Analysis of M/M/n/K Queues with Priorities

Finite and Infinite Buffers

We outline here an approach that may be used to analyse a M/M/n/K queue ($n=1,2,.....$ and $K=n,n+1,....,$ ¥) which has job classes of multiple priorities. The priority discipline followed may be either non-preemptive or preemptive in nature. When the priority discipline is *non-preemptive* in nature, a job in service is allowed to complete its service normally even if a job of higher priority enters the queue while its service is going on. In the *preemptive* case, the service to the ongoing job will be preempted by the new arrival of higher priority. If the priority discipline is *preemptive resume*, then service to the interrupted job, when it restarts, continues from the point at which the service was interrupted. For the *preeptive nonresume* case, service already provided to the interrupted job is forgotten and its service is started again from the beginning.

Note that there may be loss of work in the preemptive non-resume priority case. Such loss of work will not happen in the case of the other two priorities. Since the service times are assumed to be exponentially distributed, they will satisfy the memory-less property and that, therefore, the results will be the same both for the preemptive resume and preemptive non-resume cases.

For a P priority system, we consider jobs of class 1 to be the lowest priority and jobs of class P to be the highest priority. The job arrivals for class i come from a Poisson process at rate l_i. When a job of this class gets served by any of the servers in the queue, the service time is an exponentially distributed random variable with mean m^{-1}. The traffic of priority class i offered to the queue is denoted by $r = l/m$.

We consider here queues with both preemptive and non-preemptive priority disciplines. The approach suggested may be applied to both single-server and multi-server queues. Moreover, queues with both finite and

infinite buffers may be analysed using the suggested approach. In this section, we do not actually do a complete derivation of the state probabilities of a M/M/n/K queue. Instead, we outline the way in which such a queue may actually be analysed by solving its balance equations.

As is usual with any M/M/-/- type queue, the basic approach to its solution is to define the system-states appropriately so that a state transition diagram of the queue may be drawn. Using the exponential (memory-less) distribution of the inter-arrival and service times, the probability flows in the state transition diagram are identified. The balance equations are then obtained by choosing proper closed boundaries and equating the flow across each such boundary. These balance equations are then solved along with the normalisation condition (i.e. sum of the probabilities of all states equal to unity) to obtain the state probabilities. Once these have been obtained, the performance parameters of interest may be suitably calculated. In the following, we consider the cases of preemptive and non-preemptive priorities separately. For each case, we first give the analytical approach for the M/M/1 case and then show how it may be used to handle the M/M/1/K queue and the M/M/n/K queue.

1.1.1 M/M/-/- Queue with Premptive Priority

We first consider the simple M/M/1 queue with infinite buffers. This queue may be analysed by drawing a state transition diagram with the states given by the P-tuple (n_1,.....,n_P) where n_i is the number in the system of priority class i. Note that the server will always be engaged by a job of the highest priority class present in the system, i.e. by a job of class j with service rate μ_j if $n_j \geq 1$ and $n_{j+1}=.....=n_P=0$.

The state transition diagram for a 2-priority M/M/1 queue with infinite buffers operating with a preemptive priority discipline is shown in Fig. 1. Since all the flows are known, the equilibrium probabilities of each of the states may be found by writing and solving the corresponding balance equations with the normalisation condition. Some of the balance equations for this case are given below as an example.

$$
\begin{aligned}
&p_0(\lambda_1+\lambda_2) = p_{0,1}\mu_2 + p_{1,0}\mu_1 \\
&p_{0,1}(\lambda_1+\lambda_2+\mu_2) = p_{0,2}\mu_2 + p_0\lambda_2 \\
&p_{1,0}(\lambda_1+\lambda_2+\mu_1) = p_{2,0}\mu_1 + p_{1,1}\mu_2 + p_0\lambda_1 \qquad (1)\\
&p_{1,1}(\lambda_1+\lambda_2+\mu_2) = p_{1,0}\lambda_2 + p_{1,2}\mu_2 + p_{0,1}\lambda_1 \\
&.....
\end{aligned}
$$

