TWO QUEUES AND ONE SERVER WITH
THRESHOLD SWITCHING

BY

Mičha Hofri

Technical Report #376

October 1985
TWO QUEUES AND ONE SERVER WITH THRESHOLD SWITCHING

Micha Hofri
Department of Computer Science, Technion
Haifa, 32000 Israel.

Two queues, with Poisson arrivals and general service-time distributions are attended by a single server. When the server is positioned at a certain queue it will serve there exhaustively, and at busy-period end will only switch to the other if the queue length there exceeds in size a predetermined threshold $m_k$. The treatment combines analytic and numerical methods, to obtain the distribution of queue lengths at different sets of epochs. Only steady-state results are presented.

A. Introduction

1. The following computational problem came up in the context of investigating the optimal control properties of queueing systems:

Two independent time-homogeneous Poisson processes trigger arrivals to two waiting-lines attended by a single server. There is no bound on the size of the waiting-lines. The server operates according to the following "threshold switching" policy:

- it will never switch from a non-empty queue, nor idle. This implies that a waiting-line is served exhaustively once it captured the attention of the server;
- the server will only switch, from line $i$ to line $i'$ ($i, i' \in \{1, 2\}$) when line $i$ empties, and line $i'$ exceeds $m_k$, where the $m_k, k = 1, 2$ are predetermined non-negative integers.

The values of the line lengths $X_1, X_2$ are assumed to be instantaneously available to the server.

A customer from the waiting line $i$ occupies the server for a duration $\sim S_i$, with distribution $F_i(\cdot)$, density $f_i(\cdot)$, and LST $S_i(\cdot)$. Switching from line $i$ to line $i'$ requires a duration $\sim A_{i}$, with LST $A_{i}(\cdot)$.

The rates of the above Poisson processes will be denoted by $\lambda_i$. The symbol $U_y$ will denote the number of arrivals to line $y$ during a period $\sim U_y$; $U_y(\cdot)$ denotes its probability generating function (pgf), and we have $U_y(\check{\varepsilon}) = U_y(\lambda y (1-\varepsilon))$.

The following symbols will also be employed:

$X_i$ - the number of customers in line $i$ (including the one in service, if there is one).
\( \tau_i(\cdot) \) - pmf for \( X_i \) at departure epochs from line \( i \).
\( G_i(\cdot) \) - generating function for the \( \tau_i(\cdot) \).
\( Q_i(\cdot) \) - joint pgf for \( X_1 \) and \( X_2 \) at a random departure epoch.
\( L_i \) - the value of \( X_i \) when the server starts there a busy-period, with pgf \( l_i(\cdot) \).
\( q_i(\cdot) \) - joint pmf for \( X_i \) and \( X_2 \) at queue \( i \) busy-period initiation.
\( Q_i(\cdot) \) - generating function for \( q_i(\cdot) \).
\( B_i \) - duration of a busy-period for line \( i \), initiated by one customer, with LST \( B_i(\cdot) \).
\( V_i \) - interarrival time for line \( i \), with LST \( \nu_i(s) = \lambda_i/(\lambda_i+s) \).
\( \nu_i \) - duration of a busy-period for line \( i \), initiated by one customer, with LST \( \nu_i(s) = \lambda_i/(\lambda_i+s) \).
\( H_i = B_i + \nu_i \).
\( \bar{W}_i \) - Sojourn time for a customer in line \( i \), with LST \( \bar{W}_i(\cdot) \).

**2.** The quantities of interest in our analysis are the customary ones: queue lengths as viewed by departing or arriving customers (or long-time average distributions), waiting times and switching frequency. The latter is of interest when the control object includes a charge for the switching action.

**3.** The analysis of this set-up follows the traditional route of observing the system at suitable epochs; when the states form a Markov chain. The natural epochs are departures, from both queues. It is not hard to write transition probabilities for this chain, but the resultant matrix is quite unwieldy; since the above thresholds induce a complicated interaction between the queues-length processes. Thus we elected a somewhat different approach; the interaction is captured by a "coarser" chain, i.e., one with rarer transitions (essentially at ends of busy-periods). This \( t \)-chain is defined and analyzed in Section C. It carries the brunt of the work. Section B shows the marginal distributions of the queue-lengths, which are naturally quite close to those of \( M/G/1 \) systems, and in Sections D and E we show how the above two may be tied together, via the joint distribution at busy-period beginnings. Numerical computations were important in this work. In the Appendix we describe the procedures we used in conjunction with this analysis.

**4.** The analysis bears some fleeting resemblance to those in [Eisenberg, 1971, 1972]. There, however, the itinerary of the server was entirely deterministic; in our notation it would correspond to setting both thresholds at zero: the server switches as soon as it empties a queue. It will appear that when the switching depends on the state of the queues, it is much harder to obtain the desired results. Indeed, we only look at two queues (see also Section F), while in [Eisenberg, 1972] an arbitrary number is easily handled. The results of Sections B and C have been reported in part in [Hofri, 1985].

**B. The marginal distributions**

1. Consider the system as defined above, in A, and view queue \( i \) in isolation. Proceeding precisely as in the classroom treatment of an \( M/G/1 \) queue we observe the chain formed by \( X_i \) at departure epochs from that queue.

Technion - Computer Science Department - Technical Report CS0376 - 1985
Assuming for the moment the stability of this chain (we shall discuss it in Section C), we may write for the steady-state probabilities:

\[ \text{Prob}(X_t \text{ at departure has length } x) = \tau_t(x) \]

\[ = \sum_{j=1}^{x+1} \tau_t(j) \text{Prob}(S_u = x-j+1) + \tau_t(0) \sum_{j=1}^{x+1} \text{Prob}(I_t = j) \text{Prob}(S_u = x-j + 1) \quad x \geq 0 \]  

and the pgf is immediate

\[ G_t(x) = \sum_{z \geq 0} \tau_t(z) z^x = \frac{1}{x} S_u(z)[G_t(z)-\tau_t(0)] + \tau_t(0) \frac{S_u(z)}{z} L_t(z) \]

2. Note that by setting \( L_t(z) = z \) the familiar expression for \( M/G/1 \) reappears. Setting \( z=1 \) in (2) provides \( \tau_t(0) \) as the reciprocal of the mean number of services performed in a busy-period:

\[ \tau_t(0) = (1-\rho_t) / E(L_t), \quad \rho_t = \lambda_t E(S_i) \]  

3. The usual following properties of \( M/G/1 \) hold here as well:

- The sojourn-time distribution is given in terms of the above \( G_t(\cdot) \):

\[ W_t(s) = G_t(1-s / \lambda_t) \]

- \( G_t(\cdot) \) is also the pgf for the queue length observed by customers arriving to queue \( t \), and hence also the "random time" line length pgf, which is intuitively quaint since \( G_t(\cdot) \) was based on a relatively sparse information.

4. It only remains to determine \( L_t(z) \) to completely specify the marginal distribution of \( X_t \). \( L_t \) however, depends on the evolution of both queues, and thus we have to consider now their interaction.

C. The two-queue interaction

1. Since the interaction between the two queues is rather complicated when viewed at successive departure epochs, we use a more parsimonious description that only encodes the states when the queues actually interact. These epochs we call \( t \)-points, and there are two types of such points:

- end of a busy-period (type b point)
- beginning of switching by an idling server (type c point).

