

# A Push Pull Queueing System\*

Gideon Weiss      Anat Kopzon

Department of Statistics  
The University of Haifa  
Mount Carmel 31905, Israel.  
gweiss@stat.haifa.ac.il

August 15, 2001

## Abstract

We consider a two node multiclass queueing network given by two machines each with two classes. There are two streams of jobs: One stream originates in machine 1, which feeds it for further processing to machine 2, and the other stream moves in the opposite direction. We describe a policy for this system which is stable and which keeps both machines busy at all times. We obtain explicit expressions for its steady state behavior under M/M/· and under M/G/· assumptions. We also describe a similar  $M$  machine system.

Keywords: Queueing, Vacation Models, Manufacturing, Priority Scheduling Policies.

## 1 Introduction

We consider a system with two machines, engaged in two production processes which are moving through the machines in opposite order. Each job in stream 1 requires first operation at machine 1, with processing rate  $\lambda_1$ , and second operation at machine 2, with processing rate  $\mu_2$ , while jobs in stream 2 move in the opposite direction, with rates  $\lambda_2, \mu_1$ ; see Figure 1. We assume throughout that  $\lambda_1 < \mu_2, \lambda_2 < \mu_1$ .

In this paper we find a policy to operate this *push pull system* so that it is stable, and the machines are busy at all times. We consider two models, the M/M/· model where the processing times of all operations are memoryless, and the M/G/· model, where the first operations of each process are memoryless while the second operations are i.i.d. with general distributions. For both models we obtain explicit expressions for the steady state distributions of the queues between the machines. We also describe a generalization of the two machine push pull system to  $M$  machines.

We study this two machine push pull system because we believe it is the simplest case of a multiclass queueing network which models some of the most important features of manufacturing systems. These features are not usually found in the queueing models of communications or service systems.

---

\*Research supported in part by German-Israel GIF grant I-564-246/06/97

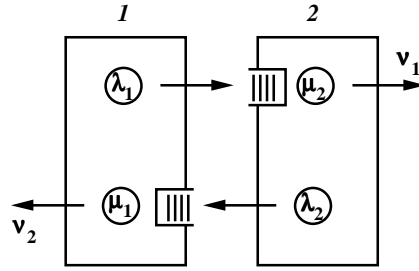


Figure 1: The Two Machine Push Pull System

To clarify our motivation, consider the following five queueing models:

**Single server queue** : There is a stream of customers which arrive at the system with rate  $\lambda$ , and which require processing by the single server with rate  $\mu$ . This system is stable if and only if  $\lambda < \mu$ , in which case the server will be idle for a fraction  $1 - \lambda/\mu$  of the time.

**Single machine manufacturing system** : There is a single machine which produces parts at a rate  $\mu$ . Instead of a random stream of arrivals, we assume more realistically that whenever the machine finishes a job, it will immediately be supplied with another job. In a well run manufacturing system this would correspond to a full order book for the product. It can be modeled by an infinite queue in front of the machine — we call it a virtual infinite queue. This system produces at rate  $\mu$  and is busy all the time.

**Manufacturing system with two machines in tandem** : There is a virtual infinite queue of jobs in front of the first machine. The first machine performs a first operation at rate  $\lambda$ . The second machine performs a second operation at rate  $\mu$ . The output of the first machine acts as an input stream to the second, and the second machine acts like the single server queue. There is a queue of parts between the two machines, and unless we operate the first machine at a rate  $\lambda < \mu$ , this queue will be unstable. If  $\lambda < \mu$  the queue will be stable, jobs will be produced at a rate  $\lambda$ , and the second machine will be idle a fraction  $1 - \lambda/\mu$  of the time.

**Tandem system with additional stream for the second machine** : If we wish to utilize the second machine fully, we can find another product which requires only a single operation, at rate  $\lambda_2$  from the second machine. When the second machine runs out of jobs of the first product, it produces the second product. At each job completion the second machine will then check its queue of the first product, and if it is non-empty it will return to produce the first product. Here the second machine works like a single server with vacations, and the system is stable (see Yechiali [8] for an early suggestion of a vacation model to utilize excess machine capacity). The first product is produced at rate  $\lambda$ , while the second product is produced at rate  $\lambda_2(1 - \lambda/\mu)$ . In this system both machines are busy all the time.

**The push pull system** : Generalizes the previous model, by assuming that both products require the two machines, in opposite order. What we shall see is that if  $\lambda_1 < \mu_2$ ,  $\lambda_2 < \mu_1$  then this system can be operated by a policy which will keep it stable, and will keep both machines busy at all times.

The important features of the manufacturing system are: Virtual infinite queues for each of the manufacturing processes (this corresponds to full order books for the plant), and several routes through the machines. These different routes, with virtual infinite queues, enable us to use a balanced policy, so that the entire system is stable, and all the machines are fully utilized.

