	.250	1.800
7	2	0.500	1.000	0.500	.250	1.800	.250	1.800	.250	1.800	.250	1.800
8	4	0.500	1.000	0.500	.250	1.800	.250	1.800	.250	1.800	.250	1.800
9	8	0.500	1.000	0.500	.250	1.800	.250	1.800	.250	1.800	.250	1.800
10	2	0.	1.000	1.000	.500	1.800	.500	1.800				
11	4	0.	1.000	1.000	.500	1.800	.500	1.800				
12	8	0.	1.000	1.000	.500	1.800	.500	1.800				
13	2	0.	1.000	1.000	.125	4.000	.125	4.000	.125	4.000	.125	4.000
					.125	4.000	.125	4.000	.125	4.000	.125	4.000
14	4	0.	1.000	1.000	.125	4.000	.125	4.000	.125	4.000	.125	4.000
					.125	4.000	.125	4.000	.125	4.000	.125	4.000
15	8	0.	1.000	1.000	.125	4.000	.125	4.000	.125	4.000	.125	4.000
					.125	4.000	.125	4.000	.125	4.000	.125	4.000
16	2	0.500	1.000	0.500	.250	2.500	.250	2.500	.250	2.500	.250	2.500
17	4	0.500	1.000	0.500	.250	2.500	.250	2.500	.250	2.500	.250	2.500
18	8	0.500	1.000	0.500	.250	2.500	.250	2.500	.250	2.500	.250	2.500
19	2	0.900	1.000	0.100	.250	2.500	.250	2.500	.250	2.500	.250	2.500
20	4	0.900	1.000	0.100	.250	2.500	.250	2.500	.250	2.500	.250	2.500
21	8	0.900	1.000	0.100	.250	2.500	.250	2.500	.250	2.500	.250	2.500
22	2	0.500	0.500	1.000	.250	2.500	.250	2.500	.250	2.500	.250	2.500
23	4	0.500	0.500	1.000	.250	2.500	.250	2.500	.250	2.500	.250	2.500
24	8	0.500	0.500	1.000	.250	2.500	.250	2.500	.250	2.500	.250	2.500
25	2	0.	1.000	1.000	.100	6.000	.100	7.500	.500	1.500	.300	2.000
26	4	0.	1.000	1.000	.100	6.000	.100	7.500	.500	1.500	.300	2.000
27	8	0.	1.000	1.000	.100	6.000	.100	7.500	.500	1.500	.300	2.000

Table 5.1 - CPU-I/O Overlap Model Descriptions

Model	No. Cust.	CPU			I/O 1(5)		I/O 2(6)		I/O 3(7)		I/O 4(8)	
		Mean1	Mean2	Prob. Overlap	Prob	Mean	Prob	Mean	Prob	Mean	Prob	Mean
28	2	0.	1.000	1.000	.250	6.000	.250	6.000	.250	6.000	.250	6.000
29	4	0.	1.000	1.000	.250	6.000	.250	6.000	.250	6.000	.250	6.000
30	8	0.	1.000	1.000	.250	6.000	.250	6.000	.250	6.000	.250	6.000
31	2	0.500	1.000	0.500	.250	6.000	.250	6.000	.250	6.000	.250	6.000
32	4	0.500	1.000	0.500	.250	6.000	.250	6.000	.250	6.000	.250	6.000
33	8	0.500	1.000	0.500	.250	6.000	.250	6.000	.250	6.000	.250	6.000
34	2	0.	1.000	1.000	f	1.800	f	1.800				
35	4	0.	1.000	1.000	f	1.800	f	1.800				
36	8	0.	1.000	1.000	f	1.800	f	1.800				

Table 5.2 - CPU-I/O Overlap Model Results

Model	Confidence		CPU Util.			CPU M.Q.L.			CPU M.W.T.		
	App	Non-over	App	Sim	Non	App	Sim	Non	App	Sim	Non
1	90	1	.55	.54	.47	.81	.74	.61	1.47	1.36	1.29
2	90	1	.75	.74	.70	1.74	1.64	1.37	2.32	2.17	1.97
3	90	1	.92	.92	.89	4.06	3.99	3.46	4.43	4.37	3.86
4	90	1	.68	.68	.58	1.07	1.01	.79	1.58	1.50	1.36
5	90	1	.88	.87	.82	2.43	2.32	1.89	2.77	2.69	2.30
6	90	1	.98	.98	.97	5.80	5.59	4.99	5.90	5.65	5.13
7	90	90	.63	.62	.58	.92	.88	.79	1.47	1.41	1.36
8	90	90	.85	.84	.82	2.16	2.05	1.89	2.54	2.40	2.30
9	90	1	.98	.98	.97	5.39	5.39	4.99	5.51	5.59	5.12
10	90	1	.61	.60	.54	.94	.89	.73	1.54	1.46	1.36
11	90	1	.76	.75	.71	1.83	1.77	1.51	2.42	2.33	2.11
12	90	50	.87	.87	.85	3.64	3.62	3.22	4.18	4.21	3.77
13	90	1	.41	.41	.36	.55	.52	.43	1.36	1.28	1.20
14	90	1	.64	.64	.58	1.25	1.17	.99	1.97	1.81	1.69
15	90	90	.87	.86	.84	3.14	2.98	2.63	3.63	3.42	3.15
16	90	90	.51	.51	.47	.70	.68	.61	1.38	1.36	1.29
17	90	90	.72	.71	.70	1.56	1.49	1.37	2.15	2.04	1.97
18	90	90	.91	.91	.89	3.76	3.65	3.45	4.15	4.04	3.86
19	90	90	.48	.48	.47	.63	.62	.61	1.31	1.29	1.29
20	90	90	.70	.71	.70	1.41	1.43	1.37	2.01	2.00	1.97
21	90	90	.90	.91	.89	3.51	3.62	3.45	3.92	4.02	3.86
22	90	90	.52	.52	.48	.69	.67	.59	1.32	1.28	1.24
23	90	90	.74	.75	.71	1.49	1.49	1.32	2.00	2.00	1.85
24	90	1	.93	.94	.91	3.68	3.88	3.37	3.97	4.18	3.69
25	90	1	.54	.52	.46	.78	.72	.59	1.46	1.36	1.28
26	90	1	.74	.73	.68	1.67	1.57	1.33	2.27	2.13	1.94
27	90	1	.90	.93	.88	3.84	3.94	3.28	4.26	4.30	3.73
28	90	90	.26	.26	.24	.32	.30	.27	1.24	1.17	1.14
29	90	90	.37	.36	.35	.55	.52	.49	1.49	1.42	1.39
30	90	90	.48	.48	.47	.87	.85	.82	1.82	1.80	1.75
31	90	90	.25	.24	.24	.29	.28	.27	1.19	1.13	1.14
32	90	90	.36	.36	.35	.52	.53	.49	1.44	1.43	1.39
33	90	90	.47	.48	.47	.84	.86	.81	1.79	1.81	1.75
34	90	1	.76	.77	.53	1.26	1.16	.73	1.67	1.54	1.36
35	90	1	.84	.85	.71	2.25	2.22	1.50	2.64	2.60	2.11
36	90	1	.91	.93	.85	4.47	4.63	3.21	4.91	5.05	3.77

Table 5.3 - I/O-I/O Overlap Model Descriptions

Model	No. Cust.	CPU Mean1	CPU Mean2	Prob. Overlap	I/O 1(5) Prob	I/O 1(5) Mean	I/O 2(6) Prob	I/O 2(6) Mean	I/O 3(7) Prob	I/O 3(7) Mean	I/O 4(8) Prob	I/O 4(8) Mean
1	2	0.	1.000	1.000	1.000	1.600						
2	4	0.	1.000	1.000	1.000	1.600						
3	8	0.	1.000	1.000	1.000	1.600						
4	2	1.000	0.	1.000	1.000	1.600						
5	4	1.000	0.	1.000	1.000	1.600						
6	8	1.000	0.	1.000	1.000	1.600						
7	2	1.000	0.	1.000	1.000	1.000						
8	4	1.000	0.	1.000	1.000	1.000						
9	8	1.000	0.	1.000	1.000	1.000						
10	2	0.	1.000	1.000	1.000	1.000						
11	4	0.	1.000	1.000	1.000	1.000						
12	8	0.	1.000	1.000	1.000	1.000						
13	2	0.	1.000	1.000	1.000	.400						
14	4	0.	1.000	1.000	1.000	.400						
15	8	0.	1.000	1.000	1.000	.400						
16	2	1.000	0.	1.000	1.000	.400						
17	4	1.000	0.	1.000	1.000	.400						
18	8	1.000	0.	1.000	1.000	.400						
19	2	0.	1.000	1.000	.250	1.000	.250	1.000	.250	1.000	.250	1.000
20	4	0.	1.000	1.000	.250	1.000	.250	1.000	.250	1.000	.250	1.000
21	8	0.	1.000	1.000	.250	1.000	.250	1.000	.250	1.000	.250	1.000
22	2	1.000	0.	1.000	.250	1.000	.250	1.000	.250	1.000	.250	1.000
23	4	1.000	0.	1.000	.250	1.000	.250	1.000	.250	1.000	.250	1.000
24	8	1.000	0.	1.000	.250	1.000	.250	1.000	.250	1.000	.250	1.000
25	2	0.	10.000	.100	.250	1.600	.250	1.600	.250	1.600	.250	1.600
26	4	0.	10.000	.100	.250	1.600	.250	1.600	.250	1.600	.250	1.600
27	8	0.	10.000	.100	.250	1.600	.250	1.600	.250	1.600	.250	1.600
28	2	0.	2.000	.500	.250	1.600	.250	1.600	.250	1.600	.250	1.600
29	4	0.	2.000	.500	.250	1.600	.250	1.600	.250	1.600	.250	1.600
30	8	0.	2.000	.500	.250	1.600	.250	1.600	.250	1.600	.250	1.600

