Optimal Routing in M/M/1/K and M/M/C/K Queueing Networks with Production Blocking

Hemant D. Gosavi
J. MacGregor Smith *
email:gosavi@ecs.umass.edu jmsmith@ecs.umass.edu

March 29, 1997

Abstract — **In an earlier paper we developed a dynamic programming algorithm to obtain the suboptimal routing vector in directed, acyclic, tree networks. This algorithm works well if the first buffer in each parallel stream after a split junction is relatively large or infinite. However the algorithm has limited application to networks where these buffers are finite and can cause blocking across the splits. In this paper we examine the routing problem in finite queueing networks where blocking can occur across split junctions. We describe the departure processes from a series of finite nodes and utilize these results in quantifying the arrivals to the split junction. The blocking phenomenon causes dependence among the different parallel streams resulting in interrupted arrival processes to individual streams. We propose a method to determine each interrupted arrival process and construct an approximate model for the throughput of such systems. Finally we develop an algorithm which utilizes Powell's unconstrained optimization algorithm to solve for the suboptimal routing probabilities in order to maximize throughput. The algorithm is robust and the results are verified with the help of extensive analytical and simulation experiments.**
Keywords: Open Networks, Throughput, Optimal Routing Probability

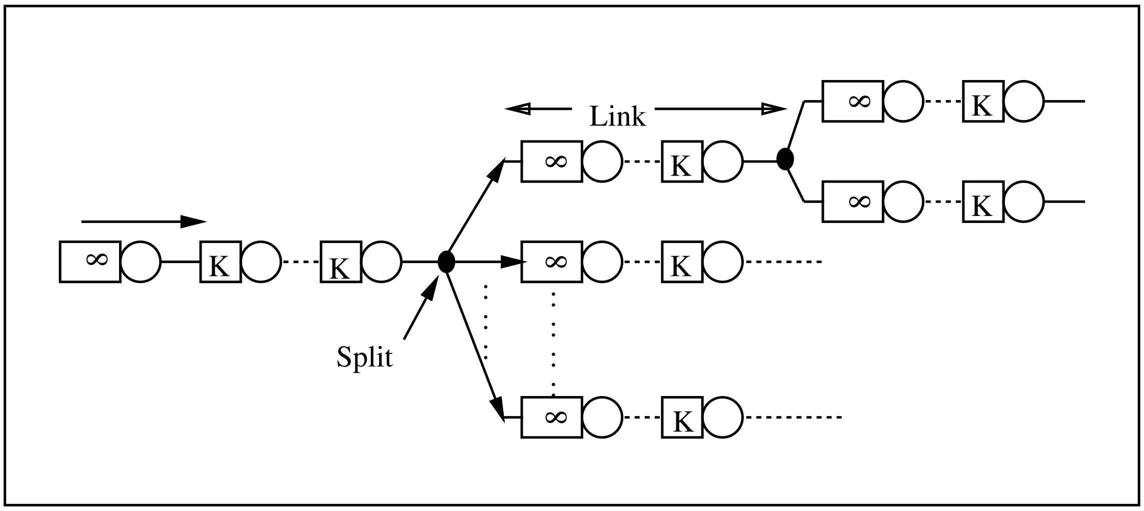

Figure 1: A tree topology network with split junctions

*Department of Mechanical and Industrial Engineering, University of Massachusetts, Amherst MA 01003

1 Introduction

Finite queueing networks are notoriously difficult for analytical treatment and closed form expressions are not easily constructed for the performance measures of such systems. Due to the absence of closed form expressions, the optimization problems associated with finite queueing networks are formidable for treatment by conventional optimization techniques. There are several optimization problems associated with finite queueing networks such as the *the optimal topology network design problem*, *the optimal routing network design problem*, and the *the optimal capacitated network design problem*[14].

- *Topological network design* ($\mathcal{TND}$): is concerned with the number and location of *nodes* (e.g. workstations, warehouses, shipment and delivery points), identification of their interconnecting *edges* (e.g. corridors, conveyors, elevators, tow lines, escalators and lifts) and the general configuration of both components.
- *Routing network design* ($\mathcal{RND}$): is concerned with the sequencing and routing of various units (e.g. products, vehicles, customers) across the network.
- *Capacitated network design* ($\mathcal{CND}$): is concerned with the optimal sizing of nodes and edges to accommodate requisite flow volumes.

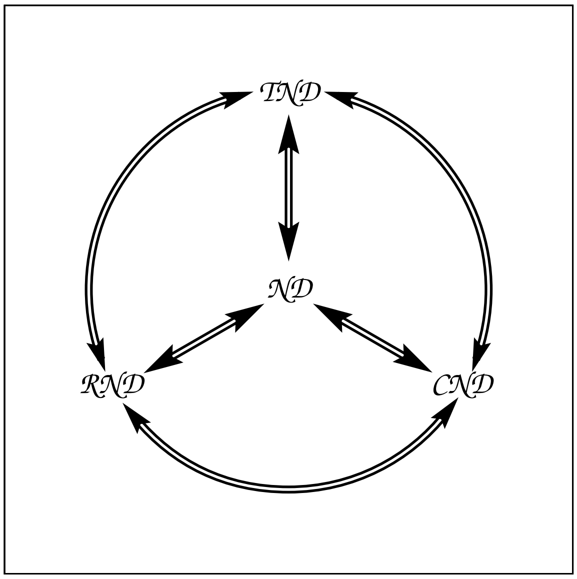

Figure 2: Network Design Problems

In this paper we examine the problem of optimal static routing in networks with split junctions in order to maximize system throughput.

The problem of optimal routing has many dimensions, and there is a wide variety of routing schemes that have been explored in the literature, whose performance and flexibility depend upon the topology and resources of the queueing network and the type of information required during their implementation. Daskalaki distinguishes three cases of the routing problem[5]:

1. *The static routing problem, where the topology and network resources are specified, and the problem is to determine the proportion α_{ij} of the traffic of customers at node i that will be directed to node j.*
2. *The quasi-dynamic routing problem, where the information about topology and traffic rates is periodically updated, and the static routing scheme is applied during a given interval.*
3. *The dynamic routing problem, where changes in topology, traffic rates and in routing assignments is possible. In this problem one needs to determine the time dependent routing vectors, which depend on the state of the system.*

Most of the literature on routing deals with data communication networks [3]. In general, the objective function for the routing algorithm is expressed in some functional form and the derivatives are used in the optimization process, with the help of stochastic approximation techniques[5].

In an earlier paper[6], we analyzed the optimal routing problem for the class of networks shown in Figure 1. It may be noted that in this class of networks, the first buffer immediately after a split junction in every parallel stream is assumed to be infinite because there is no blocking across the split junctions. We have developed a dynamic programming algorithm to yield the suboptimal routing probabilities for such networks. In developing this algorithm we have used the concept of *throughput capacity* which was also developed in an earlier paper[7]. Since we are only able to approximate throughput capacities, the algorithm can only yield *suboptimal* routing probabilities to maximize the throughput.

The development of Flexible Manufacturing Systems (FMS) opened a new application area for the routing problem. Cassandras[4] proposed a routing control scheme based on hierarchical control levels. The flow pattern was described by solving a linear program, which was then applied to the network using a proposed routing strategy. Yao[15] proposed a probabilistic shortest queue discipline, which sends customers with the highest probability to the shortest queue.

Kelly[12] has proposed a routing scheme where a customer blocked on one route may try another route, which also gives rise to overflow schemes wherein a server receives the overflow of another server. Houck[9] has applied simulation to study the threshold values that should be set for different servers, however, it is almost impossible to complete an exhaustive search along these lines.

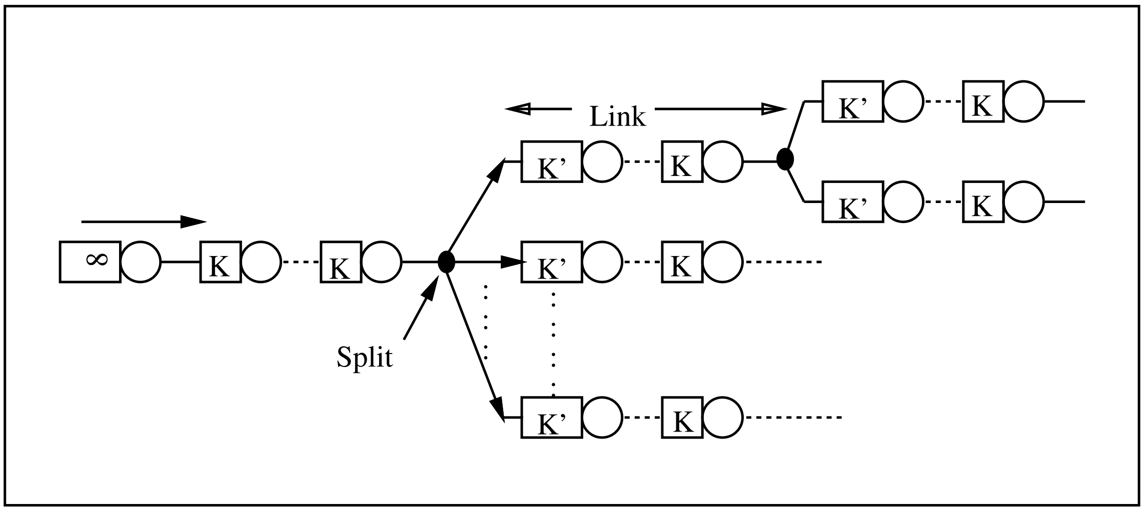

Figure 3: A tree topology network with blocking across split junctions

The motivation for this paper stems from the limitations of the previous algorithm to a more general class of networks as shown in Figure 3. The previous algorithm works very well in instances where the first buffers K are relatively large and cause minimal blocking across the split junctions. However, if the first buffers in each parallel stream are small, they can cause blocking across the split junctions resulting in interrupted arrival processes to each parallel stream. In this paper we will examine the routing problem in such networks where the blocking in one stream interrupts arrivals to the other streams, which in turn causes dependencies among all the parallel streams. It may be noted that in the networks of Figure 1 we were able to isolate each parallel stream for analysis within the algorithm.