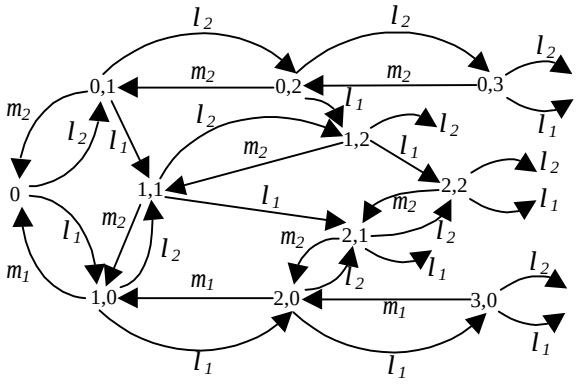

Figure 1. State Transition Diagram for 2-Priority M/M/1 Queue with Preemptive Priority

Since the rules for transition between the states may be stated logically, we can actually use a computer program to generate the actual balance equations. These equations may then be given to an appropriate mathematical package to get the state probabilities. For the infinite buffer case, we may have some computational difficulties because of the infinite number of states. If this happens to be the case, then a convenient approximation would be to limit the state space to some large values of the total number in the queue and solve this approximate model instead.

This would also actually be the way one can solve a M/M/-/- preemptive priority system with finite buffers. As an example, consider the 2-priority M/M/1/3 queue with preemptive priority. In this case, the state transition diagram will be as shown in Fig. 2. Note that unlike Fig. 1, Fig. 2 shows *all* the states of the system. Moreover, the circled states are the ones where the system is full and any jobs arriving in these states will be lost.

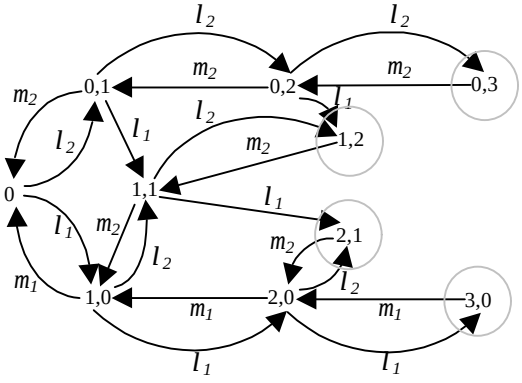

Figure 2. State Transition Diagram for a 2-Priority M/M/1/3 Queue with Preemptive Priority

The following balance equations may be written for this M/M/1/3 system.

$$
\begin{aligned}
p_0(l_1 + l_2) &= p_{0,1}m_2 + p_{1,0}m_1 \\
p_{0,1}(l_1 + l_2 + m_2) &= p_{0,2}m_2 + p_0 l_2 \\
p_{1,0}(l_1 + l_2 + m_1) &= p_{2,0}m_1 + p_{1,1}m_2 + p_0 l_1 \\
p_{1,1}(l_1 + l_2 + m_2) &= p_{1,0} l_2 + p_{1,2}m_2 + p_{0,1} l_1 \\
p_{1,2}m_2 &= p_{1,1} l_2 + p_{0,2} l_1 \\
p_{2,1}m_2 &= p_{1,1} l_1 + p_{2,0} l_2 \\
p_{0,2}(l_1 + l_2 + m_2) &= p_{0,1} l_2 + p_{0,3}m_2 \\
p_{2,0}(l_1 + l_2 + m_1) &= p_{1,0} l_1 + p_{2,1}m_2 + p_{3,0}m_1 \\
p_{0,3}m_2 &= p_{0,2} l_2 \\
p_{3,0}m_1 &= p_{2,0} l_1
\end{aligned}
\qquad (2)
$$

with the normalisation equation as

$$
\begin{aligned}
p_0 + p_{0,1} + p_{1,0} + p_{1,1} + p_{0,2} + p_{2,0} + p_{1,2} + p_{2,1} \\
p_{0,3} + p_{3,0} = 1
\end{aligned}
\qquad (3)
$$

To solve for the state probabilities in this case, we need any nine of the ten equations of (2) and the normalisation condition. Note that in this case, the job loss probability (or the blocking probability) will be $p_{1,2}+p_{2,1}+p_{3,0}+p_{0,3}$.

