R.K. Boel, J.H. van Schuppen

Distributed routing for load balancing

Department of Operations Research and System Theory Report OS-R8820 November

Distributed Routing for Load Balancing

R.K. Boel
Research Fellow NFWO (Belgian National Foundation for Scientific Research)
Laboratorium voor Theoretische Electriciteit, Rijksuniversiteit Gent
Grote Steenweg Noord 2, B9710 Gent (Zwijnaarde), Belgium

J.H. van Schuppen
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

This paper discusses some open-loop and closed-loop control algorithms for an example of a discrete-event system, viz. the routing of arriving tasks from different arrival streams among several possible service stations. We show that it is possible to design open-loop policies that give good performance in a way which is very robust with respect to large changes in the arrival rates. This is possible even though we assume that there is no on-line coordination between the routing algorithms for the different arrival streams. Some further improvements of the performance are possible when a simple feedback policy - overflow routing - is implemented. This also gives reasonable robustness of the performance with respect to changes in the service rates. Several different analysis techniques are needed to obtain the analytical and numerical solutions necessary to compare the performance of the different policies.

1980 Mathematics Subject Classification (1985 Revision): 93E20, 90B22, 68C15. 69C24

Key Words & Phrases: load balancing, distributed communication systems, control, team theory, queueing theory.

Note: This paper has been submitted for publication.

1. Introduction

In this paper we discuss several approaches to the control of a discrete-event system. In particular we discuss the problem of how to design routing units for a queueing network consisting of several service stations with their associated waiting room and of several arrival streams of tasks with their associated routing units, see Fig. 1. This problem is studied from the point of view of a control engineer, i.e. we are interested in how much information each routing unit should have available at the time of a decision in order to achieve good performance. In this paper it is assumed that routing units make a decision on where to send an arriving task at the time of the arrival. We compare open-loop (feedforward) policies with simple closed-loop (feedback) policies where the routing unit at the time of an arrival is allowed to look at one queue length only. Thus we limit ourselves to decentralized policies without on-line coordination, and, as control engineering experience suggests, we cannot expect to be able to find an "optimal policy". What can be achieved, it turns out from the simple cases which we have solved analytically or numerically, is reasonable performance (in the sense of small average waiting times and low blocking probabilities) in a way which is robust with respect to large changes in the arrival rates and - to a smaller extent - with respect to changes of the service rates.

Problems of the type we discuss are useful in the design of distributed processors [5, 9-11, 17, 28, 30, 33, 39] in particular for distributed telephone switches [3, 16] and in the design of routing

Report OS-R8820
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

algorithms for communication or manufacturing networks [20,35,38,43]. A good survey paper on these issues is that by Wang and Morris [44], where many more policies are presented and analyzed than the ones we consider here. For example, in [44] policies are considered where servers with little work to do request newly assigned tasks to be sent to them, rather than having routing units associated to arrival streams assign tasks. We do not consider this firstly because in our control engineers' approach the communications overhead is probably quite high, [18], and secondly because we cannot give an easily calculated performance measure bounding this extra communication cost.

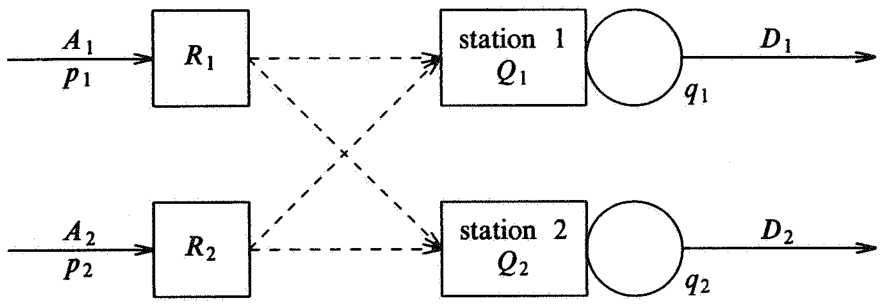

Fig. 1. Model of load balancing system with two stations.

We have also limited ourselves to routing policies for which we can calculate, at least for not too large a system, the performance measures analytically or numerically efficiently. In particular we have avoided the independence assumption usually made in papers on random routing [18,32,43,47]. One phenomenon we are interested in is the performance improvement resulting from the more regular arrival streams [20] obtained by open-loop, cyclic routing. This effect would be masked by the independence assumption. This is different from the case of random routing where there is no regularisation of the arrival streams.

In order to obtain robustly good performance we will find that the routing unit should at least have the following properties: Traffic loads - different arrival rates - should be balanced evenly over the different service stations proportionally to their respective nominal service rates. This can be achieved via random routing [11,41]. It does not matter then how many arrival streams there are, and hence the results of Bonomi and Kumar [6] can be used, showing that the optimal solution, in the sense of minimum mean waiting time, balances the idle times of the different service stations. However it is well known that, in the case of one arrival stream, significant improvements are possible by alternating arrival streams in a predetermined way between the different service stations [20]. In the case of one arrival stream it is also known how to calculate the optimal alternating pattern [27]. Even before the improvement was theoretically recognized it was already used in adaptive overflow routing of telephone traffic under the names call-gapping and Selective Dynamic Overload Control (SDOC) [24]. This improvement has to do with the fact that the arrival streams at each service station are now renewal processes that are more regular than the Poisson process, i.e. with a smaller coefficient of variation of the interarrival times [25].

Unfortunately the superposition of several independent renewal processes is not a renewal process. Such a superposition occurs when several uncoordinated routing units deterministically alternate the service stations to which they send tasks. Moreover intuition suggests that superposing "regular" renewal processes will lead to a less regular process, i.e. superposition will increase the coefficient of variation of the interarrival times. Nevertheless we do find a few cases which can be analyzed explicitly and for which it turns out that deterministically alternating routing is advantageous. This confirms the results of [3] where very large systems were simulated. As far as we know the open-loop routing algorithms for several arrival streams as discussed in this paper have not yet been analyzed theoretically. They do provide good robustness of the performance with repect to large changes in the arrival rates.

In order to also achieve robustness with respect to changes in the service rates, some state information, i.e. some knowledge of the queue lengths, is necessary. Again when all the routing units have the same information at all times and hence can be coordinated, then it does not matter that there is more than one arrival stream. Optimal control results such as "join the shortest queue" (JSQ) [45,46] can be used. Several extensions of this closed-loop optimal routing have been studied for more complicated systems [14,15,26,31,34]. Analytical and numerical solutions for these cases have also been studied intensively [4,13,14,19,22,23]. As soon as the assumption is dropped that all the routing units have the same information set, the problem becomes a team problem. Very few analytical results are known; the only attempts we know of to apply the results of team theory to problems similar to load balancing are [1,8]. Even there the required information sharing is probably unrealistic and the optimal strategies are not implementable. Therefore we limit ourselves to very simple strategies where each routing unit follows an open-loop policy except when the queue length at the service station it would normally route its task too, is longer than a predetermined threshold. We find that these overflow strategies do indeed give good performance but we cannot draw any conclusion on how much closer to the JSQ performance lower bound we could come by actually solving the optimal team problem.

The mathematical model we have used to analyze the problem described above is the following stochastically driven discrete-event system. Assume that we have stochastically specified the family of point processes $(N_{k,t},\ k=1,...,n+m)$, viz. the n arrival streams $(A_{i,t},\ i=1,...,n)$ and the m potential service completions $(D_{j,t},\ j=1,...,m)$. The internal state of the system at each time is $X_t \in \mathbb{X}$. At each jump time $T_{k,l}$ of the counting process $N_{k,t}$ the state changes from $X_{T_{k,l}-}$ to $G_k(X_{T_{k,l}-})=X_{T_{k,l}+}$. Hence we obtain the following simple dynamical system representation

$$\begin{aligned} dX_t &= \sum_{k=1}^{n+m}(G_k(X_{t-})-X_{t-})dN_{k,t} \qquad (1.1)\\ &= \sum_{k=1}^{n+m}(G_k(X_{t-})-X_{t-})\lambda_k(t)dt + \sum_{k=1}^{n+m}(G_k(X_{t-})-X_{t-})(dN_{k,t}-\lambda_k(t)dt), \end{aligned}$$

where $\lambda_k(t)$ is the rate of the point process $N_{k,t}$. We do assume that each of these point processes is self-excited so that $\lambda_k(t)$, $\forall k$, does not depend explicitly on the state X_t. The choice of the routing units is then equivalent to choosing the functions G_k, $k=1,...,n$ corresponding to the n arrival streams. The form of G_k is constrained by the information available to routing unit r_k. The functions G_k, $k=n+1,...,n+m$ correspond to departures and are assumed fixed.

Ideally in a control problem we would like to be able to analyze the transient behaviour of X_t, e.g. discuss how fast the waiting times of tasks converge to the long term average when the initial condition is X_0. This is only very rarely possible. Usually we have to limit attention to calculate the equilibrium distribution $\lim_{t\to\infty}P(X_t=x)=p(x)$ assuming it exists. The performance measures, such as mean waiting time, variance of waiting time, blocking probability, rerouting probability, etc. are then all functionals of this equilibrium distribution.

In this paper we have only analyzed the case where $A_{i,t}$ are Poisson processes, and the service times are exponential. Then X_t only should contain information about the queue lengths and about the internal state of the routing units. The model we describe then reduces to a countable state Markov process. However the methods of calculation of performance measures are very easily adapted to more general arrival processes (even non-renewal). General service times are harder to deal with because the departure processes will no longer be selfexcited (in other words $\lambda_k(t,X_t)$ for $k=n+1,...,n+m$).