## 2 The Buffer Priority Policy

In our push pull system each machine has a virtual infinite queue of jobs waiting for their first operation, and so none of the machines needs ever to be idle. When a machine is performing this first operation we say that it is producing (jobs which it feeds to the other machine). When a machine is performing the second operation on a job we say it is serving (the queue of jobs which the other machine produced). If the system is to be stable, the rate of jobs produced and served on each route must be the same, and so for stable operation with both machines busy all the time we will have:

$$\begin{aligned}\nu_1 &= (1 - \alpha_1)\lambda_1 = \alpha_2\mu_2, \\ \nu_2 &= (1 - \alpha_2)\lambda_2 = \alpha_1\mu_1,\end{aligned}\tag{2.1}$$

where  $\alpha_i$  is the fraction of time that machine  $i$  is serving, and  $\nu_i$  is the rate of production of stream  $i$ . Hence:

$$\begin{aligned}\alpha_1 &= \frac{\lambda_2(\mu_2 - \lambda_1)}{\mu_1\mu_2 - \lambda_1\lambda_2}, \\ \alpha_2 &= \frac{\lambda_1(\mu_1 - \lambda_2)}{\mu_1\mu_2 - \lambda_1\lambda_2}, \\ \nu_1 &= \frac{\lambda_1\mu_2(\mu_1 - \lambda_2)}{\mu_1\mu_2 - \lambda_1\lambda_2}, \\ \nu_2 &= \frac{\lambda_2\mu_1(\mu_2 - \lambda_1)}{\mu_1\mu_2 - \lambda_1\lambda_2}.\end{aligned}\tag{2.2}$$

We will show that the following policy will achieve this:

**Buffer Priority Policy:** For machine  $i$ , when machine  $i$  becomes free (at completion of production or service job) do the following:

- If the job completed was a service job, job leaves the system.
- check the queue of machine  $i$ .
- If it is empty, feed a job to machine  $3 - i$ , and start a production job.

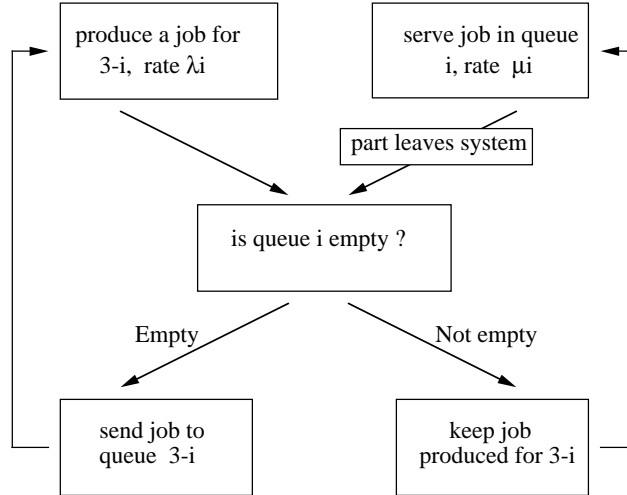


Figure 2: The Buffer Priority Policy, for Machine  $i$

- If it is not empty, start a service job.

Note the following important feature of our policy: If machine  $i$  finishes the production of a job, and it finds that it has a non-empty queue, then it will keep the job which it has produced, and will only send it to the queue of machine  $3 - i$  when it completes the service of all the customers in its queue.

**Lemma 2.1** *After an initial period, at any time exactly one of the queues of the two machines is not empty.*

**Proof.** Assume the queues at both machines are not empty at time 0. Then, while both queues are not empty, they will both not grow. Both machines will start serving their queues right away or when they complete the operation which they were processing at time 0. Once they start processing they will serve jobs which will leave the system at rates  $\mu_1, \mu_2$ . Hence there will (almost surely) be a finite earliest time  $T_1$  at which one of the machines, say machine  $i$ , will have an empty queue, and the queue of machine  $3 - i$  will not be empty.

Following this, while the queue of machine  $3 - i$  is not empty, it will not send any work to machine  $i$ . Hence at  $T_1$  will start a period during which the queue of machine  $i$  remains empty, and the queue of machine  $3 - i$  will be fed at rate  $\lambda_i$  by the production of machine  $i$ . Machine  $3 - i$  will become available at time  $T_2 \geq T_1$ , when it completes the operation which it was performing at  $T_1$ . Note that even if the operation completed at  $T_2$  was a production operation (which must have started at time  $\leq 0$ ), this job will not be sent to machine  $i$ , because the queue of machine  $3 - i$  is not empty. Starting at time  $T_2$  machine  $3 - i$  will serve its queue at rate  $\mu_{3-i}$ , and new jobs will arrive at rate  $\lambda_i$ . Since  $T_2$  is finite (almost surely), the queue at machine  $3 - i$  at time  $T_2$  is finite (almost surely). Since  $\lambda_i < \mu_{3-i}$ , there will be (almost surely) an earliest finite time  $T_3 > T_2$ , at which the queue of machine  $3 - i$  will be empty.

