Computer Science Journal of Moldova, vol.**29**, no.1(85), 2021

On a modified M/M/m/n queueing model *

Mario Lefebvre

Abstract

The classic M/M/m/n queueing model is modified by allowing a given task to require up to m servers to be performed. Moreover, the maximum time that a task can wait in the queue before being executed is a random variable having an exponential distribution. Both FIFO (First In, First Out) and priority disciplines are considered. The case when $m = 2$ is treated: the state space needed to fully describe the system is given, its size is calculated and the balance equations are presented when $n = 1, 2$ and 3. The queueing process can be used to model cluster-type multiprocessor computing systems.

Keywords: State space, balance equations, queue discipline, priority, cluster-type multiprocessor computing systems.

MSC 2010: 68M20, 60K25, 60K30, 90B22.

1 Introduction

We consider a system consisting of m (≥ 1) servers. For instance, it could be a computing system having m processors. Tasks arrive at the system according to a Poisson process with rate λ. For every server, the service time is an exponential random variable with mean μ. The times between the arrivals and the service times are assumed to be independent random variables. The capacity of the system is n (≥ 1). Finally, the queue discipline is FIFO (First In, First Out). Therefore, using Kendall's notation, the system can be denoted by M/M/m/n.

* This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC)

In a recent paper, Vardanyan and Sahakyan [5] (see also [4]) proposed the above model for cluster-type multiprocessor computing systems, with the following modifications: firstly, any task can require up to m servers to be executed. More precisely, the number of servers required to perform a task is a random variable K having a discrete uniform distribution over the set $\{1, 2, \ldots, m\}$. Secondly, the maximum time Ω that a task can wait in the queue before being executed is an exponentially distributed random variable with parameter ω.

The authors defined the states (i, j), where i is the number of tasks being serviced and j is the number of tasks waiting for service in the queue at a given time instant. They assumed that $0 \leq i \leq m$ and $0 \leq j \leq n$. There are $m \times (n+1) + 1$ possible states in the state space (since $i = 0$ implies that j is also equal to zero). Notice that n is taken to be the capacity of the *waiting queue*, rather than that of the system in their paper.

Remark 1.1. *(i) The authors of [5] actually wrote that there are* $m \times n + 1$ *possible states. But if* $m = n = 1$*, the possible states are* $(0, 0)$*,* $(1, 0)$ *and* $(1, 1)$*. Thus,* $3 = 1 \times (1 + 1) + 1$*.*
(ii) For the classic M/M/m/n model, $m = n$ *means that there is no waiting space, and the system is a so-called* loss system*. Here, because any task can require more than one server (if* $m > 1$*), the incoming tasks will only be lost (on arrival) when the system is in state* (m, m)*, so that all the* (m) *servers are servicing a different task. In particular, all the servers could be servicing the same task, and there could be* $m-1$ $(= n-1)$ *tasks waiting for service in the queue. We could also assume, as in [5], that* n *is the capacity of the waiting queue, so that the case* $n = 0$ *would indeed correspond to a loss system. Finally, note that there is no waiting if* $n < m$*.*

In [5], the authors computed the limiting probabilities of the system. However, their definition of the various states does not fully describe the different possible states of the system. Indeed, the value of i only gives us the *number of tasks* being serviced, so that the number of servers working on each task is unknown. With this important addi-

tional information, the number of possible states is much larger than $m \times (n+1)+1$.

In this note, we first generalise the model considered by Vardanyan and Sahakyan in [5]: the service time is an exponential random variable with mean μ_k, for $k = 1, 2, \ldots, m$. That is, the service time may depend on the server. Moreover, if two or more servers are working on a task, the service time may also be different. Next, instead of assuming that the random variable K has a discrete uniform distribution over the set $\{1, 2, \ldots, m\}$, we denote the probability $P[K = k]$ by $p_k \in [0, 1]$ for $k = 1, \ldots, m$. Finally, the maximum time Ω_k that a task requiring k servers can wait in the queue before being executed is an exponential random variable with parameter ω_k.

Remark 1.2. *(i) Contrary to the definition in [5], here the system capacity, n, is the maximum total number of tasks in the system at any one time; that is, the number of tasks being serviced or awaiting service. (ii) When a task requiring the two servers is completed and the first two tasks in the queue only require one server each, then they will both go into service at the same time.*

This problem was also considered by Green [1], but with different assumptions: firstly, $n = \infty$; secondly, there is no maximum time that a task can wait for service in the queue; thirdly, the mean service time is equal to $1/\mu$ for every server; finally, and more importantly, servers working together on a given task complete service at independent times, so that they do not end service together. It follows that the service times are not exponentially distributed random variables; instead, they are distributed as the maximum of a random number of exponentially distributed random variables. Consequently, the stochastic process $\{X(t), t \geq 0\}$, where $X(t)$ is the state of the system at time t, is not a continuous-time Markov chain.