Summarizing we can say that this paper tries to find good routing policies r_i, $i=1,...,n$ or equivalently good transition functions G_i, $i=1,...,n$ such that, given nominal arrival rates λ_i, $i=1,...,n$ and nominal service rates μ_j, $j=1,...,m$ the important stationary performance measures remain small even for fairly large changes away from the nominal rates. In section 2 we discuss the simple case of a discrete-time system with Bernoulli arrivals. This model even allows some simple transient results. In section 3 we discuss open-loop policies and in section 4 overflow policies for the case with Poisson arrival streams and exponential service times. In section 5 we discuss some open problems and draw some conclusions.

2. A SIMPLE DISCRETE-TIME MODEL

To illustrate the concepts discussed in this paper, we first consider a very simple discrete time queueing system, see Fig. 1. Assume a weak form of coordination between the different subsystems of the model, namely the presence of perfectly synchronized clocks. There are two Bernoulli arrival streams of tasks $a_{1,n}$ and $a_{2,n}$, independent of each other and each having one arrival per time unit with probability p_1 respectively p_2, independently of arrivals in other time intervals. There are two service stations which can process one task per time interval with probability q_1, respectively q_2. Associated with each arrival stream i there is a routing unit R_i which assigns arriving tasks to one of the infinite waiting rooms upon their arrival. We want to compare the performance under different assignment policies.

First we will discuss how the performance depends on p_1 and p_2, for the case of perfect service stations: $q_1=q_2=1$. Then we will discuss the robustness of the performance with respect to service failures. Since waiting times are exactly equal to the number of customers in front of a task times one time interval, we study only the queue lengths $(Q_{1,n})$ and $(Q_{2,n})$, i.e. the number of customers waiting or in service.

Fixed routing

Consider first the case where tasks of arrival stream $A_{i,n}=\sum a_{i,j}$ are sent to the waiting room of server i. The queue lengths $(Q_{1,n})$ and $(Q_{2,n})$ are independent stochastic processes. Let $Q_{i0}>1$. Then the transient behaviour of this system can be described as follows. Let

$$G_{i,n} = \sum_{j=1}^{n}(1 - a_{i,j}) = n - A_{i,n},$$

be the process counting the time intervals when there is no arrival on stream i. The time between two jumps of size $+1$ of $G_{i,n}$ is geometrically distributed, with mean $1/(1-p_i)$. One easily sees that $Q_{i,n} = Q_{i,0} - G_{i,n-1}$ as long as $Q_{i,n}\geq 1$. As soon as $G_{i,n}=Q_{i0}$ the system starts operating under stationary conditions: $Q_{i,n} = A_{i,n-1}$. For the two-dimensional Markov process $(Q_{1,n}, Q_{2,n})$ only the states (0,0), (1,0), (0,1), and (1,1) are ergodic; all the other states are transient. The mean queue length is p_1, respectively p_2.

Cyclic routing

The next routing policy to be analysed is cyclic routing: when $A_{i,n}$ is even, then send the arriving task to the waiting room of service station i, when $A_{i,n}$ is odd then route the arriving task to the waiting room of the other station. The performance can be analyzed as follows. Define

$$D_1 = inf\{k>0\,|\,a_{1,k}=0, \ and/or\ a_{2,k}=0\}$$

and

$$D_k = inf\{k>D_{k-1}\,|\,a_{1,k}a_{2,k}=0\}.$$

During the interval $\{1,...,D_1\}$ each station receives exacty one task in each interval, provided one takes care of sending tasks to different stations at time 1. The queue lengths remain constant in this interval. The distribution of D_1 is geometric with mean $1/(1-p_1p_2)$:

$$P(D_1=d) = (p_1p_2)^{d-1}(1-p_1p_2).$$

If at D_1 there is no arriving task on both arrival streams, an event which occurs with probability $(1-p_1)(1-p_2)/(1-p_1p_2)$, then both queue lengths $Q_{1,n}$ and $Q_{2,n}$ decrease by 1 if possible. The behaviour of the system during the next interval $(D_1+1, \ldots, D_1+D_2)$, is exactly as before. If however at D_1 there is no arrival on stream 1 but there is an arrival on stream 2 (this happens at D_1 with probability $(1-p_1)p_2/(1-p_1p_2)$) then the queue length at station number D_1 *mod* 2 remains the same between D_1 and D_1+1, while the other queue length decreases by 1 if possible. During the next interval $(D_1+1, \ldots, D_2)$, while there are at each time interval arriving tasks on both streams, the system works as follows. At D_1+1 there are two arrivals at station (D_1+1) *mod* 2 and no arrival at station D_1 *mod* 2; at D_1+2 (if $D_2 \geq 2$) there are 2 arrivals at station D_1 *mod* 2 and none at the other station, and so on. Thus the queue lengths alternately increase and decrease by 1 jumping between $Q_{1,0}-1$ and $Q_{1,0}$. The behaviour in subsequent intervals is described in the same way. This transient behaviour continues until both queue lengths become less than 2.

Clearly now all states with $Q_{1,n} \leq 2$ and $Q_{2,n} \leq 2$ are ergodic. All other states are transient. The transient behaviour is similar to that for fixed routing, except that the length of the intervals is now geometrically distributed with parameter $1-(p_1+p_2)/2$, instead of p_1, respectively p_2. The mean queue length is now the same for both queues, viz.

$$\frac{p_1+p_2}{2P(Q_1>0)}$$

according to Little's law. When p_1 and p_2 are close to 1 this will be comparable to the queue length in the fixed routing case.

Periodic routing

The problem with cyclic routing is obviously the fact that the arrival process at each station actually becomes more irregular, at least when p_1 and/or p_2 are large. The following periodic routing policy overcomes this easily: arriving tasks on stream i at even times n are sent to station i; at odd times, route arrivals on stream i to station not i. The behaviour is now exactly as with fixed routing, except that the mean arrival rates at each service station are now $(p_1+p_2)/2$ (but the arrival rates are time-dependent), instead of being p_1, respectively p_2. Again the ergodic states are (0,0), (1,0), (0,1), and (1,1); all other states are transient. The mean queue length for both queues is now $(p_1+p_2)/2$. The transient behaviour is characterized by intervals during which the queue length remains constant, with an approximately geometrically distributed duration with parameter $(p_1+p_2)/2$. At the end of each interval the queue length decreases by -1.

Random routing

Finally one can consider random routing, where each arrival is routed to either of the two stations based on independent tosses of a fair coin. Then

$$P(Q_{i,n}=Q_{i,n-1}+1 \mid Q_{i,n-1}) = p_1p_2/4,$$
$$P(Q_{i,n}=(Q_{i,n-1}-1)\vee 0 \mid Q_{i,n-1}) = (1-p_1/2)(1-p_2/2).$$

Because of symmetry considerations one sees that the marginal equilibrium distributions of $Q_{1,n}$ and of $Q_{2,n}$ are the same as the equilibrium distribution of an $M/M/1$-queue with

$$\rho = \frac{p_1p_2}{(2-p_1)(2-p_2)}, \quad E[Q] = \frac{p_1p_2}{2(2-p_1-p_2)}.$$

All states are now ergodic.

Join-the-shortest-queue
Of course, one should compare the performance of these open-loop policies with that of the closed-loop policies, which always assigns arriving tasks to the shortest queue (JSQ). In fact one can show that for the same arrival processes $(A_{1,n})$ and $(A_{2,n})$ the queue lengths of cyclic routing and of periodic routing are, for each n, at most one larger than the queue lengths with the JSQ-policy, provided that the queues are initially empty. However the transient behaviour will be different, if one starts with long queues. Fig. 2 shows why the transient behaviour of the JSQ-strategy is better than that of cyclic or periodic routing which in turn for $p_1 \neq p_2$ are slightly better than that of fixed routing.

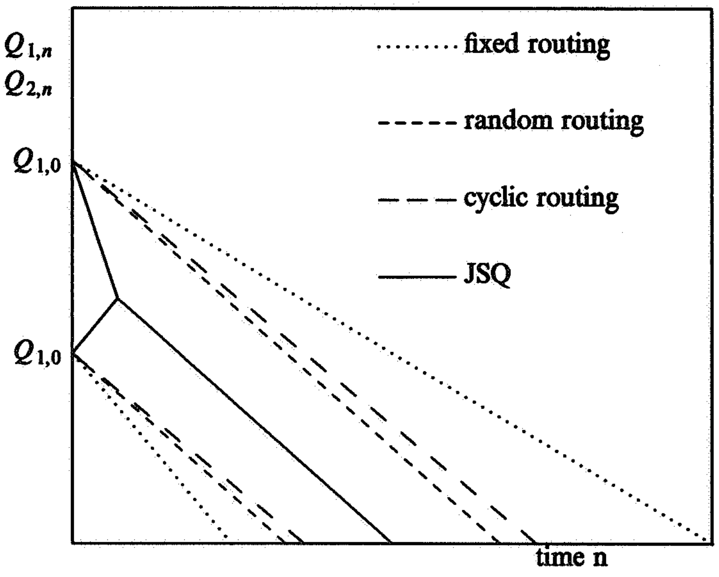

Fig. 2. The transient behaviour of several open-loop policies.

Note that the actual sample paths will, with high probability, be closer to the average behaviour for cyclic or periodic routing compared to fixed routing because there are more intervals of decrease: $(p_1 p_2 \leq (p_1 \wedge p_2))$. The advantage of cyclic and periodic routing over fixed routing clearly is a matter of averaging out overload and thus reducing the variance, rather than an improvement in mean behaviour. For *JSQ* the behaviour is initially entirely different: the longer queue is decremented by -1 at every time unit, and all arriving tasks are routed to the shorter queue untill both become equal; from then on the average behaviour is identical to that of cyclic and periodic routing, but because the two queue lengths can never be farther apart than one unit, the variance is even smaller and the sample paths are even closer to the mean behaviour. This variance reduction during transient intervals is clearly the main advantage of the JSQ policy. Under stationary conditions the only disadvantage of fixed routing compared with the other strategies is the unfairness when $p_1 \neq p_2$. In fact the variance of the queue length in equilibrium for fixed routing is equal to that for the case of JSQ or periodic routing and slightly smaller than for cyclic or random routing.