Table 5.3 - I/O-I/O Overlap Model Descriptions

Model	No. Cust.	CPU			I/O	1(5)	I/O	2(6)	I/O	3(7)	I/O	4(8)
		Mean1	Mean2	Prob. Overlap	Prob	Mean	Prob	Mean	Prob	Mean	Prob	Mean
31	2	.667	0.	.500	.250	1.600	.250	1.600	.250	1.600	.250	1.600
32	4	.667	0.	.500	.250	1.600	.250	1.600	.250	1.600	.250	1.600
33	8	.667	0.	.500	.250	1.600	.250	1.600	.250	1.600	.250	1.600
34	2	.500	.500	1.000	.250	1.600	.250	1.600	.250	1.600	.250	1.600
35	4	.500	.500	1.000	.250	1.600	.250	1.600	.250	1.600	.250	1.600
36	8	.500	.500	1.000	.250	1.600	.250	1.600	.250	1.600	.250	1.600
37	2	0.	1.000	1.000	.250	2.000	.250	2.000	.250	2.000	.250	2.000
38	4	0.	1.000	1.000	.250	2.000	.250	2.000	.250	2.000	.250	2.000
39	8	0.	1.000	1.000	.250	2.000	.250	2.000	.250	2.000	.250	2.000
40	2	1.000	0.	1.000	.250	2.000	.250	2.000	.250	2.000	.250	2.000
41	4	1.000	0.	1.000	.250	2.000	.250	2.000	.250	2.000	.250	2.000
42	8	1.000	0.	1.000	.250	2.000	.250	2.000	.250	2.000	.250	2.000
43	2	1.000	0.	1.000	.125	3.000	.125	3.000	.250	1.500	.500	.800
44	4	1.000	0.	1.000	.125	3.000	.125	3.000	.250	1.500	.500	.800
45	8	1.000	0.	1.000	.125	3.000	.125	3.000	.250	1.500	.500	.800
46	2	0.	1.000	1.000	.125	3.000	.125	3.000	.250	1.500	.500	.800
47	4	0.	1.000	1.000	.125	3.000	.125	3.000	.250	1.500	.500	.800
48	8	0.	1.000	1.000	.125	3.000	.125	3.000	.250	1.500	.500	.800
49	2	0.	1.000	1.000	.250	4.000	.250	4.000	.250	4.000	.250	4.000
50	4	0.	1.000	1.000	.250	4.000	.250	4.000	.250	4.000	.250	4.000
51	8	0.	1.000	1.000	.250	4.000	.250	4.000	.250	4.000	.250	4.000
52	2	0.	1.000	1.000	.125	2.500	.125	2.500	.125	2.500	.125	2.500
					.125	2.500	.125	2.500	.125	2.500	.125	2.500
53	4	0.	1.000	1.000	.125	2.500	.125	2.500	.125	2.500	.125	2.500
					.125	2.500	.125	2.500	.125	2.500	.125	2.500
54	8	0.	1.000	1.000	.125	2.500	.125	2.500	.125	2.500	.125	2.500
					.125	2.500	.125	2.500	.125	2.500	.125	2.500
55	2	1.000	0.	1.000	.125	2.500	.125	2.500	.125	2.500	.125	2.500
					.125	2.500	.125	2.500	.125	2.500	.125	2.500

Table 5.3 - I/O-I/O Overlap Model Descriptions

Model	No. Cust.	CPU			I/O	1(5)	I/O	2(6)	I/O	3(7)	I/O	4(8)
		Mean1	Mean2	Prob. Overlap	Prob	Mean	Prob	Mean	Prob	Mean	Prob	Mean
56	4	1.000	0.	1.000	.125	2.500	.125	2.500	.125	2.500	.125	2.500
					.125	2.500	.125	2.500	.125	2.500	.125	2.500
57	8	1.000	0.	1.000	.125	2.500	.125	2.500	.125	2.500	.125	2.500
					.125	2.500	.125	2.500	.125	2.500	.125	2.500
58	2	0.	1.000	1.000	f	1.000	f	1.000				
59	4	0.	1.000	1.000	f	1.000	f	1.000				
60	8	0.	1.000	1.000	f	1.000	f	1.000				

Table 5.4 - I/O-I/O Overlap Model Results

Model	Confidence		CPU Util.			CPU M.Q.L.			CPU M.W.T.		
	App	Non-over	App	Sim	Non	App	Sim	Non	App	Sim	Non
1	90	90	.30	.31	.28	.36	.38	.38	1.21	1.21	1.34
2	90	90	.31	.33	.30	.43	.40	.51	1.39	1.26	1.67
3	90	90	.31	.32	.31	.45	.37	.58	1.45	1.28	1.87
4	90	90	.29	.30	.28	.36	.33	.37	1.25	1.11	1.31
5	90	90	.31	.32	.31	.44	.37	.49	1.43	1.20	1.60
6	90	90	.31	.31	.31	.46	.34	.54	1.45	1.23	1.74
7	90	90	.43	.46	.42	.58	.55	.58	1.34	1.23	1.41
8	90	90	.48	.51	.47	.85	.74	.91	1.75	1.50	1.94
9	90	1	.50	.50	.49	.98	.76	1.21	1.96	1.52	2.45
10	90	1	.45	.47	.41	.59	.61	.59	1.32	1.31	1.43
11	90	1	.49	.49	.46	.84	.75	.93	1.72	1.53	2.00
12	90	1	.50	.51	.50	.99	.86	1.28	1.98	1.70	2.60
13	90	1	.78	.81	.71	1.23	1.29	1.14	1.57	1.58	1.62
14	90	1	.89	.92	.83	2.52	2.61	2.34	2.84	2.84	2.83
15	1	1	.96	.98	.92	5.50	5.91	4.95	5.73	6.03	5.40
16	90	1	.74	.79	.71	1.17	1.23	1.14	1.57	1.57	1.61
17	90	1	.88	.92	.83	2.46	2.49	2.35	2.80	2.70	2.82
18	90	1	.96	.98	.92	5.42	5.66	5.00	5.65	5.76	5.42
19	90	1	.65	.66	.53	.92	.94	.76	1.41	1.42	1.43
20	90	1	.86	.86	.74	2.12	2.12	1.78	2.47	2.46	2.39
21	90	1	.98	.98	.92	5.30	5.56	4.43	5.44	5.66	4.83
22	90	1	.61	.62	.53	.84	.85	.75	1.38	1.38	1.42
23	90	1	.84	.84	.75	1.97	1.96	1.77	2.36	2.32	2.36
24	90	1	.97	.98	.92	5.08	5.09	4.44	5.22	5.21	4.80
25	90	1	.52	.51	.37	.96	.94	.57	1.85	1.83	1.53
26	90	1	.65	.64	.51	2.25	2.07	1.31	3.49	3.27	2.57
27	90	1	.76	.77	.64	5.07	5.15	2.92	6.67	6.69	4.59
28	90	1	.53	.53	.38	.79	.80	.54	1.49	1.52	1.42
29	90	1	.71	.71	.54	1.80	1.83	1.22	2.53	2.56	2.25
30	90	1	.88	.88	.70	4.29	4.36	2.77	4.90	4.98	3.94
31	90	90	.40	.40	.39	.50	.50	.50	.84	.84	.85
32	90	90	.59	.59	.58	1.03	1.02	1.07	1.19	1.15	1.22
33	90	90	.77	.79	.77	2.21	2.34	2.43	1.96	1.99	2.10
34	90	1	.48	.48	.40	.59	.60	.49	1.24	1.24	1.24
35	90	1	.68	.69	.59	1.21	1.22	1.04	1.78	1.77	1.78
36	90	1	.85	.87	.78	2.70	2.85	2.35	3.17	3.27	3.00
37	90	1	.41	.41	.33	.50	.52	.42	1.24	1.26	1.30
38	90	1	.57	.58	.48	.99	1.05	.90	1.74	1.80	1.86
39	90	1	.72	.73	.64	1.96	1.98	1.90	2.73	2.71	2.97
40	90	1	.38	.38	.33	.47	.46	.42	1.23	1.20	1.27
41	90	1	.55	.57	.49	.94	.95	.88	1.70	1.68	1.82
42	90	1	.71	.71	.64	1.90	1.75	1.85	2.66	2.46	2.87
43	90	1	.47	.46	.40	.60	.59	.53	1.28	1.27	1.32
44	90	1	.67	.67	.59	1.29	1.24	1.18	1.91	1.83	2.00
45	90	1	.85	.86	.76	2.99	3.01	2.72	3.50	3.55	3.51

Table 5.4 - I/O-I/O Overlap Model Results

Model	Confidence		CPU Util.			CPU M.Q.L.			CPU M.W.T.		
	App	Non-over	App	Sim	Non	App	Sim	Non	App	Sim	Non
46	90	1	.50	.50	.40	.65	.66	.54	1.30	1.32	1.35
47	90	1	.69	.70	.59	1.37	1.39	1.20	1.99	2.00	2.06
48	90	1	.86	.85	.77	3.15	3.06	2.77	3.68	3.58	3.62
49	90	90	.22	.22	.18	.25	.25	.22	1.13	1.14	1.18
50	90	50	.31	.32	.27	.41	.43	.40	1.33	1.34	1.48
51	90	90	.38	.38	.35	.60	.58	.67	1.56	1.52	1.89
52	90	1	.38	.38	.29	.46	.46	.37	1.21	1.21	1.26
53	90	1	.59	.59	.48	1.02	1.03	.86	1.72	1.74	1.80
54	90	1	.81	.81	.70	2.51	2.49	2.15	3.10	3.06	3.08
55	90	1	.36	.36	.30	.43	.42	.36	1.19	1.18	1.23
56	90	1	.57	.57	.48	.95	.92	.84	1.65	1.60	1.74
57	90	1	.80	.82	.70	2.35	2.52	2.08	2.92	3.05	2.96
58	90	1	.64	.66	.50	.88	.93	.67	1.37	1.31	1.33
59	90	1	.78	.81	.67	1.70	1.84	1.33	2.15	2.25	2.00
60	90	1	.89	.92	.80	3.74	4.12	2.66	4.20	4.55	3.33

in a system with I/O-I/O overlap than in one with CPU-I/O overlap, it is expected that a non-overlap model would be less successful. This is borne by the fact that only 13 of the 60 non-overlap models fit the hypothesis with confidence greater than 90%.