At  $T_3$  machine  $i$  will be in the middle of producing a job for queue  $3 - i$ . Machine  $3 - i$  will be free and when it examines its queue it will find it empty. It will then send a job to queue  $i$  if it has a job ready, and it will start producing the next job for queue  $i$ . So immediately after  $T_3$  both machines will be producing. There may already be a job waiting at machine  $i$  (sent at  $T_3$  by machine  $3 - i$ ). Otherwise after the earlier of the two machines completes its production job, it will send that job to the other machine. So at time  $T_4$ , where  $T_4 = T_3$  or  $T_4 =$  completion of production by earlier machine, there will be a job waiting at the queue of one of the machines, while the other machine will have an empty queue, and at  $T_4$  both machines are producing.

We claim that from time  $T_4$  onwards, exactly one queue is non-empty at all times. This is because from this time onwards, if machine  $i$  has a non-empty queue, then until it is empty, no jobs will be sent to the other queue, which will therefore remain empty. However, at the same instant that the queue of machine  $i$  will become empty, machine  $i$  will send a job to the queue of machine  $3 - i$ , and from that moment, the queue of machine  $i$  will remain empty until the queue of machine  $3 - i$  reaches 0. ■

Clearly, under our buffer priority policy, both machines are busy all the time. In the next two sections we derive the steady state behavior of this system, for the M/M/· and for the M/G/· models. This will show that the system is stable under the buffer priority policy.

### 3 Analysis of the M/M/· model, via balance equations

We now assume that all the four types of operations in our system are memoryless, so that all the processing times are independent, and exponentially distributed with the appropriate rates. Our system is then described by a discrete state continuous time Markov process (see [10, 12]). This process will, after some initial period as described in Lemma 2.1, have the following states:

- (**a,i**) : Both machines are producing and the queue of machine 1 is non-empty, with  $i$  jobs in the queue.
- (**A,i**) : Machine 1 is serving, and machine 2 is producing, and there are  $i$  jobs in the queue of machine 1, including the job which is in service.
- (**b,i**) : Both machines are producing and the queue of machine 2 is non-empty, with  $i$  jobs in the queue.
- (**B,i**) : Machine 2 is serving, and machine 1 is producing, and there are  $i$  jobs in the queue of machine 2, including the job which is in service.

The transition rates between the states are described in the following Figure 3. Denote the steady state probabilities

$$s_i = P(a, i), \quad u_i = P(A, i), \quad r_i = P(b, i), \quad v_i = P(B, i), \quad i = 1, 2, \dots$$

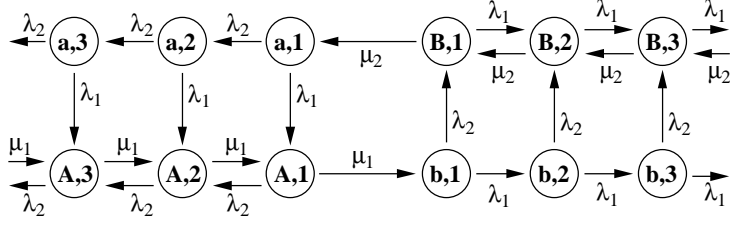


Figure 3: States and Transition Rates for the Push Pull System

define the partial generating functions:

$$S(Z) = \sum_{i=1}^{\infty} s_i Z^i, \quad R(Z) = \sum_{i=1}^{\infty} r_i Z^i, \quad U(Z) = \sum_{i=1}^{\infty} u_i Z^i, \quad V(Z) = \sum_{i=1}^{\infty} v_i Z^i,$$

and let:

$$\begin{aligned} \Pi_a &= P(a \text{ states}) = S(1), & \Pi_b &= P(b \text{ states}) = R(1), \\ \Pi_A &= P(A \text{ states}) = U(1), & \Pi_B &= P(B \text{ states}) = V(1). \end{aligned}$$

