Report 2001-025

Routing in Queues with Delayed Information

N. Litvak and U. Yechiali

ISSN 1389-2355

Routing in Queues with Delayed Information

Nelly Litvak

EURANDOM, P.O. Box 513, 5600 MB Eindhoven, The Netherlands

litvak@eurandom.tue.nl

Uri Yechiali

Dept. of Statistics and Operations Research, School of Mathematical Sciences,

Tel-Aviv University, Tel-Aviv 69978, Israel

uriy@post.tau.ac.il

August 6, 2001

Abstract

A controller of a c-channel queueing system assigns arriving messages to the various routes (servers). Information on service completions reaches the controller only after some (random or constant) delay. For such an $M/M/c$-type system we consider two possible routing strategies by the controller: (i) holding jobs in a common buffer and assigning awaiting jobs to idle servers only when the information on service completion is fully obtained, or (ii) dispatching jobs to the various channels immediately upon their arrival. We show that, under the first strategy, the delays on information can be interpreted as servers' vacations, and, as long as the mean delay is below a calculated threshold, this strategy is superior. If information is delayed on arrival instants, rather than on service completions, the delays can be viewed as preliminary service. Such an interpretation leads to a generalization of a decomposition-type result, obtained by Altman, Kofman and Yechiali, regarding the system's queue-size.

Keywords: Multiple-server queues, Routing, Delayed information, Vacation models, Decomposition

1 Introduction

Analysis and control of queueing systems with delayed information, either on service completions or on arrivals, are complex issues that have been studied very little in the literature (see e.g. [1], [4]). These issues are crucial in high-speed communication networks where routing decisions have to be made based on delayed information on the actual state of down-stream nodes. The lack of full information makes the problem of *optimal* routing of packets, among various possible channels, extremely difficult.

Consider a system with c parallel channels (routes, links) and a controller. Variable-length messages (jobs) arrive randomly and the controller has to route (assign) them to the various channels. There are two basic possible allocation procedures: either the controller holds a common buffer for all arriving messages, dispatching them, one at a time, to the various channels, whenever the procedure calls for; or the controller routes each message, immediately upon its arrival, to some selected queue. In the latter case the routing mechanism could follow any rule, such as shortest queue, minimum expected waiting time, random assignment or cyclic (round-robin) allocation. Clearly, the second procedure implies that every channel maintains a separate buffer for its own queueing messages (jobs).

If the controller has full information on the state of each channel (server), than holding a central single buffer for all queues and assigning a job to a server as soon as the latter becomes available, is the best policy in terms of minimizing waiting times and queue lengths. However, if the information about the actual state of each queue reaches the controller only after some considerable delay, then the problem of optimal assignment of jobs to the various queues becomes much more complex.

Suppose, indeed, that the information about each service completion reaches the controller only after some random delay. Then one can think of two possible routing policies. The first is to hold all arriving messages in a single common buffer and forward a message to a server (channel) only when the controller knows for sure that the server is free. This procedure implies that there are no separate buffers for the individual servers and that each arriving message will be assigned to a server only after an additional random delay, which clearly increases waiting times of the messages (jobs). The alternative policy for the controller is to assign jobs to the various queues 'blindly', as soon as they arrive, without fully knowing the state of each server. In such a case he can assign them either randomly, or by using a round-robin (cyclic) procedure. The cyclic procedure has been shown to be much better than the (independent and) uniform random assignment in terms of minimizing waiting times (see e.g. [8]). Another random assignment method is advocated in [4]. The procedure studied there is, giving the *delayed* information on the queue size of each of the c servers, the router, upon each arrival of a new job, selects randomly two out of the c processors, and dispatches the newly arriving job to the least (delayed-known) loaded processor among the two. By applying a fluid-limit approach, leading to a deterministic model corresponding to the limiting system as $c \to \infty$ it is demonstrated, via simulations, that "the strategy ... performs well under a large range of system parameters".

In this work we'll mainly analyze the first policy (i.e. holding all arriving jobs in a common buffer), which heavily depends on the delays in obtaining information on service completions. We will show that when the mean delay of obtaining that information is below a threshold value, this policy is better than the second one (i.e. assigning the jobs 'blindly'), thus partially answering the question of efficient routing when information on service completions is delayed.

The motivation for that first policy stems from the following observation. Consider a $G_1/M/c$ queueing system with i.i.d. inter-arrival times T having probability distribution function (p.d.f.) $G_1(t) = P(T \leq t)$ with mean $E(T) = 1/\lambda$. Service times B are i.i.d. with exponential p.d.f. and mean $E(B) = 1/\mu$. The system is stable iff $\rho = \lambda/(c\mu) < 1$. Call this configuration 'system-1'. Suppose that system-1 has to be partitioned into c separate (probabilistically identical) queues, each forming a $G_2/M/1$ queue with i.i.d. inter-arrival times X having p.d.f. $G_2(t) = P(X \leq t)$ and mean $E(X) = c/\lambda$. Call this configuration 'system-2'. Clearly, the traffic intensity for each separate queue in system-2 is $\rho = (\lambda/c)/\mu$, and system-2 is stable iff $\rho < 1$, similarly to system-1. The method by which X is generated from T could be either a probabilistic assignment, by which a newly arrived customer is routed to queue i ($i = 1, 2, ..., c$) with probability $1/c$, or a cyclic (round-robin) mechanism by which the $(kc + i)$-th arrival is assigned to queue i ($k = 0, 1, ...$). It is important to indicate that in system-1 there is a single common buffer where all arriving jobs are accumulated and from which, having full information on the state of each server (busy or idle), the controller assigns jobs to free servers; whereas in system-2 there are c separate queues and the controller, regardless of whether or not he maintains real-time information on the various queue sizes or the state of each individual server, assigns each new arrival to a channel as soon as the job arrives.