Imperfect service stations
Large queue lengths can occur due to occasional failures of one of the servers, a very realistic model. Assume that one of the service stations, say station 1, is down during N time intervals. Under fixed routing then, at the end of the breakdown the queue length at station 1 will be

$$Q_{1,0} + Binom(N,p_1)$$

in which the distribution of the random variables has the parameters N and p_1; $Q_{2,N}$ will be 0 or 1. With cyclic routing

$$Q_{1,N} = Q_{1,0} + Binom(\lfloor N/2 \rfloor, p_1) + Binom(N - \lfloor N/2 \rfloor, p_2),$$

or

$$Q_{1,N} = Q_{1,0} + Binom(N - \lfloor N/2 \rfloor, p_1) + Binom(\lfloor N/2 \rfloor, p_2),$$

depending on the state of the system at time 0; similar expressions can be written down for $Q_{2,N}$. The same result holds for periodic routing. For JSQ-routing

$$Q_{1,N} = Q_{1,0} + X_1, \quad Q_{2,N} = Q_{2,0} + X_2,$$

whereby

$$X_1 + X_2 = Binom(N,p_1) + Binom(N,p_2),$$

and $|Q_{1,N} - Q_{2,N}| \leq 1$ for $N > |Q_{1,0} - Q_{2,0}|$. Since the average waiting time during the busy period following a breakdown is proportional to the area under the queue-length trajectory, it is clear from Fig. 2 and the above argument that load balancing (JSQ or cyclic or periodic routing) will approximately halve this cost with respect to fixed routing.

When on the other hand large queue lengths are due to the fact that $q_1 < 1$ and/or $q_2 < 1$ then one should analyse the queues with the techniques of discrete time queues [7,42]. All states are now ergodic (in $\mathbb{Z}_+^2$) for all five policies provided the respective stability criteria are satisfied: $p_1 < q_1$, $p_2 < q_2$ for fixed routing, $(p_1 + p_2)/2 < (q_1 \wedge q_2)$ for cyclic, periodic and random routing, and $p_1 + p_2 < q_1 + q_2$ for JSQ, see Fig. 3.

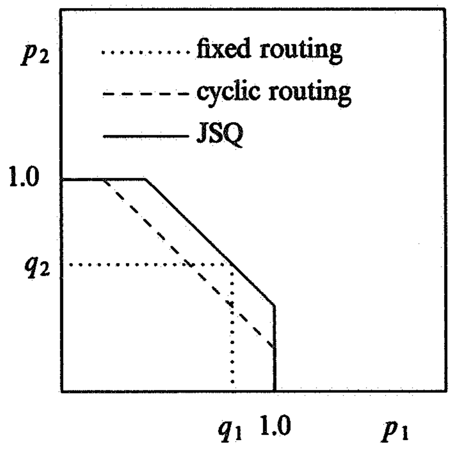

Fig. 3. The ergodicity regions for several open-loop policies.

3. Open-loop routing of several Poisson arrival streams

In this section we consider n independent Poisson arrival streams $(A_{i,t})$ with rates λ_i, $i=1,\ldots,n$. The tasks generated by these arrival streams have to be processed by one of m independent servers. Server j has a waiting room with N_j-1, $(N_j \leqslant \infty)$ waiting places. Tasks which are routed or rerouted to a full waiting room are immediately rejected and supposed never to return to the system. We assume for simplicity that the service time of any task at server j is exponential with mean $1/\mu_j$, and this service time is independent of all other service times and of all the arrival streams.

To each arrival stream i is associated a routing unit described by the function $r_i(n)$ which assigns the n-th task to server $r_i(n)$ upon its arrival. The only information available to routing unit i is the history of arrival process $A_i(s)$, $s \leqslant t$, up to the present time t and the value of time t (i.e. they know that $A_k(0)=0$, for all k). The routing units are designed in a cooperative way, so as to achieve good performance for all arrivals, and as such this can be considered a team problem. However no on-line information exchange is possible between the routing units and hence there is no on-line coordination between them. Moreover the routing strategies are *open-loop* or *feedforward* in the sense that no information at all is available about the state (the queue length) of the servers.

Random routing

The simplest possible strategy, besides fixed routing, is to use random routing. Each routing unit r_i assigns a task to server j with probability p_{ij}, independently of all other assignments or of any state information. The queue at server j then behaves like a $M/M/1/N_j$ queue with arrival intensity $\overline{\lambda}_j(p) = \sum_{i=1}^{n} p_{ij}\lambda_i$ (here p is the matrix of all elements p_{ij}). All performance measures of interest can be calculated explicitly from knowledge of $\overline{\lambda}_j(p)$ and μ_j.

Suppose we want to design the system such that the weighted mean waiting time

$$J(p) = \sum_{i=1}^{n} w_i E[W_i] = \sum_{i=1}^{n} w_i \sum_{j=1}^{m} p_{ij}\; E[W_j]$$

is as small as possible. Clearly it does not matter which arrival stream a task comes from. Hence we can take all the Poisson streams $A_i(t)$ together to obtain one arrival stream with rate $\lambda = \sum_{i=1}^{n}\lambda_j$. For the case $N_j=\infty$, $\forall j$, Bonomi and Kumar [6] have recently shown that it is possible to find $p_j^* = \overline{\lambda}_j^* / \lambda$ such that J is minimized, by solving a quadratic optimization problem in p_j. This minization reduces to balancing the idle times.

In order to find the p_{ij} one then has to solve the system of equations in p_{ij}, in $n \times m$ unknowns,

$$\sum_{j=1}^{n} p_{ij} = 1,\; p_{ij} \geqslant 0,\; \sum_{i=1}^{n} p_{ij}\lambda_i = \overline{\lambda}_j^*.$$

In general there will be infinitely many solutions, one being $p_{i,j}=p_j^*$, $\forall i$. Technologically the cheapest solution will be the one with as few non-zero values p_{ij} as possible. However there will be very little mixing of arrival streams then and hence the performance will not be robust with respect to changes in the arrival rate. Suppose the system, designed for a value λ_i^0 of the arrival rate of stream i, is to operate with arrival rate $\lambda_i = \lambda_i^0 + \Delta\lambda_i$. Then each ρ_j is increased by $p_{ij}\Delta\lambda_j/\mu_j$. Since the mean waiting time is convex-up as a function of ρ_j it is clearly advantageous to distribute this increase in load over all the servers. This suggests the choice $p_{ij}=p_j^*$. Exact sensitivity calculations require calculations of

$$\frac{dJ}{d\lambda_i} = \sum_{i=1}^{n} w_i \sum_{j=1}^{m} p_{ij} \frac{d}{d\lambda_i}\left(\frac{\overline{\lambda}_j(p)}{\mu_j(\mu_j - \overline{\lambda}_j(p))}\frac{1}{(1-\rho_j(p)^{N_j+1})}\right),$$

a very complicated function indeed.

A completely analogous development can be made for the blocking probability (i.e.

$\lim_{t \to \infty} P(Q_{r_i(n),t} = N_{r_i(n)})$) as performance measure:

$$p_{b,i} = \sum_{i=1}^{m} p_{ij} \frac{\rho_j(p)^{N_j}}{(1-\rho_j(p)^{N_j+1})}.$$

Cyclic routing
For the case $n=1$ it is known that random routing is not the best open-loop strategy. Ephremides, Varaiya and Walrand [20] have shown that, with $m=2$, assigning arrivals alternatingly to each of the servers, considerably reduces the mean waiting time. In fact, using results of Hajek [27] one could, for $n=1$, any m and any set of values $\mu_1, \ldots, \mu_m$, find a function $r\colon \mathbb{Z}_+ \to \{1, \ldots, m\}$ which assigns the n-th arriving task to server $r(n)$, such that the mean waiting time is minimized. When all the ratios μ_l / μ_k are rational with smallest common denominator t, then the optimal r will be periodic with period t and the optimal routing will be relatively easy to implement. With $m=2$ and $\mu_1 = \mu_2$ one does indeed find that alternating routing is optimal: $r(n) = ((n+1) \ mod \ 2) + 1$.

Heuristically the fact that the deterministically alternating routing policies as above are better than random routing, can be explained as follows. Alternating routing sends newly arriving tasks to those servers which have not been sent a task for a longer time than most other servers, i.e. to those servers whose waiting line is expected to be short given the previous assignments. Of course when $n>1$ and when the routing units are not coordinated, then routing unit i intending to assign a task to server j does not know whether or not other units have recently assigned a task to server j. After a long time, $t \to \infty$, the superposition of the arrival stream of tasks from $A_l(t)$, for all $l \neq i$, into server j, will look to routing unit i as a stationary (non-renewal) point process with average arrival intensity

$$\sum_{l=1, l \neq i}^{n} \lambda_l \left(\lim_{N \to \infty} \frac{1}{N} \sum_{n=1}^{N} I_{\{r_l(n)=j\}} \right).$$

The argument in favour of deterministic alternating routing remains true even though the improvement will be weaker because our prediction of the different queue lengths will be less accurate.

A second reason why deterministic alternating routing decreases the average waiting times is that the arrival process, with $n=1$, at each server j becomes more regular. Take as the simplest example $m=2$ and $N_1 = \infty$. Then the interarrival times at each server are Erlang-2 distributed with a smaller coefficient of variation than the exponential distribution. The arrival process is a renewal process so that general $GI/M/1$-results can be applied. It is known then that the equilibrium distribution, and hence the mean waiting time, blocking probabilities etc. are determined entirely by the unique root in the open interval $(0,1)$ of $\sigma = \hat{A}(\mu - \mu\sigma)$ where $\hat{A}(s)$ is the moment-generating function of the interarrival times. Then

$$\lim_{k \to \infty} P(Q_{i,T_k} = n) = \rho(1-\sigma)\sigma^{n-1},$$

$$E[W] = \frac{\sigma}{\mu(1-\sigma)},$$

in which T_k is the time of the k-th arrival. The case $N_1 < \infty$ can also be solved analytically (see Takacs [40]), but is too complicated to be interpreted easily. For the case $N_1 = \infty$ however, Hajek [25] has interpreted the above results by showing that σ should be as small as possible for good performance. Using the convexity of

$$\hat{A}^*(s) = E[\exp(-s(T_{n+1} - T_n))]$$

he shows that small coefficients of variation lead to small σ's, the smallest σ corresponding to deterministic (scheduled) arrivals. However, σ also depends on the higher moments of the interarrival times.