Such networks are frequently encountered in practice, in single and multi-commodity flow systems where arriving customers are not lost and there is a unidirectional, divergent flow along the arcs of the network. A common application is in the modeling of high volume production systems, where there is divergent material flow and one has to select a scheme to route parts along any of several sequences of parallel processing stations with finite waiting spaces. If there are finite buffers at the processing stations immediately after a split junction, these are likely to block the stations immediately preceding the split junction.

Figure 4 shows a typical situation where the static routing problem is obvious. Raw material is processed at each of stations 1 and 2, where operations common to two different products A & B are performed. The final two operations are different on the two products, each of which have a dedicated set of processing stations. The problem is to determine the right proportion of the two products to be produced, so that some profit function is maximized. Since each station has finite buffers there is blocking within each link, as well as across the split junction. It is easily realized that determining the optimal ratio of the two products is equivalent to finding the optimal routing probabilities along each arc leading to the parallel streams of service stations. In this paper we will assume that the profit function is a function of

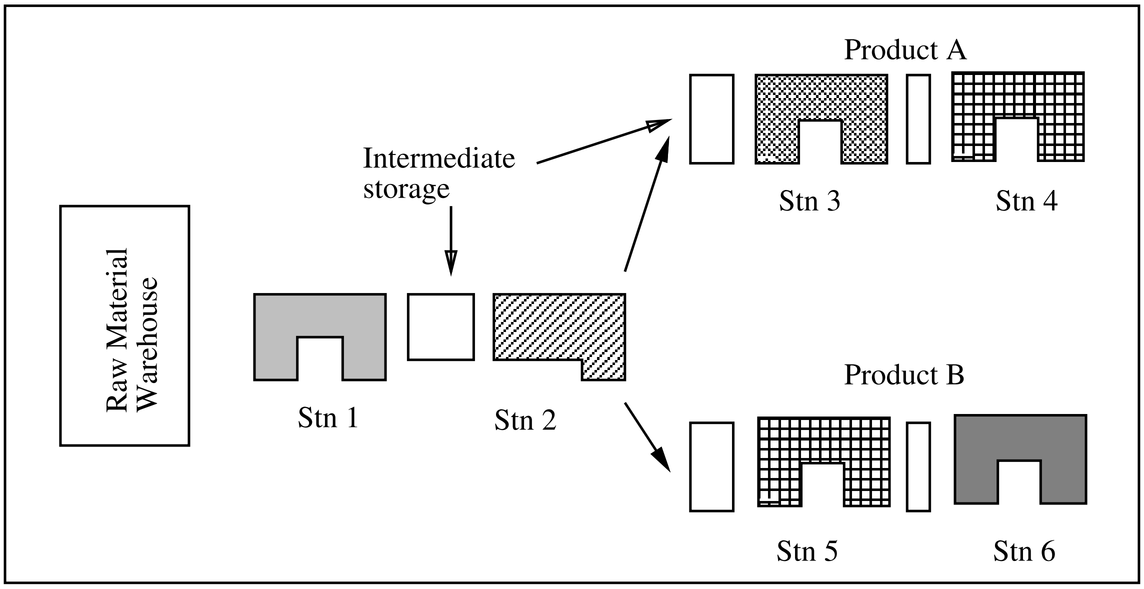

Figure 4: An application of the routing problem

the total throughput of the network.

We will now introduce the notation and discuss the assumptions made in this paper. We will analyze the routing problem for elementary networks of the type shown in Figure 5. We can later integrate such networks to solve tree topology networks with more than one split.

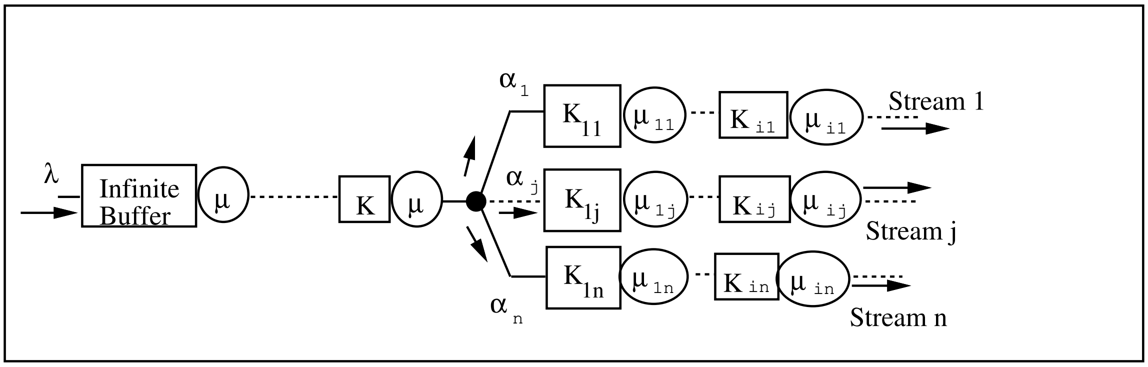

Figure 5: A Network with one split junction

The following notation is used throughout the paper.

Λ = Mean arrival rate outside the system.
$\tilde{\lambda}_j$ = Mean arrival rate to stream j.
$\tilde{\lambda}$ = Mean arrival rate to the split, $\tilde{\lambda} = \sum_{j=1}^{n} \tilde{\lambda}_j$
μ_{ij} = Mean service rate of server i in parallel stream j,where $i = 1, .., m_j, j = 1, ..., n$.
K_{ij} = Finite Buffer size at server i in parallel stream j, where $i = 1, .., m_j$, $j = 1, ..., n$.
α_j = Routing Probability to parallel stream j.
α_j^* = Suboptimal routing probability to stream j.
θ_j = Mean Throughput rate from parallel stream j.
Θ = $\sum_{j=1}^{n} \theta_j$= Total system throughput.
θ_j^* = Maximum throughput or Asymptotic Bound

on θ_j.
θ_{UB_j} = Upper Bound on Throughput Rate in stream j.

The assumptions made are summarized below:

1. *Arrivals to the network are stored in an infinite buffer until they can proceed to the first finite queue.*
2. *Queues are arranged in a split topology with parallel streams of servers, with finite buffers preceding each server.*
3. *Production Blocking will occur if a buffer is saturated, i.e. server i will be blocked if it completes service on a customer, but the customer cannot proceed to server $i + 1$ due to the next buffer being saturated.*
4. *We will assume that service rates and arrival rates are drawn from an exponential distribution. Therefore, the first queue where arrivals occur is $M/M/1/\infty$, while all the other queues are $M/M/1/K$.*
5. *At each station the service discipline is First-Come First-Served (FCFS).*
6. *Routing within the network is deterministic or static. A routing probability is assigned along each arc of the network. Therefore, if the first finite buffer in a parallel stream is saturated, and the next customer at a split junction is assigned to the same stream, the other parallel streams will be starved until this customer can be released.*

The optimal routing problem may now be stated more formally for networks with one split junction as follows:

Determine the optimal values of α_j in a split topology network, with a single split junction and n parallel streams of servers, where the first server is $M/M/1/\infty$ and all other servers are $M/M/1/K$ or $M/M/C/K$, such that the total throughput from the network $\Theta = \sum_{j=1}^{n} \theta_j$ is maximized. Obviously,

$$\sum_{j=1}^{n} \alpha_j = 1.$$

2 The departure process of a tandem network

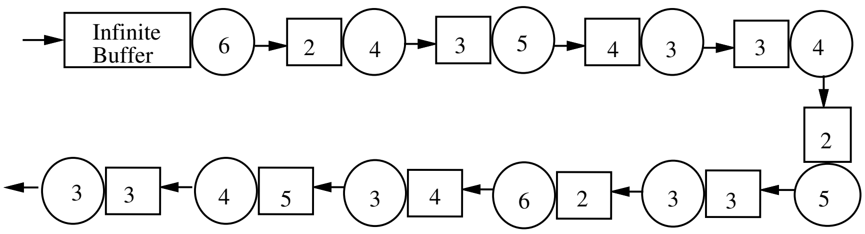

Figure 6: A network with 11 tandem queues

In order to determine the nature of arrivals to the split junction shown in Figure 5, one needs to evaluate the departure process from the last node in the sequence of tandem queues before the split junction. We will assume that all servers in the series of tandem queues have exponential service rates and that the inter-arrival times to the network are also drawn from an exponential distribution. The blocking within the network creates dependencies between the different nodes and it is not clear what distribution the departure process might take.