Denote by $E[W_1]$ the mean waiting (queueing) time in system-1, and by $E[W_2]$ the corresponding value in system-2. Let $r(\rho) = E[W_2]/E[W_1]$. Then it has been shown in [8] that for G_1 belonging to the family of $\Gamma(n, \lambda n)$ distributions (i.e. Erlang (Gamma) p.d.f. with n exponential phases, each having mean $1/(\lambda n)$), the ratio $r(\rho)$ possesses the following properties:

(1) $r(\rho)$ is a monotone decreasing function of ρ, $0 \leq \rho \leq 1$.

(2) For a probabilistic assignment, $r(\rho)$ tends to infinity as ρ approaches 0, and

$$\lim_{\rho \to 1} r(\rho) = [2cn - n + 1]/[n + 1].$$

(3) For a cyclic (round-robin) assignment

$$\lim_{\rho \to 0} r(\rho) = c(c!)^n,$$

whereas

$$\lim_{\rho \to 1} r(\rho) = [cn + 1]/[n + 1].$$

That is, under probabilistic (random) assignment, the smallest value of $r(\rho)$ is $r(1) = c$ ($n = 1$, T exponential), while $r(1) \to 2c - 1$ when T becomes a deterministic random variable ($n = \infty$). However, for any n, $r(0) = \infty$. Under the cyclic assignment $r(0) = c(c!)$ for $n = 1$ (T exponential) and is approaching infinity as n becomes large (T deterministic), whereas $r(1) = [c + 1]/2$ for $n = 1$, while $r(1) = c$ for $n = \infty$.

To summarize, for the family of Erlang inter-arrival distributions, the mean waiting time of an individual job in system-2 is at least $[c + 1]/2$ times larger than its corresponding waiting time in system-1. Indeed, a much smaller mean waiting time for individual jobs can be achieved if information on the system's state is available when assigning them to servers.

Now, consider the situation where information about each service completion does not reach the controller instantaneously, as it is assumed in most queueing models, but rather after some random delay. The controller then faces the dichotomy of either to assign each new message, 'blindly', as soon as it arrives, to one of the routes (employing a probabilistic or cyclic procedure), or holding the jobs in a common buffer, waiting for the delayed information on service completions to arrive, and only then to assign jobs to 'starving' servers.

As indicated above, we'll show that, as long as the mean delay of information on service completions is below a threshold value $1/\gamma^*$, it is better to hold jobs in a common buffer until full information is attained, and only then to route them to non-busy servers. This policy, imitating the multi-server queue with a common buffer and full information on system's state, leads to a special vacation model where each server, after completing serving a job, takes a random-length vacation.

Thus, we present in Section 2 an $M/M/c$-type queueing model where each server, after every service completion, takes an exponentially distributed vacation. In Section 3 we present two methods of analysis for this model: (1) we use a (partial) generating functions approach (see e.g. Mitrany and Avi-Itzhak [3], Levy and Yechiali [2]) which requires finding roots of a polynomial in order to be able to calculate the (two-dimensional) steady-state probabilities; (2) we employ Neuts's [5] matrix-geometric formulation, which also requires numerical calculation of a matrix R, which is the solution of a quadratic matrix equation in R.

For calculation purposes, however, we use in Section 4 an approximation formula for calculating mean queue size and waiting times and derive a threshold value such that, if the mean delay is below that value, it is better to wait for the delayed information to arrive, before routing jobs to various channels, rather than routing them, as soon as they arrive, but with lack of information on system's state. We then extend the calculations to the case where the delay is *deterministic* and calculate the threshold value for this case as well. Finally, in Section 5, we briefly discuss some issues regarding delayed information on arrivals.

2 The model

Consider an $M/M/c$ - type queueing system with c parallel servers, each capable of serving jobs at a rate of μ jobs per unit time. There is a common buffer from which the controller dispatches waiting jobs to servers. Such an assignment is performed only after the controller obtains information that the designated server is free and idle. However, contrary to the classical $M/M/c$ queue where information on service completions becomes available instantaneously, in our case this information is delayed. We assume that the length of the delay is an exponentially distributed random variable with mean $1/\gamma$. For the controller, the server becomes available only when the delay is over. This state of affairs leads to a two-dimensional Birth-and Death process as follows. We interpret the delay of information on service completion as a vacation: following each service completion of a job, the server takes an exponentially distributed (with mean $1/\gamma$) vacation, at the termination of which it becomes available again. We say that a server is 'operative' if it is not on vacation.

The present model differs from that of Levy and Yechiali [2] in that in the latter a server takes a vacation only when the common buffer is empty, whereas in our case a server takes a vacation after each service completion, regardless of whether the common buffer is empty or not.