Unfortunately, when $n>1$, the above heuristic argument again becomes less clear. First of all, the arrival stream at one server is a superposition of n independent renewal processes and hence is never a renewal process (excepts when all the renewal processes are Poisson). Hardly any analytical results are known for queues with non-renewal arrival processes. Moreover, superposing n (>1) independent, fairly regular, arrival streams will lead to a less regular stream. Hence the advantage of deterministically alternating traffic is partly lost.

Nevertheless, it turns out that using a properly chosen set of deterministic routing policies $(r_i(n))$ leads to a considerable performance improvement over fixed or random routing. Indeed it is usually found that the equilibrium distribution is of the form $\lim_{t\to\infty} P(Q_{i,t}=k) \approx c \times \sigma^k$, for k away from 0 and N_j, and for a value of σ considerably smaller than ρ. This leads to a reduction of the mean waiting time and, for $N_1<\infty$, of the blocking probability.

The numerical results, for periodic functions $r_i(n)$, have been studied via the matrix-geometric method. Indeed, the process

$$X_t = (Z_t, Q_{1,t}, \ldots, Q_{m,t}),$$

where

$$Z_t = (r_1(A_{1,t}), \ldots, r_n(A_{n,t})),$$

is itself a Markov process. Note that, given Z_t, the queue lengths $Q_{1,t}, \ldots, Q_{m,t}$, are conditionally independent. Hence it suffices to study each Markov process $(Z_t, Q_{j,t})$ independently. Using the matrix-geometric methods of Neuts [36] one can reduce this problem, both for $N_1=\infty$ and for $N_1<\infty$, to solving a system of linear equations of dimension equal to the cardinality of the state space of Z_t. Roughly speaking the problem of finding the performance of the system is reduced to the problem of recursively solving the 2nd order difference equation

$$\begin{aligned}
&(A_0-\rho I)p_0 + p_1 = 0, \\
&A_1 p_{k-1} + [A_0-(1+\rho)I]\, p_k + p_{k+1} = 0, \quad k=2,3,\ldots,N_j-1, \\
&A_1 p_{N_j-1} + [A_0+A_1-(1+\rho)I]\, p_{N_j} = 0, \\
&\sum 1^T p_k = 1,
\end{aligned} \tag{3.1}$$

where

$$p_l = \lim_{t\to\infty} (P(Z_t=z \mid Q_{jt}=l)) \in R^{m^n},$$

$A_0+A_1=P$ is the transition matrix of (Z_t) and A_0 respectively A_1 contain those elements of P which correspond to sending arriving tasks to servers $\neq j$, respectively to server j. This recursive solution works very well for all reasonable N_j values. When $N_j\to\infty$ however stability problems are encountered. It is then better to reduce (3.1) to a first order difference equation

$$\begin{aligned}
&x_k = \begin{bmatrix} p_k \\ p_{k+1} \end{bmatrix} \in \mathbb{R}^{2m^n}, \\
&x_{k+1} = \begin{bmatrix} 0 & I \\ -A_1 & (1+\rho)I-A_0 \end{bmatrix} x_k = \Lambda x_k, \quad x_0 = \begin{bmatrix} p_0 \\ p_1 \end{bmatrix} = \begin{bmatrix} I \\ \rho I - A_0 \end{bmatrix} p_0, \\
&\sum 1^T \begin{bmatrix} I \\ 0 \end{bmatrix} x_k = 1.
\end{aligned} \tag{3.2}$$

When $N_j=\infty$, then x_0 has to lie in the space spanned by the stable eigenvalues of Λ (i.e. $|\lambda_i(\Lambda)|<1$ of which there are m^n) because otherwise the normalization condition can never be satisfied. The largest stable eigenvalue turns out from the numerical and simulation experiments to agree very well with the apparent load factor σ (i.e. for k away from 0, $p_k \approx c.\sigma^k$). For $n=2$, $m=2$, cyclic routing, we find that, for $1-\rho$ small, $\sigma \approx 1-4(1-\rho)/3+o(1-\rho)$. Since $E[Q]$ and $E[W]$ contain a factor approximately equal to $\sigma/(1-\sigma)$, this explains the 25% reduction we find in mean waiting for N_j large. Of course for N_j large this will also reduce the blocking probability by

$$\left[\frac{\sigma}{\rho}\right]^{N_j} \approx \left[\frac{4}{3}-\frac{1}{3\rho}\right]^{N_j}.$$

For $N_j<\infty$ the numerical solution of (3.1) does not pose any problem as long as $(\lambda_{\max})^{N_j}$ is less than 2^w, where $\lambda_{\max}$ is the largest eigenvalue of Λ and where w is the wordsize of the computer used. This is quite reasonable if one thinks of the solution x_k of (3.2) as a linear combination of eigenvectors of Λ:

$$x_k = \sum_{i=1}^{2m^n} (\lambda_i)^k e_i.$$

Note that x_0 is now not in the stable subspace of Λ. Instead x_0 is determined by the boundary condition at N_j. For $\rho=0.7$, $n=2$, $m=2$, cyclic routing, we find that $\lambda_{\max}\approx 3.1$, and hence on a 32-bit computer we find very easy and fast algorithms to analyze the performance as long as $N_j<21\approx 32/\log_2(3.1)$. For $N_j>21$ using the results for $N_j=\infty$ usually, but not always, leads to a sufficiently accurate approximation.

Time-varying routing
It has been explained earlier that the regularity of the arrival streams at servers is reduced due to the superposition of independent streams. Independence is a result of a lack of coordination. It is possible to achieve coordination to some extent in an open-loop policy by making the routing algorithms time-dependent (cf. the discrete-time case in section 2). However, if one simply uses $r_i(t)$, then there will only be load balancing but no regularisation of the arrival streams. Simple simulation experiments do indeed indicate that it is not possible to achieve significant performance improvements over fixed or random routing, by using $r_i(t)$ as routing algorithm. Considerable improvement, even with respect to the generalized cyclic routing $r_i(A_{i,t})$, can undoubtedly be achieved by using routing algorithms of the form $r_i(A_{i,t},t)$. This should be compared to the use of time dependent control strategies for removing fixed modes in decentralized control of linear systems [2]. Another good strategy, requiring some memory in each routing unit, would be to use $r_i(A_{i,t}-A_{i,kT})$ for $kT\leq t<(k+1)T$. This also introduces coordination between the different routing units in open loop. We have not yet been able to study the performance of these time-dependent routing units analytically. On the other hand, it is not feasible to properly design the routing units via simulation because there are too many free parameters. Therefore we will not elaborate further on this type of time-dependent routing.

Performance for specific examples
To conclude this section on open-loop routing strategies we illustrate for some examples the performance improvement which can be obtained. In the figures 4.a, 4.b and 4.c we plot, on the basis of numerical experiments, $E[Q]$, $Var(Q)$ and $\log(p_b)$ as function of the server load $\rho=\lambda/\mu$ for the case $n=2$, $m=2$, $\lambda_1=\lambda_2$ $\mu_1=\mu_2$, $N_1=N_2=10$, and $r_1(A_{1,t}) = (A_{1,t} \; mod \; 2)+1$, $r_2(A_{2,t}) = ((A_{2,t}+1) \; mod \; 2)+1$. Note that the mean waiting time for accepted tasks is

$$E[W] = \left[\frac{1-p_b}{\mu}\right] E[Q].$$

We clearly obtain an improvement of up to 25% in performance ($E[Q]$, $Var(Q)$, $E[W]$, $Var(W)$, p_b) for moderate values of ρ ($\rho\leq 0.8$). The blocking probability is actually reduced for all values of ρ. The fact that for large $\rho(\approx 0.9)$ the improvement becomes insignificant is due to the fact that the queues are almost never empty. Indeed, for $N_i=\infty$, regularizing the arrival stream really has only one advantage: the chance of sending a task to an empty queue increases. Note also that the fact that for $\rho=0.9$, $E[Q]$ is larger for cyclic routing than for fixed

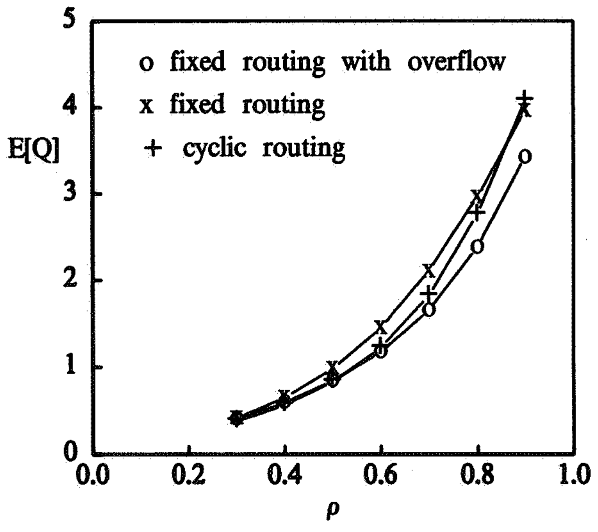

Fig. 4.a. Mean queue length versus load
for two open-loop routing policies (Fixed routing and Cyclic routing)
and one closed-loop routing policy (Fixed routing with overflow).