In the following set of experiments, we have quantified the departure process from a network of 11 tandem queues for a range of arrival rates. Figure 6 shows the system of 11 tandem queues. The average

#	λ	# Cust.	Max Value	time τ	Std Dev	Mean Θ
1	1.00	3405	520.1	60.1638	61.6796	0.9973
2	2.00	6721	235.5	30.4808	29.9081	2.0061
3	2.20	7170	332.8	27.2727	26.7806	2.2000
4	2.28	7041	437.0	26.7219	27.0330	2.2453
5	2.50	6847	306.7	26.5678	25.9554	2.2584
6	2.75	6813	236.7	26.4880	26.0032	2.2652
7	3.00	6745	332.9	26.5548	25.8236	2.2595
8	3.50	6683	241.8	26.3708	25.2930	2.2752

Table 1: Details of the departure process over 200000 time units

throughput rates measured over 200000 time units are shown in Table 1, where it may be observed that the mean and the standard deviation is approximately equal over different arrival rates. The discretized probability density function and cumulative distribution function of the mean inter-departure times (τ) are plotted for different arrival rates in Figures 21, 22, 23, 24 and 25 which are included as an appendix to this paper. The departure process is also quantified by the average throughput rate (Θ). Finally, the experimental cumulative distribution function (cdf) is compared to the cdf of a theoretical exponential distribution with parameter Θ.

#	λ	intervals k	d.f. ν	χ_0^2	$\chi^2_{0.05,\nu}$	Result
1	1.00	62	60	69.4371	79.1	Accept
2	2.00	82	80	95.5038	101.9	Accept
3	2.20	92	90	112.8188	113.1	Accept
4	2.28	92	90	98.5846	113.1	Accept
5	2.50	102	100	108.5483	124.3	Accept
6	2.75	102	100	87.9902	124.3	Accept
7	3.00	102	100	105.3804	124.3	Accept
8	3.50	102	100	102.5869	124.3	Accept

Table 2: Chi Square Test on inter-departure times of tandem network (Figure 4)

We tested the hypothesis that the inter-departure times indeed conform to exponential distributions with the help of the Chi-Square Test. The data from each of the above experiments was classified into class intervals that were equal in expected probability[2]. For each interval, the expected number of observations (based on the average departure time for each experiment) and the actual number of observations were recorded, and this information was used to compute the χ_0^2 test statistic. Finally, χ_0^2 was compared with $\chi^2_{0.05}, k-s-1$, where k is the number of class intervals and s represents the number of parameters of the theoretical distribution ($s = 1$ for the exponential distribution).

Table 2 shows the results of the χ_0^2 test for experiments with different values of λ. In each case, regardless of the traffic intensity, we were able to accept the hypothesis that the inter-departure times were exponentially distributed at a level of significance of $\alpha = 0.05$. Moreover, it was also observed that there was little auto-correlation (with several lags) between the inter-departure times.

We can therefore conjecture that the arrivals to the split junction are also approximately exponential and hence we may be able to use $M/M/1/K$ and $M/M/C/K$ standard results in computing the blocking probabilities at the split. This probably occurs under our assumption of Poisson arrivals to the systtem itself, but may not hold under more general arrival processes.

3 Determination of arrival processes into individual streams

Let us refer to the network in Figure 5 once again. We now know that the mean arrival rate to the split junction ($\tilde{\lambda}$) is equal to Θ (the throughput rate of the tandem sub-network prior to the split junction), and that the arrivals to the split junction are approximately Poisson.

However, it may be recalled that the last server prior to the split junction can be blocked if the customer that has just been served has been routed to a parallel stream whose first buffer is saturated. Notice again that this phenomenon is typical of deterministic routing situations, and during this period of blocking no departures can take place from the blocked server. Therefore, the individual arrivals into each parallel stream are dependent on the arrivals into the split junction as well as the blocking caused by the parallel streams.

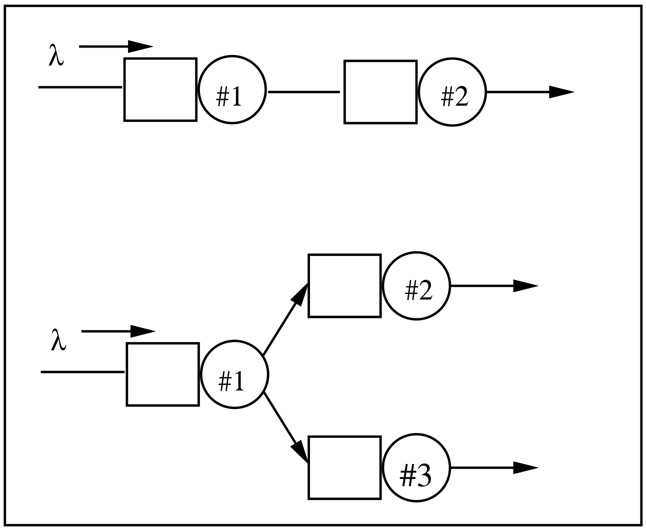

Figure 7: Interrupted arrival processes

It may now be seen that the individual arrival processes into each parallel stream are interrupted processes, due to the blocking caused by all the other parallel streams. This phenomenon can be intuitively explained with the help of Figure 7. If we consider a two node tandem network as shown in this figure, it is clear that the departure rate from node 1 is the arrival rate into node 2. However, if there is a split junction across which blocking can occur, then the departures from node 1 destined to be routed to node 2 are interrupted by the blocking caused by node 3. Similarly, those departures routed to node 3 are blocked by saturation at node 2. These interruptions may be estimated by the respective blocking probabilities at those nodes.

Therefore, if we approximate pk_i as the blocking probability at the split junction due to saturation in the ith parallel stream, then the blocking probability causing an interruption in the arrivals to the ith stream may be represented by $\sum_{j \neq i} pk_j = \sum_{j=1}^{n} pk_j - pk_i$. Therefore, the mean arrival rate into the ith stream may be represented by

$$\tilde{\lambda_i} = \tilde{\lambda}\alpha_i(\sum_{i=1}^{n} pk_i - pk_i) \tag{1}$$

where $\tilde{\lambda}$ is the arrival rate to the split junction.

If each parallel stream consists of only one exponential server as seen in Figure 8, then we have approximate $M/M/1/K$ queues in each stream. Therefore, in this case one may approximate the blocking probability by the following equations:

$$pk_i = \frac{(1-\rho_i)\rho_i^{K_i+1}}{1-\rho_i^{K_i+2}} \tag{2}$$

where

$$\rho_i = \frac{\tilde{\lambda_i}}{\mu_i} \tag{3}$$

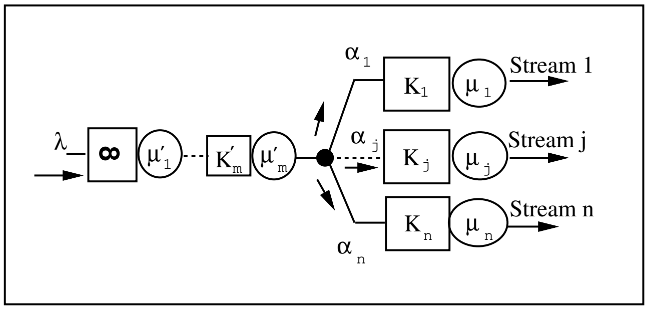

Figure 8: An elementary split network

Notice that in Equation 2 the buffer sizes have been augmented by 1 to account for the blocked server. Finally, the throughput of a typical network shown in Figure 8 may be estimated by

$$\hat{\Theta} = \tilde{\lambda}(1 - \sum_{i=1}^{n} pk_i) \tag{4}$$

Equations 1, 2 and 3 represent a system of n non-linear equations, and given a vector $(\alpha_i, \;\; i = 1, .., n)$ such that $\sum_{i=1}^{n} \alpha_i = 1$, there exists a solution $pk_i, \;\; i = 1, .., n$ which yields the exact blocking probabilities due to saturation in each parallel stream.

4 The Optimization Routine

It is clear that given α_i, $i = 1, ..n$, we can solve Equations 1, 2 and 3 to yield pk_i, $\forall i$ and consequently $\hat{\Theta}$. Therefore, a model of the optimal routing problem may be stated as follows:

$$\begin{aligned}
\textbf{Maximize } \hat{\Theta} &= \tilde{\lambda}(1 - \sum_{i=1}^{n} pk_i) \\
\textbf{s.t.} & \\
pk_i &= \frac{(1-\rho_i)\rho_i^{K_i+1}}{1-\rho_i^{K_i+2}} \;\; \forall \;\; i = 1, .., n \\
\rho_i &= \frac{\tilde{\lambda}_i}{\mu_i} \;\; \forall \;\; i = 1, .., n \\
\tilde{\lambda}_i &= \tilde{\lambda}\alpha_i(\sum_{i=1}^{n} pk_i - pk_i) \;\; \forall \;\; i = 1, .., n \\
\sum_{i=1}^{n} \alpha_i &= 1
\end{aligned}$$

The constraints represented by Equations 1, 2 and 3 are n simultaneous non-linear equations in n variables, and hence need to be solved for every value of the decision vector α_i, which satisfies the final constraint.

Figure 9 represents the basic structure of the *MINBLOCK* algorithm which we have developed to solve the above maximization problem.

Once we have a routing vector $\alpha_i, \;\; i = 1, .., n$, then we can solve the non-linear simultaneous equations to obtain the blocking probabilities due to each stream i. We have used the IMSL subroutine NEQNF to solve these equations. NEQNF solves the system of non-linear equations using the Levenberg-Marquardt algorithm and a finite difference approximation to the Jacobian[10].