A type c point occurs following the event that a server finishes a busy-period (itself a type b point), observes that the queue at the other position is below the switching threshold, and therefore idles. However, arrivals to that other queue let it reach the threshold before there is an arrival to the queue where the idling server is stationed,
and it starts to switch over. A state of the t-chain will be denoted by \((i, x, b, c)\) with \(i \in \{1,2\}\) the position of the server, and \(x\) the length of the other queue \((X_t = 0\) then). Note that for a type \(c\) point only \(x = m_z\) is possible, with \(i\) being the alternate of \(i\). The transition probabilities for this chain may be read off the following equations, which we write for clarity for the specific value of \(i = 1\):

\[
p(1, x, b) = \sum_{k=0}^{\infty} p(1, k, b) P(U_{12} = x + k) + p(2, m_1, c) P(A_{12} + B_{12}^{m_1+1} = x) \]

\[+ \sum_{j=m_1}^{\infty} p(2, j, b) P(A_{12} + B_{12}^{j+1} = x), \quad 0 \leq x < m_2 \]

where a superscript \(r\) denotes \(r\)-fold convolution, and

\[
p(1, x, b) = \sum_{i_1=0}^{m_z-1} p(1, i_1, b) \sum_{i_2=0}^{m_z-1} P(V_{12} = i_2) P(B_{12} = x - i_1 - i_2) \]

\[+ p(2, m_1, c) P(A_{12} + B_{12}^{m_1+1} = x) + \sum_{j=m_1}^{\infty} p(2, j, b) P(A_{12} + B_{12}^{j+1} = x), \quad x \geq m_2 \]

and lastly

\[
p(1, x, c) = \delta(x - m_2) \sum_{j=0}^{m_2-1} p(1, j, b) P(V_{12} = m_2 - j = 0). \]

To simplify the notation we shall use \(p_i(x)\) instead of \(p(i, x, b)\).

Note that the \(A_{11}\) and \(A_{12}\) in equations (4) or (5) are dependent through the same realization of \(A_{11}\). The corresponding equations and transition probabilities for \(i = 2\) are obtained from the above by completely symmetrizing over the indices 1 and 2.

2. Define

\[g_i(z) = \sum_{x=0}^{x} p_i(x)z^x\]

and observe that

\[P(V_{12} = k) = \frac{\lambda_1 x^k}{(\lambda_1 + \lambda_2)^{k+1}}, \quad P(V_{21} = 0) = \left(\frac{\lambda_2}{\lambda_1 + \lambda_2}\right)^{m_2} \]

Substituting (4), (6) and (8) into (7) and rearranging we obtain

\[g_i(z) = A_i(\lambda_1(1-B_{12}(z)) + \lambda_2(i-z)) [p(2, m_1, c)B_{12}^{m_1}(z)] \]

\[+ g_2(B_{12}(z)) - \sum_{j=0}^{m_1-1} p_2(j)B_{12}^{j}(z)] + \frac{\lambda_1 B_{12}(z)}{\lambda_1 + \lambda_2} \sum_{i=0}^{m_1-1} z^i \frac{p_1(i)}{1 - \left(\frac{\lambda_2}{\lambda_1 + \lambda_2}\right)^{i-m_1}} \]

and the dual equation:

\[g_2(z) = A_2(\lambda_2(1-B_{21}(z)) + \lambda_1(1-z)) [p(1, m_2, c)B_{21}^{m_2}(z)] \]

\[+ g_1(B_{21}(z)) - \sum_{j=0}^{m_2-1} p_1(j)B_{21}^{j}(z)] + \frac{\lambda_2 B_{21}(z)}{\lambda_2 + \lambda_1} \sum_{i=0}^{m_2-1} z^i \frac{p_2(i)}{1 - \left(\frac{\lambda_1}{\lambda_2 + \lambda_1}\right)^{i-m_2}} \]

Denote by \(h_i(z)\) the low-order part of \(g_i(z)\). Specifically

\[h_i(z) = \sum_{x=0}^{m_1-1} p_i(x)z^x\]

\[\text{This chain is "minimal" in the sense that for low enough arrival rates it has no transient states.}\]
One server and threshold switching...

\[ h_i(z) = \sum_{x=0}^{m_i-1} p_i(z)z^x \]

Note that substituting \( z = 1 \) in (9.1) (or (9.2)) yields

\[ p(1,m_2,c) + g_1(1) = h_1(1) = p(2,m_1,c) + g_2(1) = h_2(1) \tag{10} \]

which merely states that under steady-state conditions, the frequency of switching from queue 1 to queue 2 is equal to the frequency of switching in the reverse direction. This obvious observation will provide us with an equation for the "boundary" probabilities \( p_i(z), z < m_i \). Another equation is supplied by the natural normalization

\[ p(1,m_2,c) + p(2,m_1,c) + g_1(1) + g_2(1) = 1 \tag{11} \]

3. We turn now to the solution of equations (9). To keep the size of the expressions manageable we shall introduce a raft of functions, as we go along. Define

\[ \alpha_i(z) = \lambda_i(1-B_i(z)) + \lambda_e(1-z) \]

Substituting (6) into (9.1) we obtain:

\[ g_1(z) = \alpha_i(z)g_2(B_{1i}(z)) + \sum_{i=0}^{m_i-1} p_i(i)\beta_{1i}(z) + \sum_{i=0}^{m_i-1} p_i(i)\gamma_{1i}(z) \tag{12.1} \]

with

\[ \beta_{1i}(z) = \frac{\lambda_1 z B_{1i}(z)}{\lambda_1 + \lambda_2 - \lambda_2 z} \left[ 1 - \left( \frac{\lambda_2 z}{\lambda_1 + \lambda_2} \right)^{m_i} \right] \]

\[ \gamma_{1i}(z) = \alpha_i(z) \left[ B_{1i}^{m_i}(z) \left( \frac{\lambda_1}{\lambda_1 + \lambda_2} \right)^{m_i} - B_{1i}(z) \right] \tag{13} \]

A dual equation is likewise obtained from (9.2):

\[ g_2(z) = \alpha_2(z)g_1(B_{2i}(z)) + \sum_{i=0}^{m_2-1} p_2(i)\beta_{2i}(z) + \sum_{i=0}^{m_2-1} p_2(i)\gamma_{2i}(z) \tag{12.2} \]

with the \( \beta_{2i}(z) \) and \( \gamma_{2i}(z) \) being obtained from their correspondents in equation (13) by the exchange 1 \(<->2\).

Using (12.2) to evaluate \( g_2(B_{1i}(z)) \) we get from (12.1) an equation involving \( g_1(\cdot) \) only, with \( m_1 + m_2 \) "boundary" state probabilities:

\[ g_1(z) = \delta_1(z)\gamma_{1i}(z) + \sum_{i=0}^{m_1-1} p_1(i)\beta_{1i}(z) + \sum_{i=0}^{m_1-1} p_2(i)\gamma_{1i}(z) \tag{14} \]

with
One server and threshold switching...

\[ \delta_1(x) = \alpha_1(x) \beta_2(B_1(x)) \]
\[ \eta_1(x) = \beta_1(B_1(x)) \]
\[ \varphi_{1,4}(x) = \beta_{1,4}(x) + \alpha_1(x) \gamma_{2,1}(B_1(x)) \]
\[ \zeta_{1,4}(x) = \gamma_{1,4}(x) + \alpha_1(x) \beta_{2,1}(B_1(x)) \]

Clearly a dual equation for \( g_2(\cdot) \) is produced when one symmetrizes over 1 and 2 in \( g_1(\cdot) \).

So far we have proceeded with blithe disregard to the question of stability or stationarity of the two queue-system, expecting that analysis under the assumption of ergodicity will produce equations that, inter alia, display, or at least determine the conditions under which stationarity will in fact prevail.

We have not been disappointed. Considering equation (14) one may ask under what conditions does this equation define a function \( g_1(z) \), which is analytic for \(|z| < 1\), and continuous for \(|z| \leq 1\), as a proper pgf. should be? Since when \( g_1(\cdot) \) is analytic we may use (14) for recursive substitution, one may ask instead what conditions are required to guarantee that such a procedure, continued ad infinitum, shall yield an analytic result. To this question we have an answer. Let \( \eta^{(j)}(z) \) denote the \( j \)-th functional iterate of \( \eta_1(z) \), i.e.

\[ \eta^{(0)}(z) = z \]
\[ \eta^{(j)}(z) = \eta_1(\eta^{(j-1)}(z)) \]

Carrying out the recursive substitution procedure we formally obtain

\[ g_1(z) = \lim_{j \to \infty} \eta^{(j)}(z) \delta_1(\eta^{(j)}(z)) \]
\[ + \sum_{i=0}^{m_1-1} \sum_{j=0}^{m_2-1} \frac{\beta_{1,4}(z)}{z} \sum_{i=0}^{m_1-1} \frac{\beta_{2,1}(z)}{z} \]

Now the conditions can be stated explicitly: Equation (14) is viable iff the following hold:

a) \( \eta^{(j)}(z) \), for \(|z| \leq 1\) and \( j \to \infty \) converges to a point where \( g_1(z) \) is defined. Call this point \( a \).