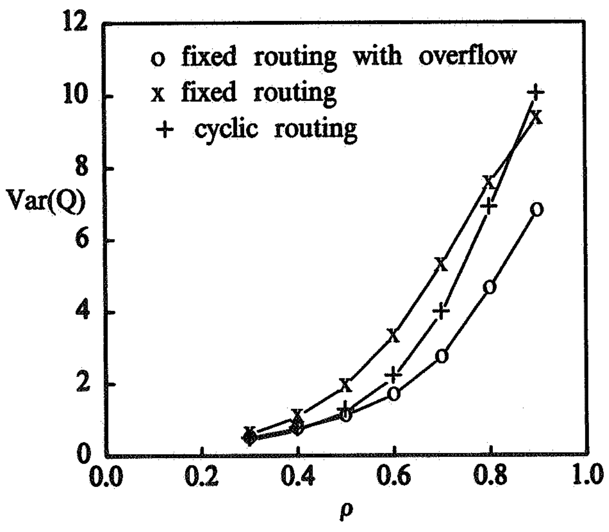

Fig. 4.b. Variance of queue length versus load for two open-loop
routing policies and one closed-loop routing policy.

routing, is misleading; it is due to the heavier load of the queue with cyclic routing since more tasks are accepted (p_b decreases). Indeed the phenomenon disappears when N_j increases.

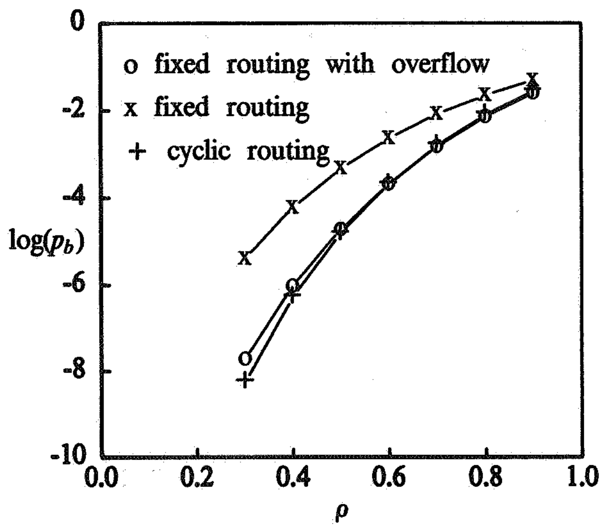

Fig. 4.c. Logarithm of blocking probability versus load for two open-loop routing policies and one closed-loop routing policy.

Robustness of performance
The main purpose of routing strategies, different from fixed routing, is to make the performance robust with respect to changes in the arrival parameters. With fixed routing there is a significant increase in mean waiting time, blocking probability, etc. when the arrival rates become unequal. With random routing on the other hand there is no change in performance as long as the total arrival rate remains unchanged. For cyclic routing with constant total arrival rate, there is actually a small reduction in mean waiting as one arrival stream becomes dominant. For example, with $\rho=0.7$ we find that for $\lambda_1=\lambda_2=0.7\mu$, $E[Q]=1.8115$, for $\lambda_1=0.5\mu$, $\lambda_2=0.9\mu$ we obtain $E[Q]=1.8109$, a reduction of less than 0.1%. For $Var(Q)$ and p_b the reduction is less than 1%. It is actually intuitively reasonable that for cyclic routing the performance improves as the load becomes more unbalanced: as one of the two, regularized, superimposed renewal processes becomes more dominating, the superposition itself becomes more regular. The performance approaches that of an $E_2/M/1$ queue. These results are confirmed for other parameter values. In figure 5.a we indicate how $E[Q]$ varies as a function of λ_2 for $n=m=2$, cyclic routing, $\lambda_1=0.7$, $\mu_1=\mu_2=1$, $N_1=10$. Figure 5.b displays $E[Q]$ as a function of λ_1 in case $\lambda_1+\lambda_2=1.4$ and figure 5.c displays the logarithm of the blocking probability in the same case. Here also we find good robustness properties.

Of course all the previous examples were for the case $\mu_1=\mu_2$, when the alternating routing strategy is optimal. As soon as $\mu_1\neq\mu_2$ it becomes necessary to use more complicated routing strategies. We will describe these strategies by giving the sequence of servers over one period, e.g. $1\to1\to2\to1\to2\to\cdots$. The arrival streams at the servers are less regular now. Mathematically this leads to larger values of the largest stable eigenvalue of Λ. Nevertheless we still do find a significant improvement over fixed routing if we choose routing algorithms properly. Consider the case $\lambda_1=\lambda_2=0.7$, $\mu_1=1$, $\mu_2=1.5$, $N_1=10$ then we find:

a. for policies $r_1:1\to2$, $r_2:1\to2$, $E[Q_1]=1.0379$, $E[Q_2]=1.1117$, and a queue length of 1.0748 seen by each arrival stream 2. Compare this to $E[Q_1]=2.111$ and $E[Q_2]=0.862$ for fixed routing and $E[Q_1]=E[Q_2]=1.244$ for random routing.
b. for policies $r_1:1\to1\to2$, $r_2:1\to1\to2$, $E[Q_1]=1.4497$, $E[Q_2]=0.6948$, or a mean queue length as seen by each arrival stream of 1.198.

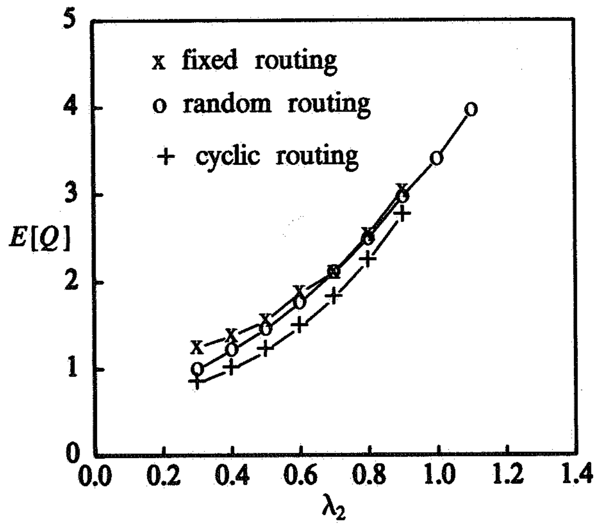

Fig. 5.a. Mean queue length versus λ_2 for several open-loop routing policies.

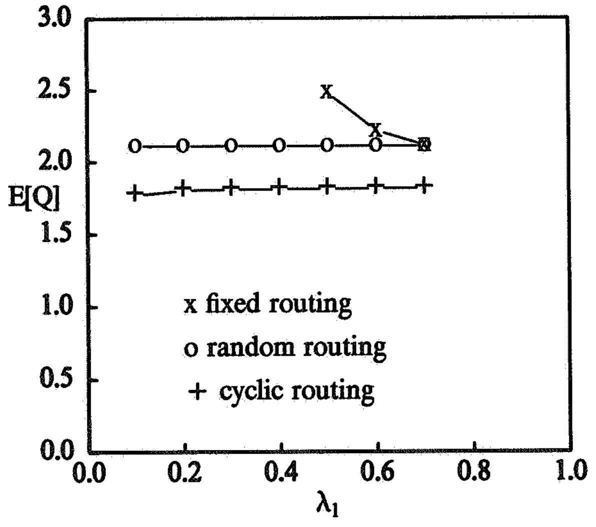

Fig. 5.b. Mean queue length versus λ_1 for several open-loop routing policies in the case $\lambda_1+\lambda_2=1.4$.

For the case $\lambda_1=\lambda_2=0.7$, $\mu_1=1$, $\mu_2=2$, $N_1=10$ we consider the policies:

a. r_1:$1\rightarrow1\rightarrow2$, r_2:$1\rightarrow1\rightarrow2$, $E[Q_1]$=0.8011, $E[Q_2]$=0.6948, $E[Q]$=0.7657 as seen by both arrival streams.

b. r_1:$1\rightarrow1\rightarrow2$, r_2:$1\rightarrow1\rightarrow1\rightarrow2$, $E[Q_1]$=0.9118, $E[Q_2]$=0.5556, $E[Q]$=0.7931 as seen by arrival stream 1, $E[Q]$=0.8228 as seen by arrival stream 2.

These values should be compared to:

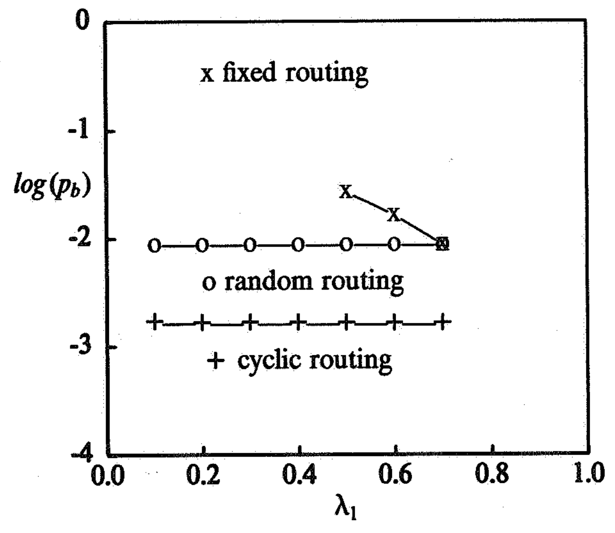

Fig. 5.c. Logarithm of blocking probability versus λ_1 for several open-loop routing policies in the case $\lambda_1 + \lambda_2 = 1.4$.

- fixed routing: $E[Q_1]=2.11$, $E[Q_2]=0.535$;
- random routing: $E[Q_1]=E[Q_2]=0.86$.

It should be noted that in each case the blocking probability $p_{b,i}$ is smallest for the most balanced load case, i.e. for policy a. in both examples. The performance is also very robust in these examples with respect to load changes, just as it was in the case of alternating routing.