The blocking probabilities thus obtained are then used to compute the objective function value. Powell's method[13][8] is then used to generate a new set of conjugate directions (new α_i vectors) depending on

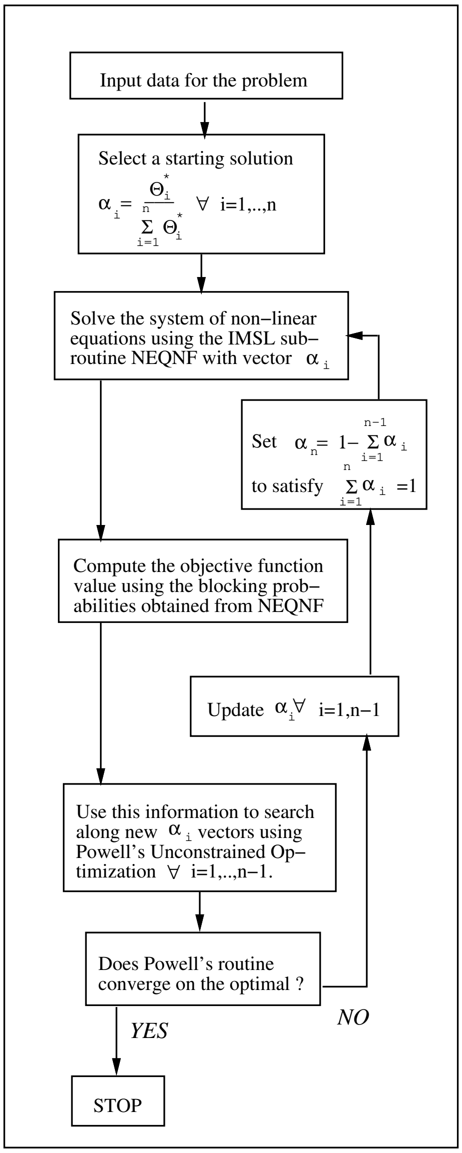

Figure 9: Structure of the *MINBLOCK* algorithm

the value of the objective function. These multi-dimensional searches are continued until the optimal value of the objective function is reached. It must be mentioned here that Powell's method is essentially an unconstrained optimization routine, whereas we have adapted it here for a constrained optimization problem. Powell's method finds the minimum of $f(\bar{\alpha})$ of a non-linear function by using unidimensional searches from an initial point $\alpha_0^{(k)}$ along a set of conjugate directions generated by the procedure itself. The idea behind this method rests mainly on the concept that if the minimum of a quadratic function is found along each of p conjugate directions in one stage of the search, and if a suitable step is made in each of the p directions, the overall step from beginning to the p^{th} step is conjugate to all of the p subdirections of the search. The advanatage of Powell's algorithm is that it exhibits quasi-quadratic convergence which means that for a quadratic objective function, Powell's method yields quadratic convergence. This is advantgeous to us here in that the concavity of the objective function would be closely approximated by a quadratic in the vicinity of an optimum.

We shall illustate this in the experimental section, but offer no proof of the concavity of the objective function. A very detailed discussion of Powell's method is not being included here, however, for an in-depth treatment of this subject area refer to [13]. We have built in the constraint requiring the sum of routing probabilities as follows: Powell's routine generates the search directions α_i only for $i = 1, ..n-1$. We then set $\alpha_n = 1 - \sum_{i}^{n-1} \alpha_i$. Since the starting solution is computed as $\alpha_i = \frac{\theta_i^*}{\sum_i^n \theta_i^*}$, we have a starting solution in the vicinity of the optimal. If the step size in the Powell's method is chosen to be appropriately small, the method searches only along feasible directions and the optimal solution is quickly reached due to the concavity of the objective function.

The *MINBLOCK* algorithm, which is coded in FORTRAN 77, is extremely efficient and yields the optimal routing vector with very little computation time, when compared with a search procedure based on simulating the network with different routing vectors.

5 Experimental Results

5.1 One split junction and two parallel servers

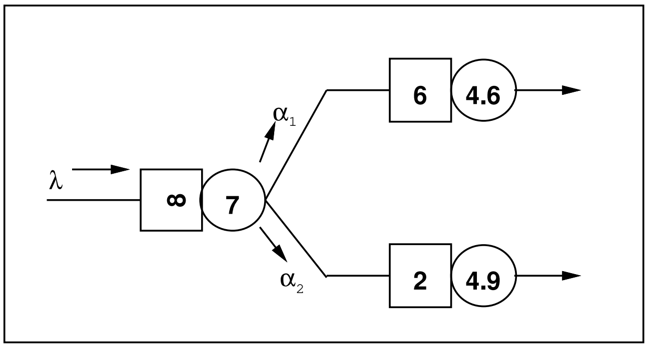

Figure 10: *Network A*: Network with 2 parallel servers

Θ analytical	α_1	α_2
5.8796282	0.48421052	0.51578951
6.0175390	0.58148593	0.41851407
6.0175395	0.58148593	0.46851408
6.0175447	0.58203834	0.41796166
6.0175447	0.58213836	0.41786164
7 Function Evaluations Within Powell Routine And 25 Function Evaluations During The Line Searches.		
Optimal Values		
$\mathbf{\Theta = 6.0175447}$	$\mathbf{\alpha_1 = 0.58213836}$	$\mathbf{\alpha_2 = 0.41786164}$

Table 3: Network A: Path taken by the MinBlock Program

In the experiment shown in Figure 10, a network with one split junction and two parallel servers was evaluated. The MinBlock algorithm was used to predict the suboptimal routing probabilities and the expected throughput rate. The path taken by the algorithm to search for the optimal solution is shown in Table 3. Table 4 presents the simulation results obtained with the suboptimal routing vector provided by the algorithm, as well as a range of neighboring routing vectors.

It was found that the algorithm yielded a routing probability vector (0.5821,0.4179) that was close to the vector (0.59,0.41) that maximized system throughput in the simulation and minimized the total blocking at the split junction. It may also be noted that the predicted value of the maximum throughput rate (6.0175) is very close to the maximum simulated throughput (5.979), a difference of 0.6398 %. The predicted value of maximum throughput (6.0175) obtained with the suboptimal routing vector $\bar{\alpha}^* = (0.5821, 0.4179)$ is very close to the upper limit of the 95% confidence interval of the maximum simulated throughput(5.9644,5.9926). Figure 11 shows graphically the analytical throughput curve and the simulated

α_1	α_2	Θ	σ	pk_1	pk_2	pk
Suboptimal Vector by MINBLOCK algorithm						
0.5821	0.4179	5.970	0.0528	0.0588	0.0896	0.1484
Routing vectors in the immediate neighborhood of suboptimal						
0.56	0.44	5.965	0.027	0.0461	0.1016	0.1476
0.57	0.43	5.971	0.037	0.0485	0.0968	0.1452
0.58	0.42	5.976	0.025	0.0563	0.0915	0.1478
0.59	**0.41**	**5.979**	0.018	0.0634	0.0864	0.1498
0.60	0.40	5.954	0.019	0.0670	0.0805	0.1475
0.61	0.39	5.964	0.028	0.0713	0.0750	0.1464
Arbitrarily selected routing vectors						
0.48	0.52	5.821	0.042	0.0178	0.1486	0.1664
0.49	0.51	5.868	0.039	0.0225	0.1423	0.1648
0.50	0.50	5.883	0.041	0.0239	0.1360	0.1599
0.51	0.49	5.893	0.045	0.0258	0.1295	0.1553
0.52	0.48	5.907	0.040	0.0328	0.1224	0.1552
0.53	0.47	5.931	0.019	0.0331	0.1196	0.1527
0.54	0.46	5.950	0.032	0.0382	0.1127	0.1509
0.55	0.45	5.959	0.029	0.0412	0.1073	0.1485
0.62	0.38	5.954	0.028	0.0802	0.0689	0.1491

Table 4: Simulation results for Network A

throughput values. It may be noticed that the starting solution for this example was the routing vector (0.4842,0.5158), and the algorithm compensated for the unequal buffer sizes to provide a suboptimal routing vector of (0.5821,0.4179).

The CPU time taken by the Minblock algorithm was 5.21 seconds (on the Vax 8600), while the time taken by a single simulation experiment for one routing vector on the same system was 11.03 minutes. The simulation was run over 10 replications, each for 200000 time units, so as to obtain the desired confidence on the results. To measure the throughput obtained with different routing vectors, one would have to run this experiment many times over which could results in excessive run times.

5.2 One split junction and three parallel servers

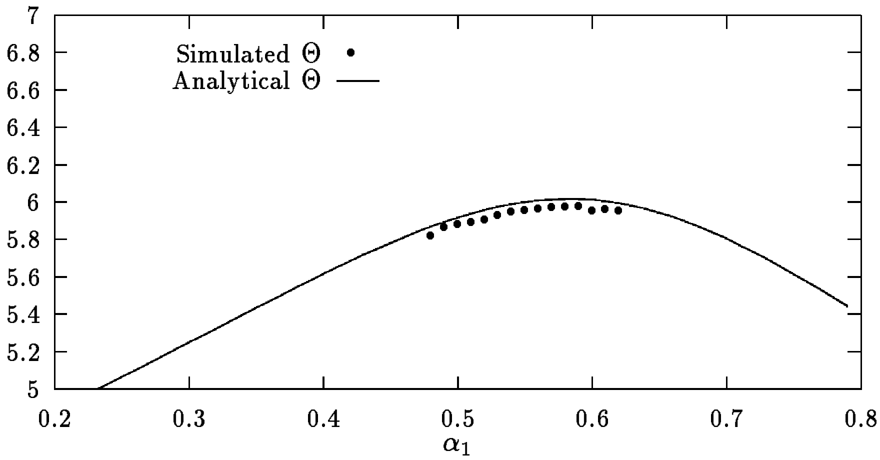

Figure 11: *Network A*: Simulated and Analytical Θ

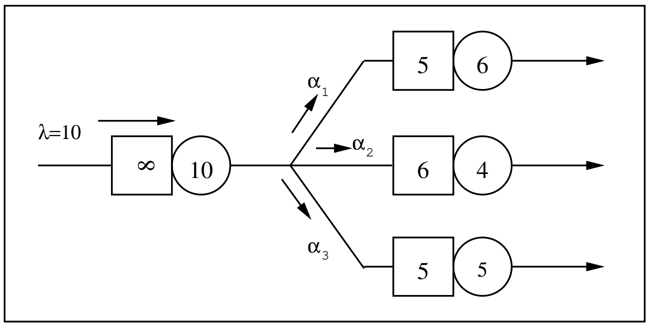

Figure 12: *Network B*: Network with 3 parallel servers

Figure 12 shows a network with one split junction and three parallel servers. The analytical throughput curve, which is the objective function to be maximized, is shown in Figure 13. The MinBlock algorithm was used to predict the suboptimal routing probabilities as well as the expected throughput rate and the path taken by the algorithm to search for the optimal solution is shown in Table 5. Table 6 presents the simulation results obtained with the suboptimal routing vector provided by the algorithm, as well as a range of neighboring and arbitrary routing vectors.