Algorithms 5.2 - 5.5 are coded as FORTRAN routines and run on a CDC 6600. The CPU-I/O overlap models execute in less than 20ms. and the I/O-I/O overlap models, for a level of multiprogramming of 8, in less than 200ms.

5.6 Validation against Synthetic Job Mixes

We have run six synthetic job mixes on a CDC 6400 configuration for further validation of the overlap approximation models. The configuration consists of a CDC 6400 mainframe with 65536 60 bit words of main memory, 7 peripheral processors (PP's), and two CDC 808 disk drives with associated controllers and channels. The job mixes were designed to fit entirely into main memory, allowing us to model the system as a simple two I/O central server network with a constant level of multiprogramming. Each I/O uses a FCFS discipline, and the CPU uses a round robin discipline with a fixed quantum of 16ms. We approximate the CPU schedules by FCFS. The effects of this approximation are discussed later.

We run three job mixes for each model, composed of 1, 2, or 4 identical synthetic programs. The level of multiprogramming 4 is chosen as being representative of the real world. The levels 1 and 2 were chosen to show maximum effects of overlap. The programs are written in FORTRAN and COMPASS using a local I/O software package

called IOP [C6]. IOP allows the user the option of non-overlapped I/O (Figure 5.4) or CPU-I/O overlap (Figure 5.5). IOP allows us to write two synthetic programs, one for each model, with the probability of overlap set to 1. The CPU-I/O overlapped synthetic program is written so that the time for CPU_2 is approximately exponential, and the time for CPU_1 is the time necessary for IOP to set up the actual I/O transfer (a constant 8ms.). The I/O-I/O overlapped program is written so that CPU_1 is approximately exponential and CPU_2, the time to set up a data transfer, is a constant 9ms. In both programs, I/O consists of transferring 512 60 bit words either from the disk or to the disk. Each I/O requests consists of a seek, rotational delay, and the actual data transfer. The first two I/O stages are uniformly distributed and the transfer is constant. This results in highly hypoexponential I/O times. The I/O-I/O overlap program is written so that the two I/O requests are to separate disks. The exact parameter values for the models are obtained from event trace data and are found in Table 5.5.

In the solution of the models, the composite I/O rates are not determined as in Algorithms 5.3 and 5.5. The job mix consisting 2 CPU-I/O overlap programs is set up so that each program accesses a separate disk. Thus, the service rate of the I/O for two jobs is twice the service rate for one job. This is also true for the job mix consisting of one I/O-I/O program. The job mix of 4 CPU-I/O overlap programs is set up so that two programs access each disk. The I/O rates can be determined by using Norton's Theorem on the network with two customer classes, each with two customers and each accessing only

one disk. The service rates are set to the total throughputs of the I/O system. The I/O service rates for the job mixes of 2 and 4 I/O-I/O overlap programs are determined in the same way.

The model results are found in Table 5.6. The results are remarkably good considering the various approximations. In all cases the tolerance is less than .05. Let us examine some of the approximations and their probably effects. We are modelling a round robin discipline with a FCFS discipline. Studies with non-overlapped models [B1] indicate that with hypoexponential CPU distributions, the substitution of a FCFS discipline for a round robin discipline will result in higher utilizations. Second, we approximate the CPU hypoexponential distributions with exponential distributions. Studies [B1] indicate that predicted utilizations for non-overlapped systems should be lower than actual utilizations with this approximation. These two approximations apparently cancel each other out. This leaves the approximation of the I/O distributions as being exponential. The trace tape data indicate I/O distributions with coefficients of variation ranging from .34 to .48. This approximation should result in lower utilizations and probably explains this pattern.

The non-overlap model produces utilizations appreciably lower than the actual measured ones (.07 to .22 lower). We feel that only the approximation models of this chapter give reasonable predictions.

5.7 CPU-I/O Overlap vs. No Overlap

Given that CPU-I/O overlap can be effected without an in-

Table 5.5 - Job Mix Parameters

CPU-I/O

No. Cust.	CPU			I/O 1		I/O 2	
	Mean 1	Mean 2	Prob. Overlap	Prob	Mean	Prob	Mean
1	8.0ms	26.6ms	1.00	1.0	46.1ms		
2	8.0	26.2	1.00	0.5	42.0	0.5	43.5ms
4	8.0	25.7	1.00	0.5	43.9	0.5	37.6

I/O-I/O

No. Cust.	CPU			I/O 1		I/O 2	
	Mean 1	Mean 2	Prob. Overlap	Prob	Mean	Prob	Mean
1	78.6	9.0	1.00	0.5	35.2	0.5	35.8
2	67.5	9.0	1.00	0.5	39.9	0.5	39.6
4	67.6	9.0	1.00	0.5	38.0	0.5	45.8

Table 5.6 - Comparison of Trace and Model Statistics

CPU-I/O

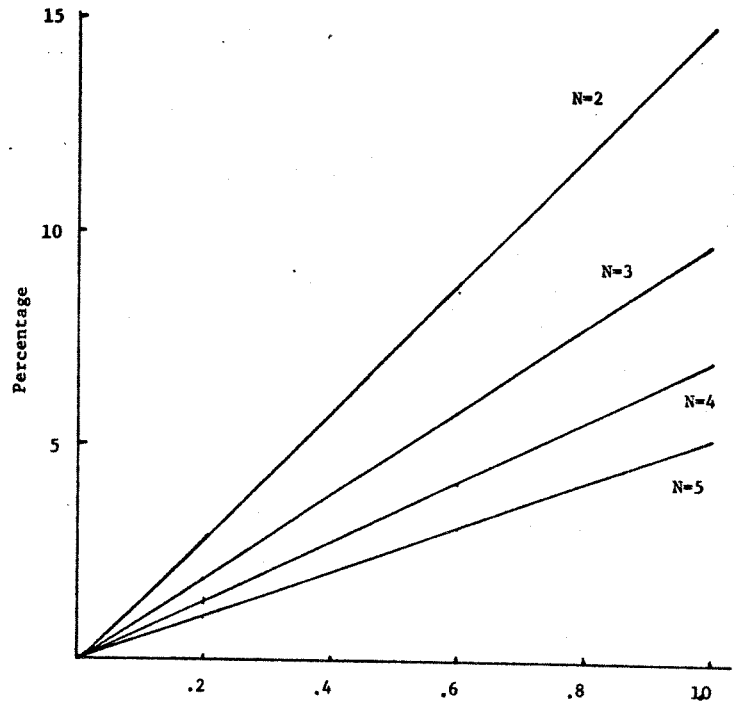
No. Cust.	CPU Util.			CPU M.Q.L.			CPU M.W.T.		
	app	trace	non	app	trace	non	app	trace	non
1	.57	.57	.45	.57	.57	.45	34.6	34.6	34.6
2	.87	.91	.75	1.43	*	1.06	57.0	*	51.6
4	.97	.99	.92	3.14	3.18	2.60	109.0	108.5	104.6

I/O-I/O

No. Cust.	CPU Util.			CPU M.Q.L.			CPU M.W.T.		
	app	trace	non	app	trace	non	app	trace	non
1	.67	.72	.50	.67	.72	.50	87.6	87.6	87.6
2	.85	.89	.73	1.31	1.25	1.06	117.8	107.8	111.2
4	.96	.99	.90	2.95	2.72	2.42	234.0	186.0	206.0

* Unobtainable due to garbage on tape.

crease in overhead or buffer space, throughput will be higher than for a system with no overlap. The question becomes how much improvement. Consider a well balanced system with three I/O devices. Figure 5.13 graphically illustrates the percentage relative improvement in throughput of a system with CPU-I/O overlap over a system without overlap as the probability of overlap p is varied for the levels of multiprogramming 2, 3, 4, and 5. The values of $1/\mu_1$ and $1/\mu_2$ are such that these curves represent the optimal improvements. If we set 5% as the minimum relative improvement in throughput necessary before considering the overlap as worthwhile, then it is worthless for a level of multiprogramming 5. For a level of multiprogramming of 4, it is necessary that at least 72% of all I/O activity be overlapped with processing activity. For a large general purpose, large user community computing system such as the University of Texas, UT2D, system, the levels of multiprogramming are high, and it would be impossible to have a high percentage of overlapped I/O activity. These systems would find the expected improvement too low relative to the cost of widespread introduction of overlapped CPU-I/O processing. However CPU-I/O overlap might be worthwhile for a balanced production system where: 1) the level of multiprogramming is low (2 or 3), or 2) the set of application programs making up the workload is small in number such that a high value of p could be attained economically.



Relative Throughput Improvement of
CPU-I/O Overlap over Non-Overlap

Figure 5.13

CHAPTER VI

SUMMARY AND CONCLUSIONS

We have made three contributions with this work. First, we have presented a fairly complete characterization of single server queues which have product form solutions. We have discovered relationships between the product form, local balance, and station balance.

Second, using the above results, we have presented a large set of queueing networks which have product form solutions. These include networks with several customer classes and simple state dependent routing strategies. Examples have been given which show the utility of the models with routing strategies to the modeling of computer systems. These models should be useful for modeling I/O levelling schemes. Preliminary validation has shown reasonable accuracy for our models. Further work is necessary to show the range of applicability and accuracy of these models. Additionally, we have shown that complex networks satisfying local balance may be replaced by an "equivalent" two queue network (Norton's Theorem). Norton's Theorem has widespread application in the solution of complex central server models.

Finally, we have developed models of device sharing by individual jobs and applied these to the cases of CPU-I/O and I/O-I/O processing overlap behavior within one program. These models along with Norton's Theorem form the basis for an accurate and efficient

set of algorithms for the approximate solution of a large set of central server models with device sharing characteristics. 96 models have been compared to simulation for validation. Additionally, six synthetic job mixes have been run on a CDC 6400 configuration for further validation. Finally, we have shown that CPU-I/O overlap processing increases system throughput over non-overlap processing only minimally for levels of multiprogramming greater than three. These models should prove useful in modeling multiprogrammed computer systems with device sharing capabilities. They may also be used with techniques for modeling passive resources such as memory [B3, I1]. The feasibility of extending the models to include non-exponential I/O distributions should be studied.