4. Overflow routing strategies

An obvious weakness of the open-loop routing strategies of the previous section is the fact that they are not robust with respect to changes in the service capacity. If at some time T_0 one of the servers, say server i, suddenly fails, so that from T_0 on the service rate drops from μ_j to μ_{j0}, then a long queue will quickly build up behind it. Our open-loop routing mechanism will not adapt to this at all. It will keep sending tasks to the failed server; Q_{jt} may well, if $\mu_{j,0}$ is less than the arrival rate at server i, remain nearly full all the time. The blocking rate will therefore rise drastically.

To make the system robust against such server failures, it is necessary to let the routing units use information on the queue lengths. The routing strategy now becomes a *closed-loop* or a *feedback* policy. Often it is either too costly or technically not feasible to give all routing units access to complete state information. If this were possible each routing unit would at each time send an arriving task to the server with the shortest queue in case $\mu_1 = \mu_2$. This minimizes the expected waiting time in the class of all policies which assign a task upon its arrival to a server [20,45,46]. Of course one can do even better if one postpones the assignment decision until a server becomes free, or equivalently if unlimited jockeying between the queues is possible (See [4] for a discussion on how much further improvement in performance this can give). This last case is simply an $M/M/m$ queue.

Since the policies of the preceding paragraph are too costly to implement, they only serve as benckmarks against which to compare the following policies. Given any of the open-loop policies of the previous section, let the only information available for routing unit i at the arrival time T_n^i of the n-th task be the binary value "is $Q_{r_i(n)}(t)$ larger than a threshold level $k_{r_i(n)}$ or not". If not, then the arriving task is assigned to server $r_i(n)$; if yes, then the arriving task is assigned to server $r_i(n)+1$. Such a policy is called an *overflow* policy. It will be analyzed in this section. The synthesis problem of designing a good overflow policy therefore reduces to the choice of good threshold levels k_j, $j=1,\dots,m$, corresponding to nominal arrival rates λ_i, $i=1,\dots,n$, to nominal service rates μ_j, $j=1,..,m$, and to routing functions $r_i(n)$.

Performance measures
For the case of two arrival streams and two service stations we have calculated the following performance measures for various parameter values:

1. The average queue length $E[Q_j]$ at each server, and directly related to it, the average waiting time at each server.
2. The blocking probability $p_{b,j}=\lim_{t\to\infty} P(Q_j(t)=N_j)$ at each server. Notice that the policy described above may reroute an arriving task from a waiting line $r_i(n)$ with $k_{r_i(n)}<Q_{r_i(n)}<N_{r_i(n)}$ to waiting line $r_i(n)+1$, which may be full, $Q_{r_i(n)}+1=N_{r_i(n)+1}$. If blocking is a serious problem in the system to be designed, then it is probably sensible to install extra hardware and software to avoid this. Thus the blocking probabilities we calculate can be reduced further at little extra cost.
3. The fraction of rerouted traffic $p_{r,j}=\lim_{t\to\infty} P(Q_j(t)>k_j)$.

These three performance measures should remain small when the arrival rates and/or the service rates deviate from the nominal values. The case of three arrival streams and three service stations can be computed by the same method as for the case of two arrival streams and two stations.

The rerouting fractions should be small because there is an obvious communications cost associated with it. In some systems there will be a separate communication network for transmitting rerouted tasks. The bandwidth, and hence the cost, of this network will grow with $p_{r,j}$, and in fact also with the burstyness of the rerouting traffic. In other systems rerouting uses the servers and will reduce the service rate of both sending and receiving server [32]. Since it will be very difficult to model these costs exactly we prefer to simply use $p_{r,j}$, $j=1,..,m$ as a performance measure to be kept small.

There is actually another reason why $p_{r,j}$ should be small, and consequently why the overflow threshold levels k_j should be large enough. If k_j is chosen too small and when server $j+1$ breaks down (i.e. μ_{j+1} becomes considerably smaller than its nominal value used for design purposes), then the following undesirable situation will occur eventually: due to normal random fluctuations $Q_j(t)>k_j$ and the next arriving task at server j is rerouted to server $j+1$, where it experiences a long average waiting time; the queue length at the server $j+1$ increases further causing more rerouting; this may eventually lead to instability in case of infinite buffers. Similar problems occur in the feedback control of classical, continuous, noisy nonlinear systems. The fact that limited observations do not allow the controller to distinguish large deviations due to noise from systematic deterioration forces the designer of a practical control system to use dead zones and saturation, i.e. policies very similar to the overflow strategies discussed in this section [37].

Synthesis of overflow routing
It should be noted that in the overflow strategy described here each decisionmaker, i.e. each routing unit, has a different information set at each time. Therefore we deal with a team problem, for which no dynamic programming algorithm is available. Therefore there is no way of proving the optimality of a bang-bang strategy of the type: reroute when $Q_{r_i(n)}>k_{r_i(n)}$. In fact, one expects that the optimal strategies for this team problem will be randomized. For the case where n and m are small, the following analysis shows that the overflow policies can achieve good performance in a way which is robust with respect to changes in service rates. No claim of optimality can be made however.

Performance evaluation
To keep the amount of computer time required for the calculations small, we have mainly concentrated on the cases $n=m=2$. A few runs with $n=3$ and/or $m=3$ have shown that extension of the methods is feasible. For an analytic treatment see [21].

First we consider, for $n=m=2$, the case of fixed routing, $r_i(n)=i$, with overflow. It is then fairly easy to write the problem of calculating the equilibrium distribution in matrix-geometric form, or at least in a form similar to that used in the analysis of cyclic routing with overflow. The blocks to be worked with are now of dimension N_1 (or N_2, choose whatever is smaller). In the symmetric case, $\lambda_1=\lambda_2$, $\mu_1=\mu_2$, $N_1=N_2$, we find that, as one expects, $k=k_1=k_2$ leads to good performance. It turns out that a good choice of k should increase with the load $\rho=\lambda_1/\mu_1$.

Secondly we considered for $n=m=2$ the case of cyclic routing with overflow where $r_i(n)$ is chosen to give good open-loop performance as explained in section 3. In principle it is still possible to apply the matrix-geometric method, and its extension to finite queues, but the blocksizes now become too large. Each block again has a lot of structure, i.e. it can again be divided in similar blocks, so that one can hope that a two-step hierarchical application of the matrix-geometric method will lead to efficient numerical solution methods. This program has not been carried out yet. However we did find, when N_1 and N_2 are limited to sizes up to about 10 or 15, that then a Gauss-Seidel solution of the Kolmogorov equations is feasible provided one uses, as suggested in [29], the following tricks to speed up the convergence:

1. use as initial value for the state distribution, the one corresponding to cyclic routing without overflow
2. use a selective under relaxation method, where the optimal relaxation coefficient is found to be about 0.95 in all cases.

A few computerruns have also been made for $n=m=3$ and for cases with $\mu_1 \neq \mu_2$ nominally and hence with more complicated routing strategies. These confirm both that the Gauss-Seidel approach remains feasible, and that the general conclusions of the following discussion remain valid. As far as mean queue length and rerouting probabilities are concerned, simulations using the simulation package *QNAP* with $n=m=8$ for several different routing policies $r_i(n)$, have shown that roughly the same results continue to hold. However more sophisticated programming techniques - perturbation analysis and importance sampling in particular - will be necessary to draw clear cut conclusions about blocking probabilities in this case.

Numerical results
To illustrate the performance achievable with overflow routing we plot in the figures 6.a and 6.b the performance measures $E[Q]$ and $E[W]$ respectively for fixed routing with overflow and cyclic routing with overflow, for $\lambda_1=\lambda_2=0.7$, $\mu_1=\mu_2=1.0$, $N_1=N_2=10$ as a function of $k_1=k_2=k=0,1,2,3,4,5,8$. For the sake of comparison we also indicate in the figures the levels achievable with fixed routing (equal to the performance of random routing in this case), with cyclic routing with overflow, with "join the shortest queue" and finally as best possible performance the results for an $M/M/2/20$ queue with $\rho=0.7$. Finally in Fig. 6.c the rerouting probability versus overflow threshold is indicated. One notices immediately that $k=2$ gives good performance (very close to the minimal value achievable) in terms of $E[Q]$, $E[W]$ (and also $Var(Q)$ and $Var(W)$ it turns out). However the rerouting fraction remains excessive. Especially for fixed routing, using an overflow policy with $k>2$ leads to a significant improvement in terms of rerouting fractions with only minor degradations of the performance measures $E[Q]$, $E[W]$ and p_b.

To illustrate the robustness of the performance, compare the results of Table 1 for $\mu_1=\mu_2=1$, $N_1=N_2=10$, $k_1=k_2=2$. Obviously the robustness against disturbances in the arrival rate is completely determined by the open-loop policy.

To study the robustness of the overflow controllers with respect to changes of the service rates we have compared several cases of reasonably well designed nominal systems. The results are illustrated in Table 2 for the following example: $\lambda_1=\lambda_2=0.7$, $N_1=N_2=10$, $k_1=k_2=2$ and $\mu_1=\mu_2=1$ vs.

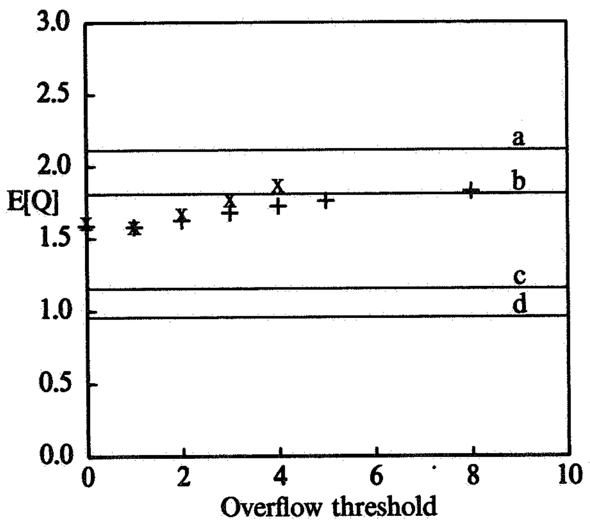

Fig. 6.a. Mean queue length versus overflow threshold for several routing policies:
Fixed routing with overflow (x), Cyclic routing with overflow (+),
Fixed routing (a), Cyclic routing (b), JSQ (c), and $M/M/2/20$ (d).