**Proposition 3.1** *The steady state probabilities for the M/M/. push pull system are:*

$$\begin{aligned} s_i &= \frac{(\mu_2 - \lambda_1)(\mu_1 - \lambda_2)}{(\mu_1 \mu_2 - \lambda_1 \lambda_2)} \left( \frac{\lambda_1}{\lambda_1 + \lambda_2} \right) \left( \frac{\lambda_2}{\lambda_1 + \lambda_2} \right)^i, & u_i &= \frac{(\mu_2 - \lambda_1)(\mu_1 - \lambda_2) \lambda_1}{(\mu_1 \mu_2 - \lambda_1 \lambda_2)(\lambda_1 + \lambda_2 - \mu_1)} \left[ \left( \frac{\lambda_2}{\mu_1} \right)^i - \left( \frac{\lambda_2}{\lambda_1 + \lambda_2} \right)^i \right], \\ r_i &= \frac{(\mu_2 - \lambda_1)(\mu_1 - \lambda_2)}{(\mu_1 \mu_2 - \lambda_1 \lambda_2)} \left( \frac{\lambda_2}{\lambda_1 + \lambda_2} \right) \left( \frac{\lambda_1}{\lambda_1 + \lambda_2} \right)^i, & v_i &= \frac{(\mu_2 - \lambda_1)(\mu_1 - \lambda_2) \lambda_2}{(\mu_1 \mu_2 - \lambda_1 \lambda_2)(\lambda_1 + \lambda_2 - \mu_2)} \left[ \left( \frac{\lambda_1}{\mu_2} \right)^i - \left( \frac{\lambda_1}{\lambda_1 + \lambda_2} \right)^i \right]. \end{aligned}$$

**Proof.** We set up balance equations for the steady state probabilities based on the transition rates:

$$\begin{aligned} s_i(\lambda_1 + \lambda_2) &= s_{i-1} \lambda_2, & u_i(\lambda_2 + \mu_1) &= s_i \lambda_1 + u_{i+1} \mu_1 + u_{i-1} \lambda_2, \\ r_i(\lambda_1 + \lambda_2) &= r_{i-1} \lambda_1, & v_i(\lambda_1 + \mu_2) &= r_i \lambda_2 + v_{i+1} \mu_2 + v_{i-1} \lambda_1, \end{aligned} \quad i = 2, \dots$$

with equations for the boundaries:

$$\begin{aligned} s_1(\lambda_1 + \lambda_2) &= v_1 \mu_2, & u_1(\lambda_2 + \mu_1) &= s_1 \lambda_1 + u_2 \mu_1, \\ r_1(\lambda_1 + \lambda_2) &= u_1 \mu_1, & v_1(\lambda_1 + \mu_2) &= r_1 \lambda_2 + v_2 \mu_2. \end{aligned}$$

Using the standard technique, multiply equation for item  $i$  by  $Z^i$  and sum up, we get equations for the generating functions, which are solved by:

$$\begin{aligned} S(Z) &= \frac{v_1 \mu_2 Z}{\lambda_1 + \lambda_2(1-Z)}, & U(Z) &= \frac{\lambda_1 S(Z) - u_1 \mu_1}{\lambda_2(1-Z) + \mu_1(1-1/Z)}, \\ R(Z) &= \frac{u_1 \mu_1 Z}{\lambda_2 + \lambda_1(1-Z)}, & V(Z) &= \frac{\lambda_2 R(Z) - v_1 \mu_2}{\lambda_1(1-Z) + \mu_2(1-1/Z)}. \end{aligned} \quad (3.1)$$

Note that  $S(1) = \frac{v_1 \mu_2}{\lambda_1}$ , and that the denominator of  $U(Z)$  vanishes at  $Z = 1$ . Since in a steady state solution  $0 < U(1) < 1$ , the numerator of  $U(Z)$  at  $Z = 1$ , which is  $v_1 \mu_2 - u_1 \mu_1$ , must also vanish. Hence we get

$$v_1 \mu_2 = u_1 \mu_1.$$

Substituting this into (3.1) we get:

$$U(Z) = \frac{u_1 \mu_1 (\lambda_1 + \lambda_2) Z}{(\lambda_1 + \lambda_2 (1-Z))(\mu_1 - \lambda_2 Z)}, \quad V(Z) = \frac{u_1 \mu_1 (\lambda_1 + \lambda_2) Z}{(\lambda_2 + \lambda_1 (1-Z))(\mu_2 - \lambda_1 Z)}. \quad (3.2)$$

We note that  $\Pi_A = U(1)$  is exactly the fraction of time  $\alpha_1$  that machine 1 is serving its queue. Hence, from (3.2) and (2.2):

$$\frac{u_1 \mu_1 (\lambda_1 + \lambda_2)}{\lambda_1 (\mu_1 - \lambda_2)} = \frac{\lambda_2 (\mu_2 - \lambda_1)}{\mu_1 \mu_2 - \lambda_1 \lambda_2} \Rightarrow u_1 = \frac{(\mu_2 - \lambda_1)(\mu_1 - \lambda_2) \lambda_1 \lambda_2}{(\mu_1 \mu_2 - \lambda_1 \lambda_2)(\lambda_1 + \lambda_2 - \mu_1)(\lambda_1 + \lambda_2) \mu_1}.$$