Θ analytical	α_1	α_2	α_3
9.3324070	0.40000001	0.26666668	0.33333331
9.3343296	0.40854108	0.26666668	0.37479222
9.3344431	0.40791857	0.26847008	0.30638179
9.3344707	0.40694427	0.26858237	0.32028732
9.3344698	0.40704426	0.26868236	0.32427335
9.3344717	0.40689725	0.26869932	0.32440341
8 Function Evaluations Within Powell Routine And 53 Function Evaluations During The Line Searches.			
optimal Values			
$\mathbf{\Theta = 9.3344717}$	$\boldsymbol{\alpha_1 = 0.40689725}$	$\boldsymbol{\alpha_2 = 0.26869932}$	$\boldsymbol{\alpha_3 = 0.32440341}$

Table 5: Network B: Path taken by the MinBlock Program

It was found that the algorithm yielded a routing vector (0.4067,0.2687,0.3244) that was close to the vector (0.41,0.25,0.34) that maximized system throughput as well as minimized the total blocking at the split junction. It may also be noticed that the predicted value of the maximum throughput rate (9.3334) is very close to the maximum simulated value (9.308), a difference of 0.27%. The predicted value of maximum throughput falls within the 95% confidence interval (9.2545,9.3615) of the simulated maximum throughput.

The Minblock algorithm yielded the optimal routing vector at the expense of 5.21 C.P.U seconds, while a simulation experiment (10 replications, 200000 time units) which measured throughput for a single routing vector for the same network took 17.47 C.P.U minutes. Again, to find the optimal routing vector via simulation, the same experiment would have to be run many times over.

6 General Networks with one split junction

Consider a network such as the one shown in Figure 5 with one split junction, n parallel streams and where all queues are $M/M/1/K$ except the first queue where arrivals occur, which is $M/M/1/\infty$. As in the earlier models, due to finite buffers at the beginning of each parallel stream, there will be blocking across the splits, which will adversely affect the arrival rates into each stream.

We would like to apply the model developed in the preceding sections to such type of networks. To be

α_1	α_2	α_3	Θ	σ	pk_1	pk_2	pk_3	pk
Suboptimal Vector by MINBLOCK algorithm								
0.4067	**0.2687**	**0.3244**	**9.304**	0.054	.0297	.0169	.0240	**.0706**
Routing vectors in the immediate neighborhood of suboptimal								
0.40	0.27	0.33	9.290	0.040	.0281	.0175	.0263	.0719
0.40	0.26	0.34	9.296	0.065	.0284	.0148	.0291	.0723
0.40	0.25	0.35	9.285	0.055	.0279	.0103	.0332	.0715
0.41	0.28	0.31	9.292	0.048	.0319	.0210	.0197	.0726
0.41	0.27	0.32	9.290	0.061	.0323	.0188	.0221	.0732
0.41	0.26	0.33	9.290	0.059	.0308	.0142	.0262	.0711
0.41	**0.25**	**0.34**	**9.308**	0.071	.0309	.0125	.0293	.0727
0.42	0.27	0.31	9.287	0.057	.0352	.0183	.0188	.0723
0.42	0.26	0.32	9.290	0.060	.0350	.0146	.0214	.0709
0.42	0.25	0.33	9.289	0.074	.0354	.0119	.0260	.0733
0.43	0.26	0.31	9.280	0.032	.0396	.0143	.0192	.0731
Other routing vectors in the neighborhood of suboptimal								
0.45	0.25	0.30	9.251	0.053	.0475	.0113	.0166	.0754
0.43	0.25	0.32	9.269	0.051	.0398	.0111	.0219	.0729
0.43	0.28	0.29	9.269	0.063	.0388	.0207	.0139	.0734
0.43	0.27	0.30	9.282	0.044	.0384	.0183	.0157	.0725
0.42	0.29	0.29	9.245	0.053	.0346	.0249	.0144	.0739
0.42	0.28	0.30	9.280	0.045	.0359	.0217	.0156	.0731
0.41	0.29	0.30	9.263	0.063	.0315	.0264	.0166	.0746
0.40	0.29	0.31	9.274	0.063	.0273	.0268	.0186	.0728
0.40	0.28	0.32	9.276	0.049	.0277	.0212	.0242	.0732
0.39	0.29	0.32	9.255	0.066	.0230	.0272	.0230	.0732
0.39	0.28	0.33	9.284	0.046	.0247	.0224	.0244	.0714
0.39	0.27	0.34	9.277	0.034	.0239	.0174	.0309	.0723
0.39	0.26	0.35	9.262	0.051	.0241	.0146	.0345	.0732
0.39	0.25	0.36	9.265	0.062	.0255	.0105	.0418	.0778
Arbitrarily selected routing vectors								
0.35	0.40	0.25	8.612	0.061	.0124	.1211	.0055	.1390
0.35	0.35	0.30	9.061	0.049	.0129	.0663	.0152	.0944

Table 6: Simulation results for Network B

able to do so, we will reduce such networks into equivalent networks with one split junction and $n+1$ servers, with one $M/M/1/\infty$ server before the split and n $M/M/1/K$ parallel servers after the split junction. In the following section we will discuss the sufficiency of studying the routing problem under heavy traffic conditions, which in turn will suggest an appropriate service rate for each server in the reduced model.

6.1 Heavy Traffic Conditions

We will apply a piecewise linear upper bound to the entire network shown in Figure 5. We know that for a given vector of routing probabilities $\bar{\alpha}$, there exists an upper bound on throughput[7, 1] as follows:

$$\theta_{\bar{\alpha}}^{UB} = \begin{cases} \lambda & \text{if } \lambda < \theta_{\bar{\alpha}}^* \\ \theta^* & \text{if } \lambda \geq \theta_{\bar{\alpha}}^* \end{cases}$$

Since it has been shown that the function θ is concave in $\bar{\alpha}$[5], there exists an optimal routing vector $\bar{\alpha}^\star$. Let,

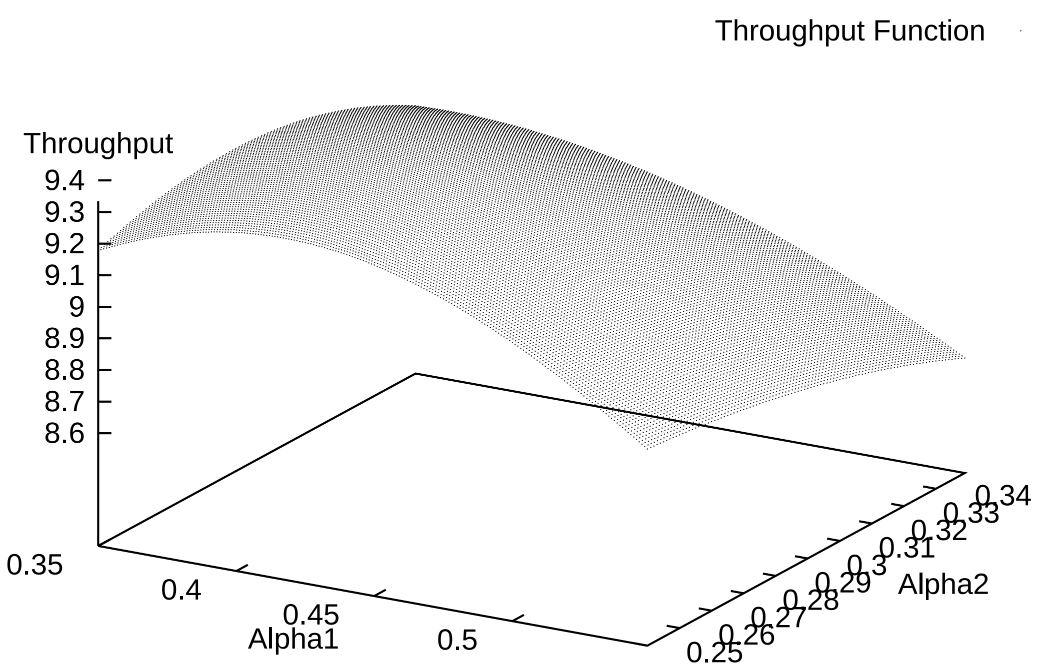

Figure 13: *Network B*: Analytical throughput function

$\Theta^*_{\bar{\alpha}^\star}$ = throughput capacity of the network with the routing vector $\bar{\alpha}^\star$ and
$\Theta^*_{\bar{\alpha}}$ = throughput capacity of the network with the routing vector $\bar{\alpha}$.