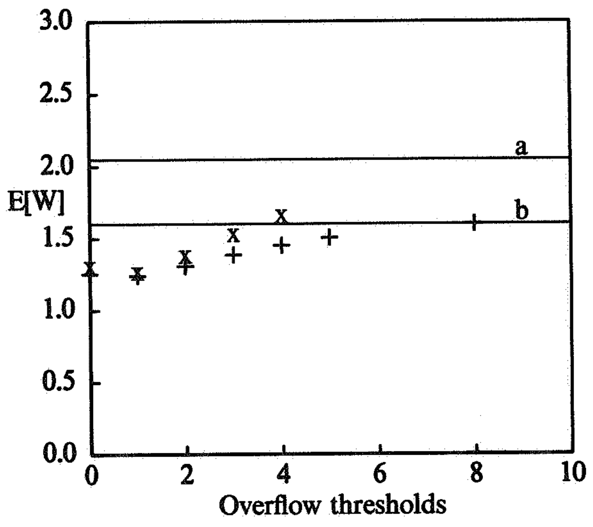

Fig. 6.b. Mean waiting time versus overflow threshold for several routing policies:
Fixed routing with overflow (x), Cyclic routing with overflow (+),
Fixed routing (a), and Cyclic routing (b).

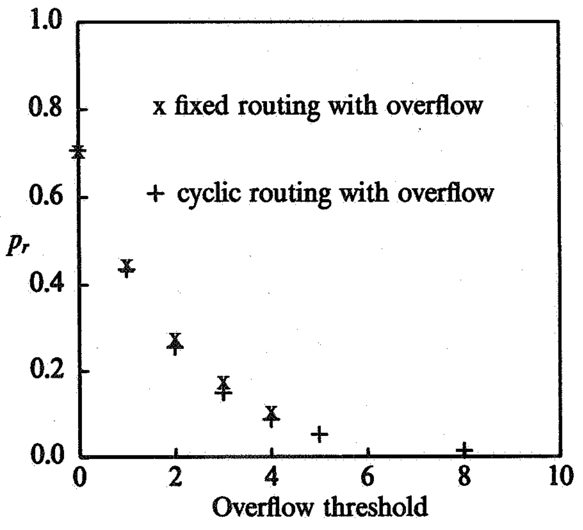

Fig. 6.c. Rerouting probability versus overflow threshold for two routing policies.

Routing policy	$E[W_1]=E[W_2]$	$E[W_1]$	$E[W_2]$
	$\lambda_1=\lambda_2=0.7$	$\lambda_1=0.3$	$\lambda_2=1.1$
Fixed routing	2.043	0.450	∞
Random routing	2.043	2.043	2.043
Cyclic routing	1.596	1.589	1.589
Fixed routing with overflow	1.274	1.10	1.54
Cyclic routing with overflow	1.222	1.206	1.206
JSQ	1.108		
$M/M/2/20$	0.966	0.966	0.966

Table 1. Performance of several routing policies.

$\mu_1=0.5$, $\mu_2=1.5$.

Routing policy	$E[W]$	p_b	$E[W_1]$	$E[W_2]$	$p_{b,1}$	$p_{b,2}$
	$\mu_1=\mu_2=1.0$		$\mu_1=0.5$, $\mu_2=1.5$			
Fixed	2.043	0.086	∞	1.863	1	0.00075
Fixed with overflow	1.37	0.0015	2.12	1.79		
Cyclic	1.596	0.000206	3.84	1.55	0.000580	0.000580
Cyclic with overflow	1.222	0.000173	2.281	1.159	0.000313	0.000313
JSQ	1.108					
$M/M/2/20$	0.966		0.966	0.966		

Table 2. Robustness of performance for several routing policies.

Clearly if the rerouting cost is significant it is difficult to build a very robust system, with overflow control, since there always is significant rerouting from the slow server. Apart from rerouting considerations we find that $k=2$ is about optimal in the above example. The mean waiting time at each server and the blocking probability are indeed minimal for $k=2$ in the unbalanced case ($\mu_1=0.5$, $\mu_2=1.5$). Notice that in the balanced case $k=1$ was optimal (see Fig. 5a, 5b) but that the results were not very sensitive to an increase in k. For the unbalanced case the results of Table 3 show that the dependence on k is quite strong for the case: $\lambda_1=\lambda_2=0.7$, $N_1=N_2=10$; $\mu_1=0.5$, $\mu_2=1.5$; cyclic routing with overflow.

Overflow	$E[W_1]$	$E[W_2]$	$p_{b,1}=p_{b,2}$	$p_{r,1}$	$p_{r,2}$
threshold					
1	2.482	0.959	0.000383	0.731	0.346
2	2.281	1.159	0.000313	0.579	0.203
3	2.500	1.326	0.000346	0.486	0.125
4	3.152	1.455	0.000434	0.430	0.079
5	3.837	1.553	0.000580	0.394	0.051

Table 3. Performance of cyclic routing with overflow as function of overflow threshold.

Clearly when rerouting is expensive it will be worthwhile to pay for a communication network that allows for more centralized control, or for processing capacity that will detect server failures quickly and modify the overflow limit of other servers accordingly. This last suggestion amounts to a decentralized adaptive controller. This raises new, unsolved, stability problems.

To understand better how the overflow policy operates we plot in the figures 7.a and 7.b $E[Q_1\,|\,Q_2=n]$ and $E[Q_2\,|\,Q_1=n]$ respectively for the case of cyclic routing with overflow, $\lambda_1=\lambda_2=0.7$, $N_1=N_2=10$, $k_1=k_2=2$, the balanced case $\mu_1=\mu_2=1.0$ vs. the unbalanced case $\mu_1=0.5$, $\mu_2=1.5$.

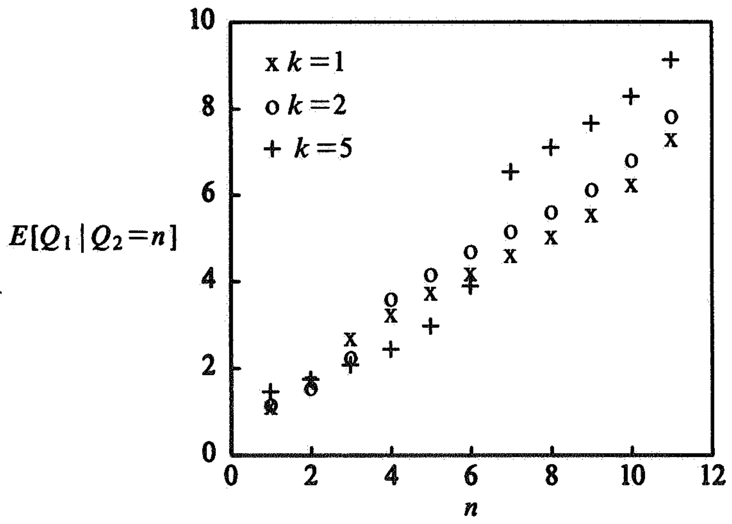

Fig. 7.a. Conditional queue length $E[Q_1\,|\,Q_2=n]$ versus n for overflow thresholds $k=1,2,5$ in case of cyclic routing with overflow.

Notice the tempting conjecture that for an "optimal" choice of the overflow limit one should have that $E_k[Q_1\,|\,Q_2=n]>n$ for $n\leqslant k$ and $E_k[Q_1\,|\,Q_2=n]<n$ for $n>k$. However this is not properly formulated since the conditional expectations themselves depend on the choice of k. In fact this

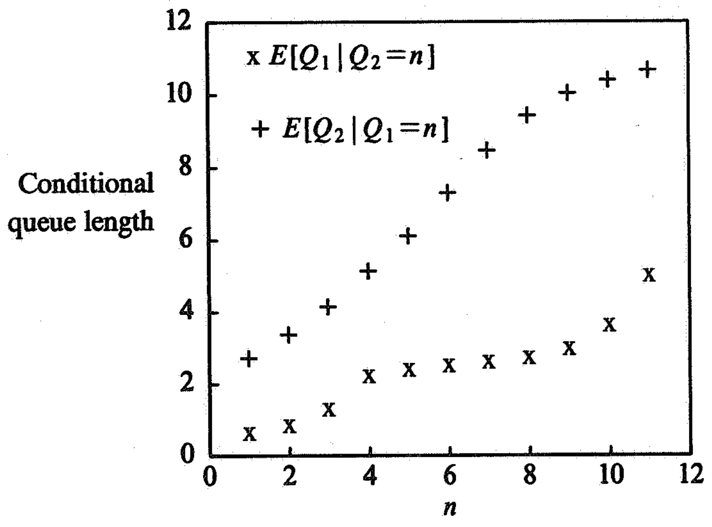

Fig. 7.b. Conditional queue length versus measured queue length
for overflow threshold $k=2$ in case of cyclic routing with overflow and $\mu_1=0.5$ and $\mu_2=1.5$.

suggested conjecture is an equilibrium condition in game theoretic sense and thus an individual optimization rule. There is no reason to believe that it will give a good social optimum, and we have indeed found good choices of k for which the conjecture is false.

5. Conclusions

In the previous sections we have shown that, at least for the simple models which we could analyze, open-loop cyclic routing and closed-loop overflow routing can provide good performance reasonably robustly. However a lot of open problems remain. First of all we would like to be able to analyze cyclic routing with overflow more efficiently, e.g. by the modified matrix-geometric method. For the case $m=2$, with infinite buffers, $N_1=N_2=\infty$ [12] opens up the possibility of obtaining an analytic solution via reduction to a boundary value problem [13].