Substituting back into (3.1,3.2) we get explicit expressions for the various  $\Pi$ :

$$\begin{aligned} \Pi_a &= \frac{(\mu_2 - \lambda_1)(\mu_1 - \lambda_2) \lambda_2}{(\mu_1 \mu_2 - \lambda_1 \lambda_2)(\lambda_1 + \lambda_2 - \mu_1)(\lambda_1 + \lambda_2)}, & \Pi_A &= \frac{(\mu_2 - \lambda_1) \lambda_2}{(\mu_1 \mu_2 - \lambda_1 \lambda_2)(\lambda_1 + \lambda_2 - \mu_1)}, \\ \Pi_b &= \frac{(\mu_2 - \lambda_1)(\mu_1 - \lambda_2) \lambda_1}{(\mu_1 \mu_2 - \lambda_1 \lambda_2)(\lambda_1 + \lambda_2 - \mu_2)(\lambda_1 + \lambda_2)}, & \Pi_B &= \frac{(\mu_1 - \lambda_2) \lambda_1}{(\mu_1 \mu_2 - \lambda_1 \lambda_2)(\lambda_1 + \lambda_2 - \mu_2)}. \end{aligned} \quad (3.3)$$

Explicit expressions for the generating functions follow:

$$\begin{aligned} S(Z) &= \Pi_a \frac{\lambda_1 Z}{\lambda_1 + \lambda_2 (1-Z)}, & U(Z) &= \Pi_A \frac{\lambda_1 (\mu_1 - \lambda_2) Z}{(\lambda_1 + \lambda_2 (1-Z))(\mu_1 - \lambda_2 Z)}, \\ R(Z) &= \Pi_b \frac{\lambda_2 Z}{\lambda_2 + \lambda_1 (1-Z)}, & V(Z) &= \Pi_B \frac{\lambda_2 (\mu_2 - \lambda_1) Z}{(\lambda_2 + \lambda_1 (1-Z))(\mu_2 - \lambda_1 Z)}. \end{aligned} \quad (3.4)$$

Recall that the generating function of a geometric random variable that counts the number of  $p$ -trials until the first success, which we denote  $\sim \text{Geom}_1$ , is  $pZ/1 - (1-p)Z$ , and the generating function of a geometric random variable that counts the number of  $p$ -failures until the first success, which we denote  $\sim \text{Geom}_0$ , is  $p/1 - (1-p)Z$ . Hence, we see that:

$$\begin{aligned} P(a, i) &= \Pi_a P(X = i), & X &\sim \text{Geom}_1 \left( \frac{\lambda_1}{\lambda_1 + \lambda_2} \right), \\ P(A, i) &= \Pi_A P(X + Z = i), & Z &\sim \text{Geom}_0 \left( 1 - \frac{\lambda_2}{\mu_1} \right), \\ P(b, i) &= \Pi_b P(Y = i), & Y &\sim \text{Geom}_1 \left( \frac{\lambda_2}{\lambda_1 + \lambda_2} \right), \\ P(B, i) &= \Pi_B P(Y + W = i), & W &\sim \text{Geom}_0 \left( 1 - \frac{\lambda_1}{\mu_2} \right). \end{aligned}$$

and the proposition now follows. ■

## 4 Analysis of the M/G/· case, via vacation models

We now assume that the service times at machines 1,2 are independent identically distributed, independent of the production stream, with some general distribution function  $G_i(t)$ ,  $i = 1, 2$ , with expected production time  $1/\mu_i$ , and with moment generating function  $G_i^*(s)$ . We analyze this more general case by using the model of a server with vacations.

We note first that the push pull system under our buffer priority policy operates in cycles: Entry to state  $(a, 1)$  (when the queue of machine 2 empties) is followed by an  $a$ -period, with

states  $(a, i)$ , until machine 1 finishes production and returns to find a non-empty queue and a transition to state  $(A, i)$  occurs, which starts an  $A$ -period. The  $A$ -period continues until the queue at machine 1 is empty, when a  $b$ -period starts, followed by a  $B$ -period, analogous to the  $a, A$ -periods, but with the queue of machine 2 not empty. Hence the cycles move in the order  $a \rightarrow A \rightarrow b \rightarrow B$ . Figure 4 illustrates the cyclic sample path of the push pull system

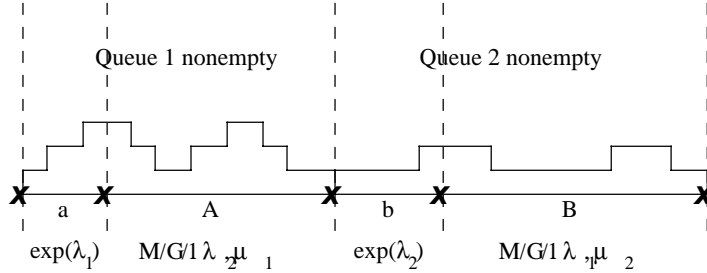


Figure 4: Cyclic Sample Path of the Push Pull System

Consider now a complete cycle, from the view of machine 1. Machine 1 is serving its cycle during the  $A$ -period, and it is not serving during the following  $b \rightarrow B \rightarrow a$  periods. In these  $b, B, a$ -periods, arrivals to the queue of machine 1 occur as follows: There are no arrivals (and the queue is empty) during the  $b, B$ -periods. Then there is one arrival at the start of the  $a$ -period, and additional arrivals, according to a Poisson process of rate  $\lambda_2$ , during the  $a$ -period.