If the M/M/-/- preemptive multi-priority queue has more than one server, then that can be taken into account while drawing the state transition diagram. Specifically, consider the situation where there are n_P jobs of priority class P and c servers in the system. If $c£ n_P$ then all the servers will be taken by c of the n_P (class P) jobs and no free servers will be available for lower priority jobs (if any). In this case, the overall service rate will be cm_P. On the other hand, if $c>n_P$, then the n_P high priority jobs will each get a server and their overall service rate will be $n_P m_P$. The remaining $(c-n_P)$ servers will be available for the jobs of priority class P-1 and lower and will be used by these jobs according to the number in these classes. This approach will allow us to define the flows between each of the states in the state transition diagram. Note that the representation of states in the state transition diagram may still be done as the P-tuple $(n_1,.....,n_P)$ where n_i is the number in the system of priority class i. Since the priority discipline is preemptive, we do not need to separately specify the priority classes of the jobs engaging the servers. An example of this approach for obtaining the

state transition diagram of a M/M/2/3 queue is shown in Fig. 3. Using this, one can follow the same solution approach as the one outlined for the M/M/1/3 queue earlier to obtain the various state probabilities.

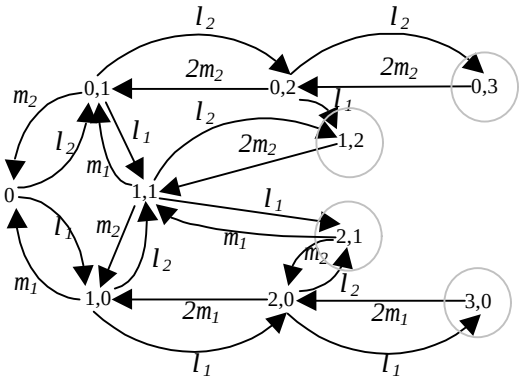

Figure 3. State Transition Diagram for a Preemptive 2-Priority M/M/2/3 Queue

1.1.2 M/M/-/- Queue with Non-Premptive Priority

The basic approach to analysing a M/M/-/- queue with a non-preemptive priority discipline would be fairly similar to the approach suggested for the case where the priority discipline is preemptive in nature. The state definition would need to be modified in this case in order to describe the job class currently being served for each of the servers. This is required, as unlike the M/M/-/- queue with preemptive priority, the ongoing service (if any) is not interrupted in the non-preemptive case because of the arrival of a job of a higher priority class than the one currently being served at a server (when no free servers are available).

For a M/M/c/K queue with P priority classes of jobs, a possible state descriptor would be a (P+c)-tuple of the following form.

State Representation $(n_1, ..., n_P, s_1, ..., s_c)$
where n_j = number of jobs of class j in the system $j=0,1,....,P$
s_k = priority class of the service currently on-going at server k
$k=1,.....,c$
Note that $n_1+.....+n_P \pounds K$ for a finite capacity system

An alternative representation using a 2P-tuple may give a more compact representation if the number of priority classes is less than the number of

servers. In this alternative approach, the state description will be of the following form.

State Representation $(n_1, ..., n_P, s_1, ..., s_P)$
where n_j = number of jobs of class j in the system $j=0,1,....,P$
s_k = number of servers currently busy serving jobs of priority class k, $k=1,.....,P$

As an example, we consider using the first method of representation to represent the states of a M/M/1/3 2-priority queue following a non-preemptive priority discipline. The state diagram for this queue is shown in Fig. 4.