The analysis of systems with overflow routing would be much easier if the queues behind different service stations remained conditionally independent. For the small systems ($n=2$ or 3, $m=2$ or 3) which we could study, this independence assumption turns out not to be a good approximation. When n and m are large, it might be possible to design the routing units in such a way that the independence assumption becomes a good approximation. To achive this, design the open-loop cyclic routing units in such a way that tasks from arrival stream i are sent to a service station in K_i, where K_i is a small subset of $\{1,...,m\}$. Let $\bar{K}_i=\{l \in \mathbb{N} \mid K_l \cap K_i \neq \emptyset\}$ be the indices of all arrival streams which use at least one service station in common with arrival stream i. Let the overflow routing be organized in such a way that overflow tasks from arrival stream i, i.e. overflowing from K_i, are distributed over the service stations which do not belong to $\cup_{l \in \bar{K}_i} K_l$, i.e. those service stations which do not receive any non-overflow tasks interacting with the tasks of arrival stream i. Clearly if the rerouting probability is sufficiently small then the independence assumption will be a good approximation. This will greatly simplify the design of the routing units. Moreover on intuitive grounds we conjecture that, for n and m large enough so that both K_i and $(\cup_{l \in \bar{K}_i} K_l)^c$ contain sufficiently many elements to achieve good load balancing, the robustness and stability of such a system will actually be better than was the case for the strategies discussed in section 4.

ACKNOWLEDGEMENT
The authors are grateful to the Flemish Department of Education of the Belgian Government, and to The Netherlands' Government which through their cultural exchange agreement provided financial support for their cooperation. They also thank Mr. F. van den Broeck for implementing some of the algorithms of this paper.

REFERENCES

1. M. AICARDI, F. DAVOLI, and R. MINCARDI (1987). Decentralized optimal control of Markov chains with a common past information set, *IEEE Trans. Automatic Control*, 32, 1028-1031.
2. B.D.O. ANDERSON and J.B. MOORE (1981). Time-varying feedback laws for decentralized controls, *IEEE Trans. Automatic Control*, 26, 1133-1139.
3. R.N. ANDRIES, M. GRUSZECKI, J. MASSANT, G.H. PETIT, and P. VAN ESBROECK (1985). Simulation of distributed microprocessor control in digital switching systems, in *Proceedings ITC 11, Paper 5.10-1*, ed. A. Minoru, Elsevier Science Publishers B.V..
4. J.P.C. BLANC (1985). *A note on waiting times in systems with queues in parallel,* Technical Report 85-46, Department of Mathematics and Informatics, Technical University of Delft, Delft.
5. S.H. BOKHARI (1979). Dual processor scheduling with dynamic reassignment, *IEEE Trans. Software Engrg.*, 5, 341-349.
6. F. BONOMI and A. KUMAR (1988). *Adaptive optimal load balancing in a non-homogeneous multiserver system with a central job scheduler,* Preprint, A.T.&T. Bell Laboratories, Holmdel.
7. H. BRUNEEL (1986). A general treatment of discrete-time buffers with one randomly interrupted output line, *European J. Oper. Res.*, 27, 67-81.
8. G. CASALINO, F. DAVOLI, R. MINCARDI, P.P. PULIAFITA, and R. ZAPPOLI (1984). Partially nested information structures with a common past, *IEEE Trans. Automatic Control*, 29, 846-850.
9. L.M. CASEY (1981). Decentralized scheduling, *Australian Comput. J.*, 13, 58-63.
10. T.C.K. CHOU and J.A. ABRAHAM (1982). Load balancing in distributed systems, *IEEE Trans. Software Engrg.*, 8, 401-412.
11. Y.C. CHOW and W. KOHLER (1979). Models of dynamic load balancing in a heterogeneous multi processor system, *IEEE Trans. Comm.*, 28, 354-361.
12. J.W. COHEN (1988). A two-queue model with semi-exhaustive alternating service, in *Performance '87*, ed. P.J. Courtois, G. Latouche, Elsevier Science Publishers B.V., Amsterdam.
13. J.W. COHEN and O.J. BOXMA (1983). *Boundary value problems in queueing systems analysis*, North-Holland Publ. Co., Amsterdam.
14. B.W. CONOLLY (1984). The autostrada queueing problem, *J. Appl. Probab.*, 21, 394-403.
15. C. COURCOUBETIS and P. VARAIYA (1985). Optimal resource allocation for two processes, *AT&T Technical Journal*, 64, 1-14.
16. J.R. DE LOS MOZOS MARQUES and A. BUCHHEISTER (1981). ITT 1240 Digital exchange traffic handling capacity, *Electrical Communication*, 56, 207-217.
17. E. DE SOUZA E SILVA and M. GERLA (1984). Load balancing in distributed systems with multiple classes and site constraints, in *Performance '84*, 17-33.
18. D.L. EAGER, E.D. LAZOWSKA, and J. ZAHORJAN (1986). Adaptive load sharing in homogeneous distributed systems, *IEEE Trans. Software Engrg.*, 12, 662-675.
19. M. EISENBERG (1979). Two queues with alternating service, *SIAM J. Appl. Math.*, 36, 287-303.
20. A. EPHREMIDES, P. VARAIYA, and J. WALRAND (1980). A simple dynamic routing problem, *IEEE Trans. Automatic Control*, 25, 690-693.
21. G. FAYOLLE, P.J.B. KING, and I. MITRANI (1982). The solution of certain two-dimensional Markov models, *Adv. Appl. Prob.*, 14, 295-308.
22. L. FLATTO and H.P. MCKEAN (1977). Two queues in parallel, *Comm. Pure & Appl. Math.*, 30, 255-263.
23. I. GERTSBAKH (1984). The shorter queue problem: A numerical study using the matrix-geometric solution, *Eur. J. Oper. Res.*, 15, 374-381.

24. D.G. HAENSCHKE, D.A. KETTLER, and E. OBERER (1981). Network management and congestion in the U.S. telecommunications network, *IEEE Trans. Comm.*, 29, 376-385.
25. B. HAJEK (1983). The proof of a folk theorem on queueing delay with applications to routing in networks, *J. A.C.M.*, 30, 834-851.
26. B. HAJEK (1984). Optimal control of two interacting service stations, *IEEE Trans. Automatic Control*, 29, 491-499.
27. B. HAJEK (1985). Extremal splittings of point processes, *Math. OR*, 10, 543-.
28. K. HWANG, W.J. CROFT, G.H. GOBLE, B.W. WAH, F.A. BRIGGS, W.R. SIMMONS, and C.L. COATES (1982). A UNIX-based local computer network with load balancing, *IEEE Computer*, 15 (1982, April), 55-66.
29. L. KAUFMAN, B. GOPINATH, and E.F. WUNDERLICH (1981). Analysis of packet network congestion control using sparse matrix algorithms, *IEEE Trans. Comm.*, 29, 453-466.
30. P. KRUEGER and M. LIVNY. *Load balancing, load sharing and performance in distributed systems*, University of Wisconsin-Madison.
31. A. KUMAR (1986). Adaptive load control of the central processor in a distributed system with a star topology, in *IEEE Conference on Decision and Control*, 1697-1699, IEEE Press.
32. K.J. LEE and D. TOWSLEY (1986). A comparison of priority-based decentralized load balancing policies, *Perform Eval. Rev.*, 14, 70-77.
33. G. LE LANN (1981). A distributed system for real-time transaction processing, *IEEE Computer*, 14 (1981, Feb.), 43-48.
34. W. LIN and P.R. KUMAR (1982). Stochastic control of a queue with two servers of different rates, in *Analysis and Optimization of Systems*, 719-728, ed. A. Bensoussan, J.L. Lions, Springer-Verlag, Berlin.
35. D. MITRA and R. CIESLAK (1986). *Randomized parallel communications on an extension of the Omega network*, AT&T.
36. M.F. NEUTS (1981). *Matrix-geometric solution in stochastic models - An algorithmic approach*, The John Hopkins University Press, Baltimore.
37. B.B. PETERSON and K.S. NARENDRA (1982). Bounded error adaptive control, *IEEE Trans. Automatic Control*, 27, 1161-1168.
38. P. SARACHIK (1984). Congestion reducing dynamic routing strategies for multidestination traffic networks, in *Proceedings of the 23rd Conference on Decision and Control*, 1383-1387, IEEE Press.
39. H.S. STONE (1978). Critical load factors in two distributed systems, *IEEE Trans. Software Engrg.*, 4, 254-258.
40. L. TAKACS (1962). *Introduction to the theory of queues*, Oxford University Press, New York.
41. A.N. TANTAWI and D. TOWSLEY (1985). Optimal static load balancing in distributed computer systems, *J. ACM*, 32, 445-465.
42. D. TOWSLEY (1980). The analysis of a statistical multiplexer with nonindependent arrivals and errors, *IEEE Trans. Comm.*, 28, 65-72.
43. D. TOWSLEY and R. MIRCHANDANEY (1987). The effect of communication delays on the performance of load balancing policies in distributed systems, in *Second International Workshop on Applied Mathematics and Performance/Reliability Models of Computer/Communication Systems*, 213-226.
44. Y.T. WANG and R.J.T. MORRIS (1985). Load sharing in distributed systems, *IEEE Trans. Computers*, 34, 204-217.
45. R.W. WEBER (1978). On optimal assignment of customers to parallel servers, *J. Appl. Probab.*, 15, 406-413.
46. W. WINSTON (1977). Optimality of the shortest line discipline, *J. Appl. Probab.*, 14, 181-189.
47. P.S. YU, S. BALSAMO, and Y.-H. LEE (1986). Dynamic load sharing in distributed database systems, in *Proc. Fall Joint Comput. Conf.*, 675-683.