It readily follows that the stability condition for that model is

$$\rho = \frac{\lambda(\mu+\gamma)}{c\gamma\mu} < 1. \tag{1}$$

Condition (1) says that for each server the mean inter-arrival time c/λ should be greater than the quantity $1/\mu+1/\gamma$, which can be interpreted as the mean duration of a generalized service time being composed of two consecutive stages: service and vacation.

Let $p_{j,n}$ be the steady-state probability that there are j operative servers and n jobs in the system. Then, for $j = 0, 1, \dots, c$, the balance equations are given by

$$\begin{aligned}
[\lambda + (c-j)\gamma + n\mu]\, p_{j,n} &= (n+1)\mu p_{j+1,n+1} + \lambda p_{j,n-1} + (c-j+1)\gamma p_{j-1,n}, \quad n < j,\\
[\lambda + (c-j)\gamma + j\mu]\, p_{j,n} &= (j+1)\mu p_{j+1,n+1} + \lambda p_{j,n-1} + (c-j+1)\gamma p_{j-1,n}, \quad n \geq j.
\end{aligned} \tag{2}$$

Here $p_{j,-1} = 0$ and $p_{-1,n} = p_{c+1,n} = 0$ for all $n \geq 0$.

To find the steady-state probabilities we have to solve equations (2).

3 Analysis

3.1 Analysis via generating functions

For $j = 0, 1, \dots, c$ define the (partial) generating function

$$G_j(z) = \sum_{n=0}^{\infty} p_{j,n} z^n.$$

Then, multiplying every equation of (2) by z^n and summing over n, we obtain

$$\begin{aligned}
[\lambda(1-z) + (c-j)\gamma + j\mu]\, G_j(z) &= (j+1)\mu z^{-1} G_{j+1}(z) + (c-j+1)\gamma G_{j-1}(z)\\
&\quad + \sum_{n=0}^{j-1} (j-n)\mu p_{j,n} z^n - z^{-1} \sum_{n=0}^{j} (j-n+1)\mu p_{j+1,n} z^n.
\end{aligned} \tag{3}$$

Set

$$
\begin{aligned}
b_0(z) &= -\mu p_{1,0};\\
b_j(z) &= z\sum_{n=0}^{j-1}(j-n)\mu p_{j,n}z^n - \sum_{n=0}^{j}(j-n+1)\mu p_{j+1,n}z^n, \quad 1 \le j \le c-1;\\
b_c(z) &= \sum_{n=0}^{c-1}(c-n)\mu p_{c,n}z^n
\end{aligned}
$$

and

$$
\begin{aligned}
f_j(z) &= z\left[\lambda(1-z) + (c-j)\gamma + j\mu\right], \quad 0 \le j \le c-1;\\
f_c(z) &= \lambda(1-z) + c\mu.
\end{aligned}
$$

Also, define the matrix $A(z)$ and the vectors $b(z)$ and $g(z)$ as

$$
A(z) = \begin{bmatrix}
f_0(z) & -\mu & 0 & 0 & \dots & 0 & 0 & 0\\
-c\gamma z & f_1(z) & -2\mu & 0 & \dots & 0 & 0 & 0\\
0 & -(c-1)\gamma z & f_2(z) & -3\mu & \dots & 0 & 0 & 0\\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots\\
0 & 0 & 0 & 0 & \dots & -2\gamma z & f_{c-1}(z) & -c\mu\\
0 & 0 & 0 & 0 & \dots & 0 & -\gamma & f_c(z)
\end{bmatrix},
$$

$$
\begin{aligned}
b(z) &= (b_0(z), b_1(z), \dots, b_c(z))^T,\\
g(z) &= (G_0(z), G_1(z), \dots, G_c(z))^T.
\end{aligned}
$$

Then (3) becomes: $A(z)g(z) = b(z)$.

To obtain $G_j(z)$ we use Cramer's rule and write $|A(z)|G_j(z) = |A_j(z)|$, $0 \le j \le c$, where $|A|$ is the determinant of the matrix A, and $A_j(z)$ is a matrix obtained from $A(z)$ by replacing the jth column by $b(z)$. It follows that the functions $G_j(z)$ are expressed in terms of $c(c+1)/2$ unknown probabilities $p_{j,n}$, $0 \le n < j \le c$, appearing in the expressions for $b_j(z)$. From the first part of (2) we have $c(c-1)/2$ linear equations for those probabilities, needing additional $(c(c+1)/2 - c(c-1)/2) = c$ equations. Those can be found by using the following theorem.

Theorem 3.1 *For any $c = 2c_0$, $c = 2c_0 + 1$, where $c_0 \ge 0$, the polynomial $|A(z)|$ has a root of multiplicity c_0 at $z_0 = 0$ and exactly $c - c_0$ roots in $(0,1)$.*

Proof. Let $q_0(z) = 1$, and define the minors of the diagonal of $A(z)$, starting from the lower right-hand side corner, as follows:

$$
q_1(z) = f_c(z), \quad q_2(z) = \begin{vmatrix} f_{c-1}(z) & -c\mu \\ -\gamma & f_c(z) \end{vmatrix}, \quad \dots, \quad q_{c+1}(z) = |A(z)|. \tag{4}
$$