Then we also need:

b) \( \delta_1(a) \leq 1 \);

c) \( \delta_{1,4}(a) \cdot \zeta_{1,4}(a) < \infty \), and if \( \delta_1(a) = 1 \), they must both vanish there.

d) Finally, the limits must be approached fast enough to assure convergence to a continuous function for \(|z| \leq 1\). The last requirement will turn out to be the key one.

In [Hofri, 1985] we show that the above requirements indeed hold when \( \eta^{(1)}(1) \) (or \( \eta^{(1)}(1) \)) is less than 1, which is equivalent to \( \rho = \lambda_1 \beta(S_1) + \lambda_2 \beta(S_2) < 1 \). Then \( a = 1 \) and the convergence is geometrical.

This condition has a simple intuitive meaning. The definition in equation (15) of \( \eta_1(z) \) can be read as
\[ \eta_1(z) = E(z^{B_1^M}) \]

This translates to the verbal statement that \( \eta_1(z) \) is the pgf of the number of arrivals to queue 2, during a sequence of busy-periods of queue 1 with multiplicity equal to a random variable with the distribution of the number of arrivals to queue 1 during a (simple) busy-period of queue 2. Clearly the expected value derived from this pgf must be less than one to assure stability. Note that the switching times play no role in the stability condition, which is natural for the case of exhaustive service. We may therefore now state

**Theorem:** When \( \rho := \lambda_1 E(S_1) + \lambda_2 E(S_2) < 1 \), the \( t \)-chain reaches a steady state with the pgf

\[
g_1(z) = g_1(1) \prod_{j=0}^{m_k-1} \delta_1(\eta^j_1(z)) + \sum_{i=0}^{m_k-1} p_1(i) \sum_{j=0}^{m_k-1} \delta_1(\eta^j_1(z)) \prod_{k=0}^{j-1} \delta_1(\eta^k_1(z))
\]

The dual \( g_2(z) \) is obtained by symmetrizing over the indices 1 and 2.

5. The determination of \( g_k(1) \) and \( p_k(j) \), \( 0 \leq j < m_k \) is in principle quite straightforward: equations (10) and (11), equation (18) (respectively, its dual), its first-\( m_2-1 \) derivatives (respectively, \( m_1-1 \)), all the derivatives evaluated at \( z=0 \), provide the required number of linear equations in those unknowns. The difficulty resides in differentiating anything as unattractive as the right-hand-side of equation (18) several times.

Two obstacles had to be overcome, beyond the sheer complexity of the expressions:

a) The expressions contain infinite sums and products. As mentioned above, these converge exponentially. In practice we iterated \( \eta_1(z) \), starting at \( z=0 \) until it approached \( z=1 \) to within \( z=10^{-6} \), and used the order of the iteration thus determined as the upper limit to all the (infinite) sums and products. It was found that changing \( \log_2 \) in the range -7 to -13 had no effect on the first six significant decimal digits in all the numbers of interest. The required number of iterations rarely exceeded 20, and never reached 100.

b) The basic functions, \( \eta_i(z) \), are available in terms of the LSTs of the busy-period length distributions. These are determined by the equations

\[
B_i(u) = S_i(u + \lambda_i(1 - B_i(u)))
\]

In the general case there is no closed-form solution to equation (19). A formal solution is given by

\[
B_i(u) = \sum_{j=1}^{\infty} \frac{(-\lambda)^{j-1}}{j!} \left( S_i(u + \lambda_i) \right)^j
\]

where a bracketed superscript \( [j] \) denotes the \( j \)-th order differentiation. From the numerical point of view our experience with equation (20) was disappointing. Although the series converges nicely in principle, the higher order terms — at least when \( S(u) \) is a rational or an exponential function — are made of very large and extremely small...
factors, and overflow or underflow usually intervened before satisfactory convergence was achieved. This can probably be overcome by careful programming, specific for each type of $S(u)$, but it turned out that the following iterative scheme provided a stable and as prompt convergence. Shedding the subscript $t$ we write:

$$B_0(u) = S(u + \lambda)$$
$$B_{j+1}(u) = S(u + \lambda(1 - B_j(u)))$$

(21)

Note that (for fixed $u$) $B_j(u)$ is monotonically increasing in $j$.

6. To obtain derivatives of $B(u)$ and the other functions that abound here one uses the Faa-di-Bruno formula:

$$J^k [g(x)]^{(k)} = \sum_{j=1}^{k} f_{j,n} [g(x)]^j \prod_{i=1}^{j} [g(x)]^{k_i}$$

(22)

with the coefficients $f_{j,n}$ given by

$$f_{j,n} = \frac{r!}{\prod_{i=1}^{j} (k_i)!}$$

and the sum over $k$ ranges over integer-valued $r$-dimensional vectors that satisfy

$$\sum_{i=1}^{j} k_i = j, \sum_{i=1}^{j} ik_i = r, \quad k_i \geq 0.$$ 

Applying (22) to equation (19) iteratively, and noting that $B^{(r)}(u)$ appears just once in the right-hand-side (with coefficient 1), the derivatives of $B(u)$ to any order were easy to compute. (See the Appendix, part 4.5). Equation (22) was used extensively throughout the computation. The entire set of the required vectors $\mathbf{k}$ and coefficients $F$ was prepared in a first phase; to order of $\max(m_1, m_2) - 1$.

7. For the special case of exponentially distributed service times $B(u)$ does have a closed-form representation

$$B_i(u) = (\mu + \lambda_4 + u - \sqrt{(\mu_4 + \lambda_4 + u)^2 - 4\lambda_4 \mu_4})/2\lambda_4$$

(23)

and in this case the iteration (21) is not required. Indeed, most of the numerical experiments were conducted for this case, as there is no apparent reason for the qualitative behavior of the system to depend in any essential way on the shape of the distributions of the service requirements.

D. Use of the t-chain

1. We are now in position to complete the calculation started in Section B. We need the $l_t(x)$ - the pgf of $X_t$ at the beginning of an $i$-queue busy period. Since we shall later need the joint queue lengths distribution at these epochs, we shall proceed directly to evaluate it.
Define
\[ q_i(z, y) = \Pr(X_1 = z, X_2 = y \text{ at the beginning of an } i\text{-busy period}). \]

Computing \( q_i \) is simple if we condition on the \( t \)-point this busy-period follows. We have to consider \( t \)-points with states \((i', m_i', c'), (i', j, b) \) \( i' \geq m_i \) and \((i, j, b) \) \( i < m_i \). Thus, simplifying notation by taking for the moment \( i = 1 \):

\[
q_1(z, y) = \frac{1}{\tau_1} \{ \Pr(z - m_1 - 1 - \text{arrivals, } y \text{ -arrivals during } A_1) p(2, m_1, c) \\
+ \sum_{j=m_1}^{m_2-1} \Pr(z - j - 1 - \text{arrivals, } y \text{ -arrivals during } A_1) p_0(j) \\
+ \delta_{z,y} \sum_{j=0}^{m_2-1} \Pr(y - j \text{ -arrivals during } V_i) p_1(j) \}
\]

The symbol \( \tau_i \) stands for the fraction of \( t \)-points that produce an \( i \)-busy-period before another \( t \)-point transpires. Routine manipulations provide

\[
Q_1(z_1, z_2) = \sum_{x=1}^{z_1} \sum_{y=0}^{z_2} q_i(x, y) 2^x 2^y
\]

\[
= \frac{1}{\tau_1} \{ \lambda_1 [\lambda_1 (1-z_1) + \lambda_2 (1-z_2)] [z_1 2^m p(2, m_1, c) + g_2(z_1) - h_2(z_1)] \\
+ \lambda_1 z_1 \lambda_2 [h_2(z_1) - z_2 2^m p(1, m_2, c)] \} 
\]

(25)