In the next section, the case when $m = 2$ will be treated. We will give a state space that is exhaustive, as well as the full balance equations in simple cases. Then, the queue discipline will be modified: instead of the FIFO discipline, we will consider priority disciplines. Again, the balance equations will be presented in a simple case.

2 A generalised model

We consider the M/M/m/n queueing model with the modifications mentioned in the previous section. The tasks are generated by a Poisson process with rate λ, and the probability that a given task will require k servers to be performed is $P[K = k] = p_k \in [0, 1]$ for $k = 1, \ldots, m$, with $p_1 + \ldots + p_m = 1$. It follows that tasks requiring k servers arrive in the system according to a Poisson process with rate $\lambda_k := p_k \lambda$. We could instead assume that the various tasks are generated by m independent Poisson processes with rates $\lambda_1, \ldots, \lambda_m$.

Because (with the FIFO discipline) a task requiring more than one server can prevent a task behind it in the waiting queue to enter service, even if a server is free, we need to determine the type of each task in the system. Therefore, the number of possible states is very large. For this reason, we will limit ourselves in this note to the case when there are two servers. The results obtained below can be generalised to any value of m.

Thus, we assume that the queueing model is an M/M/2/n queue, where $n \geq 1$ is the capacity of the system. Moreover, an arriving task requires one server with probability $p_1 \in [0, 1]$ and two servers with probability $p_2 = 1 - p_1 \in [0, 1]$. The two servers do not necessarily execute the various tasks at the same rate: the service time for server i is an exponential random variable with parameter μ_i, for $i = 1, 2$. When the system is empty, an arriving (or waiting) task requiring only one server (which we call a type 1 task) will be executed by server 1 or 2 with respective probabilities $r_1 \in [0, 1]$ and $r_2 = 1 - r_1$. Furthermore, when a task requires the two servers, the service time becomes an exponential random variable with parameter μ_3. It is indeed more realistic to suppose that the service time depends both on the server and on the type of the task. Finally, tasks requiring i server(s) can spend an exponential amount of time with parameter ω_i, for $i = 1, 2$, before entering service. Note that we assume that once a task is taken into service, it will remain in the system until it has been fully completed.

Let us start with the case when $n = 1$. Then, as mentioned above, there is no waiting. Even if a server is free, if the other server is

occupied then an incoming task will be rejected. In this case, which is the simplest possible, there are four different states of the system:

0: the system is empty
1: server 1 is executing a task, server 2 is free
2: server 2 is executing a task, server 1 is free
3: servers 1 and 2 are executing a type 2 task

Let π_j denote the limiting probability that the system will be in state j, for $j = 0, 1, 2, 3$. Note that because the number of states is finite, the limiting probabilities do exist. The balance equations of the system (see Ross [3] or Lefebvre [2]) are the following:

State j	Departure rate from j	=	Arrival rate to j
0	$\lambda\pi_0$	=	$\mu_1\pi_1 + \mu_2\pi_2 + \mu_3\pi_3$
1	$\mu_1\pi_1$	=	$r_1 p_1 \lambda\pi_0$
2	$\mu_2\pi_2$	=	$r_2 p_1 \lambda\pi_0$
3	$\mu_3\pi_3$	=	$p_2\lambda\pi_0$

To obtain the limiting probabilities, we can solve the balance equations, together with the condition $\sum_{j=0}^{3}\pi_j = 1$. Notice that we have four equations in four unknowns, plus the preceding condition. Therefore, we can drop one of the balance equations. Once the limiting probabilities have been computed, we should check that they also satisfy the equation that was neglected. We easily find that

$$\begin{aligned}
\pi_0 &= \left\{1 + \frac{r_1 p_1 \lambda}{\mu_1} + \frac{r_2 p_1 \lambda}{\mu_2} + \frac{p_2\lambda}{\mu_3}\right\}^{-1}, \\
\pi_1 &= \frac{r_1 p_1 \lambda}{\mu_1}\pi_0, \quad \pi_2 = \frac{r_2 p_1 \lambda}{\mu_2}\pi_0 \quad \text{and} \quad \pi_3 = \frac{p_2\lambda}{\mu_3}\pi_0.
\end{aligned}$$

When $n = 2$, in addition to the above defined states (for which we

must now mention that the waiting queue is empty), we have:

4: servers 1 and 2 are executing type 1 tasks
5: server 1 is executing a task, server 2 is free, a type 2 task is waiting for service
6: server 2 is executing a task, server 1 is free, a type 2 task is waiting for service
7: servers 1 and 2 are executing a type 2 task, a type 1 task is waiting for service
8: servers 1 and 2 are executing a type 2 task, a type 2 task is waiting for service

Thus, there are now nine possible states. The balance equations of the system become

State j	Departure rate	=	Arrival rate
0	$\lambda\pi_0$	=	$\mu_1\pi_1 + \mu_2\pi_2 + \mu_3\pi_3$
1	$(\mu_1+\lambda)\pi_1$	=	$r_1 p_1 \lambda\pi_0 + \mu_2\pi_4 + \omega_2\pi_5 + \mu_3 r_1\pi_7$
2	$(\mu_2+\lambda)\pi_2$	=	$r_2 p_1 \lambda\pi_0 + \mu_1\pi_4 + \omega_2\pi_6 + \mu_3 r_2\pi_7$
3	$(\mu_3+\lambda)\pi_3$	=	$p_2\lambda\pi_0 + \mu_1\pi_5 + \mu_2\pi_6 + \omega_1\pi_7 + (\mu_3+\omega_2)\pi_8$
4	$(\mu_1+\mu_2)\pi_4$	=	$p_1\lambda(\pi_1+\pi_2)$
5	$(\mu_1+\omega_2)\pi_5$	=	$p_2\lambda\pi_1$
6	$(\mu_2+\omega_2)\pi_6$	=	$p_2\lambda\pi_2$
7	$(\mu_3+\omega_1)\pi_7$	=	$p_1\lambda\pi_3$
8	$(\mu_3+\omega_2)\pi_8$	=	$p_2\lambda\pi_3$

Next, when $n = 3$, we can add a waiting task to the states 4 to 8 of the system. This waiting task can be either a type 1 or type 2 task. It follows that the number of possible states increases to $9 + 5 \times 2 = 19$. In the general case $n \geq 1$, we have the following proposition.

Proposition 2.1. *The size of the state space of the modified M/M/2/n queueing model considered in this note is given by* $4 + 5(2^{n-1} - 1)$ *for* $n \geq 1$.

Proof. We can use mathematical induction. The result holds true for $n = 1$ (as well as $n = 2$ and $n = 3$). Assume that the formula is valid for a fixed $n \geq 2$. Then, for $n + 1$ the number of states is obtained by adding any waiting task to the states that correspond to a full system when the capacity is n. There are

$$[4 + 5(2^{n-1} - 1)] - [4 + 5(2^{n-2} - 1)] = 5 \times 2^{n-2}$$

such states. Hence, we add $5 \times 2^{n-1}$ states to the state space. It implies that the total number of states is

$$4 + 5(2^{n-1} - 1) + 5 \times 2^{n-1} = 4 + 5(2^n - 1),$$

which proves the result. □

Remark 2.1. *(i) If we assume that n is the capacity of the waiting queue, rather than that of the system, then we find that the number of possible states is $5+6(2^n-1)$ for $n \geq 1$, compared to $2(n+1)+1 = 2n+3$ for the problem considered in [5].*
(ii) Maple *is able to solve the nine balance equations of the system when $n = 2$. However, the solution it provides is very involved. In the particular case when $p_1 = 3/4$, $r_1 = 1/2$, $\lambda = 5$, $\mu_1 = 2$, $\mu_2 = 2$, $\mu_3 = 3$ and $\omega_1 = \omega_2 = 0$, the solution is*

$$\begin{aligned} &\pi_0 \simeq 0.1469, \quad \pi_1 \simeq 0.1277, \quad \pi_2 \simeq 0.1277, \\ &\pi_3 \simeq 0.0745, \quad \pi_4 \simeq 0.2394, \quad \pi_5 \simeq 0.0798, \\ &\pi_6 \simeq 0.0798, \quad \pi_7 \simeq 0.0931, \quad \pi_8 \simeq 0.0310. \end{aligned}$$

We see that, in this example, state 4 is by far the most likely state in stationary regime. Moreover, $\pi_1 = \pi_2$ and $\pi_5 = \pi_6$, which follows from the fact that $\mu_1 = \mu_2$ and $r_1 = 1/2$.

Now for $n \geq 3$, with the FIFO discipline, sometimes a server is free while a type 1 task is in the waiting queue, but behind a type 2 task that is blocking access to the free server. If we then allow the type 1

task to get serviced, it will reduce the average time that type 1 tasks spend in the system and conversely increase this average time for type 2 tasks. When m and n are large, a task requiring m servers could block the system for a long time. It is therefore worth examining the effect of giving priority to type 1 or to type 2 tasks in our case.