APPENDIX A

In this appendix we develop the notation necessary to describe the behavior of a single queue (SQ) system with a general interarrival process. We define the property of local balance similar to that in Chapter III and show that local balance implies a Poisson interarrival process.

A.1 Notation

We allow K customer classes, labelled k=1,...,K. Let Y_k be a positive real random variable denoting the initial service time for a customer of class k. Y_k has the probability distribution function

$$F_k(t)=P_r(0<Y_k<t),$$

with derivative

$$f_k(t)=dF_k(t)/dt,$$

and mean

$$1/\mu_k=E[Y_k]=\int_0^\infty tf_k(t)dt.$$

At the time of arrival of a customer of class k, let Z_k be a positive real random variable denoting the time until the next arrival for a customer of that class. We assume Z_k is independent of the state of the system at the time of the last arrival and has the probability distribution function

$$H_k(t)=P_r(0<Z_k<t),$$

with derivative

$$h_k(t)=dH_k(t)/dt,$$

and mean

$$1/\lambda_k=E[Z_k]=\int_0^\infty th_k(t)dt.$$

The only constraints on F_k and H_k are that they be twice differentiable and have finite nonzero means.

We use the concept of a queue developed in Chapter III. Let $T=(T_1,\ldots,T_K)$ be a K-tuple in which T_k is the time until the next arrival of a customer of class k. Using the terms S and X_i as introduced in Chapter III, we define $X=(S,T,X_1,\ldots,X_n)$ to be the complete state description. Note that the T_k's and X_i's are continuous variables.

The behavior of a SQ system can be specified in terms of the occupancy S, the state X and the occupancies and states created either by arrival or departure of an arbitrary customer from S and X. As a result, we define the following notations

$$S-i=(k(1),\ldots,k(i-1),k(i+1),\ldots,k(n))$$

$$S+(i,k)=(k(1),\ldots,k(i-1),k,k(i),\ldots,k(n))$$

$$T\leftarrow(k,Z)=(T_1,\ldots,T_{k-1},Z,T_{k+1},\ldots,T_k)$$

$$X-(i,k,Z)=(S-i,T\leftarrow(k,Z),X_1,\ldots,X_{i-1},X_{i+1},\ldots,X_n)$$

$$X+(i,k,Y)=(S+(i,k),T,X_1,\ldots,X_{i-1},Y,X_i,\ldots,X_n).$$

S-i and X-i are the occupancy and state resulting from the departure of the customer from station i. S+(i,k) and X+(i,k,Y,Z) are the occupancy and state which result from the arrival of a customer of class k into the i^{th} station. The customer will have an initial service time requirement of Y and the time until the next arrival of a customer of class k is Z.

The queueing discipline parameters a(i|k,S) and r(i|S) are defined in Chapter III. Similarly, the meaning of the term feasibility remains unchanged. Using the above notation and following the same analysis of Chapter III we derive the following differential equations for the spdf p(t,X):

$$\frac{dp(t,X)}{dt} = \sum_{i=1}^{n} \frac{\partial p(t,X)}{\partial X_i} r(i|S) + \sum_{k=1}^{K} \frac{\partial p(t,X)}{\partial T_k} + \sum_{i=1}^{n} a(i|S-i)p(t,X-(i,k(i),0+))f_{k(i)}(X_i)h_{k(i)}(T_{k(i)}) + \sum_{k=1}^{K}\sum_{i=1}^{n+1} r(i|S+(i,k))p(t,X+(i,k,0+)).$$

for all states X. We use the notation p(X) to represent the equilibrium spdf. The balance equations for the system are

$$\sum_{i=1}^{n} \frac{\partial p(t,X)}{\partial X_i} r(i|S)+a(i|S-i)p(t,X-(i,k(i),0+))f_{k(i)}(X_i)h_{k(i)}(T_{k(i)})]=$$

$$\sum_{k=1}^{K} \frac{\partial p(t,X)}{\partial T_k} + \sum_{i=1}^{n+1} r(i|S+(i,k))p(t,X+(i,k,0+))]$$

Local balance for the system is:

$$\frac{\partial p(t,X)}{\partial T_k} + \sum_{i=1}^{n+1} r(i|S+(i,k))p(t,X+(i,k,0+))=0$$

for k=1,...,K. Local balance has the same intuitive meaning as in Chapter III.

We now present the following result:

Theorem A.1: If a system is in local balance, then the arrival process must be Poisson for each customer class.

Proof: Since the system is in local balance, the following relation must be true:

$$\sum_{i=1}^{n} [\frac{\partial p(X)}{\partial X_i} r(i|S)+a(i|S-i)p(X-(i,k(i),0+))f_{k(i)}(X_i)h_{k(i)}(T_{k(i)})]=0$$

for all feasible X, n=1,....

The solution for this partial differential equation when k(i)=k, i=1,...,n, must be of the form

$$p(X)=d(X_1,\ldots,X_n,T_1,\ldots,T_{k-1},T_{k+1},\ldots,T_k)h_k(T_k)+C(T_k) \quad k=1,\ldots,K. \quad (A.1)$$

for some function d.

Due to the constraint $\lim_{\{X_i\to\infty\}_{i=1,\ldots,n}} p(X)=0$, $C(T_k)=0$

Therefore,

$$p(X)=d(X_1,\ldots,X_n,T_1,\ldots,T_{k-1},T_{k+1},\ldots,T_k)h_k(T_k),k=1,\ldots,K.$$

Substituting into the local balance relation, for

$k(i)=k,i=1,\ldots,n$

$$\frac{\partial p(X)}{\partial T_k} = -g(X_1,\ldots,X_n,T_1,\ldots,T_{k-1},T_{k+1},\ldots,T_k)h_k(T_k),k=1,\ldots,K.$$

for some function g.

The solution is

$$p(X)=g(X_1,\ldots,X_n,T_1,\ldots,T_{k-1},T_{k+1},\ldots,T_k)\cdot$$
$$\lambda_k(1-H_k(T_k))+\ C(X_1,\ldots,X_n,T_1,\ldots,T_K)$$

Now $C(X_1,\ldots,X_n,T_1,\ldots,T_{k-1},T_{k+1},\ldots,T_K)=0$ since $\lim_{T_k\to\infty} p(X)=0$

Substituting into (A.1) results in

$$d(X_1,\ldots,X_n,T_1,\ldots,T_{k-1},T_{k+1},\ldots,T_K)h_k(T_k) =$$
$$g(X_1,\ldots,X_n,T_1,\ldots,T_{k-1},T_{k+1},\ldots,T_K)\cdot$$
$$\lambda_k(1-H_k(T_k)),k=1,\ldots,K$$

Therefore $h_k(T_k)=\lambda_k(1-H_k(T_k)),k=1,\ldots,K.$

This relationship only occurs if H_k defines a Poisson process for k=1,...,K.

APPENDIX B

Theorem 4.1: If the network satisfies the above constraint (25) and each subsystem satisfies local balance when isolated, then:

1. The equilibrium spdf of the network takes the product form

$$P(X)=\frac{1}{G}\left[\prod_{\alpha\subseteq S(U)}\ \prod_{\beta\subseteq S(K)}\ \prod_{i=1}^{N(\alpha,\beta|\bar{N})} f_{\alpha,\beta}(i)\right]\prod_{n=1}^{U} p_u(X_u) \quad (26)$$

   where G is a normalizing constant and $p_u(X_u)$ is the spdf of subsystem u in isolation with arrival rate $d_{u,k}$.

2. the network is locally balanced, and
3. each subsystem satisfying station balance when isolated satisfies network station balance.

Proof: Let $p_u(X_u)$ be the equilibrium spdf for the u^{th} subsystem when fed with a Poisson source with rate $d_{u,k}$, k=1,...,K. If the u^{th} subsystem is in local balance when in isolation, the following equations hold

$$d_{u,k}p_u(X_u)=\sum_{i=1}^{n(u)+1} r_u(i|S_u+(i,k))p_u(X_u+(i,k,0+1)),k=1,\ldots,K \quad (6)$$

and

$$\sum_{i=1}^{n(u)}\left[d_{u,k(i)}a_u(i|k(i),S_u-i)p_u(X_u-i)f_{u,k(i)}(X_i) + r_u(i|S_u\frac{\partial p_u(X_u)}{\partial X_i}\right] = 0 \quad (7)$$

for all feasible X_u.

Let us write the balance equation in the following way.

$$\sum_{u=1}^{U} [V_{1,u}(\bar{X})+V_{2,u}(\bar{X})] = \sum_{k=1}^{K_o} [V_{3,k}(\bar{X})+V_{4,k}(\bar{X})]$$

where

$$V_{1,u}(\bar{X}) = \sum_{i=1}^{n(u)} \frac{\partial p(\bar{X})}{\partial X(u,i)}\, r(u,i|\bar{S})$$

$$V_{2,u}(\bar{X}) = \sum_{i=1}^{n(u)} \Big[\sum_{v=1}^{U} \sum_{j=1}^{n(v)+1} b_{v,u,k(u,i)}(\bar{N}-(u,k(u,i)))a(u,i|k(u,i),$$

$$\bar{S}-(u,i))f_{u,k(u,i)}(X(u,i))\cdot$$

$$r(v,j|\bar{S}-(u,i)+(v,j,k(u,i)))p(\bar{X}-(u,i)+(v,j,k(u,i,0)))+$$

$$b_{0,u,k(u,i)}(\bar{N}-(u,k(u,i)))\lambda_{k(u,i)}{}^{a(u,i\ K(u,i),\bar{S}-(u,i))}\cdot$$

$$f_{u,k(u,i)}(X(u,i))p(\bar{X}-(u,i))]$$

$$V_{3,k}(\bar{X}) = y_{0,k}p(\bar{X})$$

$$V_{4,k}(\bar{X}) = -\sum_{u=1}^{U} \sum_{i=1}^{n(u)+1} p(\bar{X}+(u,i,k,0+))r(u,i|\bar{S}+(u,i,k))b_{u,0,k}(\bar{N})$$