By definition, $\Theta^*_{\bar{\alpha}^\star} \geq \Theta^*_{\bar{\alpha}}$. Therefore we get Θ Vs. λ curves as shown in Figure 14.

Note that the throughputs (by virtue of their proximity to the upper bound) obtained under the two different routing vectors satisfy the following condition:

$$\Theta_{\bar{\alpha}^\star} \geq \Theta_{\bar{\alpha}} \quad \forall \quad \lambda = 0, .., \infty$$

Therefore, it is sufficient to solve the problem of finding $\bar{\alpha}^\star$ under the condition that $\lambda \geq \Theta^*$. In other words, suppose we find $\bar{\alpha}^\star$ for a value of $\lambda \geq \theta^*$, then this $\bar{\alpha}^\star$ is also optimal for any other value of λ. Now, since we do not know how to compute Θ^* for such networks, we will use $\sum_{i=1}^{n} \theta_i^*$ as a safe estimate to choose λ. The important issue is that one can restrict the optimal routing problem to be studied under heavy traffic conditions.

6.2 Reduction to Equivalent $n+1$ node networks

Since we will restrict studying the routing problem to heavy traffic conditions, it is safe to assume that the output of each parallel stream in the network will be approximately equal to its respective throughput capacity Θ^*. We have also observed experimentally that the inter-departure times of these output processes can be closely approximated by exponential distributions with rate Θ^*.

Therefore, we are able to reduce the general networks shown in Figure 5 to equivalent networks as shown in Figure 15 under heavy traffic. For instance, a parallel stream of $j = 1, ..., m$ servers with finite

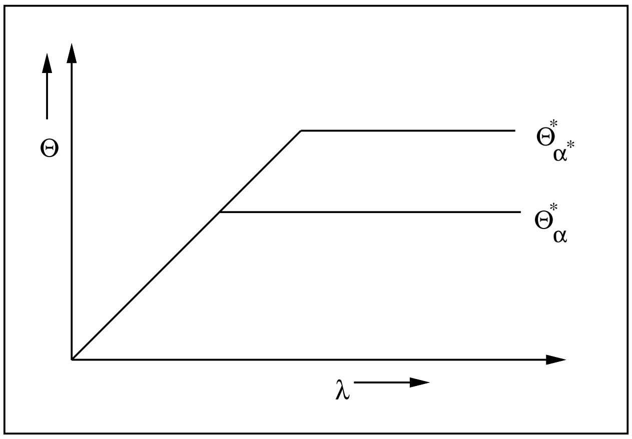

Figure 14: Θ Vs. λ curves for $\bar{\alpha}$ and $\bar{\alpha}^{\star}$

buffers $K_j,\ j = 1, .., m$ would be replaced by one node with an exponential service rate Θ^* (its throughput capacity) and a finite buffer of K_1, which is the buffer at the first node. It is not possible to isolate K_1 since it directly affects the blocking at the node prior to the split junction. It must be borne in mind that this is only an approximate model of the real system, however it goes a long way in simplifying the routing problem.

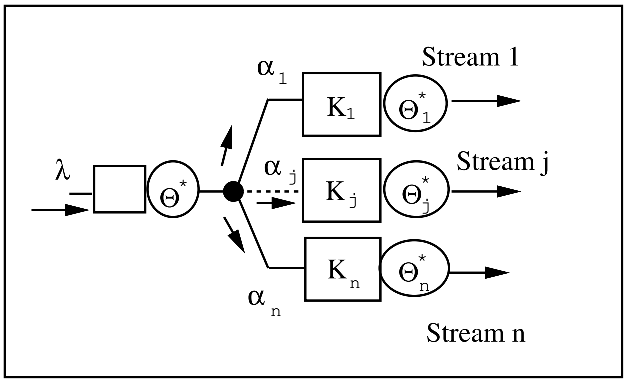

Figure 15: An equivalent reduced network

Once we have the equivalent network shown in Figure 15, the *MINBLOCK* algorithm may be applied to determine the approximate optimal routing vector. The following subsection shows some experiments designed to compare the results of the *MINBLOCK* algorithm with similar results obtained via simulation.

6.3 Experiment: A 5 node network

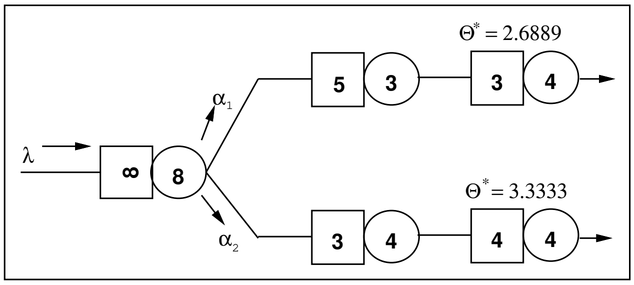

Figure 16: Network C: 5 servers

Figure 16 shows a split topology network with one node before the split junction and two nodes in each of the two parallel streams. The throughput capacities of the two parallel streams in the network are computed using the asymptotic bound method[7] and are as indicated alongside the respective streams. Assuming that the problem is solved under heavy traffic conditions, the equivalent reduced network would appear as shown in Figure 17.

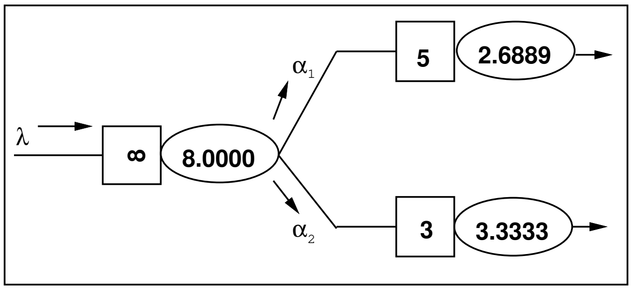

Figure 17: Network C: Equivalent Reduced Network

α_1	α_2	Θ	σ	pk_1	pk_2	pk
Suboptimal Vector by MINBLOCK algorithm						
0.4294	**0.5706**	**5.0235**	**0.0482**	**0.1105**	**0.2594**	**0.3699**
Routing vectors in the immediate neighborhood of suboptimal						
0.39	0.61	4.953	0.041	0.0741	0.3069	0.3810
0.40	0.60	4.970	0.032	0.0865	0.2933	0.3798
0.41	0.59	4.999	0.041	0.0962	0.2778	0.3740
0.42	0.58	4.999	0.035	0.1073	0.2657	0.3730
0.43	0.57	5.016	0.043	0.1202	0.2522	0.3724
0.44	0.56	5.016	0.041	0.1385	0.2336	0.3721
0.45	0.55	5.011	0.029	0.1557	0.2178	0.3735
0.46	0.54	4.994	0.053	0.1716	0.2037	0.3753
0.47	0.53	4.994	0.040	0.1862	0.1899	0.3761
Routing vectors in the neighborhood						
0.38	0.62	4.914	0.036	0.0637	0.3208	0.3845
0.48	0.52	4.954	0.046	0.2020	0.1778	0.3798
0.49	0.51	4.950	0.048	0.2158	0.1632	0.3790
0.51	0.49	4.840	0.056	0.2550	0.1381	0.3931
Routing vectors chosen arbitrarily						
0.00	1.00	3.331	0.025	0.0000	0.5846	0.5846
0.15	0.85	3.877	0.052	0.0002	0.5132	0.5134
0.30	0.70	4.589	0.030	0.0148	0.4122	0.4270
0.35	0.65	4.817	0.040	0.0407	0.3566	0.3973
0.55	0.45	4.668	0.035	0.3226	0.0937	0.4163
0.60	0.40	4.392	0.034	0.3992	0.0524	0.4516
0.75	0.25	3.391	0.024	0.5706	0.0050	0.5756
1.00	0.00	2.541	0.026	0.6829	0.0000	0.6829

Table 7: Simulation results for Network C

The optimal routing vector yielded by the MINBLOCK algorithm is $(0.4294, 0.5706)$. In the simulation experiment set up to verify this result, an arrival rate of $\lambda = 7$ was chosen, since $\theta_1 + \theta_2 = 6.0222 < 7$. The experiment was thus conducted under heavy traffic conditions. The results of the experiment, which also compares the throughputs obtained with different routing vectors, are shown in Table 7.

It can be seen that the MINBLOCK algorithm yields a routing vector which results in the maximum throughput and minimum blocking in the simulation experiment when compared with the other routing vectors. The maximum throughput estimated by the algorithm was 4.8508, while the maximum simulated throughput was 5.0235 (a difference of 3.44 %). It may also be noticed that arbitrarily chosen routing vectors result in much higher blocking and lower throughput rates.

6.4 Experiment: A 6 node network

Consider the network shown in Figure 18a where we have to determine the optimal routing vector (α_1, α_2). We will solve the problem under heavy traffic conditions because we will be able to apply the principles of the preceding subsection and reduce the network to the equivalent three node network, also shown in Figure 18b.