Consider now a single server M/G/1 model with vacations [7, 12], defined as follows: Arrivals occur at all times in a Poisson stream of rate  $\lambda_2$ . Service times are i.i.d. independent of the arrival process, with service time distribution  $G_1(t)$ . When the system is empty the server goes on a vacation, with the following rule: The server waits until the first arrival (after an  $\sim \exp(\lambda_2)$  period), and then stays on vacation for an additional  $\sim \exp(\lambda_1)$  period. It is seen immediately that this system will behave exactly like the queue of machine 1 in the push pull system, after the following change: Each  $b, B$  period is replaced by a single  $\sim \exp(\lambda_2)$  period, with an arrival at its end.

Hence the queue length process of machine 1 of the push pull system will be identical in its behavior to that of the M/G/1 server with vacations, except that its empty periods will be different.

We now use the results known about the M/G/1 server with vacations [12]: The number of customers in M/G/1 queue with vacations in steady state is distributed as the sum  $Q + V$  of two independent random variables, with  $Q$  the number of customers in the ordinary M/G/1 queue in steady state, and  $V$  the stationary number of vacation customers.

The distribution of  $V$  is obtained as follows: If  $K$  is the number of arrivals in a vacation period, then  $P(V = i) = P(K > i)/E(K), i = 0, 1, \dots$ . In our case, the number of arrivals  $K$  in one vacation period is the number of  $\lambda_2$  Poisson arrivals in a single  $\lambda_1$  Poisson interval plus the initial arrival, and this is distributed  $K \sim \text{Geom}_1\left(\frac{\lambda_1}{\lambda_1 + \lambda_2}\right)$ , with

$P(K = i) = \frac{\lambda_1}{\lambda_1 + \lambda_2} \left( \frac{\lambda_2}{\lambda_1 + \lambda_2} \right)^{i-1}$ ,  $i = 1, 2, \dots$ . One then sees immediately that the stationary number of vacation customers  $V$  is distributed  $V \sim \text{Geom}_0 \left( \frac{\lambda_1}{\lambda_1 + \lambda_2} \right)$  with

$$P(V = i) = \frac{\lambda_1}{\lambda_1 + \lambda_2} \left( \frac{\lambda_2}{\lambda_1 + \lambda_2} \right)^i, \quad i = 0, 1, \dots$$

This has generating function  $\chi(Z) = \frac{\lambda_1}{\lambda_1 + \lambda_2(1-Z)}$ .

The generating function for  $Q$ , the ordinary stationary queue length in an M/G/1 system (including the customer in service), for Poisson arrival rate  $\lambda_2$  and service distribution  $G_1$  and rate  $\mu_1$ , is given by:

$$\Theta_1(Z) = \frac{(1 - \lambda_2/\mu_1)(1 - Z)G_1^*(\lambda_2(1 - Z))}{G_1^*(\lambda_2(1 - Z)) - Z}$$

Hence, the M/G/1 server with vacations will have a stationary distribution of customers in steady state (including the one in service) given by the generating function:

$$\Gamma_1(Z) = \frac{\lambda_1}{\lambda_1 + \lambda_2(1 - Z)} \frac{(1 - \lambda_2/\mu_1)(1 - Z)G_1^*(\lambda_2(1 - Z))}{G_1^*(\lambda_2(1 - Z)) - Z}$$

From this we can get the generating function of the steady state number of customers in the system when the queue to machine 1 is not empty. All we need to do is adjust for the different empty periods, that is:

$$\sum_{i=1}^{\infty} P(a, i \cup A, i \mid \text{queue 1 non empty}) Z^i = \frac{\Gamma_1(Z) - \Gamma_1(0)}{1 - \Gamma_1(0)}.$$

Next we note that the  $a$  and the  $b$ -periods have identical stochastic behavior in the M/G/· case and in the M/M/· case, since they only involve the Poisson production processes. Also, the start of each  $a$  or  $b$ -period is a regeneration time of the system in both cases [3, 10, 12]. Hence, our calculations of  $\Pi_a, \Pi_A, \Pi_b, \Pi_B$ , and the partial generating functions for the  $a, b$ -periods,  $S(Z), R(Z)$  are valid also for the M/G/· case.