We first show that

$$\sum_{i=1}^{n(u)+1} p(\bar{X}+(u,i,k,0))r(u,i|\bar{S}+(u,i,k)) = y_{u,k}(\bar{N})p(\bar{X}) \text{ for all}$$

feasible $\bar{X}, u=1,\ldots,U, k=1,\ldots,K.$

Assuming $p(\bar{X})$ of the form (26) and substituting into the left hand side of the above expression, we obtain

$$\sum_{i=1}^{n(u)+1} p(\bar{X}+(u,i,k,0))r(u,i|\bar{S}+(u,i,k)) =$$

$$\sum_{i=1}^{n(u)+1} \frac{1}{G} \prod_{\alpha\subseteq S(U)} \prod_{\beta\subseteq S(K)} \prod_{j=0}^{N(\alpha,\beta|\bar{N}+(u,k))-1} f_{\alpha,\beta}(j) r_u(i|S_u+(i,k))\cdot$$

$$\prod_{v\neq u}^{U} p_v(X_v)\cdot p_u(X_u+(i,k,0)), k=1,\ldots,K.$$

Substituting (6) into the above term results in

$$\sum_{i=1}^{n(u)+1} p(\bar{X}+(u,i,k,0))r(u,i|\bar{S}+(u,i,k)) =$$

$$\frac{1}{G}\Big[\prod_{\alpha\subseteq S(U)} \prod_{\beta\subseteq S(K)} \prod_{j=0}^{N(\alpha,\beta|\bar{N})-1} f_{\alpha,\beta}(j)\Big] \cdot \prod_{v=1}^{U} p_v(X_v) \cdot$$

$$d_{u,k}\Big[\prod_{\alpha\subseteq S(U)} \prod_{\beta\subseteq S(K)} f_{\alpha,\beta}[N(\alpha,\beta|\bar{N})]\Big], k=1,\ldots,K.$$

$$u\in\alpha \quad k\in\beta$$

Substituting (25) and (26) into the above relation yields

$$\sum_{i=1}^{n(u)+1} p(\bar{X}+(u,i,k,0))\ r(u,i|\bar{S}+(u,i,k)) = y_{u,k}(\bar{N})\ p(\bar{X}), \qquad (41)$$

$$k=1,\ldots,K.$$

Substituting (41) into $V_{4,k}(\bar{X})$ results in

$$V_{4,k}(\bar{X}) = -\sum_{u=1}^{U} y_{u,k}(\bar{N})\ b_{u,0,k}(\bar{N})\ p(\bar{X}),\ k=1,\ldots,K.$$

Substituting (19) into the above expression yields

$$V_{4,k}(\bar{X}) = -y_{0,k}(\bar{N})\ p(\bar{X}) = -\lambda_k\ p(\bar{X})\ ,\ k=1,\ldots,K.$$

Therefore

$$\sum_{k=1}^{K^o} (V_{3,k}(\bar{X}) + V_{4,k}(\bar{X})) = 0 \ , \quad \text{for all feasible } \bar{X}.$$

Now we show that network local balance holds for each system u, in other words,

$$V_{1,u}(\bar{X}) + V_{2,u}(\bar{X}) = 0 \ , \ u=1,\ldots,U.$$

From equation (41), for k=1,...,K

$$\sum_{j=1}^{n(v)+1} p(\bar{X}-(u,i)+(v,j,k(u,i),0))\ r(v,j|\bar{S}-(u,i)+(v,j,k(u,i))) =$$

$$y_{v,k(u,i)}(\bar{N}-(u,k(u,i))\ p(\bar{X}-(u,i)) \quad .$$

Substituting into the equation for $V_{2,u}(\bar{X})$ we obtain

$$V_{2,u}(\bar{X}) = \sum_{i=1}^{n(u)} \sum_{v=0}^{U} b_{v,u,k(u,i)}(\bar{N}-(u,k(u,i))\ a(u,i\,|k(u,i),\bar{S}-(u,i))$$

$$f_{u,k(u,i)}(X(u,i))\ y_{v,k(u,i)}(\bar{N}-(u,k(u,i)))\ p(\bar{X}-(u,i)) \ .$$

Note that for $k(u,i) \quad K_o$, $b_{0,u,k(u,i)} = 0$.

Substitution of equations (19) and (20) yield

$$V_{2,u}(\bar{X}) = \sum_{i=1}^{n(u)} y_{u,k(u,i)}(\bar{N}-(u,k(u,i)))\ f_{u,k(u,i)}(X(u,i))\cdot$$

$$p(\bar{X}-(u,i))\ a(u,i\,|k(u,i),\bar{S}-(u,i)) \ .$$

Assuming (25) and (26) results in

$$V_{2,u}(\bar{X}) = \sum_{i=1}^{n(u)} \frac{1}{G} \Big[\prod_{\alpha \subseteq S(U)} \prod_{\beta \subseteq S(K)} \prod_{j=0}^{N(\alpha,\beta|\bar{N})-1} f_{\alpha,\beta}(j) \Big] \cdot$$

$$\prod_{v\neq u}^{U} p_v(X_v)\ d_{u,k}\ p_u(X_u-i) \quad .$$

Substituting in equation (7) yields

$$V_{2,u}(\bar{X}) = -\sum_{i=1}^{n(u)} \frac{\partial p(X_u)}{\partial X(u,i)}\ r(u,i\,|\bar{S})\ \frac{1}{G} \Big[\prod_{\alpha\_S(U)} \prod_{\beta\_S(K)} \prod_{j=0}^{N(\alpha,\beta|\bar{N})-1}$$

$$f_{\alpha,\beta}(j)\Big] \prod_{v\neq u}^{U} p_v(X_v) \quad .$$

Substituting back for equation (26) yields

$$V_{2,u}(\bar{X}) = -\sum_{i=1}^{n(u)} \frac{\partial p(\bar{X})}{\partial X(u,i)}\ r(u,i\,|\bar{S}) \quad .$$

Hence $V_{1,u}(\bar{X}) + V_{2,u}(\bar{X}) = 0$ for u=1,...,U.

Thus the product form solution (25) satisfies network local balance and network balance.

The proof of part 3. is similar to that of part 2.

APPENDIX C

In this appendix we give a proof to Theorem 4.2. First we restate equations (25) and (27):

$$y_{u,k}(\bar{N}) = d_{u,k} \prod_{\substack{\alpha \subseteq S(U) \\ u \in \alpha}} \prod_{\substack{\beta \subseteq S(K) \\ k \in \beta}} f_{\alpha,\beta}[N(\alpha,\beta|\bar{N}] \quad u=1,\ldots,U, k=1,\ldots,K \tag{25}$$

and

$$b_{u,v,k} = p'_{u,v,k} \prod_{\beta \subseteq S(K)} \prod_{\substack{w=1 \\ u=e(w)}}^{W} \frac{p_{m(w,\ell(w,v)),\beta}[N(m(w,\ell(w,v)),\beta|\bar{N})]}{g_{m(w),\beta}[N(m(w),\beta|\bar{N})]} \tag{27}$$

$$u,v=1,\ldots,U, \; k=1,\ldots,K$$

The proof of the theorem assumes that each node in the network is either the entry node or exit node for one or zero p-subnetworks. For example, the network in Figure 4.5 does not satisfy the assumptions of the proof. If a network has U nodes (we will use node for subsystem), u=1,...,U, of which at least one is an entry and/or an exit to more than one p-subnetwork, we can construct a second network with V>U nodes which will satisfy the earlier restriction and whose arrival rates for nodes u=1,...,U are identical to those of the original network. This "equivalent" network is constructed by applying Algorithm C.1, which we present later, on the original network. The nodes u=U+1,...,V are branching points and can be thought of as having constant holdings times of 0.

Before presenting Algorithm C.1 we introduce the term level. In section 4.4.2 we introduced the notion of nested p-subnetworks. Assume p-subnetwork w is nested within n p-subnetworks. We define level(w)=n+1. The level of a p-subnetwork w, level(w), can be determined formally as follows. We say p-subnetwork w has level(w)=1 if there exists no other p-subnetwork w', w'=1,...,W such that $m(w) \supset m(w')$. Several p-subnetworks may have level 1. A p-subnetwork w has level(w)=n+1, n=1,...,W if there exists another p-subnetwork w_1, with level(w_1)=n such that $m(w_1) \subset m(w)$ and there exists no p-subnetwork w_2 such that $m(w_1) \subset m(w_2)$ and $m(w_2) \subset m(w)$. Using this formula we can determine the level of any p-subnetwork. For notational convenience we assign a level of zero to the entire network. Given two p-subnetworks w_1 and w_2 with level(w_1)$\geq$level(w_2), either of the following relationships holds,

1) $m(w_1) \subset m(w_2)$,
2) or there exists at least one p-subnetwork (or the network) w_3 with level(w_3)<level(w_2) such that $m(w_1) \subset m(w_3)$ and $m(w_2) \subset m(w_3)$. We may now present Algorithm C.1.

Algorithm C.1: Construction of a new network with V$\geq$U nodes none of which are entries and/or exits to more than one p-subnetwork. The new and old networks have identical arrival rates for nodes u=1,...,U.

1. Let V=U, W'=W, $L'_w = L_w$ $m'_w(w,\ell)$ w=1,...,W, ℓ=1,...,L_w e'(w)=e(w), m'(w)=m(w), u=1,...,W, $b'_{u,v,k} = b_{u,v,k}$,

$u,v=0,\ldots,U$, $p'_{\alpha,\beta}=p_{\alpha,\beta}$,

$g'_{\alpha,\beta}=g_{\alpha,\beta}$, where $\alpha\_S(U),\beta\_S(K)$ and $p''_{u,v,k}=p'_{u,v,k}$

$u,v=0,\ldots,U,k=1,\ldots,K$.

In other words the parameters for our intermediate and final networks will be primed. We initialize to the original network. For simplicity we will carry the index for u and v from 0 to U even when referring to closed customer classes.

2. Let S be the set of all nodes which are entries and or exits to more than one p-subnetwork. If $S=\emptyset$ end and return the last primed network.

3. Take any $u \in S$.

If u is an entry to more than one p-subnetwork, go to 5.

If u is an exit to more than one p-subnetwork, go to 8.

4. u is an exit node to one p-subnetwork w_1 and entry to another p-subnetwork w_2.