Where, in accordance with the above calculation,

\[
\tau_1 = p(2, m_1, c) + g_2(1) - h_2(1) + h_1(1) - p(1, m_2, c).
\]

2. When \( m_1 = 0 \), the state \((2, 0, b)\) does not necessarily lead to a 1-busy-period, but only with probability \( 1 - A_{11}(0) \), thus

\[
Q_1(z_1, z_2) = \frac{1}{\tau_1} \{ \lambda_1 [\lambda_1 (1-z_1) + \lambda_2 (1-z_2)] g_2(z_1) - p_0(0) A_1 [\lambda_1 + \lambda_2 (1-z_2)] \\
+ \frac{\lambda_1 z_1}{\lambda_1 + \lambda_2 (1-z_2)} [h_2(z_1) - z_2 2^m p(1, m_2, c)] \}.
\]

(26)

3. When \( m_1 = 0 \), the state \((2, 0, b)\) does not necessarily lead to a 1-busy-period, but only with probability \( 1 - A_{11}(0) \), thus

\[
Q_1(z_1, z_2) = \frac{1}{\tau_1} \{ A_1 [\lambda_1 (z_1) + \lambda_2 (1-z_2)] g_2(z_1) - p_0(0) A_1 [\lambda_1 + \lambda_2 (1-z_2)] \\
+ \frac{\lambda_1 z_1}{\lambda_1 + \lambda_2 (1-z_2)} [h_2(z_1) - z_2 2^m p(1, m_2, c)] \}.
\]

(27)

The function \( Q_0(z_1, z_2) \) is obtained by the obvious symmetrization in equations (25)-(27).

3. We now get \( I_i(z) \) by substituting \( i \), \( Q_i(z_1, z_2) \) \( z_1 = z, z_2 = 1 \), yielding

\[
I_i(z) = \frac{1}{\tau_i} \{ z (h_i(1) - p(t, m_i, c)) + A_1 (z) [p(t, m_i, c) z^{m_i} + g_2(z) - h_2(z)] \}
\]

(28)

If \( m_i = 0 \) then instead of equation (28) we obtain:

\[
I_i(z) = \frac{1}{\tau_i} \{ z (h_i(1) - p(t, m_i, c)) + A_1 (z) g_2(z) - p_0(0) A_0(0) \}
\]

(29)

Note that when \( m_i = 0 \), \( p_0(0) \) is not one of the 'boundary' probabilities computed through.
the procedure outlined in C.5, but has to be evaluated from equation (18).

4. As was mentioned in the introduction, the purpose of the analysis was to evaluate this policy under a cost structure where the system incurs constant-rate holding charges for any customer in the queues or being serviced, as well as a one-time charge for every switch. The first component averages out $c_1 E(X_1) + c_2 E(X_2)$ which is available from the first derivative of $G(z)$, as given in equation (2) at $z=1$. To adapt the second component to this scale of average cost per time unit we need the mean frequency of switching. This can be approached in a number of ways, the simplest being probably as follows:

Consider the $t$-chain transitions. Let $\pi_i$ be the probability that the state at a transition point of the chain, selected at random, leads to an immediate switch to queue $i$ (every switch starts at a transition point of this chain). Such a switch occurs if the selected point found the chain in the states $(i,m_i,c)$ or $(i',x,b)$ with $x=m_i$. Otherwise one or more transitions will be realized before such a switch takes place. Thus

$$\pi_i = p (i,m_i,c) + g_i (t) (1 - h_i (t))$$

To convert this to frequency we need the mean time between transition points of the chain, $L$. This is straightforward to compute by counting the possible transitions: at queue $i$, from a $c$-point, the expected time to the next point (necessarily a $b$-point in queue $i'$) is $E(A_i) + [m_i + \lambda_i E(B_i)] E(B_i)$; similar is the case at a $b$-point when $X_i > m_i$, with $m_i$ in the above expression replaced by the value of $X_i$; at a $b$-point where the server idles, we have to find the mean time to the first of the two possible transitions: to a $b$-point in the same queue, or to a $c$-point there. One obtains, using the symbol $\alpha_i$ for $\lambda_i / (\lambda_1 + \lambda_2)$,

$$L = \sum_{i=1}^{2} \{ (g_i (t) - h_i (t)) E(A_i) [1 + \lambda_i E(B_i)] + [g_i (t) - h_i (t)] E(B_i) \}$$

$$+ \sum_{i=1}^{m_i - 1} \sum_{j=0}^{m_i - 1} p_i (j) \left[ \alpha_i (m_i - j) E(A_i) + \frac{m_i - j}{\lambda_1 + \lambda_2} + (m_i + \lambda_i E(B_i)) E(B_i) \right]$$

$$+ E(B_i) \left[ 1 - \alpha_i (m_i - j) \right] + \frac{1}{\lambda_i} \left[ 1 - (m_i - j + 1) \alpha_i (m_i - j + 1) + (m_i - j) \alpha_i (m_i - j + 1) \right]$$

Using equation (10), the required frequency is clearly given by $2 \pi_i / L$, for which all the necessary quantities are given, parameters or have been computed.

E. Refining the $t$-chain

1. The purpose of this section is to refine the joint distribution of $X_1$ and $X_2$, available so far at $t$-points, to arbitrary departure epochs. For this we need two preliminary results. One is the lengths of the queues at the beginning of an $i$-busy-period, and the second concerns the evolution of such a busy-period. The first one is available from D.1-2, equations (24) to (27), and we shall turn now to the second one.
One server and threshold switching...

2. We are again considering queue \( i \) in isolation. Define\(^e\)

\[
a_i(t,x;j,n) = \text{Prob}(\text{the } n\text{-th departure in a } j\text{-fold busy-period leaves } x \text{ in queue and occurs at time } t \text{ since the busy-period started}) \quad (32)
\]

\( t > 0, x \geq 0, j > 0, n > 0. \)

The definition provides the immediate recursion

\[
a_i(t,x;j,n) = \int_{s=0}^{x} \sum_{i=1}^{n} a_i(t-s,x;j,n-1) f_i(s) e^{-\lambda_i s (\lambda_i s)^{-1+1}} ds \quad j > 0, n > 1, x \geq 0.
\]

\[
a_i(t,x;j,1) = \int_{s=0}^{x} f_i(s) e^{-\lambda_i s (\lambda_i s)^{-1+1}} ds \quad x \geq j-1
\]

Define successively the transforms

\[
\varphi_i(t,x;j,n) = \sum_{z \geq 0} a_i(t,x;j,n) z^x \quad (34)
\]

\[
\psi_i(u,x;j,n) = \int_{t=0}^{x} f_i(t) e^{-\lambda_i t} dt \quad (35)
\]

\[
\chi_i(u,x;j,n) = \sum_{n \geq 1} \psi_i(u,x;j,n) u^n \quad (36)
\]

Substituting (33) into (34) yields

\[
\varphi_i(t,x;j,n) = \frac{1}{z} \int_{s=0}^{x} f_i(s)[\varphi_i(t-s,x;j,n-1) - \varphi_i(t-s,0;j,n-1)] e^{-\lambda_i (1-z)} ds \quad n > 1
\]

and

\[
\varphi_i(t,x;j,1) = z^{j-1} \int_{s=0}^{x} f_i(s) e^{-\lambda_i s (1-z)} ds
\]

Definition (35) provides now

\[
\psi_i(u,x;j,n) = \frac{S_i(u+\lambda_i(1-z))}{z} \left[ \psi_i(u,x;j,n-1) - \psi_i(u,0;j,n-1) \right] \quad n > 1
\]

\[
\psi_i(u,x;j,1) = z^{j-1} S_i(u+\lambda_i(1-z)).
\]