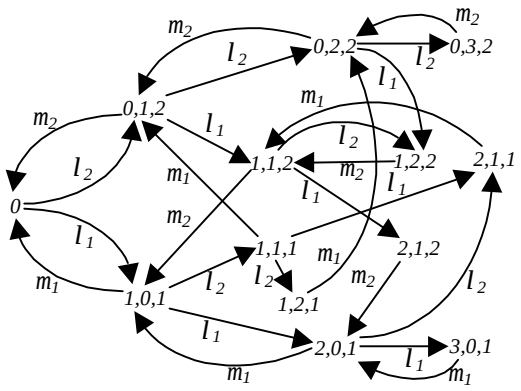

Figure 4. State Transition Diagram for a 2-Priority, Non-Preemptive M/M/1/3 Queue

For this, we can write balance equations in the same way as done earlier and in Chapter 2. These are given in (4).

$$
\begin{aligned}
p_0(l_1 + l_2) &= p_{0,1,2}m_2 + p_{1,0,1}m_1 \\
p_{0,1,2}(l_1 + l_2 + m_2) &= p_0 l_2 + p_{0,2,2}m_2 \\
p_{1,0,1}(l_1 + l_2 + m_1) &= p_0 l_1 + p_{2,0,1}m_1 + p_{1,1,2}m_2 \\
p_{1,1,1}(l_1 + l_2 + m_1) &= p_{1,0,1} l_2 \\
p_{1,1,2}(l_1 + l_2 + m_2) &= p_{0,1,2} l_1 + p_{1,2,2}m_2 + p_{2,1,1}m_1 \\
p_{0,2,2}(l_1 + l_2 + m_2) &= p_{0,1,2} l_2 + p_{1,2,1}m_1 + p_{0,3,2}m_2 \\
p_{2,0,1}(l_1 + l_2 + m_1) &= p_{1,0,1} l_1 + p_{2,1,2}m_2 + p_{3,0,1}m_1
\end{aligned}
\qquad (4)
$$

$$p_{1,2,2}\mu_2 = p_{1,1,2}\lambda_2 + p_{0,2,2}\lambda_1$$
$$p_{2,1,2}\mu_2 = p_{1,1,2}\lambda_1$$
$$p_{2,1,1}\mu_1 = p_{1,1,1}\lambda_1 + p_{2,0,1}\lambda_2$$
$$p_{1,2,1}\mu_1 = p_{1,1,1}\lambda_2$$
$$p_{0,3,2}\mu_2 = p_{0,2,2}\lambda_2$$
$$p_{3,0,1}\mu_1 = p_{2,0,1}\lambda_1$$

As for similar systems solved in Chapter 2, we would need any twelve of the thirteen equations of (4) along with the normalisation condition

$$p_0 + p_{1,0,1} + p_{0,1,2} + p_{0,2,2} + p_{0,3,2} + p_{2,0,1} + p_{3,0,1} + p_{1,1,1} + p_{1,1,2} + p_{1,2,1} + p_{2,1,1} + p_{1,2,2} + p_{2,1,1} = 1 \quad (5)$$

to obtain the actual state probabilities. Once these have been calculated, one can use them to obtain the usual queue performance parameters as in Chapter 2. Note that, in this case, the loss probability will be given by ($p_{0,3,2}$ + $p_{3,0,1}$ + $p_{1,2,2}$ + $p_{2,1,2}$ + $p_{1,2,1}$ + $p_{2,1,1}$) as the sum of the probability of all those states where the system is full, i.e. the total number in the system is three for this M/M/1/3 queue.

As for the case of the M/M/-/- queue with preemptive priority classes, the approach given above for the non-preemptive M/M/1/3 queue may also be generalised. Using the same approach, one can also analyse queues where

(a) the number of priority classes are more than two
(b) the buffer capacity is infinite
(c) the queue has more than one server.

Note that the complexity of the approach suggested here is basically because of the large number of states in the state transition diagram and the resultant complexity in the balance equations. Since the rules for transitions between states are straightforward, one can actually use a computer program to identify the transitions between states and directly write the corresponding balance equations. This set of equations, even if it is large, may be directly solved by using an appropriate numerical package to get the actual state probabilities. These may then be used to obtain the queue performance parameters required.