Add a new node V+1 and make the following changes to the branching probabilities for $k=1,\ldots,K$

$$b'_{v,V+1,k}=0 \qquad v=0,\ldots,V+1$$
$$b'_{V+1,v,k}=0 \qquad v=0,\ldots,V+1$$
$$b'_{v,V+1,k}=b'_{v,u,k} \qquad v=0,\ldots,V$$
$$b'_{V+1,u,k}=1$$
$$b'_{v,u,k}=0 \qquad v=0,\ldots,V$$

Node V+1 replaces u as the exit for p-subnetwork w_1 and u remains the entry to p-subnetwork w_2. Changing $b'_{u,v,k}$ implies changing the $p'_{\alpha,\beta}$, $g'_{\alpha,\beta}$ and $p''_{u,v,k}$ where appropriate. Make changes to $m'(w,\ell)$, $m'(w)$, and $e'(w)$, $w=1,\ldots,W$; $\ell=1,\ldots,L_w$ by adding V+1 where appropriate. When referring in the future to adding new nodes and changing $b'_{u,v,k}$, this will be understood.

$V=V+1$

Go to 2.

5. Let S(u) be all the p-subnetworks for which u is an entry. S(u) contains J nodes.

Let $n=\min\{m \mid m=\text{level}(w),\ w\in S(u)\}$

Determine the $M>0$ p-subnetworks $w_1,\ldots,w_M$ such that level $(w_i)=n$ where $w_i \in S(u), i=1,\ldots,M$.

If $M=1$ go to 7.

6. Add M new nodes $V+1,\ldots,V+M$ and make the following changes to the branching probabilities for $k=1,\ldots,K$; $i=1,\ldots,M$.

$$b'_{v,V+i,k}=0 \qquad v=0,\ldots,V+M$$
$$b'_{V+i,v,k}=0 \qquad v=0,\ldots,V+M$$
$$b'_{u,V+i,k}=\sum_{v\in q(w_i)} b'_{u,v,k}$$
$$b'_{V+i,v,k}=\frac{b'_{u,v,k}}{b'_{u,V+i,k}}, \quad v\in q(w_i)$$
$$b'_{u,v,k}=0 \qquad v\in m(w_i)$$

Note: It is easy to show that $\sum_{v\in m(w_i)} b'_{uvk}$ is constant.

Therefore the new branching probabilities still have the form of equation (27).

Now V+i is the new entry node for p-subnetwork w_i, i=1,...,M.

V=V+M

Go to 2.

7. u is an entry to J≥2 p-subnetworks in S(u).

Determine $w_1 \epsilon S(u)$ and $w_2 \epsilon S(u)$ such that level(w_2)= level(w_1)+1, $m(w_2) \subset m(w_1)$ and there is no $w \epsilon S(u)$ such that level(w)<level(w_1). Determine ℓ such that $m(w_2) \subset m(w_1, \ell)$.

Add a new node V+1 and make the following changes to the branching probabilities for k=1,...,K.

$$b_{v,V+1,k} = 0 \quad , \quad v=0,1,\ldots,V+1,$$

$$b_{V+1,v,k} = 0 \quad , \quad v=0,\ldots,V+1$$

$$\tilde{p}_{u,V+1,k} = \sum_{v \epsilon m(w_1,\ell)} \tilde{p}_{u,v,k}$$

$$b_{u,V+1,k} = \prod_{\substack{\beta \subseteq S(K) \\ k \epsilon \beta}} p'_{m'(w_1,\ell),\beta}[N(m'(w_1,\ell),\beta|\bar{N}] \cdot g'_{m'(w_1),\beta}[N(m'(w_1),\beta|\bar{N})]\, \tilde{p}_{u,V+1,k}$$

$$b_{V+1,v,k} = \frac{\tilde{p}_{u,v,k}}{\tilde{p}_{u,V+1,k}} \prod_{\substack{\beta \subseteq S(K) \\ k \epsilon \beta}} \prod_{\substack{\ell=1 \\ \text{and } v \epsilon m'(w_2,\ell)}}^{L'_{w_2}} p'_{m'(w_2,\ell),\beta}[N(m'(w_1,\ell),\beta|\bar{N}].$$

$$g'_{m(w_2),\beta}[N(m'(w_1),\beta|\bar{N})], \quad v \epsilon m(w_2)$$

$$b_{u,v,k} = 0 \qquad v \epsilon m(w_2)$$

V+1 is the new entry for w_2.

V=V+1

Go to 2.

8. Let S(u) be all the p-subnetworks for which u is the exit. S(u) contains J nodes.

Let $n = \min\{m \mid m = \text{level}(2),\ w \epsilon S(u)\}$

Determine the M>0 p-subnetwork $w_1,,,.w_M$ such that level(w_i)=n where $w_i \epsilon S(u)$, i=1,...,M.

If M=1 go to 10.

9. Add M new nodes V+1,...,V+M and make the following changes to the branching probabilities for k=1,...,K; i=1,...,M

$$b'_{v,V+i,k} = 0 \quad , \quad v=0,\ldots,V+M$$

$$b'_{V+i,v,k} = 0 \quad , \quad v=0,\ldots,V+M$$

$$b'_{v,V+i,k} = b'_{v,u,k} \quad , \quad v \epsilon m(w_i)$$

$$b'_{v,u,k} = 0 \quad , \quad v \epsilon m(w_i)$$

$$b'_{V+i,u,k} = 1$$

V+i is the new exit for p-subnetwork w_i, i=1,...,M.

V=V+M

Go to 2.

10. Determine $w_1 \epsilon S(u)$ and $w_2 \epsilon S(u)$ such that level(w_2)= level(w_1)+1, $m(w_2) \subset m(w_1)$, and there is no $w \epsilon S(u)$ with level(w)<level(w_1).

Determine ℓ_1 such that $m(w_2) \subset m(w_1, \ell_1)$.

Add a new node V+1 and make the following changes to the branching probabilities for k=1,...,K.

$$b'_{v,V+1,k}=0 \quad , \quad v=0,\ldots,V+1$$

$$b'_{V+1,v,k}=0 \quad , \quad v=0,\ldots,V+1$$

$$b'_{v,V+1,k}= b'_{v,u,k} \quad , \quad v \epsilon m(w_2)$$

V+1 is the new exit node for w_2.

V=V+1

Go to 2.

Lemma C.1 The primed network resulting from the application of Algorithm C.1 to the original network has identical arrival rates to the original network for u=1,...,U.

Proof: The proof is obtained by inducting on $n \geq 0$, the number of iterations through the algorithm.

For n=0 iterations, the arrival rates are trivially identical since the network remains unchanged. Consider the network after n>0 iterations. By the induction hypothesis the network after the $(n-1)^{th}$ iteration has arrival rates for the nodes w=1,...,U identical to those for the original network. Let

the network after the $(n-1)^{th}$ iteration have branching probabilities $b_{w,v,k}$, w,v=0,...,V; k=1,...,K. Let the network after the n^{th} iteration have branching probabilities $b'_{w,v,k}$, w,v=0,...,V+L; k=1,...,K where L depends on which path the algorithm took during the n^{th} iteration. The set of equations for the network after the $(n-1)^{th}$ iteration is

$$y_{v,k}= \sum_{w=0}^{V} b_{w,v,k} y_{w,k} \quad v=0,1,\ldots,V; \; k=1,\ldots,K$$

If we assume that the algorithm traversed step 4, and operated on node u, then L=1 and the set of equations for the new arrival rates $y'_{v,k}$ are

$$y'_{v,k}= \sum_{w=0}^{V} b_{w,v,k} y'_{w,k} \quad v=0,\ldots,u-1,u+1,\ldots,V; \; k=1,\ldots,K,$$

$$y'_{u,k}=y'_{V+1,k} \quad k=1,\ldots,K, \text{ and}$$

$$y'_{V+1,k}= \sum_{w=0}^{V} b_{w,u,k} y'_{w,k} \quad k=1,\ldots,K$$

The equation for $y'_{u,k}$ becomes

$$y'_{u,k}= \sum_{w=0}^{V} b_{w,u,k} y'_{w,k} \quad k=1,\ldots,K$$

Therefore the set of equations for the arrival rates for nodes v=1,...,V are identical to the equations for the network after the n^{th} iteration. Thus the arrival rates for the first U nodes are identical. If the algorithm traverses 6, 7, 9, or 10, the method of proof is the same.

We are now able to prove Theorem 4.2.

Theorem 4.2: Queueing networks with branching probabilities of the form (27) have subsystem arrival rates of the form (25).

Proof: We assume there are W>0 p-subnetworks. Otherwise the arrival rates are constant and trivially satisfy equation (25). We assume that the network contains no nodes which are entries or exits to more than one p-subnetwork. Otherwise Algorithm C.1 is applied to obtain a new "equivalent" network satisfying those assumptions.

Take p-subnetwork w,w=1,...,W. Let u=e(w). If s is the exit to p-subnetwork w, define

$$b'_{u,s,k} = \sum_{v \in m(w)} b_{u,v,k} \qquad k=1,\ldots,K$$

Note: It is easy to show that $b'_{u,s,k}$ is constant.

The arrival rate y_{sk}, k=1,...,K is

$$y_{s,k} = \sum_{v \notin m(w)} b_{v,s,k} y_{v,k} + \sum_{v \in m(w)} b_{v,s,k} y_{v,k}$$

The second term of $y_{s,k}$ is just the rate of job flow out of p-subnetwork w and is identical to $b'_{u,s,k} y_{u,k}$, the rate of job flow into p-subnetwork w. Therefore

$$y_{s,k} = \sum_{v \notin m(w)} b_{v,s,k} y_{v,k} + b'_{u,s,k} y_{u,k} \quad , \; k=1,\ldots,K$$

Note that $b_{v,s,k}$ and $b'_{u,s,k}$ are constant.

Now we induct on the level of the p-subnetwork to show that the functional forms are of equation (25). At level 0 (the network level) there is at least one node u such that $u \notin \bigcup_{w=1}^{W} m(w)$. If there is only one node, it must be an entry and exit to the same p-subnetwork, and so all jobs pass through it. Therefore its arrival rate is constant.