Finally, definition (36) leads to

\[
\chi_i(u,x;j,v) = u z^{j-1} S_i(u+\lambda_i(1-z)) + \frac{u}{z} S_i(u+\lambda_i(1-z)) [\chi_i(u,v;0;j,v) - \chi_i(u,0;j,v)]
\]

\[
\quad = u S_i(u+\lambda_i(1-z)) \left[ \frac{z^{j-1} - \chi_i(u,0;j,v)}{z-v S_i(u+\lambda_i(1-z))} \right]
\]

with \( \chi_i(u,0;j,v) = \chi_i(u,v;0;j,v) \) as the latter is simply the joint LST-pdf of the duration of a \( j \)-fold busy-period and the number of customers served in it. We need not evaluate here this function, as it will turn out we can make do with \( \chi_i(u,0;1,1) \) which equals \( E_i(u) \); for reference we record that similarly to (19) it is the solution of a functional equation

---

\(^e\) It is possible to define consistently \( a(0,x;j,n) = \delta_{0n} \delta_{xj} \), but there is no obvious advantage.
One server and threshold switching...

\[ \chi_i(u,0;1,v) = iS_i(u + \lambda_i - \lambda_i \chi_i(u,0;1,v)) \]  

(40)

3. Now we can tackle the joint distribution of \( X_i \) at arbitrary departures. Define, for such an epoch

\[ \tau(i,x,y) = \text{Prob}(\text{A random departure is from queue } i, \text{ with } X_i = x, X_2 = y) \]

with obvious notation:

\[ \tau'(i,x,y) = \tau(x,y|i)p(i) \]

(41)

where \( p(i) \) is the fraction of departures from queue \( i \), and equals \( \lambda_i / (\lambda_1 + \lambda_2) \).

Again, if we simplify the notation if we temporarily fix \( i = 1 \). To compute

\[ \tau_1(x,y) = \tau'(x,y|1) \]

we use the natural conditioning:

\[ \tau_1(x,y) = \sum_{k \geq 1} \sum_{j \geq 0} q_1(k,j) \frac{k}{E(1)} \text{Prob}(\text{a departure during a } k \text{-fold } 1 \text{-busy-period leaves there } x, \text{ and } y-j 2\text{-arrivals came since the busy-period started}) \]

(42)

The factor \( k/E(1) \) reflects the probability of a random 1-departure occurring during a \( k \)-fold 1-busy-period. The probability mentioned in (42) can be expressed as

\[ \sum_{n \geq 0} \int_{t \geq 0} e^{-\lambda t} \left( \frac{(\lambda t)^{n-j}}{(y-j)!} \right) da_1(t,x;1,h) \]

Hence

\[ G_1(z_1,z_2) = \sum_{x \geq 0} \sum_{y \geq 0} \tau_1(x,y) x! z_2^y \]

\[ = \sum_{k \geq 1} \sum_{j \geq 0} z_2^j k \frac{k}{E(1)} q_1(k,j) \chi_1(\lambda_2(1-z_2),x_1;1,k) \]

(43)

and using (39)

\[ G_1(z_1,z_2) = \frac{S_1(\lambda_1(1-z_1) + \lambda_2(1-z_2))}{E(1) \left[ S_1(\lambda_1(1-z_1) + \lambda_2(1-z_2)) \right]} \]

\[ \times \left[ z_1 Q_1^1(z_1,z_2) - B_{12}(z_2) Q_1^1(\tilde{B}_{12}(z_2),z_2) \right] \]

(44)

where \( Q_1^1(z_1,z_2) = \frac{\partial}{\partial z_1} Q_1(z_1,z_2) \), and is immediately available from (25) (or (27)).

If in equation (41) one sums over \( i \) the joint distribution of \( x_i \) at a random departure result,

\[ G(z_1,z_2) = \frac{1}{(\lambda_1 + \lambda_2)} \]

(45)

4: An unwelcome difference between the one- and two-line length distributions is that in the latter arrivals do not "see" the same distribution as departures, though they do "see" the same distribution as observed in random time.

Based on the last remark, the computation of the arrival time joint distribution of \( X_i \) can be done by randomizing on the state at the last proceeding departure; while in
principle the computation is straightforward, the expressions obtained are extremely cumbersome and we omit them here. One can hardly imagine it would differ materially from the one provided by \( \hat{C}(z_1, z_2) \), so for all engineering purposes the latter suffices.

F. Discussion

1. The preceding analysis raises some interesting, as well as vexing questions. The first concerns the way we chose to analyze the interaction between the two queues. The method works, but only at the expense of a highly involved and substantial amount of software, in order to obtain the boundary probabilities and get any numbers out of the equations we could write. (Note that the computation times were entirely acceptable - the appended tables are extracted from a much larger set that required less than ten minutes on the minicomputer VAX-780; it is the preparation of the programs that we would have preferred to do without - see the Appendix). Is there any other approach that would be - if less direct - easier to implement? We have not found one and confess to being highly intrigued by such a possibility.

2. The above analysis handles two queues in interaction. What about more? It is easy to see which way our approach generalizes: if the sequence of switches the server performs is predetermined, the above method applies. That limitation implies the following on the part of the server: if it is to serve in the order of stations (\( \cdots, t, t', t'' \), \( \cdots \)), and when it empties queue \( i \) it finds \( X_r<\text{em} \), no switching is done, even if \( X_r=\text{em} \). Admittedly this appears to be a peculiar constraint. Even then, however, while the preceding analysis holds, the task of generating numbers looks even more daunting. Consider that now, for more three queues, the function we need iterate to solve for \( g_1(z) \) has the following representation:

\[
\delta_1(z) = \alpha_1(z)\alpha_2(B_{12}(z))\alpha_3(B_{23}(B_{12}(z))),
\]

and compare with section 4.6 in the Appendix.

Again, we would like to see a different approach to handling the interaction between the queues, that would generalize in a less onerous way.

Abandoning the assumption of deterministic switching sequence (e.g., in the above example letting the server switch directly to \( t'' \)), leads to a different structure that we have not yet examined, but believe to be considerably harder. We remark that a well known problem, of doing exact analysis of the policy SSTF used sometimes to manage computer disks may be cast within such a model.

3. We show some of the results in Tables 1 and 2, giving the expected queue lengths at departure and at busy-period beginnings as well as the switching frequency as a function of the thresholds. It was interesting to observe that it is not always obvious a priori which thresholds would provide the shorter queues, though the overall behavior is quite intuitive. Note in particular that for quite different service distributions (exponential vs. a constant plus a uniformly distributed segments) the numerical
One server and threshold switching