The polynomials $q_j(z)$, $0 \leq j \leq c+1$, satisfy the following equations:

$$\begin{aligned} &q_1(z) = f_c(z)q_0(z),\\ &q_2(z) = f_{c-1}(z)q_1(z) - c\mu\gamma,\\ &q_{k+1}(z) = f_{c-k}(z)q_k(z) - k(c-k+1)\mu\gamma z q_{k-1}(z), \quad 2 \leq k \leq c. \end{aligned} \tag{5}$$

From (4), (5) we see that

(a) $q_0(z)$ has no roots;

(b) $q_k(z)$, where $k \geq 1$, is a polynomial of degree $2k-1$;

(c) $q_k(z)$ and $q_{k+1}(z)$ have no joint roots in $(0,\infty)$, because, if they do have such a joint root, then it is also a root of $q_{k-1}(z), q_{k-2}(z), \dots, q_0(z)$, but $q_0(z)$ possesses no roots;

(d) $q_1(0) > 0$, $q_2(0) < 0$;

(e) $q_{2l+1}(z)$ and $q_{2l+2}(z)$, where $l \geq 1$, have a root $z_0 = 0$ of multiplicity l; the l-th derivative of $q_{2l+1}(z)$ at $z = 0$ has a sign $(-1)^l$; the lth derivative of $q_{2l+2}(z)$ at $z = 0$ has a sign $(-1)^{l+1}$;

(f) if $z' > 0$ is a root of $q_k(z)$, then $q_{k+1}(z')$ and $q_{k-1}(z')$ are opposite in sign to each other;

(g) $q_k(1) = (c!\mu^k)/(c-k)!$;

(h) the sign of $q_k(\infty)$ is $(-1)^k$.

Let us now subsequently consider the roots of $q_1(z), q_2(z), \dots, q_{c+1}(z)$. Obviously, $q_1(z)$ only has the root $z_{1,1} = 1 + c\mu/\lambda > 1$. Further, $q_2(0) = -c\mu\gamma < 0$, $q_2(1) > 0$, $q_2(z_{1,1}) < 0$, $q_2(\infty) > 0$, and thus $q_2(z)$ has roots $z_{2,1} \in (0,1)$, $z_{2,2} \in (1, z_{1,1})$ and $z_{2,3} \in (z_{1,1}, \infty)$. There are no other roots, because $q_2(z)$ is of degree 3. Similarly, $q_3(z)$ is of degree 5, it has a root $z_0 = 0$, a root $z_{3,1} \in (z_{2,1}, 1)$, and there are also three other roots: $z_{3,2} \in (1, z_{2,2})$, $z_{3,3} \in (z_{2,2}, z_{2,3})$, $z_{3,4} \in (z_{2,3}, \infty)$. Now, $q_4(0) = 0$, $q_4(+0) > 0$, $q_4(z_{3,1}) < 0$, $q_4(1) > 0$. Hence, there are roots $z_0 = 0$, $z_{4,1} \in (0, z_{3,1})$ and $z_{4,2} \in (z_{3,1}, 1)$. Also, $q_4(z)$ has four other rots in $(1,\infty)$. Proceeding further, we see that $q_{c+1}(z)$, whose degree is $2c+1$, has a root of multiplicity c_0 at $z_0 = 0$, roots $z_{c+1,l} \in (0,1)$, $l = 1, 2, \dots, c - c_0$, and $c+1$ other roots in $(1,\infty)$. This completes the proof. □

Now, from Theorem 3.1 we have

$$\frac{d^k}{dz^k}|A_j(z)|\bigg|_{z=0} = 0, \quad 0 \leq j \leq c, \quad 1 \leq k \leq c_0 - 1, \tag{6}$$

$$|A_j(z_{c+1,l})| = 0, \quad 0 \leq j \leq c, \quad 1 \leq l \leq c - c_0. \tag{7}$$

The normalizing condition is

$$\sum_{j=0}^{c} G_j(1) = 1. \tag{8}$$

For $0 \leq j \leq c$, equations (6), as well as equations (7), differ only by a constant multiplier. Hence, equations (6)–(8) give exactly the c additional equations needed for a complete solution of the set (2).

3.2 Matrix-geometric approach

Following [5, Section 6.3], we construct a quasi birth-and-death process with generator $\tilde{Q}$ given by

$$\tilde{Q} = \begin{bmatrix} A_{0,0} & A_{0,1} & 0 & \cdots & & & & & & \\ A_{1,0} & A_{1,1} & A_{1,2} & \cdots & & & & & & \\ & & & \cdots & & & & & & \\ & & & & A_{c-2,0} & A_{c-2,1} & A_{c-2,2} & & & \\ & & & & & A_{c-1,0} & A_{c-1,1} & A_{c-1,2} & & \\ & & & & & & A_2 & A_1 & A_0 & \\ & & & & & & & A_2 & A_1 & \cdots \\ & & & & & & & & \cdots & \end{bmatrix}.$$

The square blocks of dimension $(c+1) \times (c+1)$ are defined as

$$(A_{0,0})_{i,h} = \begin{cases} -\lambda - (c-i)\gamma, & 0 \leq i = h \leq c, \\ (c-i)\gamma, & 0 \leq i = h-1 \leq c-1, \\ 0, & \text{otherwise}; \end{cases}$$

$$A_{0,1} = A_{1,2} = \cdots = A_{c-1,2} = A_0 = \operatorname{diag}\{\lambda, \lambda, \ldots, \lambda\};$$

$$(A_{n,0})_{i,h} = \begin{cases} \mu \min(i,n), & 0 \leq h = i-1 \leq c-1, \\ 0, & \text{otherwise}, \end{cases} \quad \text{for } 1 \leq n \leq c;$$

$$(A_{n,1})_{i,h} = \begin{cases} -\lambda - (c-i)\gamma - \mu \min(i,n), & 0 \leq i = h \leq c, \\ (c-i)\gamma, & 0 \leq i = h-1 \leq c-1, \\ 0, & \text{otherwise}, \end{cases} \quad \text{for } 1 \leq n \leq c;$$

$$A_2 = A_{c,0};$$
$$A_1 = A_{c,1}.$$

Here $(A)_{i,h}$ for $0 \leq i, h \leq c$ is the element of the ith row and hth column of the matrix A.