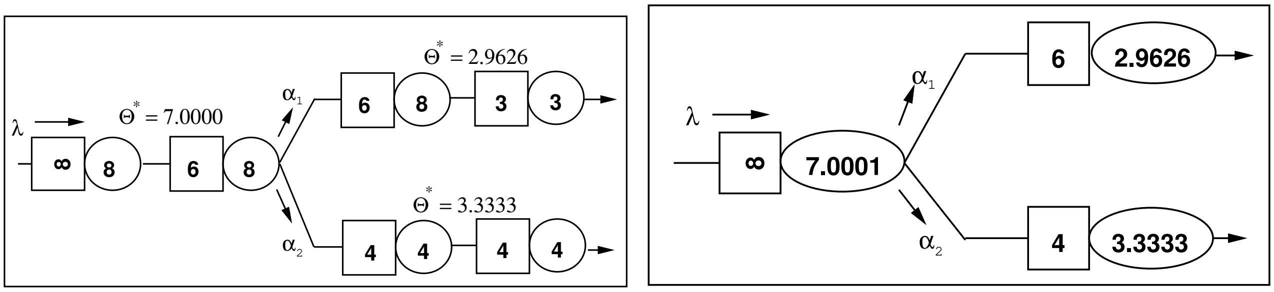

Figure 18: Network D: 6 servers & Equivalent Reduced Network

α_1	α_2	Θ	σ	pk_1	pk_2	pk
Suboptimal Vector by MINBLOCK algorithm						
0.4747	**0.5253**	**5.4363**	**0.0566**	**0.1236**	**0.1574**	**0.2809**
Routing vectors in the immediate neighborhood of suboptimal						
0.42	0.58	5.353	0.048	.0598	.1836	.2434
0.43	0.57	5.384	0.035	.0720	.1818	.2539
0.44	0.56	5.398	0.041	.0762	.1823	.2585
0.45	0.55	5.437	0.040	.0890	.1736	.2626
0.46	0.54	5.441	0.032	.1021	.1698	.2720
0.47	**0.53**	**5.450**	0.035	.1150	.1615	.2765
0.48	0.52	5.428	0.052	.1326	.1539	.2865
0.49	0.51	5.413	0.045	.1387	.1476	.2863
0.50	0.50	5.405	0.044	.1478	.1416	.2894
0.51	0.49	5.358	0.051	.1604	.1311	.2915

Table 8: Simulation results for Network D

The MINBLOCK algorithm was applied to the reduced network and the optimal routing vector was obtained as $(0.4747, 0.5253)$. In the simulation experiment set up to verify this result, an arrival rate $\lambda = 7$ was chosen since $\theta_1 + \theta_2 = 6.2959 < 7$, thus ensuring heavy traffic conditions. In the simulation experiment, the routing vector that maximized the throughput was (0.47.0.53), which is very close to the one yielded by the algorithm. As seen in the simulation results shown in Table 8 this routing vector does yield a throughput value that is very close to the maximum simulated throughput. The maximum throughput value obtained in the simulation experiment was 5.450, while the one estimated by the algorithm was 5.1597 (a difference of 5.33 %).

It may be noticed that the maximum throughput rates predicted by the algorithm are generally very close to the maximum simulated throughput rates. However as the number of nodes in the network increases there is a tendency for the difference between the two values to increase.

7 Split networks with M/M/C/K servers

The Minblock algorithm can be applied to networks with M/M/C/K servers after a split junction, if the arrival rate to the split junction is approximately Poisson. We have experimentally observed that the inter-departure times of the topology in Figure 6 fit an exponential distribution, even if all servers are M/M/2/K and M/M/5/K. We can therefore conjecture that the departure process of a series of M/M/C/K queues can be modeled with a Poisson distribution. The optimization model remains identical to the one developed in §4 for a simple split junction with n parallel M/M/1/K servers, except for the equation specifying the blocking probabilities. In general, the networks studied in this section can be represented as shown in Figure 19.

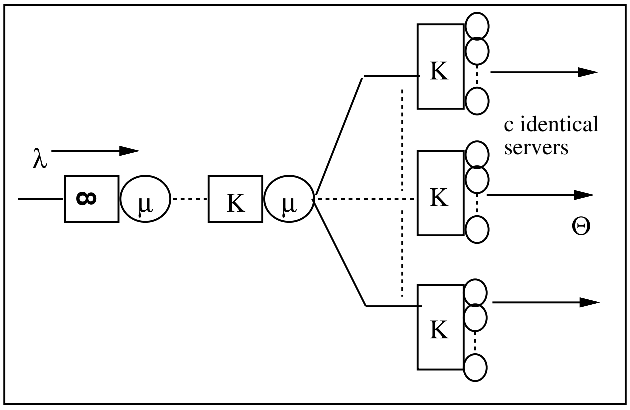

Figure 19: Split Network with M/M/C/K servers

The blocking probability for an M/M/C/K server is given by [11]:

$$p_K = \frac{1}{c^{K-c}c!}\left(\frac{\tilde{\lambda}}{\mu}\right)^K p_0$$

where for $\frac{\tilde{\lambda}}{c\mu} \neq 1$,

$$p_0 = \left[\sum_{n=0}^{c-1}\frac{1}{n!}\left(\frac{\tilde{\lambda}}{\mu}\right)^n + \frac{(\tilde{\lambda}/\mu)^c}{c!}\frac{1-(\tilde{\lambda}/c\mu)^{K-c+1}}{1-(\tilde{\lambda}/c\mu)}\right]^{-1}$$

and for $\frac{\tilde{\lambda}}{c\mu} = 1$,

$$p_0 = \left[\sum_{n=0}^{c-1}\frac{1}{n!}\left(\frac{\tilde{\lambda}}{\mu}\right)^n + \frac{(\tilde{\lambda}/\mu)^c}{c!}(K-c+1)\right]^{-1}$$

Thus, for each parallel server $i = 1,..,n$ in Figure 19, the blocking probability will be given by the above formula where

$$\tilde{\lambda}_i = \tilde{\lambda}\alpha_i(\sum_{i=1}^{n} pk_i - pk_i) \;\; \forall \;\; i = 1,..,n$$

and the objective is to maximize $\hat{\Theta}$, such that

$$\hat{\Theta} = \tilde{\lambda}(1 - \sum_{i=1}^{n} pk_i)$$

Again, for a given routing vector $\alpha_j, j = 1,..,n$, the constraints are n nonlinear equations with n unknowns which can be solved with the IMSL subroutine NEQNF to yield a set of blocking probabilities pk_j. Powell's optimization technique is then used to search for the suboptimal routing vector $\alpha_j^*, j = 1,..,n$ which maximizes Θ. Both these steps are automatically executed by a version of the Minblock algorithm, which is specially designed to account for M/M/C/K servers after the split.

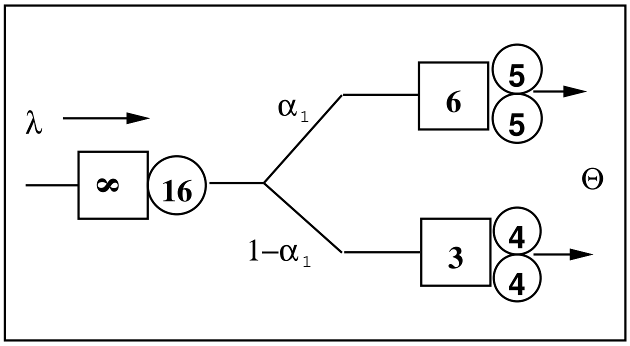

Figure 20: Network E: M/M/2/K servers after split

The solution provided by the Minblock algorithm was applied to an example with two parallel streams, each with an M/M/2/K server, as shown in Figure 20. The server prior to the split junction is M/M/1/K and blocking occurs across the split junction if the buffers at the M/M/2/K servers are saturated. As can be seen from the experimental results, the suboptimal routing vector provided by the Minblock algorithm (0.6222,3778) is also the suboptimal vector obtained through simulation. Also, the maximum throughput obtained by the Minblock algorithm (12.9401) is very close to that obtained via simulation (12.764), a difference of only 1.3609%.

Θ analytical	α_1	α_2
12.721428	0.55555558	0.44444442
12.940121	0.62240171	0.37759829
12.940117	0.62250173	0.37749827
12.940121	0.62239617	0.37760383
12.940125	0.62215447	0.37784553
7 Function Evaluations Within Powell Routine And 33 Function Evaluations During The Line Searches.		
optimal Values		
$\boldsymbol{\Theta}$ **= 12.940125**	α_1 **= 0.62215447**	α_2 **= 0.37784553**

Table 9: Network E: Path taken by the MinBlock Program

8 Summary and Conclusions

We have developed an approximate model of a finite queueing network where blocking occurs across split junctions and interrupts the arrival processes into each parallel stream after the split junction. Based on this analytical model we have constructed an algorithm to select the suboptimal routing vector which maximizes system throughput. This algorithm is computationally efficient and yields good solutions when compared with simulation experiments.

In an earlier paper[6] we had assumed that all buffers immediately after a split junction are infinite and hence do not cause blocking across the split junction. In this paper we have relaxed this assumption and constructed a model to approximate the blocking probabilities at the split junction. It may be seen that the assumption of infinite buffers is a relaxation of this model, where due to infinite buffers the blocking probabilities across the split junctions are very small, and hence can be ignored. Thus, in the earlier paper, the arrival rate into stream i after a split junction was simply $\tilde{\lambda}\alpha_i$, which helped us in developing the concept of preferred routing probabilities.

Finally, it may be mentioned that the approach developed in this paper is an approximate model which performs well under most experimental conditions, and hence is an important methodological tool for the optimal routing problem.

α_1	α_2	Θ	σ	pk_1	pk_2	pk
0.5700	0.4300	12.615	0.081	.0620	.1474	.2094
0.5800	0.4200	12.656	0.058	.0703	.1399	.2102
0.5900	0.4100	12.702	0.065	.0756	.1317	.2074
0.6000	0.4000	12.731	0.061	.0834	.1218	.2052
0.6100	0.3900	12.733	0.045	.0915	.1118	.2033
0.6200	0.3800	12.757	0.052	.0981	.1050	.2031
0.6222[a]	**0.3778**	**12.764**[b]	0.065	.0969	.1049	.2018
0.6300	0.3700	12.731	0.054	.1060	.0974	.2034
0.6400	0.3600	12.732	0.058	.1153	.0901	.2054
0.6500	0.3500	12.727	0.055	.1233	.0822	.2055
0.6600	0.3400	12.691	0.040	.1302	.0771	.2073
0.6700	0.3300	12.690	0.051	.1372	.0685	.2056

[a]Optimal routing vector by Minblock algorithm as well as by simulation
[b]Maximum simulated throughput(% error 1.3609)

Table 10: Network E: Simulation results

References

[1] Baccelli,F. and S. Foss, 1995, "On the Saturation Rule for the stability of Queues," *J. Appl. Prob.* **32**, 494-507.