Let us first assume that type 2 tasks always have priority over type 1 tasks in the waiting queue. We can consider the general case when $n \geq 1$. We will calculate the number of states needed to fully describe the state of the system. Because our aim is to determine the effect of giving priority to a type of task, we could assume that $\mu_1 = \mu_2$.

If $n = 1$, priority does not make any difference, because there is no waiting. Similarly, if $n = 2$, then there is at most one task in the waiting queue, which again implies that priority is irrelevant. Let $n \geq 3$. There are five possible states when the waiting queue is empty, that we denote by $0, 1, 2, 3$ and 4 as defined above. Then, the other states of the system can be characterised by a triple (s, n_2, n_1), where $s \in \{1, 2, 3, 4\}$ and n_i is the number of type i tasks in the waiting queue, for $i = 1, 2$. We assume that a server can be free even if there is at least one type 1 task in the waiting queue. However, we cannot have triples of the form $(s, 0, n_1)$ with $s \in \{1, 2\}$ and $n_1 > 0$.

When $s = 1$ or 2, we can write that $n_2 \geq 1$ and $n_1 + n_2 \leq n - 1$. The number of states is given by

$$\underbrace{n-1}_{n_2=1} + \underbrace{n-2}_{n_2=2} + \cdots + \underbrace{1}_{n_2=n-1} = \frac{(n-1)n}{2}.$$

If $s = 3$, we have $n - 1$ additional states of the form $(3, 0, n_1)$, with $1 \leq n_1 \leq n - 1$, so that the number of states is

$$\frac{(n-1)n}{2} + (n-1) = (n-1)\frac{n+2}{2}.$$

Finally, the number of states of the form $(4, n_2, n_1)$ is obtained by replacing n by $n-1$ in the above formula: $(n-2)(n+1)/2$. Therefore, we can state the following proposition.

Proposition 2.2. *The size of the state space of the queueing model considered in this note, when type 2 tasks have priority over type 1*

tasks in the waiting queue, is given by

$$5 + 2\frac{(n-1)n}{2} + (n-1)\frac{n+2}{2} + (n-2)\frac{n+1}{2} = 2n^2 - n + 3$$

for $n \geq 3$.

Remark 2.2. *(i) The formula is actually valid for* $n \geq 1$.

(ii) If we assume that $\mu_1 = \mu_2$, *then there is no need to distinguish between* 1 *and* 2, *and between* $(1, n_2, n_1)$ *and* $(2, n_2, n_1)$. *The number of states is reduced to*

$$4 + \frac{(n-1)n}{2} + (n-1)\frac{n+2}{2} + (n-2)\frac{n+1}{2} = \frac{3n^2 - n + 4}{2}.$$

(iii) Compared with the FIFO discipline, the size of the state space is greatly reduced when n *is relatively large: for* $n = 3$, *it goes from 19 to 18, but from 639 to 123 if* $n = 8$.

(iv) When type 1 tasks have priority over type 2 tasks in the waiting queue, we have the states $0, 1, 2, 3$ *and* 4 *as above, and triples of the form* (s, n_1, n_2), *where* $s \in \{1, 2, 3, 4\}$. *We cannot have states of the form* $(1, n_1, n_2)$ *or* $(2, n_1, n_2)$ *with* $n_1 > 0$. *For* $s = 3$ *and* $s = 4$, *the number of different states is the same as when type 2 tasks have priority. It follows that the total number of states is*

$$5 + 2(n-1) + (n-1)\frac{n+2}{2} + (n-2)\frac{n+1}{2} = n^2 + 2n + 1$$

for $n \geq 3$ *(*$n \geq 1$, *actually). If* $\mu_1 = \mu_2$, *we can reduce the number of states to*

$$4 + (n-1) + (n-1)\frac{n+2}{2} + (n-2)\frac{n+1}{2} = n^2 + n + 1.$$

We give next the balance equations when $n = 3$ and $\mu_1 = \mu_2$, so that there are 14 states. If we assume further, for the sake of simplicity, that

$\omega_1 = \omega_2 = 0$, then the system of equations is obviously much simpler.