The matrix $Q = A_0 + A_1 + A_2$ is the generator of the classical machine repair model with μ as a breakdown rate:

$$\begin{bmatrix} -c\gamma & c\gamma & 0 & 0 & \dots & 0 & 0 & 0 \\ \mu & -(c-1)\gamma-\mu & (c-1)\gamma & 0 & \dots & 0 & 0 & 0 \\ 0 & 2\mu & -(c-2)\gamma-2\mu & (c-2)\gamma & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & (c-1)\mu & -\gamma-(c-1)\mu & \gamma \\ 0 & 0 & 0 & 0 & \dots & 0 & c\mu & -c\mu \end{bmatrix}.$$

In such a machine repair model each server is considered, independently, as alternating between two phases: service (mean $1/\mu$) and breakdown, or vacation (mean $1/\gamma$). Let $\pi = (\pi_0, \pi_1, \dots, \pi_c)$ be a stationary vector of the matrix Q, i.e. $\pi Q = \mathbf{0}$, where $\mathbf{0} = (0, 0, \dots, 0)$. In that machine repair model π_j is the stationary probability that there are j operative servers. In the case that vacations start after each service completion, π_j can be interpreted as a stationary probability that there are j operative servers given that there are c or more jobs in the system. It is readily seen that

$$\pi_j = \binom{c}{j} \left(\frac{\gamma}{\gamma+\mu}\right)^j \left(\frac{\mu}{\gamma+\mu}\right)^{c-j}.$$

Substituting this expression in the stability condition from [5, p. 83]

$$\pi A_2 \mathbf{e} > \pi A_0 \mathbf{e},$$

where $\mathbf{e} = (1, 1, \dots, 1)^T$, we again yield (1) since

$$\pi A_0 \mathbf{e} = \lambda \sum_{j=0}^{c} \pi_j = \lambda \quad \text{and} \quad \pi A_2 \mathbf{e} = \sum_{j=0}^{c} \mu j \pi_j = \frac{c\mu\gamma}{\gamma+\mu}.$$

Now let $\mathbf{p}_n = (p_{0,n}, p_{1,n}, \dots, p_{c,n})$. Then

$$\mathbf{p}_n = \mathbf{p}_{c-1} R^{n-c+1}, \quad n \geq c-1.$$

Here R is a minimal solution of the matrix quadratic equation $R^2 A_0 + R A_1 + A_2 = 0$.

The vectors $\mathbf{p}_0$, $\mathbf{p}_1$, ..., $\mathbf{p}_{c-1}$ can be found from

$$\begin{aligned} &\mathbf{p}_0 A_{0,0} + \mathbf{p}_1 A_{1,0} = 0, \\ &\mathbf{p}_0 A_{0,1} + \mathbf{p}_1 A_{1,1} + \mathbf{p}_2 A_{2,0} = 0, \\ &\mathbf{p}_{n-1} A_{n-1,2} + \mathbf{p}_n A_{n,1} + \mathbf{p}_{n+1} A_{n+1,0} = 0, \quad 2 \leq n \leq c-1. \end{aligned}$$

4 Mean queue length

The mean queue size is given by $E[L_q] = \sum_{n=c+1}^{\infty}(n-c)p_{\cdot n}$, where $p_{\cdot n} = \sum_{j=0}^{c} p_{j,n}$. Using the results of Section 3, the values of $p_{j,n}$ can be calculated in two ways, following Sections 3.1 or 3.2, respectively. By Section 3.1 one has to first obtain the values of $c(c+1)/2$ unknown probabilities $p_{j,n}$ (for $0 \le n < j \le c$) and then calculate the quantities $p_{\cdot n} = \sum_{j=0}^{c} p_{j,n}$. By Section 3.2 one has to first calculate numerically the matrix R and then the vectors $\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_{c-1}$, as well as $\mathbf{p}_n = \mathbf{p}_{c-1}R^{n-c+1}$ for $n > c-1$, from which $p_{\cdot n} = \mathbf{p}_n \cdot \mathbf{e}$ are obtained. However, in both cases $E[L_q]$ is calculated numerically only after some truncating on n.