We can now obtain the partial generating function of the  $A$ -period states, in the M/G/· case, as follows:

$$\begin{aligned} U(Z) &= \sum_{i=1}^{\infty} P(A, i) Z^i = \sum_{i=1}^{\infty} P(a, i \cup A, i) Z^i - S(Z) \\ &= (\Pi_a + \Pi_A) \sum_{i=1}^{\infty} P(a, i \cup A, i \mid \text{queue 1 non empty}) Z^i - S(Z) \\ &= (\Pi_a + \Pi_A) \frac{\Gamma_1(Z) - \Gamma_1(0)}{1 - \Gamma_1(0)} - S(Z) \\ &= \frac{(\mu_2 - \lambda_1)\lambda_1\mu_1}{(\mu_1\mu_2 - \lambda_1\lambda_2)(\lambda_1 + \lambda_2 - \mu_1)} \frac{\Theta_1(Z) - (1 - \lambda_2/\mu_1)}{\lambda_1 + \lambda_2(1 - Z)} \end{aligned}$$

$$\begin{aligned}
&= \Pi_A \frac{\lambda_1 Z}{\lambda_1 + \lambda_2(1-Z)} \frac{\Theta_1(Z) - (1 - \lambda_2/\mu_1)}{Z\lambda_2/\mu_1}, \\
&= \Pi_A \frac{\lambda_1 Z}{\lambda_1 + \lambda_2(1-Z)} \frac{1 - \lambda_2/\mu_1}{\lambda_2/\mu_1} \frac{1 - G_1^*(\lambda_2(1-Z))}{Z - G_1^*(\lambda_2(1-Z))}.
\end{aligned}$$

A similar derivation will yield:

$$V(Z) = \Pi_B \frac{\lambda_2 Z}{\lambda_2 + \lambda_1(1-Z)} \frac{1 - \lambda_1/\mu_2}{\lambda_1/\mu_2} \frac{1 - G_2^*(\lambda_1(1-Z))}{Z - G_2^*(\lambda_1(1-Z))}.$$

Note that  $\frac{\Theta_1(Z) - (1 - \lambda_2/\mu_1)}{Z\lambda_2/\mu_1} = \frac{\Theta_1(Z) - \Theta_1(0)}{Z(1 - \Theta_1(0))}$  is the generating function of the steady state queue length of a standard M/G/1 queue, conditional on the system being non-empty (i.e. the server is busy). Also,  $\frac{\lambda_1 Z}{\lambda_1 + \lambda_2(1-Z)}$  is the generating function of  $V + 1$  which is  $\sim \text{Geom}_1\left(\frac{\lambda_1}{\lambda_1 + \lambda_2}\right)$ . Hence we have obtained, in analogy with the results of the previous section:

$$\begin{aligned}
P(a, i) &= \Pi_a P(X = i), \quad X \sim \text{Geom}_1\left(\frac{\lambda_1}{\lambda_1 + \lambda_2}\right), \\
P(b, i) &= \Pi_b P(Y = i), \quad Y \sim \text{Geom}_1\left(\frac{\lambda_2}{\lambda_1 + \lambda_2}\right), \\
P(A, i) &= \Pi_A P(X + Z = i), \quad Z \sim \text{Stationary Queue} | \text{Nonempty in M/G/1}(\lambda_2, \mu_1), \\
P(B, i) &= \Pi_B P(Y + W = i), \quad W \sim \text{Stationary Queue} | \text{Nonempty in M/G/1}(\lambda_1, \mu_2).
\end{aligned}$$

## 5 Generalization to an $M$ machine system

Assume now there are  $j = 1, \dots, M$  machines, and machine  $j$  is capable of performing a production operation at rate  $\lambda_j$ , where the production times are memoryless, and a service operation at rate  $\mu_j$ , with i.i.d. service times. The system operates as follows:

After some initial time period, at any time exactly one of the machines, say  $j$ , has a non-empty queue, and is not sending work to the other machines, while all the other machines have empty queues and they are producing work for the non-empty machine. Hence the total rate of production into machine  $j$  is  $\Lambda_j = \sum_{i \neq j} \lambda_i$ , where the combined arrivals form a memoryless process.

When the non-empty machine  $j$  empties its queue, it will immediately send a stored job, which it produced before it started to serve its queue, to machine  $k$  according to i.i.d. changeover probability  $p_{jk}$ . From that instant machine  $k$  will become the non-empty machine, and all the remaining machines  $l \neq j, k$  as well as machine  $j$  will now send all their produced jobs to machine  $k$ .

It is immediately seen that the non-empty period of machine  $j$  consists of an initial period when machine  $j$  is producing the job which it will store till the end of the period (this lasts for a duration  $\sim \exp \lambda_j$ ), followed by a period when it serves its queue, at the rate  $\mu_j$ , until it

is empty. Throughout both these two periods, which we will denote  $j, J$ , there will be Poisson arrivals to the queue of machine  $j$ , at rate  $\Lambda_j$ .