Assume there are two or more nodes at level 0 not contained in $\bigcup_{w=1}^{W} m(w)$. The equations for their arrival rates $y_{u,k}$ will be

$$y_{u,k} = \sum_{v=0}^{U} b_{v,u,k} y_{v,k} \quad \text{for u not an exit node}$$

$$y_{u,k} = \sum_{v \notin m(w)} b_{v,u,k} y_{u,k} + b'_{u,u'k} y_{u',k} \quad \text{for u an exit to p-subnetwork w and } e(w)=u'$$

At level 0, all $b_{v,u,k}$'s and $b'_{v,u,k}$'s are constant and so the y_{uk}'s are constant. Therefore they satisfy equation (25).

We now look at a p-subnetwork w with level(w)>0 with $\ell=1,\ldots,L_w$ branches and entry e(w)=u. Let us look at branch ℓ. Define S,

$$S=\{v \mid v \in m(w,\ell), \quad v \notin \bigcup_{w'=1}^{W} m(w') \text{ and level } (w')>\text{level}(w)\}$$

where S contains the nodes which are in the ℓ^{th} branch of p-subnetwork w but not in any higher level p-subnetworks

(excluding entries and exits).

The arrival rate equations for the elements of S are for k=1,...,K

$$y_{v,k} = \sum_{\substack{\beta \subseteq S(K) \\ k \in \beta}} p_{m(w,\ell),\beta}[N(m(w,\ell),\beta \mid \bar{N})] \cdot g_{m(w),\beta}[N(m(w),\beta \mid \bar{N})] \cdot p'_{u,v,k} \cdot y_{u,k} + \sum_{s \in S} b_{s,v,k} y_{s,k}$$

where $v \in S$ and v is not an exit to any p-subnetwork.

$$y_{v,k} = \prod_{\substack{\beta \subseteq S(K) \\ k \in \beta}} p_{m(w,\ell),\beta}[N(m(w,\ell),\beta \mid \bar{N})] \cdot g_{m(w),\beta}[N(m(w),\beta \mid \bar{N})] + \sum_{s \in S} b_{s,v,k} \cdot y_{s,k} + b'_{r,v,k} \cdot y_{s,k}$$

where $v \in S$, v is an exit to p-subnetwork w', $r = e(w')$.

By the induction hypothesis y_{uk} is of the form equation (25) y_{vk} must be of the form

$$y_{v,k} = \sum_{\substack{\beta \subseteq S(K) \\ k \in \beta}} p_{m(w,\ell),\beta}[N(m(w,\ell),\beta \mid \bar{N})] \cdot g_{m(w),\beta}[N(m(w),\beta \mid \bar{N})] \cdot y_{u,k} \cdot y'_{v,k}, \quad v \in S;\ k = 1,\ldots,K.$$

where $y'_{v,k}$'s are the solutions for k=1,...,K to

$y'_{v,k} = p'_{u,v,k} + \sum_{s \in S} b_{s,v,k} y_{s,k}$ for $v \in S$ and v not an exit

and

$$y'_{v,k} = p'_{u,v,k} + \sum_{s \in S} b_{s,v,k} y_{s,k} + b'_{r,v,k} \cdot y_{s,k}$$

for $v \in S$, v an exit to w', and $e(w') = r$.

Since all the coefficients are constant, $y'_{v,k}$ is constant and $y_{v,k}$ is of the form (25).

APPENDIX D

We are interested in modeling a central server network with two I/O's, N identical customers, exponentially distributed service times and FCFS disciplines. The mean service times for the CPU is $1/\mu$ and for each I/O is $1/\lambda$. After exiting the CPU, jobs enter the I/O device with the shortest queue length; otherwise jobs enter either with equal probability.

We can model this problem by a discrete state continuous transition ergodic Markovian process. The equilibrium state probabilities can be determined from the balance equations for the state spaces (see Drake [D1]. Let (n_1,n_2,n_3) be the state of the system where n_1 is the number of jobs in the CPU and n_2 and n_3 are the number of jobs in the two I/O devices. Let $\lfloor X \rfloor$ have as its value the greatest integer not exceeding X. For notation we define $M=\lfloor (N+1)/2 \rfloor$. The constraints on (n_1,n_2,n_3) are $n_1,n_2 \geq 0, n_1,n_2 \leq M$, $n_1+n_2+n_3=N$. Figure D.1 gives the state transition diagram for N=3. Let $P(n_1,n_2,n_3)$ be the probability that the system will be in state (n_1,n_2,n_3) at equilibrium. The balance equations for state (n_1,n_2,n_3) under the above constraints will be

$$[\mu+2\lambda(1-\delta_{0,m})]P(N-m,m,m)=\lambda[P(N-m-1,m+1,m)+P(N-m-1,m,m+1)]$$
$$+\mu[P(N-m+1,m,m-1)+P(N-m+1,m-1,m)],\ m=1,\ldots,M-1,$$

$$[\mu(1-\delta_{0,N-2m-1})+\lambda(2-\delta_{0,m})]. \quad P(N-2m-1,m,m+1)=\lambda[P(N-2m-2,m,m+2)+$$
$$P(N-2m-2,m+1,m+1)]+\frac{\mu}{2}[P(N-2m,m,m)+2P(N-2m,m-1,m+1)],\ m=1,\ldots,M-1,$$

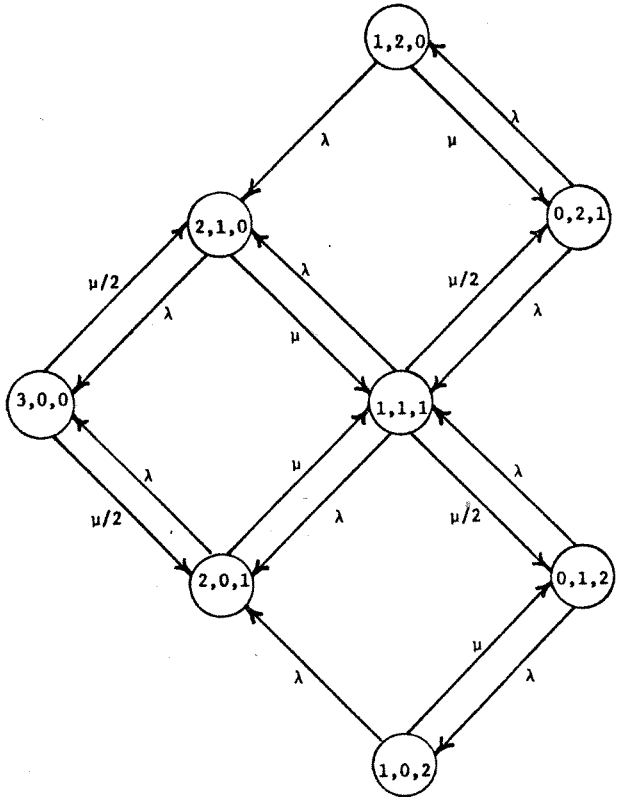


Figure D.1

$$[\mu(1-\delta_{0,N-2m-1})+\lambda(2-\delta_{0,m})]P(N-2m-1,m+1,m)=\lambda[P(N-2m-2,m+1,m+1)+$$

$$P(N-2m-2,m+1,m)]+\frac{\mu}{2}[P(N-2m,m,m)+2P(N-2m,m+1,m-1)],m=1,\ldots,M-1,$$

$$[\mu+\lambda(2-\delta_{0,m})]P(N-m-n,m,n)=\lambda[P(N-m-n-1,m,n+1)+P(N-m-n-1,m+1,n)]$$

$$+\mu P(N-m-n+1,m-1,n),m=0,1,\ldots,M-2;n=m+2,\ldots,M,$$

$$[\mu+\lambda(2-\delta_{0,m})]P(N-m-n,n,m)=\lambda[P(N-m-n-1,n+1,m)+P(N-m-n-1,n,m+1)]$$

$$+\mu P(N-m-n+1,m-1,n),m=0,1,\ldots,-2;n-m+2,\ldots,M,$$

where $P(n_1,n_2,n_3)$ is assumed to be identically zero for impossible states.

If $N=2\cdot M$ then there is an additional equation

$$2\lambda P(0,M,M)=\mu[P(1,M-1,M)+P(1,M,M-1)]$$

Finally, since the above set of equations are dependent, it is necessary to substitute the normalization criterion

$$\sum_{\substack{\text{all feasible}\\ \text{states } (n_1,n_2,n_3)}} P(n_1,n_2,n_3) = 1$$

for one of the balance equations. This is the set of equations we solve numerically to obtain CPU throughput, and utilization for the above model.

APPENDIX E

Algorithm 5.4 Determination of $P(N+j,j,0,1),j=0,1,\ldots,N$

(assume N>1)

Through most of this algorithm we will be dealing with column vectors of length N+1. We will let e_i represent the column vector with all elements 0 except the i^{th} element, which is 1.

1. Initialization
   a. S=0
   b. For j=0,1,...,N, do step 1b.

      $P(N+j,j,0,1)=e_j$

      $S=S+P(N+j,j,0,1)$
   c. $P(2N-1,N-1,1,2)=P(2N,N,0,1)\lambda/\mu_2$

      $P(2N-2,N-1,1,1)=P(2N-1,N-1,1,2)(\lambda+\mu_2)/p\mu_1)$

      $P(2N-2,N-1,1,2)=P(2N-1,N-1,1,2)\lambda/((\lambda+\mu_2)(2N-1))$
   d. For j=N-2,...,0, do step 1d.

$$P(N+j,j,1,2)=\frac{\lambda(P(N+j+1,j+1,0,1)-P(N+j+2,j+2,0,1)}{\mu_2}$$

$$-P(N+j,j+1,1,2)$$

$$P(N+j-1,j,1,1)=(P(N+j,j,1,2)(\lambda+\mu_2)-P(N+j+1,j+1,1,2)\cdot$$

$$2(j+1)\lambda/(N+j+1))/(p\mu_1)$$

$$P(N+j-1,j,1,2)=\frac{\lambda(P(N+j,j,1,2)/(N+j)+P(N+j,j+1,1,2))}{(\lambda+\mu_2)}$$

$S=S+P(N+j,j,1,2)+P(N+j-1,j,1,1)+P(N+j-1,j,1,2)$

2. For $j=N-2,\ldots,0$, do step 2.

   a. $P(2j+1,j,N-j,2)=(P(2(j+1),j+1,N-j-1,1)(\lambda+\mu_1)-$
   $P(2j+3,j+1,N-j-2,1)\lambda/(2j+3))/\mu_2$

   $P(2j,j,N-j,1) =$

   $$\frac{(P(2j+1,j,N-j,2)(\lambda+\mu_2)-P(2(j+1),j,N-j-1,2)\lambda/(2(j+1)))}{p\mu_1}$$

   $P(2j,j,N-j,2) =$

   $$\frac{\lambda(P(2j+1,j,N-j,2)/(2j+1)+P(2(j+1),j,N-j-1,2)/(2(j+1)))}{(\mu_2+\lambda)}$$

   $S=S+P(2j+1,j,N-j,2)+P(2j,j,N-j,1)+P(2j,j,N-j,2)$

   b. For $n=j-1,\ldots,0$, do 2b.