We use, instead, a more direct approximation method as follows. As mentioned earlier, our model, where every server takes vacation after each service completion, can be interpreted as an $M/G/c$ queue with a generalized service time S composed of two consecutive phases - actual service B and vacation V. Thus, in order to calculate $E[L_q]$ we use a two-moment approximation (see Tijms [7, p. 297]):

$$E\left[L_q^{app}\right] = \left(1 - c_S^2\right) E\left[L_q(\det)\right] + c_S^2 E\left[L_q(\exp)\right], \tag{9}$$

where $c_S^2 = Var[S]/E^2[S]$ is the variation coefficient of the generalized service time $S = B + V$; $E[L_q(\exp)]$ is the mean queue size in a $M/M/c$ queue with arrival rate λ and mean service time $E[S]$, and $E[L_q(\det)]$ is the corresponding mean for an $M/D/c$ queue for which we use Cosmetatos approximation

$$E\left[L_q(\det)\right] \approx E\left[L_q^{app}(\det)\right] = \frac{1}{2}\beta E\left[L_q(\exp)\right].$$

Here

$$\beta \equiv 1 + (1-\rho)(c-1)\frac{\sqrt{4+5c}-2}{16\rho c}$$

and $\rho = \lambda E[S]/c$. Further, for the $M/M/c$ queue with mean service time $E[S]$ we have

$$E\left[L_q(\exp)\right] = \frac{\rho(c\rho)^c}{c!(1-\rho)^2}p_0,$$

where

$$p_0 = \left[\sum_{k=0}^{c-1}\frac{(c\rho)^k}{k!} + \frac{(c\rho)^c}{c!(1-\rho)}\right]^{-1}.$$

For exponential service times and vacation durations, where $E[S] = 1/\mu + 1/\gamma$ and $Var[S] = 1/\mu^2 + 1/\gamma^2$, we get $c_S^2 = (\gamma^2+\mu^2)/(\gamma+\mu)^2$. This enables us to write the right-hand side of equation (9) as a function of ρ and the ratio $E[V]/E[B] = (1/\gamma)/(1/\mu) = \mu/\gamma$:

$$E\left[L_q^{app}\right] = E\left[L_q^{app}(\text{exponential delay})\right] = \left(1+\frac{\mu}{\gamma}\right)^{-2}\left[\frac{\mu}{\gamma}\beta + 1 + \frac{\mu^2}{\gamma^2}\right] E\left[L_q(\exp)\right]. \tag{10}$$

Another approximation for the mean queue size in $M/G/c$ system is given by Nozaki and Ross [6] as follows:

$$E\left[L_q^{app}(M/G(S)/c)\right] \approx \frac{1+c_S^2}{2} E\left[L_q(M/M(S)/c)\right] = \frac{1+c_S^2}{2} E\left[L_q(\exp)\right].$$

Clearly, for $E[V] = 1/\gamma = 0$ we readily obtain, for both approximations, the classical $M/M/c$ queue with arrival rate λ, mean service time $E[S] = 1/\mu$ and utilization factor $\rho_0 = \lambda/(c\mu)$. Note that if $c_S^2 < 1$, then the Cosmetatos approximation gives higher $E\left[L_q^{app}\right]$ values than the Nozaki-Ross approximation, and vice versa. This follows since $\beta > 1$. Note also that, when $c = 1$, $\beta = 1$, making the two approximations equal.

To ensure stability, condition (1) requires that the ratio μ/γ must be smaller than a critical value

$$\mu/\gamma^{crit} = \frac{1-\rho_0}{\rho_0}.$$

Moreover, for fixed μ, $E\left[L_q^{app}\right]$ increases with μ/γ.

Let us now compare system-1, where the controller assigns a newly arrived job to a server only (and immediately) after the former obtains the (delayed) information that the server has completed serving an earlier job, with system-2, where the controller assigns jobs to servers 'blindly', as soon as they arrive, following the round-robin procedure. In the latter case there are c separate sub-systems, each being an $E_c/M/1$ queue with mean arrival rate λ/c, mean service time $1/\mu$ and $\rho_0 = \lambda/(c\mu)$. The mean queueing time is given by (see [8])

$$E\left[W_q(E_c/M/1)\right] = \frac{\alpha}{\mu(1-\alpha)},$$

where α is the unique root in $(0,1)$ of the equation

$$z = (\lambda/(\mu(1-z)+\lambda))^c = (c\rho_0/(1-z+c\rho_0))^c.$$

Thus, the mean queue length for each individual queue is

$$E\left[L_q(E_c/M/1)\right] = \frac{\lambda}{c} E\left[W_q(E_c/M/1)\right] = \frac{\rho_0\alpha}{1-\alpha}.$$

It follows that the total mean queue size among all c separate queues is

$$E\left[L_q(\text{cyclic})\right] = \frac{c\rho_0\alpha}{1-\alpha}.$$

Since, for any $c > 1$, the value $E\left[L_q(\text{cyclic})\right]$ is greater than $E\left[L_q(M/M/c)\right]$ with the same ρ_0 (see [8]), there exists a threshold value $\mu/\gamma^* \in (0, \mu/\gamma^{crit})$ such that $E\left[L_q^{app}\right]$ becomes equal to $E\left[L_q(\text{cyclic})\right]$. Thus, as long as $\mu/\gamma < \mu/\gamma^*$, it is better to keep all arriving jobs in a common buffer and operate the system by using the delayed information on service completions, rather than to assign arriving jobs 'blindly', immediately upon arrival, to the various servers, even if it is done in a cyclic manner. However, if $\mu/\gamma > \mu/\gamma^*$, then the delays are too long, and it is better to have a separate buffer for each server and assign new jobs to the various servers following the round-robin procedure, without waiting for the delayed information to arrive.