State	Departure rate	=	Arrival rate
0	$\lambda\pi_0$	=	$\mu_1\pi_1 + \mu_3\pi_3$
$1\,(=2)$	$(\lambda+\mu_1)\pi_1$	=	$p_1\lambda\pi_0 + 2\mu_1\pi_4 + \omega_2\pi_5 + \mu_3\pi_8$
3	$(\lambda+\mu_3)\pi_3$	=	$p_2\lambda\pi_0 + \mu_1\pi_5 + \omega_1\pi_8 + (\mu_3+\omega_2)\pi_9$
4	$(\lambda+2\mu_1)\pi_4$	=	$p_1\lambda\pi_1 + \omega_2(\pi_7+\pi_{14}) + \mu_3\pi_{11} + (2\mu_1+\omega_1)\pi_{13}$
$(1,1,0) := 5$	$(\lambda+\mu_1+\omega_2)\pi_5$	=	$p_2\lambda\pi_1 + 2\omega_2\pi_6 + \omega_1\pi_7 + 2\mu_1\pi_{14}$
$(1,2,0) := 6$	$(\mu_1+2\omega_2)\pi_6$	=	$p_2\lambda\pi_5$
$(1,1,1) := 7$	$(\mu_1+\omega_1+\omega_2)\pi_7$	=	$p_1\lambda\pi_5$
$(3,0,1) := 8$	$(\lambda+\mu_3+\omega_1)\pi_8$	=	$p_1\lambda\pi_3 + \mu_1\pi_7 + 2\omega_1\pi_{11} + (\mu_3+\omega_2)\pi_{12}$
$(3,1,0) := 9$	$(\lambda+\mu_3+\omega_2)\pi_9$	=	$p_2\lambda\pi_3 + \mu_1\pi_6 + (\mu_3+2\omega_2)\pi_{10} + \omega_1\pi_{12}$
$(3,2,0) := 10$	$(\mu_3+2\omega_2)\pi_{10}$	=	$p_2\lambda\pi_9$
$(3,0,2) := 11$	$(\mu_3+2\omega_1)\pi_{11}$	=	$p_1\lambda\pi_8$
$(3,1,1) := 12$	$(\mu_3+\omega_1+\omega_2)\pi_{12}$	=	$p_2\lambda\pi_8 + p_1\lambda\pi_9$
$(4,0,1) := 13$	$(2\mu_1+\omega_1)\pi_{13}$	=	$p_1\lambda\pi_4$
$(4,1,0) := 14$	$(2\mu_1+\omega_2)\pi_{14}$	=	$p_2\lambda\pi_4$

3 Concluding remarks

In this note, a modified M/M/m/n queueing model that can be used for cluster-type multiprocessor computing systems was considered. The main modification is that any task can require up to m servers to be executed. Moreover, each task can spend a random (exponentially distributed) maximum amount of time in the waiting queue before being taken into service. The default queue discipline is FIFO.

We saw that to fully describe the state of the system, we need a very large number of states. For this reason, we limited ourselves to the case

when $m = 2$. There are then two types of tasks: type 1 (respectively 2) tasks require only one (respectively two) servers. If priority in the waiting queue is given to either type 1 or type 2 tasks, then the size of the state space is greatly reduced, at least when n is relatively large. We presented the detailed balance equations in simple cases.

In any particular case, if n is small enough, we can use a mathematical software such as *Maple* to solve the balance equations, which are a system of linear equations. For n relatively large, if the number of states is too large to obtain precise numerical solutions, then one could perhaps use simulation to estimate the various limiting probabilities.

Finally, an important question is whether one should use the FIFO discipline or a priority discipline, particularly because in the general case, with m large enough, a task requiring the m servers could block the system for a large amount of time. The answer to this question depends on the aim of the system administrators. Do they prioritise the average number of tasks being serviced, or the average time spent in the system for each (or a particular) type of task? Another important criterion is the percentage of rejected tasks.

Acknowledgements

The author wishes to thank the anonymous reviewer of this note for his/her constructive comments.

References

[1] L. Green, "A queueing system in which customers require a random number of servers," *Operations Research*, vol. 28, no. 6, pp. 1335–1346, 1980.

[2] M. Lefebvre, *Applied Stochastic Processes,* New York, NY: Springer, 2007, 382 p.

[3] S. M. Ross, *Introduction to Probability Models,* 12th ed., Amsterdam: Elsevier/Academic Press, 2019, 826 p.

[4] V. G. Sahakyan, Y. H. Shoukourian, and H. V. Astsatryan, "About some queueing models for computational grid systems," *Mathematical Problems of Computer Science*, vol. 46, pp. 55–58, 2016.

[5] A. P. Vardanyan and V. G. Sahakyan, "The queue distribution in multiprocessor systems with the waiting time restriction," *Mathematical Problems of Computer Science*, vol. 51, pp. 82–89, 2019.

Mario Lefebvre

Received November 25, 2020
Accepted March 11, 2021

Polytechnique Montréal
Department of Mathematics and Industrial Engineering
2500, chemin de Polytechnique
Montréal (Québec), Canada H3T 1J4
E–mail: mlefebvre@polymtl.ca