For this policy to be stable, we assume the obviously necessary condition  $\Lambda_j < \mu_j$ . Consider now the  $M$  machine system, in a  $j, J$  period in which the queue of machine  $j$  is non-empty. The queues of all the machines  $k \neq j$  are empty, and the queue length process at machine  $j$  behaves exactly as in the two machine push pull system, say in an  $a, A$  period (with the relevant parameters  $\mu_j, \lambda_j, \Lambda_j$ ). The order of these machine busy periods will no longer be cyclic, a  $j, J$  period will now be followed by a randomly chosen  $k, K$  period with probability  $p_{j,k}$ . Let  $\pi_j$  be the steady state probability of  $j$  in the Markov chain with transitions  $p_{j,k}$ . Then the overall long term fraction of  $j, J$  periods will be  $\pi_j$ .

Clearly, the instants at which the non-empty queues change over form an embedded Markov process [3]. Behavior between these time points is analogous to that of the two machine push pull system. Hence the  $M$  machine system is stable. It follows immediately from the results of Section 4 that the steady state partial generating functions of this process are:

$$\begin{aligned} \Pi_j = P(j \text{ states}) &= \pi_j \frac{1}{\lambda_j} / \sum_{k=1}^M \pi_k \frac{\mu_k + \lambda_k}{\lambda_k(\mu_k - \Lambda_k)}, \\ \Pi_J = P(J \text{ states}) &= \pi_j \frac{\lambda_j + \Lambda_j}{\lambda_j(\mu_j - \Lambda_j)} / \sum_{k=1}^M \pi_k \frac{\mu_k + \lambda_k}{\lambda_k(\mu_k - \Lambda_k)}, \\ S_j(Z) = \sum_{i=1}^{\infty} P(j, i) Z^i &= \Pi_j \frac{\lambda_j Z}{\lambda_j + \Lambda_j(1 - Z)}, \\ U_j(Z) = \sum_{i=1}^{\infty} P(J, i) Z^i &= \Pi_J \frac{\lambda_j Z}{\lambda_j + \Lambda_j(1 - Z)} \frac{1 - \Lambda_j/\mu_j}{\Lambda_j/\mu_j} \frac{1 - G_j^*(\Lambda_j(1 - Z))}{Z - G_j^*(\Lambda_j(1 - Z))}, \quad j = 1, \dots, M. \end{aligned}$$

## 6 Discussion

In this paper we have obtained explicit expressions for the steady state behavior of a push pull system under M/M/· and M/G/· assumptions. We believe that our push pull system remains stable under the more general GI/G/· assumptions of i.i.d. non-exponential processing times for all four operations. This should follow by use of fluid model techniques, as in [4, 5], though of course no explicit expressions for any of the steady state parameters can be expected for that case.

The special feature of the push pull system is that it is a stable system which operates with full utilization of all the machines. We believe that this example is just a first simple example of MCQN (multi class queueing networks) which can operate with  $\rho = 1$  and yet not be in heavy traffic regime. Our hope is that such systems could be naturally approximated by fluid models, and these fluid models optimized, and translated into asymptotically optimal stable policies for control of the MCQN. For some indications of these ideas, and related results see [11, 1, 2, 6, 9].

## References

- [1] Bertsimas, D. and Gamarnik, D. (1999) Asymptotically optimal algorithms for job shop scheduling and packet routing. *J. of Algorithms* **33**, 296–318.
- [2] Boudoukh, T., Penn, M. and Weiss, G. (1999) Scheduling jobshops with some identical or similar jobs. *J. of Scheduling* To appear.
- [3] Cinlar, E. (1975). *Introduction to Stochastic Processes*, Prentice Hall, New Jersey.
- [4] Dai, J.G. 1995. On positive Harris recurrence of multi-class queueing networks: a unified approach via fluid limit models. *Annals of Applied Probability* **5**, 49–77.
- [5] Dai, J. G. and Weiss, G. (1994). Stability and Instability of fluid models for re-entrant lines. *Mathematics of Operations Research* **21**, 115–134.
- [6] Dai, J.G., Weiss G. 1999. A fluid heuristic for minimizing makespan in job-shops. *Operations Research* to appear.
- [7] Fuhrman, S.W. and Cooper, R.B. (1985) Stochastic decomposition in the general M/G/1 queue with general vacations *Operations Research* **33**, 1117–1129.
- [8] Levy, Y. and Yechiali, U. (1975) Utilization of idle time in an M/G/1 queueing system *Management Science* **22**, 202–211.
- [9] Nazarathy, Y. and Weiss, G. (2001) A simulation study of fluid imitation multi-class job-shop scheduling heuristics. In preparation.
- [10] Ross, S.M. (1989). *Introduction to Probability Models*. Academic Press, New York.
- [11] Weiss, G. (1999) Scheduling and control of manufacturing systems — a fluid approach *Proceedings of the 37 Allerton Conference, 21–24 September, 1999, Monticello, Illinois*, 577-586.
- [12] Wolff, R.W. (1989). *Stochastic Modeling and the Theory of Queues*. Prentice Hall, New Jersey.