In Table 1 below we give the values of μ/γ^*, for some different values of ρ_0 and c. Since $c_S^2 < 1$ when the delay is exponential (or deterministic), we use the two-moment approximation (9) for calculations. The results show that, for example, if $\rho_0 = 0.8$ then, for stability, we must have $E[V] = 1/\gamma < 1/\gamma^{crit} = 0.25(1/\mu) = 0.25E[B]$, while, for $c = 10$, as long as $1/\gamma < 1/\gamma^* = 0.1934(1/\mu)$, it is preferred to operate the system by using the delayed information. It is seen that, for a given value of ρ_0, μ/γ^* increases with growing numbers of servers and that, for fixed c, μ/γ^* decreases when ρ_0 increases.

Table 1. Threshold values μ/γ^* when delays are exponential

		$c=1$	$c=2$	$c=3$	$c=5$	$c=10$
$\rho_0 = 0.2$	$E[L_q(M/M/c)]$	0.05	0.0167	0.0062	0.0096	0.00001
$\mu/\gamma^{crit} = 4$	$E[L_q(\text{cyclic})]$	0.05	0.0414	0.0376	0.0353	0.0376
	$E[V]/E[B] = \mu/\gamma^*$	0	0.4263	0.6781	1.0332	1.5723
$\rho_0 = 0.5$	$E[L_q(M/M/c)]$	0.5	0.3333	0.2368	0.1304	0.0361
$\mu/\gamma^{crit} = 1$	$E[L_q(\text{cyclic})]$	0.5	0.6180	0.7406	0.9905	1.6232
	μ/γ^*	0	0.2190	0.3373	0.4769	0.6411
$\rho_0 = 0.8$	$E[L_q(M/M/c)]$	3.2	2.8444	2.5888	2.2170	1.6367
$\mu/\gamma^{crit} = 0.25$	$E[L_q(\text{cyclic})]$	3.2	4.5564	5.9026	8.6090	15.3782
	μ/γ^*	0	0.0744	0.1128	0.1534	0.1934
$\rho_0 = 0.9$	$E[L_q(M/M/c)]$	8.1	7.6737	7.3535	6.8624	6.0186
$\mu/\gamma^{crit} = 0.1111$	$E[L_q(\text{cyclic})]$	8.1	11.8589	15.6188	23.1394	41.9425
	μ/γ^*	0	0.0352	0.0531	0.0715	0.0888

In fact, the above calculations can be extended to the case where the delay V is deterministic with $V = E[V] = 1/\gamma$. In such a case $E[S] = 1/\mu + 1/\gamma$, as before, but $Var[S] = 1/\mu^2$. This implies that $c_S^2 = (1+\mu/\gamma)^{-2}$ and $1 - c_S^2 = (1+\mu/\gamma)^{-2}\,(2\mu/\gamma + (\mu/\gamma)^2)$. Thus, equation (9) leads to

$$E\left[L_q^{app}(\text{deterministic delay})\right] = \left(1+\frac{\mu}{\gamma}\right)^{-2}\left[\left(\frac{\mu}{\gamma}+\frac{1}{2}\frac{\mu^2}{\gamma^2}\right)\beta+1\right]E\left[L_q(\exp)\right].$$

It follows that (see (10))

$$\begin{aligned} E\left[L_q^{app}(\text{exponential delay})\right] &- E\left[L_q^{app}(\text{deterministic delay})\right] \\ &= \left(1+\frac{\mu}{\gamma}\right)^{-2}\left[\frac{\mu^2}{\gamma^2}\left(1-\frac{\beta}{2}\right)\right] E\left[L_q(\exp)\right]. \end{aligned}$$

Table 2 below gives, similarly to Table 1, values of μ/γ^* for various combinations of ρ_0 and c. It is seen that μ/γ^* in Table 2 is slightly bigger than in Table 1, but the difference decreases when ρ_0 increases.

Table 2. Threshold values μ/γ^* when delays are deterministic

		$c=1$	$c=2$	$c=3$	$c=5$	$c=10$
$\rho_0 = 0.2$ $\mu/\gamma^{crit} = 4$	μ/γ^*	0	0.4532	0.7199	1.0877	1.6341
$\rho_0 = 0.5$ $\mu/\gamma^{crit} = 1$	μ/γ^*	0	0.2256	0.3487	0.4924	0.6581
$\rho_0 = 0.8$ $\mu/\gamma^{crit} = 0.25$	μ/γ^*	0	0.0747	0.1134	0.1541	0.1942
$\rho_0 = 0.9$ $\mu/\gamma^{crit} = 0.1111$	μ/γ^*	0	0.0353	0.0532	0.0716	0.0889

5 Delayed information on arrivals

In this section we discuss general queueing systems where information on arrivals, rather than on service completions, is delayed. We consider an arbitrary structure of arrival flow and arbitrary service discipline. Furthermore, we don't specify the number of servers or the distribution of the service times. It is assumed that the delays are independent and are all distributed as some random variable τ. The control policy is such that a job is routed to a server only when the controller knows for sure that the common buffer, which holds all arriving messages, is not empty.