<table>
<thead>
<tr>
<th>$m_1$</th>
<th>$m_2$</th>
<th>$E(I_1)$</th>
<th>$E(I_2)$</th>
<th>$P(X_1=0)$</th>
<th>$P(X_2=0)$</th>
<th>$E[X_1]$</th>
<th>$E[X_2]$</th>
<th>$E[X_1+X_2]$</th>
<th>swfr</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>6.9647</td>
<td>5.9829</td>
<td>0.1029</td>
<td>0.1270</td>
<td>4.0605</td>
<td>3.5112</td>
<td>7.5717</td>
<td>0.0613</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>6.7680</td>
<td>5.9721</td>
<td>0.1034</td>
<td>0.1273</td>
<td>4.0539</td>
<td>3.5063</td>
<td>7.5601</td>
<td>0.0611</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>6.3851</td>
<td>6.0779</td>
<td>0.1096</td>
<td>0.1250</td>
<td>4.0275</td>
<td>3.5007</td>
<td>7.5282</td>
<td>0.0600</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>5.6769</td>
<td>6.3239</td>
<td>0.1233</td>
<td>0.1202</td>
<td>3.9742</td>
<td>3.5231</td>
<td>7.5479</td>
<td>0.0577</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>4.8644</td>
<td>7.3686</td>
<td>0.1439</td>
<td>0.1128</td>
<td>3.9011</td>
<td>3.6125</td>
<td>7.5135</td>
<td>0.0542</td>
</tr>
<tr>
<td>0</td>
<td>5</td>
<td>4.1530</td>
<td>7.3066</td>
<td>0.1686</td>
<td>0.1040</td>
<td>3.8234</td>
<td>3.7909</td>
<td>7.6143</td>
<td>0.0499</td>
</tr>
<tr>
<td>0</td>
<td>6</td>
<td>3.6096</td>
<td>8.0635</td>
<td>0.1939</td>
<td>0.0950</td>
<td>3.7545</td>
<td>4.6561</td>
<td>8.3756</td>
<td>0.0456</td>
</tr>
<tr>
<td>0</td>
<td>7</td>
<td>3.2130</td>
<td>8.7946</td>
<td>0.2179</td>
<td>0.0664</td>
<td>3.7011</td>
<td>4.3918</td>
<td>8.0929</td>
<td>0.0415</td>
</tr>
<tr>
<td>0</td>
<td>8</td>
<td>2.9243</td>
<td>9.6521</td>
<td>0.2394</td>
<td>0.0767</td>
<td>3.6647</td>
<td>4.7972</td>
<td>8.4439</td>
<td>0.0378</td>
</tr>
<tr>
<td>0</td>
<td>9</td>
<td>2.7105</td>
<td>10.5553</td>
<td>0.2583</td>
<td>0.0720</td>
<td>3.6440</td>
<td>5.2026</td>
<td>8.3466</td>
<td>0.0346</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
<td>2.5466</td>
<td>11.4985</td>
<td>0.2746</td>
<td>0.0661</td>
<td>3.6389</td>
<td>5.6502</td>
<td>8.3267</td>
<td>0.0318</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>6.9093</td>
<td>5.8459</td>
<td>0.1013</td>
<td>0.1300</td>
<td>4.0608</td>
<td>3.5015</td>
<td>7.5624</td>
<td>0.0608</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>6.6139</td>
<td>5.8365</td>
<td>0.1027</td>
<td>0.1302</td>
<td>4.0544</td>
<td>3.4970</td>
<td>7.5514</td>
<td>0.0606</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>6.4321</td>
<td>5.9468</td>
<td>0.1088</td>
<td>0.1278</td>
<td>4.0287</td>
<td>3.4923</td>
<td>7.5210</td>
<td>0.0596</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>5.7211</td>
<td>6.1978</td>
<td>0.1224</td>
<td>0.1226</td>
<td>3.9763</td>
<td>3.5155</td>
<td>7.4918</td>
<td>0.0573</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>4.8956</td>
<td>6.1815</td>
<td>0.1430</td>
<td>0.1148</td>
<td>3.9035</td>
<td>3.6054</td>
<td>7.5090</td>
<td>0.0539</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>4.1729</td>
<td>7.1987</td>
<td>0.1676</td>
<td>0.1058</td>
<td>3.8257</td>
<td>3.7945</td>
<td>7.6102</td>
<td>0.0498</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>3.6212</td>
<td>7.9081</td>
<td>0.1933</td>
<td>0.0961</td>
<td>3.7564</td>
<td>4.0506</td>
<td>7.8070</td>
<td>0.0455</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>3.2196</td>
<td>8.7124</td>
<td>0.2174</td>
<td>0.0872</td>
<td>3.7025</td>
<td>4.3873</td>
<td>8.0989</td>
<td>0.0414</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>2.9278</td>
<td>9.5828</td>
<td>0.2391</td>
<td>0.0793</td>
<td>3.6657</td>
<td>4.7757</td>
<td>8.4414</td>
<td>0.0378</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>2.7124</td>
<td>10.4979</td>
<td>0.2581</td>
<td>0.0724</td>
<td>3.6448</td>
<td>5.1998</td>
<td>8.4446</td>
<td>0.0345</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>2.5495</td>
<td>11.4486</td>
<td>0.2746</td>
<td>0.0661</td>
<td>3.6374</td>
<td>5.6481</td>
<td>8.2995</td>
<td>0.0317</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>9.5677</td>
<td>3.4281</td>
<td>0.0732</td>
<td>0.2217</td>
<td>4.9687</td>
<td>3.3244</td>
<td>8.2931</td>
<td>0.0439</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>9.5034</td>
<td>3.4258</td>
<td>0.0737</td>
<td>0.2219</td>
<td>4.9652</td>
<td>3.3224</td>
<td>8.2886</td>
<td>0.0439</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>9.1825</td>
<td>3.4546</td>
<td>0.0762</td>
<td>0.2200</td>
<td>4.9475</td>
<td>3.3237</td>
<td>8.2713</td>
<td>0.0437</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>8.4098</td>
<td>3.5355</td>
<td>0.0832</td>
<td>0.2150</td>
<td>4.9006</td>
<td>3.3936</td>
<td>8.2402</td>
<td>0.0431</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>7.2529</td>
<td>4.0690</td>
<td>0.0995</td>
<td>0.2055</td>
<td>4.8145</td>
<td>3.3954</td>
<td>8.2129</td>
<td>0.0421</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>6.0256</td>
<td>3.9648</td>
<td>0.1162</td>
<td>0.1917</td>
<td>4.6921</td>
<td>3.5299</td>
<td>8.2218</td>
<td>0.0406</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>4.9814</td>
<td>4.3488</td>
<td>0.1405</td>
<td>0.1748</td>
<td>4.5475</td>
<td>3.7506</td>
<td>8.2991</td>
<td>0.0386</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>4.1904</td>
<td>4.8537</td>
<td>0.1670</td>
<td>0.1566</td>
<td>4.3993</td>
<td>4.0592</td>
<td>8.4585</td>
<td>0.0364</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>3.6184</td>
<td>5.4768</td>
<td>0.1935</td>
<td>0.1388</td>
<td>4.2623</td>
<td>4.4408</td>
<td>8.7030</td>
<td>0.0341</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>3.2076</td>
<td>6.2118</td>
<td>0.2182</td>
<td>0.1223</td>
<td>4.1453</td>
<td>4.8756</td>
<td>9.0210</td>
<td>0.0318</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>2.9094</td>
<td>7.0502</td>
<td>0.2406</td>
<td>0.1076</td>
<td>4.0514</td>
<td>5.3465</td>
<td>9.3980</td>
<td>0.0298</td>
</tr>
</tbody>
</table>

Table I: Expected values for constant switching times

$A_1 = 10, A_2 = 5, \lambda_1 = 0.3, \lambda_2 = 0.24$

and exponential services $\mu_1 = \mu_2 = 1.0$. 
One server and threshold switching ...

<table>
<thead>
<tr>
<th>( m )</th>
<th>( n )</th>
<th>( E(I_1) )</th>
<th>( E(I_2) )</th>
<th>( P(X_1 = 0) )</th>
<th>( P(X_2 = 0) )</th>
<th>( E[X_1] )</th>
<th>( E[X_2] )</th>
<th>( E[X_1 + X_2] )</th>
<th>( \text{swfr} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0,0</td>
<td>0,0</td>
<td>6.8862</td>
<td>5.9742</td>
<td>0.1020</td>
<td>0.1272</td>
<td>3.9115</td>
<td>3.3749</td>
<td>7.2865</td>
<td>0.0613</td>
</tr>
<tr>
<td>0,1</td>
<td>0,1</td>
<td>6.7934</td>
<td>5.9658</td>
<td>0.1031</td>
<td>0.1274</td>
<td>3.9056</td>
<td>3.3713</td>
<td>7.2779</td>
<td>0.0611</td>
</tr>
<tr>
<td>0,2</td>
<td>0,2</td>
<td>6.4751</td>
<td>6.0307</td>
<td>0.1281</td>
<td>0.1256</td>
<td>3.8653</td>
<td>3.3689</td>
<td>7.2541</td>
<td>0.0603</td>
</tr>
<tr>
<td>0,3</td>
<td>0,3</td>
<td>6.3203</td>
<td>6.2853</td>
<td>0.1200</td>
<td>0.1214</td>
<td>3.8389</td>
<td>3.3880</td>
<td>7.2250</td>
<td>0.0583</td>
</tr>
<tr>
<td>0,4</td>
<td>0,4</td>
<td>5.9273</td>
<td>6.6379</td>
<td>0.1392</td>
<td>0.1145</td>
<td>3.7705</td>
<td>3.4686</td>
<td>7.2391</td>
<td>0.0550</td>
</tr>
<tr>
<td>0,5</td>
<td>0,5</td>
<td>5.4452</td>
<td>7.1764</td>
<td>0.1634</td>
<td>0.1059</td>
<td>3.7942</td>
<td>3.6414</td>
<td>7.4356</td>
<td>0.0508</td>
</tr>
<tr>
<td>0,6</td>
<td>0,6</td>
<td>5.0254</td>
<td>7.6588</td>
<td>0.1891</td>
<td>0.0967</td>
<td>3.8240</td>
<td>3.9064</td>
<td>7.7303</td>
<td>0.0464</td>
</tr>
<tr>
<td>0,7</td>
<td>0,7</td>
<td>4.5954</td>
<td>8.6452</td>
<td>0.2137</td>
<td>0.0879</td>
<td>3.8563</td>
<td>4.2475</td>
<td>8.1015</td>
<td>0.0422</td>
</tr>
<tr>
<td>0,8</td>
<td>0,8</td>
<td>4.1957</td>
<td>9.5066</td>
<td>0.2380</td>
<td>0.0799</td>
<td>3.8928</td>
<td>4.6440</td>
<td>8.5368</td>
<td>0.0394</td>
</tr>
<tr>
<td>0,9</td>
<td>0,9</td>
<td>3.7933</td>
<td>10.4184</td>
<td>0.2556</td>
<td>0.0729</td>
<td>3.9376</td>
<td>5.0778</td>
<td>8.5504</td>
<td>0.0350</td>
</tr>
<tr>
<td>1,0</td>
<td>1,0</td>
<td>3.5677</td>
<td>11.3583</td>
<td>0.2726</td>
<td>0.0669</td>
<td>3.4994</td>
<td>5.3595</td>
<td>8.0592</td>
<td>0.0321</td>
</tr>
</tbody>
</table>