[2] Banks, Jerry and John S. Carson, *Discrete Event System Simulation*, Prentice Hall, Inc, 1972.

[3] Bertsekas, D. and R. Gallagher, 1987. *Data Networks,* Prentice-Hall, Inc. Englewood Cliffs, N.J.

[4] Cassandras, C., A Hierarchical Routing Control Scheme for MHS, *Proceedings of ORSA/TIMS Meeting on FMS*, Univ. of Michigan, August 1984.

[5] Daskalaki, Sophia., "Static and Real time Routing in Open Finite Queueing Networks", PhD Thesis, Dept. of Industrial Engineering and Operations Research, University of Massachusetts, Amherst.

[6] Gosavi, Hemant and J. MacGregor Smith, *An algorithm for sub-optimal routing in series parallel queueing networks*, accepted in *The International Journal of Production Research.*

[7] Gosavi, Hemant and J. MacGregor Smith, 1995. *Asymptotic bounds of throughput in open finite queueing networks*, *Computers and Operations Research*__22__(10), 1057-1073.

[8] Himmelblau, David M., 1972, *Applied Non-linear Programming*, McGraw Hill Inc.

[9] Houck, D. J., Comparison of Policies for Routing Customers to Parallel Queueing Systems, *Ops Res.*, **35 (2)**, 306-310 (1987).

[10] IMSL Software Systems, 1987, *MATH/Library, Fortran Subroutines for mathematical Applications.*

[11] Jain, Sushant and J. MacGregor Smith, 1994, *Open Finite Queueing Networks with M/M/C/K Parallel Servers*, Computers and Operations Research Vol. 21, No. 3, pp. 297-317.

[12] Kelly, F. P., Routing and Capacity Allocation in Networks with Trunk Reservations, in review.

[13] Powell, M.J.D., *An efficient method for finding the minimum of a function of several variables without calculating derivatives*, Computer Journal 7, pp. 155-162, 1964.

[14] Smith, J. MacGregor, R.J. Graves and L. Kerbache, *QNET: An open queueing network model for material handling system analysis*, Material Flow, 3(1986) 225-242.

[15] Yao, David, D., A FMS Network Model with State-Dependent Routing, in *Proceedings of the ORSA/TIMS Meeting on FMS*, Univ. of Michigan, August 1984.

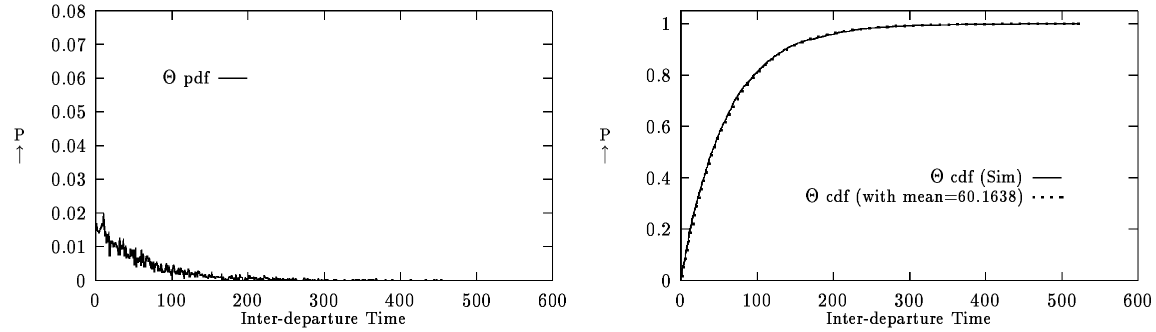

Figure 21: Plots with $\lambda = 1$

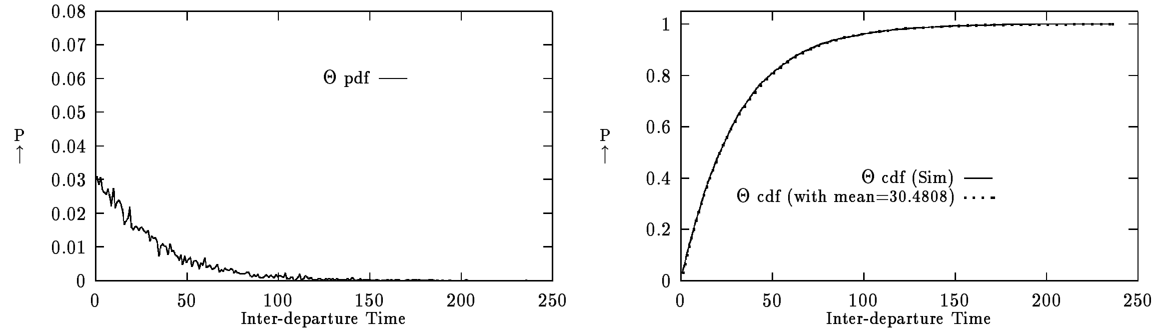

Figure 22: Plots with $\lambda = 2$

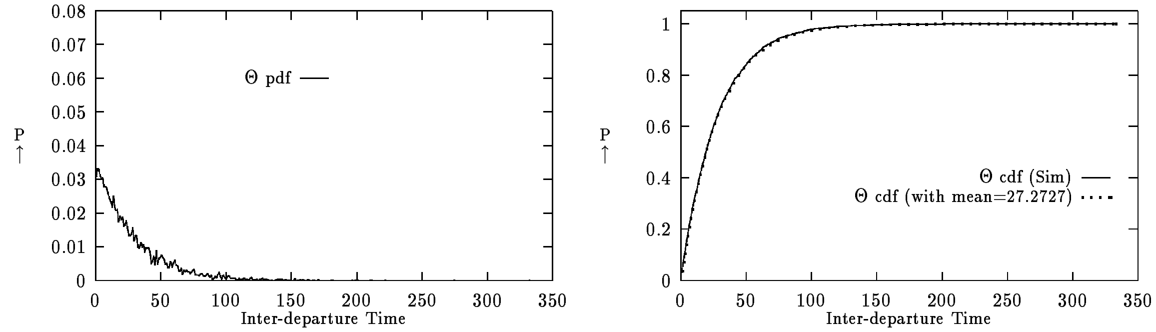

Figure 23: Plots with $\lambda = 2.2$

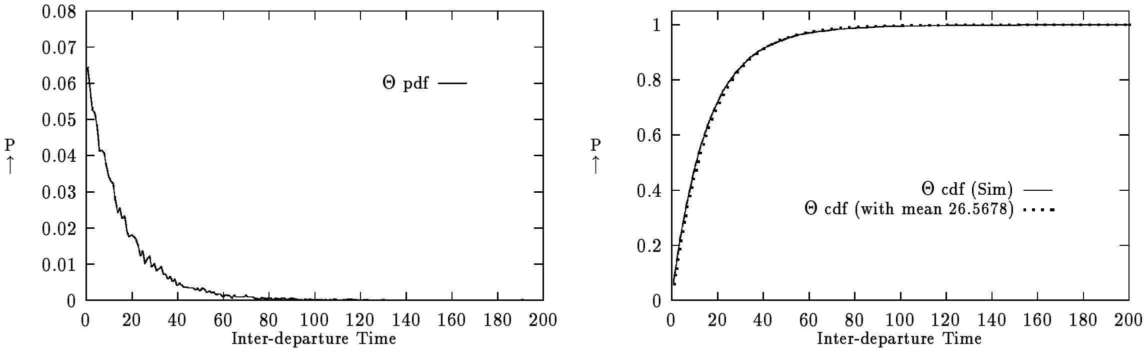

Figure 24: Plots with $\lambda = 2.5$

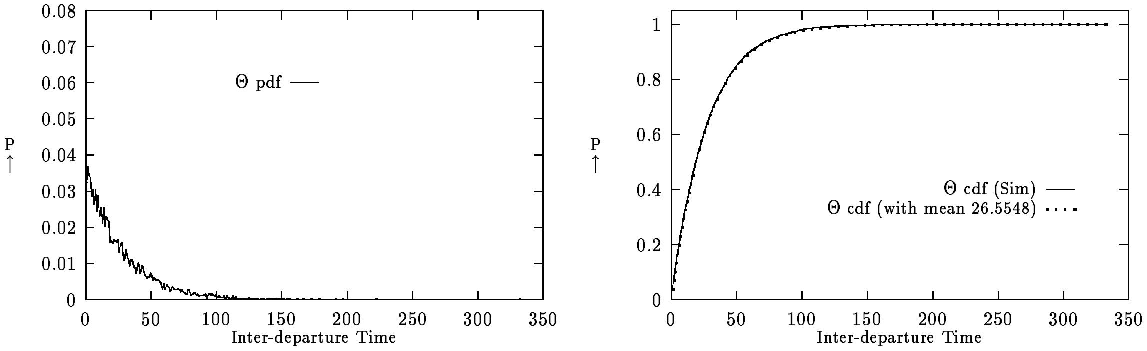

Figure 25: Plots with $\lambda = 3.0$