Altman, Kofman and Yechiali [1] studied a queue length process $\{X_n\}$, $n = -K, -K+1, \ldots, -1, 0, 1, \ldots$, in a discrete-time single-server queue, satisfying the conditions above. They considered a stationary process $\{Y_n\}$ of batch arrivals (Y_n is the number of jobs arriving at time slot n) and arbitrary service times. For such a system they proved that if information on arrivals is delayed by K slots, then

$$X_n \overset{d}{=} X_n^0 + \sum_{i=1}^{K} Y_{n-i+K}, \tag{11}$$

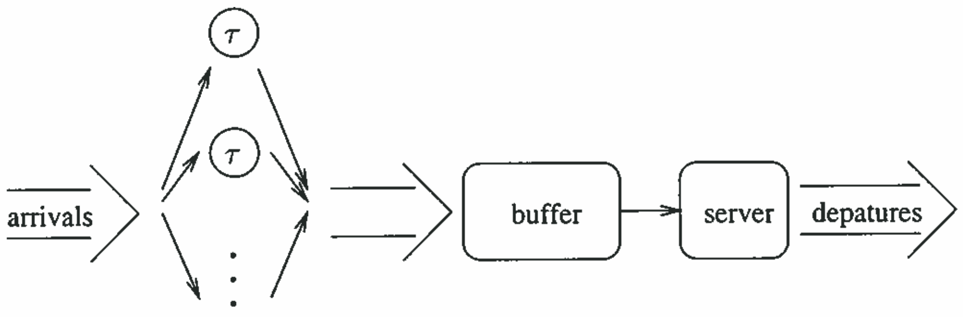

Figure 1: A delayed information on arrivals can be seen as a 'preliminary' service.

where $\{X_n^0\}$, $n = 0, 1, \ldots$, is the queue length process corresponding to no delay on arrival information (K=0), and $X_{-K} \overset{d}{=} X_0^0$. To explain formula (11) on an intuitive level as well, the authors point out that the delays of information on arrivals may be looked upon as extra server's vacations, since the server may stay idle even if there are jobs present in the buffer. This observation is still true for more general systems with delayed information on arrivals. Thus, once again, there exists a strong connection between models with server's vacations and models with delayed information.

However, for delayed information on arrivals, the interpretation via vacation model is not that natural as for delayed information on service completions. For example, for the system studied in [1] the description of the analogous vacation model may read as follows: *"The server takes a vacation at time n if the buffer became empty at time $n-K$. The server becomes available again at time $n+l$, if the first arrival after time $n-K$ occurred at time $n-K+l$"*. Within such a vacation model formula (11) expresses a direct decomposition, where the first term of the right-hand side is the queue length in the beginning of an arbitrary time slot in the system without vacations, and the second term is the queue length at the beginning of an arbitrary non-serving slot. Yet, this formula is still not very intuitive, and formally it requires a proof, which is of the same difficulty as the direct proof for the system with delayed information.

Using the general model, we propose another outlook of delayed information on arrivals: when a job arrives to the buffer, the controller does not know about its existence during some random delay. When the delay is over, the controller recognizes the job, and operates as if this job has just arrived. From the point of view of the controller, this system differs from the system without delayed information only by the characteristics of the arrival flow. Thus, if each delay equals the same given constant K, than the controller observes the same arrival flow, shifted by K time units. Actually, the controller may have no idea about the delays. However, in reality, the delays dictate that new arrivals have to waste additional time, hanging in the system, causing an increased queue length.

This situation can be better visualized with the aid of Figure 1. A new job first arrives to a 'preliminary' queue with infinite number of servers and immediately gets served, where its service time is distributed as τ. Completing this delay, the job immediately proceeds to the main queue which is the same as our original system, but without delays (and, of course, with a modified structure of the arrival flow). The queue length in the original system with delayed information equals the number of jobs in the first queue in Figure 1 plus the buffer content of the second queue. In general, such a tandem queueing system may not allow an explicit probabilistic analysis. Nevertheless, result (11) follows immediately from the above interpretation. One just has to note that, at the beginning of a time slot n, the number of jobs in the first queue is $\sum_{i=1}^{K} Y_{n-1-K+i}$, and the arrival flow to the second queue is the same as the original one, shifted by K slots.

Acknowledgement

We would like to acknowledge Daniel Kofman for discussions that motivated this work.

References

[1] E. Altman, D. Kofman and U. Yechiali, Discrete time queues with delayed information, Queueing Systems 19 (1995) 361–376.

[2] Y. Levy and U. Yechiali, An $M/M/s$ queue with server's vacations, INFOR 14 (1976) 153–163.

[3] I.L. Mitrany and B. Avi-Itzhak, A many server queue with service interruptions, Oper. Res. 16 (1968) 628–638.

[4] M. Mitzenmacher, How useful is old information, IEEE Trans. Parallel Distrib. Systems 11 (2000) 6–20.

[5] M.F. Neuts, *Matrix-geometric Solutions in Stochastic Models - An Algorithmic Approach,* Johns Hopkins, Baltimore, 1981.

[6] S.A. Nozaki and S.M. Ross, Approximation in finite capacity multi-server queues with Poisson arrivals, J. Appl. Prob. 15 (1978) 826–834.

[7] H.C. Tijms, *Stochastic Models: an Algorithmic Approach,* Wiley, 1994.

[8] U. Yechiali, On the relative waiting times in the $GI/M/s$ and the $GI/M/1$ queueing systems, Oper. Res. Quart. 28 (1977) 325–337.