   $P(N+n,n,N-j,2)=$
   $(P(N+n+1,n+1,N-j-1,1)(\lambda+\mu_1)$
   $-P(N+n+2,n+1,N-j-2,1)\lambda(N-n)/(N+n+2)$
   $-P(N+n+2,n+2,N-j-1,1)2(n+2)\lambda/(N+n+2)$
   $-P(N+n,n+1,N-j,2)\mu_2-P(N+n,n+1,N-j,1)p\mu_1)/\mu_2$

   $P(N+n-1,n,N-j,1)=$
   $(P(N+n,n,N-j,2)(\lambda+\mu_2)$
   $-P(N+n+1,n,N-j-1,2)\lambda(N-n)(N+n+1)$
   $-P(N+n+1,n+1,N-j-1,2)2(n+1)\lambda/(N+n+1))/(p\mu_1)$

   $P(N+n-1,n,N-j,2)=$
   $(P(N+n,n,N-j-1,2)(N-n)/(N+n)$
   $+P(N+n,n+1,N-j-1,2)(n+1)+P(N+n-j,2)$

   $S=S+P(N+n,n,N-j,2)+P(N+n-1,n,N-j,1)$
   $+P(N+n-1,n,N-j,2)$

3. Determination of $P(2N,N,0,1),\ldots,P(N,0,0,1)$ as scalars

   a. $S=S-P(0,0,N,2)$

   $P(0,0,N,2)=P(0,0,N,2)\mu_2/(\lambda+\mu_2)$

   $S=S+P(0,0,N,2)$

   b. For $n=1,\ldots,N-1$, do 3b.

   $D_n=P(N-n,0,n,0,1)(\mu_1+\lambda)-P(N-n+1,0,n-1,1)\lambda$
   $-P(N-n+1,1,n,1)2\lambda/(N-n+1)-P(N-n-1,0,n+1,2)\mu_2$

   c. $D_N=P(0,0,N,1)\mu_1-P(1,0,N-1,1)\lambda$

   d. Solve

$$\begin{bmatrix} D_1^T \\ \cdot \\ \cdot \\ \cdot \\ D_N^T \\ S^T \end{bmatrix} x = \begin{bmatrix} 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \\ 1 \end{bmatrix}$$

   For $n=0,\ldots,N$ $\quad P(N+n,n,0,1)=X_{n+1}$

Our computer algorithm allows λ to be dependent on the queue length of the I/O queue.

BIBLIOGRAPHY

B1. Baskett, F., Mathematical Models of Multiprogrammed Computer Systems. TSN-17, Computation Center, The University of Texas at Austin, 1971.

B2. Baskett, F., Chandy, K. M., Muntz, R. R., and Palacios-Gomez, F., "Open, Closed and Mixed Networks of Queues with Different Classes of Customers", JACM 22, 2 (1975), pp. 248-260.

B3. Brown, R. M., An Analytic Model of a Large Scale Interactive System Including the Effects of Finite Main Memory, TR-31, Department of Computer Sciences, University of Texas at Austin, 1974.

B4. Browne, J. C., Chandy, K. M., Brown, R. M., Keller, T. K., Towsley, D. F., and Dissly, C. W., "Hierarchical Techniques for Development of Realistic Models of Complex Computer Systems", Proceeding of the IEEE 63, 6 (June 1975), pp. 966-975.

B5. Buzen, J., Queueing Network Models of Multiprogramming, Ph.D. Dissertation, Division of Engineering and Applied Physics, Harvard University, 1971.

C1. Chandy, K. M., "The Analysis and Solutions for General Queueing Networks", Proceedings Sixth Annual Princeton Conference on Information Sciences, Princeton University, 1972.

C2. Chandy, K. M., Herzog, U., and Woo, L., "Parametric Analysis of Queueing Network Models", IBM Journal of Research and Development 19, 1 (January 1975), pp. 36-42.

C3. Chandy, K. M., Herzog, U., and Woo, L., "Approximate Analysis of General Queueing Networks", IBM Journal of Research and Development 19, 1 (January 1975), pp. 43-49.

C4. Chandy, K. M., Howard, J. H. and Towsley, D. F., "Station Balance", submitted JACM, 1975.

C5. Chen, P. P. S., "Queueing Network Model of Interactive Computing Systems", Proceedings of the IEEE 63, 6 (June 1975), pp. 954-957.

C6. Computation Center, User's Manual, Computation Center, University of Texas at Austin, 1974.

C7. Control Data Corporation, A Simulation Process-Oriented Language (ASPOL) Reference Manual, publ. 17314200, 1972.

C8. Cotten, L. W., and Abd-Alla, A. M., "Processing Times for Segmented Jobs with I/O Compute Overlap", JACM 21, 1 (1974), pp. 18-30.

C9. Courtois, P. J. and Georges, J., "On a Single Server Finite Queueing Model with State Dependent Arrival and Service Processes", Operations Research, (1971) pp. 424-434.

D1. Drake, A. W., Fundamentals of Applied Probability Theory, McGraw-Hill Book Company, 1967.

F1. Foster, D. V., File Assignment in Memory Hierarchies, TR-48, Department of Computer Sciences, University of Texas at Austin, 1975.

F2. Fultz, G. L., Adaptive Routing Techniques for Message Switching Computer-Communication Networks, Computer Science Department, University of California at Los Angeles, 1972.

G1. Gaver, D. P., "Probability Models for Multiprogrammed Computer Systems", JACM 14, 3 (1967), pp. 423-438.

G2. Gaver, D. P. and Shedler, G. S., "Approximate Models for Processor Utilization in Multiprogrammed Computer Systems", SIAM Journal of Computing 2, 3 (1973), pp. 183-192.

G3. Gordon, W. J. and Newell, G. F., "Closed Queueing Systems with Exponential Servers", Operations Research 15, 2 (April 1967), pp. 254-265.

H1. Hellerman, H., and Smith, H. J., Jr., "Throughput Analysis of Some Idealized Input, Output, and Compute Overlap Configurations", Computing Surveys 2, 2 (June 1970), pp. 111-118.

H2. Herzog, U., Woo, L., and Chandy, K. M., "Solution of Queueing Problems by a Recursive Technique", IBM Journal of Research and Development 19, 3 (May 1975), pp. 295-300.

I1. Information Research Associates (IRA), "A Network Model for a CDC CYBER-70 Computer System under the SCOPE 3.4 Operating System", Austin, Texas, 1975.

J1. Jackson, J. R., "Jobshop-like Queueing Systems", Management Science 10, 1 (1963), pp. 131-142.

K1. Keller, T. K., Models of Computer Systems with Passive Resources, Forthcoming Ph.D. Dissertation, Department of Computer Sciences, University of Texas at Austin.

K2. Kobayashi, H., "Applications of the Diffusion Approximation to Queueing Networks I: Equilibrium Queue Distributions", JACM 21, 2 (1974), pp. 316-328.

K3. Kobayashi, H., "Applications of the Diffusion Approximation to Queueing Networks II: Nonequilibrium Distributions and Applications to Computer Modeling", JACM 21, 3 (1974), pp. 459-469.

L1. Lavenberg, S. S., Efficient Estimation via Simulation of Work-Rates in Closed Queueing Networks, IBM Research Report, RJ1390, 1974.

L2. Leitch, D. F., Tools Useful in the Study of Queues and Queueing Networks, M.A. Thesis, Department of Computer Sciences, University of Texas at Austin, 1973.

L3. Little, J. D. C., "A Proof for the Queueing Formula L=λw", Operations Research 9 (1966), pp. 383-387.

M1. Mood, A. M. and Graybill, F. A., Introduction to the Theory of Statistics, McGraw Hill, 1963.

M2. Morse, P. M., Queues, Inventories and Maintenance, John Wiley and Sons, 1958.

M3. Muntz, R. R., Poisson Departure Processes and Queueing Networks, IBM Research Report RC4145, 1972.

M4. Muntz, R. R. and Wong, J., "Asymptotic Properties of Closed Queueing Network Models", Proceedings Eighth Annual Princeton Conference on Information Sciences and Systems, 1974.

R1. Reiser, M. and Kobayashi, H., "Queueing Networks with Multiple Closed Chains: Theory and Computational Algorithms", IBM Journal of Research and Development 19, 3 (May 1975), pp. 283-294.

S1. Sauer, C. H., Configuration of Computing Systems: An Approach Using Queueing Network Models. Ph.D. Dissertation, Department of Computer Sciences, University of Texas at Austin, 1975.

S2. Shedler, G. S., A Cyclic Queue Model of a Paging Machine, IBM Research Report, RC 2814, Yorktown Heights, New York, March 1970.

S3. Smith, W. L., "Renewal Theory and Its Ramifications", J. Roy. Statist. Soc. B 20, (1958) pp. 243-302.

T1. Thornton, J. E., The Design of a Computer, The Control Data 6600, Glenview, Illinois, Scott-Foresman, 1970.

W1. Wallace, V. L. and Rosenberg, R. S., "Markovian Models and Numerical Analysis of Computer System Behavior", Proceedings Spring Joint Computer Conference, 1966.