Table II: Expected values for constant switching times \( A_1 = 10, A_2 = 5, \)
\( A_1 = 0.3, A_2 = 0.24 \) and services are \( S = \sigma + U(0,H) \),
\( C_1 = 0.7, C_2 = 0.25, B_1 = 0.6, B_2 = 1.5 \)

differences in the performance measures are small, while the displayed tendencies are the same, when just the means of the service requirements are equal.

References

Appendix

1. Since the design and checking out of the numerical procedures required to implement the results reported here are not entirely trivial - at least in part - we deemed it worthwhile to describe the latter in some detail. The numerical work involved two steps:

a) Computing the boundary probabilities \( p_i(j) \) and the \( g_i(1) \), by setting up the \( m_1 + m_2 + 2 \) linear equations described in C.5 and solving them.

b) Evaluating the first moments of \( X_i \), \( I_i \), the empty queue probabilities and the switching frequency.

Part a) above is by far the more involved, and nearly all of the following is concerned with it. In part b), we only computed first moments, but the additional effort required to obtain higher ones is marginal.

2. A certain amount of reprogramming is necessary when one changes the type of the underlying service requirement distributions (not just its parameters), or that of the switching times. With respect to the latter we chose to implement the program for constant times; the only places that it matters is in part 4.1, end of 4.6 below, and in the functions \( a(x) \) that compute the LST values, \( A_i(x) \). The dependence on the service times is more complex.

The main program was prepared to depend on those latter distributions to a very limited extent: The program starts by reading input parameters and computing the first two moments of the service times, the required parameters, and the moments clearly depend on the type of the distribution. In addition to this, the service times are manifest through two routines (for each queue), that return values of the LST and its derivatives. The names of these routines are recognized in the permanent procedures as serlstd and seflstd (for queue \( t \)), respectively.

3. Two "combinatorial" functions were prepared: One \( (fleib) \) implements the Leibnitz rule of differentiation: given an integer \( r \) and two vectors \( a \) and \( b \), where \( a(t) \) \( (b(t)) \) are the values of the derivatives of order \( i \) of the function \( f \) \( (g(t)) \), at some implicit point \( u \), it returns \( [f(x)g(x)]_u = \). This function requires one initial entry per run to set up an array of binomial coefficients (the call is to a sister subroutine \( fleib \)).

The second function \( (fsum) \) implements equation (22). It also acts on an integer \( r \) and two vectors, one containing the values \( f[a]^r(g(u)) \) and the second contains \( g[a]^r(u) \). 

Both \( fleib \) and \( fsum \) work along vectors, However one can also transfer slices of arrays, provided they are so stored that the required derivatives are contiguous in storage. This possibility is rather important in terms of the running time of the program.
4. The computing algorithm: (the description also carries the names of the variables used in the actual program. The names have no semantic value, resulting from the several phases the program went through. A listing is available from the author).

1) Read input:

- \( mx \) (allows computing with thresholds up to \( mx + 1 \)).
- \( \lambda \) values for both queues.

Service parameters, according to the type of the distributions, these reading instructions may have to be modified.

Switching times. The programs we prepared assume constant switching times. These instructions would need be changed if a different distribution was desired.

Threshold values.

2) Initial calls (to \( f_{leibi} \) and \( f_{aa} \)) to set up “combinatorial” arrays. Fix \( \varepsilon \).

3) Loop over input parameters, compute service moments and exclude values of \( \rho \geq 1 \).

4) Prepare in \( \eta^j(t) \) the values of \( \eta^j(0) \), up to \( j = \text{tm} \), where \( \text{tm} \) is the smallest value for which

\[
| \eta_{i-1}(t-1)(0) - 1 | < \varepsilon
\]

5) This is the more involved step of the entire process. For \( 0 \leq r \leq mx \), \( 0 \leq j \leq \text{tm} \) compute \( \eta^j(t) \mid r \mid_{z=0} \) through the relation

\[
\eta^j(t)(z) = \begin{cases} z & \text{j}=0 \\ \eta_i(\eta_{i-1}(z)) & \text{j}>0 \end{cases}
\]

which is used recursively over \( j \) and within it over \( r \). The simultaneous recursion requires some care in the sequencing of computations.

We use equation (22) for (A1), writing symbolically:

\[
\eta^j(t)(z) | r | = f sum(\eta^j(t)(z), \eta_{j-1}(z), r) \quad 1 \leq r \leq mx
\]

The second vector is available recursively in \( u_{13i}(r,j-1) \). The first one is done as follows. We specialize for \( i = 1 \), to simplify notation, but everything is duplicated for \( i = 2 \) as well:

\[
\eta^j(t)(z) | r | = f sum(B_{z1}(\eta^j(t)(z), B_{z1}(\eta^j(t)(z)), B_{z1}(\eta^j(t)(z)), r)
\]

where

\[
B_{z1}(r)(\eta^j(t)(z)) = (-\lambda_2)B^{[r]}(\lambda_2 - \lambda_2 \eta^j(t)(z))
\]

and

\[
B_{z1}(r)(B_{z1}(\eta^j(t)(z))) = (-\lambda)_2 B^{[r]}(\lambda_2 - \lambda_1 B_{z1}(\eta^j(t)(z)))
\]

To evaluate \( u_{31i} \) and \( u_{41i} \) we differentiate equation (19) \( r \) times,

\[
B^{[r]}(u) = f sum(S^{[r]}(\varphi(u)), \varphi^{[r]}(u), r)
\]

where
\[ \phi(u) = u + \lambda - \lambda B(u) \] (A7)

The values for \( r = 0, 1 \) were done separately: for \( r = 0 \) the iteration (21) provided the values, and for \( r = 1 \) we have, as a special case of (A8) below,

\[ B^r(u) = S^r(\varphi(u))/(1 + \lambda S^r(\varphi(u))). \] For \( r \geq 2 \) the derivatives \( \psi^{[r]}'(u) \) behave regularly, then,

\[ B^{[r]}(u) = f \sum_0^0 (S^{[k]}(\varphi(u)), \psi^{[k]}(u), \tau)/(1 + \lambda S^{[r]}(\varphi(u))). \] (A8)

where \( f \sum_0^0 \) signifies that the array entry corresponding to \( B^{[r]} \) was first zeroed (so that the sum over \( j \) in (22) goes effectively only in the range \( 2 \leq j \leq r \)).

Equation (A8) is done for the values

\[ \psi_a(u) = u + \lambda_1 \psi_1(u) \]
\[ \psi_b(u) = u + \lambda_2 \psi_2(u) \]

The derivatives \( S(\varphi(u)) \) are evaluated by the subroutines \( \text{serlstd} \), and now one can resubstitute through (A4), (A5), (A3) into (A2).

Since for later stages the derivatives \( B_{18}(\eta^{[j]}(z))^{[r]} \) are needed their evaluation is incorporated into the above recursion (the values are stored in the array \( \psi_1(r, j) \)),

6) Compute the derivatives \( \delta_i(\eta^{[j]}(z))^{[r]} \). The \( \delta_i \) are defined in equation (15). The computation is straightforward, though lengthy. For convenience we also display only for \( i = 1, \) and omit the specification \( z = 0 \) below:

\[ \delta_1(\eta^{[j]}(z))^{[r]} = f \sum((\alpha_1(\eta^{[j]}(z)))^{[k]}, \alpha_2(B_{18}(\eta^{[j]}(z)))^{[k]}, r) \]

where
\[ \alpha_1(\eta^{[j]}(z))^{[r]} = f \sum(\alpha_1^{[k]}(\eta^{[j]}(z)), \eta^{[j]}(z)^{[k]}, r), \]
\[ \alpha_2(B_{18}(\eta^{[j]}(z)))^{[r]} = f \sum(\alpha_2^{[k]}(B_{18}(\eta^{[j]}(z))), B_{18}(\eta^{[j]}(z)))^{[k]}, r) \]

Introducing the notation \( \psi_1(z) = \lambda_1(1 - B_{18}(z)) + \lambda_2(1 - z) \) we now have

\[ \alpha_1(\eta^{[j]}(z)) = A_1(\psi_1(\eta^{[j]}(z))), \]
\[ \alpha_2(B_{18}(\eta^{[j]}(z))) = A_2(\psi_2(B_{18}(\eta^{[j]}(z)))), \] (A14)

and so
\[ \alpha_1^{[r]}(\eta^{[j]}(z)) = f \sum(A_1^{[k]}(\psi_1(\eta^{[j]}(z))), \psi_1^{[k]}(\eta^{[j]}(z)), r), \]
\[ \alpha_2^{[r]}(B_{18}(\eta^{[j]}(z))) = f \sum(A_2^{[k]}(\psi_2(B_{18}(\eta^{[j]}(z))), \psi_2^{[k]}(B_{18}(\eta^{[j]}(z))), r). \]

Now
One server and threshold switching...

\[ \psi^t(r_{\beta}) = \begin{cases} -\lambda_2 - \lambda_1 B_{12}(\eta_{\beta}^t(z)) & r = 1 \\ -\lambda_1 B_{32}^t(\eta_{\beta}^t(z)) & r \geq 2 \end{cases} \]  

where the right-hand-side is available from \( u_{31} \), and

\[ \psi^t_{\beta}(B_{12}(\eta_{\beta}^t(z))) = \begin{cases} -\lambda_1 - \lambda_2 B_{21}(B_{12}(\eta_{\beta}^t(z))) & r = 1 \\ -\lambda_2 B_{31}^t(B_{12}(\eta_{\beta}^t(z))) & r \geq 2 \end{cases} \]  

where the right-hand-side is available from \( u_{41} \).

Since our computations were specialized for \( \mathcal{A}_s = e^{-a} \), the values \( A^{[k]}(\psi_1(\eta_{\beta}^t(z))) \) and \( A^{[k]}(\psi_2(B_{12}(\eta_{\beta}^t(z)))) \) were immediate to store in \( u_{151}(k,j) \) and \( u_{201}(k,j) \), respectively. The entries \( u_{221}(r,j) \) can then be evaluated.

This is the last stage in the computation that depends explicitly on the model distributions.

7) The next step is to evaluate \( \text{epsa}_4(k,z) = \prod_{i=0}^{k-1} \delta_i(\eta_{\beta}^t(z)) \) and its derivatives at \( z = 0 \).

Since we have already the derivatives of the factors, this is immediate: the empty product is 1, its derivatives vanish, and recursively...

\[ \text{epsa}^t(k,z) = f_{\text{leib}}(\text{epsa}^t_{[1]}(k-1,z),\delta_i(\eta_{\beta}^t(z)),r) \]  

where the second factor is now available in \( u_{221} \). This is carried for \( 1 \leq k \leq \text{itm} \), and the last values, \( v_{21}(r,\text{itm}) \) are also stored in \( u_{251}(r) \) for later convenience.

6) The stage is set to compute the derivatives of the three parts of equation (18) (denoted by \( \text{epsa}_1, \text{epsb}_1, \text{epsc}_1 \), respectively). This requires the derivatives of \( \beta_{i,j} \) and \( \gamma_{i,j} \), to prepare those of \( \delta_{i,j} \) and \( \xi_{i,j} \). For convenience in perusing the listing we display here the variables we used. (Only the \( i = 1 \) case is shown; all the derivatives are with respect to \( z \), at \( z = 0 \), unless otherwise noted).

\[ v_{41}(r,j) = \left( \frac{\lambda_1 B_{12}(\eta_{\beta}^t(z))}{\lambda_1 + \lambda_2 (1 - \eta_{\beta}^t(z))} \right)^{[r]} \]  

\[ v_{81}(r,j) = (B_{12}(\eta_{\beta}^t(z)))^{[r]} \]  

\[ v_{91}(r,j) = (1/\lambda_1 + \lambda_2 (1 - \eta_{\beta}^t(z)))^{[r]} \]  

\[ v_{101}(r,j) = (\lambda_1 - \lambda_2 (1 - \eta_{\beta}^t(z)))^{[r]} \]  

\[ v_{111}(r,j) = \frac{\partial^r}{\partial z^r} (1/\lambda_1 + \lambda_2 (1 - \eta_{\beta}^t(z))) \]  

\[ v_{121}(r,i,j) = (\eta_{\beta}^t(z))^{[r]} \]
One server and threshold switching ...

\[ \psi_{151}(r,i,j) = \left( B_{15}^r(\eta^j_I(z)) \right)^{[r]} \]
\[ v_{161}(r,j) = \left( g_{15}(\eta^j)(z) \right)^{[r]} \]
\[ v_{171}(r,j) = \left( \frac{\eta^j(z)}{\lambda_1 + \lambda_1 - \lambda_1 B_{15}(\eta^j_I(z))} \right)^{[r]} \]
\[ v_{181}(r,j) = \left( \frac{1}{(\lambda_2 + \lambda_1 - \lambda_1 B_{15}(\eta^j_I(z)))} \right)^{[r]} \]
\[ v_{191}(r,j) = \left( \frac{1}{\lambda_2 + \lambda_1 - \lambda_1 B_{15}(\eta^j_I(z))} \right)^{[r]} \]
\[ v_{201}(r,j) = \frac{\partial}{\partial z^r} \left( \frac{1}{z} \right) \bigg|_{z = \lambda_2 + \lambda_1 - \lambda_1 B_{15}(\eta^j_I(z))} \]
\[ v_{221}(r,j) = \left( \xi_{1,1}(\eta^j_I(z)) \right)^{[r]} \]

Note that the cost of making the above computation could be substantially reduced; if it were done as a loop on the iteration level, \( j \), of \( \eta^j_I(z) \), as we could reduce the dimensionality of most of the arrays. We preferred, however, to retain the flexibility the above structure allowed.

9) Finally one can set up the equations. Note that only now it is necessary to consider separately the different threshold values. The vector of unknowns is assumed to have the structure \( g_1(1), g_2(1), p_1(j) \ (0 \leq j \leq m_2 - 1), p_2(j) \ (0 \leq j \leq m_1 - 1) \). The first two equations are as given by equations (10) and (11), then follow the equations for the derivatives of \( g_1(z) \), and then those for the derivatives of \( g_2(z) \).

While lengthy, the computations are immediate from equation (13) and its dual. The solution of the linear system is done by our favorite linear solver (reproduced from [Forsythe, Malcolm and Moler, 1977]).

10) The rest of the computations merit no special comment. Evaluating the derivatives at \( z = 1 \) is by far easier than at \( z = 0 \); the derivatives of equations (2) and (28) pose no difficulty. In this part we could afford mildly mnemonic variable names, as seen in the listing.