Convolution with Load-Dependent Service Rates*

Bart Sinclair

This work is produced by OpenStax-CNX and licensed under the
Creative Commons Attribution License 1.0†

The Convolution Algorithm presented in the module on convolution is only applicable to networks of queues with service rates that do not depend on the number of jobs in the queues. We can extend this basic algorithm to incorporate queues with load-dependent service rates as well.

For any product-form queueing network with M queues, the probability of being in a state with n_m jobs in queue m is given by

$$\pi_{n_1 n_2 \ldots n_M} = \frac{\pi_1(n_1)\,\pi_2(n_2)\cdots\pi_M(n_M)}{K} \tag{1}$$

where $\pi_m(n_m)$ is the probability of n_m jobs in queue m if the queue were in isolation with a Poisson input process, and K is a normalization constant. In the case of an open network, K is always 1. For closed networks, K is determined by the constraint that the state probabilities sum to 1; i.e.,

$$K = \sum_{\text{all states}\in\left\{n'_1,n'_2,\ldots,n'_M\right\},\left(n'_M\geq 0\right)\ \wedge\ (1\leq m\leq M)\ \wedge\ \left(\sum_{m=1}^{M} n'_m = N\right)} \pi_1(n'_1)\,\pi_2(n'_2)\cdots\pi_M(n'_M) \tag{2}$$

with N being the total number of jobs in the system.

The state probability distributions $\{\pi_m(n_m)\}$ are the solutions to birth-and-death processes with arrival rates $\{\lambda_m\}$ and possibly load-dependent service rates $\{\mu_m(n_m)\}$; that is,

$$\begin{aligned} \pi_m(n_m) &= \pi_m(0)\prod_{j=0}^{n_m-1}\frac{\lambda_m}{\mu_m(j+1)} \\ &= \pi_m(0)\prod_{j=1}^{n_m}\frac{\lambda_m}{\mu_m(j)} \end{aligned} \tag{3}$$

Define

$$A_m(n) = \prod_{k=1}^{n}\frac{\mu_m(k)}{\mu_m(1)}$$

Then,

$$\begin{aligned} \pi_m(n_m) &= \pi_m(0)\prod_{j=1}^{n_m}\frac{\lambda_m}{\mu_m(1)}\frac{\mu_m(1)}{\mu_m(j)} \\ &= \pi_m(0)\,{\rho_m}^{n_m}\prod_{j=1}^{n_m}\frac{\mu_m(1)}{\mu_m(j)} \\ &= \frac{\pi_m(0){\rho_m}^{n_m}}{A_m(n_m)} \end{aligned} \tag{4}$$

*Version 2.2: Jun 9, 2005 10:15 am +0000

†http://creativecommons.org/licenses/by/1.0

where $\rho_m = \frac{\lambda_m}{\mu_m(1)}$ may or may not be the actual utilization of queue m. Hence,

$$\begin{aligned} \pi_{n_1 n_2 \ldots n_M} &= \frac{\frac{\pi_1(0)\rho_1{}^{n_1}}{A_1(n_1)} \frac{\pi_2(0)\rho_2{}^{n_2}}{A_2(n_2)} \cdots \frac{\pi_M(0)\rho_M{}^{n_M}}{A_M(n_M)}}{K} \\ &= \frac{\frac{\rho_1{}^{n_1}}{A_1(n_1)} \frac{\rho_2{}^{n_2}}{A_2(n_2)} \cdots \frac{\rho_M{}^{n_M}}{A_M(n_M)}}{C} \end{aligned} \tag{5}$$

and $C = \frac{K}{\pi_1(0)\pi_2(0)\pi_M(0)}$ is another constant. Let $\forall m, 1 \leq m \leq M : (\rho_m = \alpha\mu_m)$. Then as we saw in the case of single-server queues,

$$\pi_{n_1 n_2 \ldots n_M} = \frac{\frac{u_1{}^{n_1}}{A_1(n_1)} \frac{u_2{}^{n_2}}{A_2(n_2)} \cdots \frac{u_M{}^{n_M}}{A_M(n_M)}}{G(N)} \tag{6}$$

$G(N) = \frac{C}{\alpha^N}$ is the scaled normalization constant.

$$G(N) = \sum_{\text{all states} \in \left\{n'_1, n'_2, \ldots, n'_M\right\}, (n'_m \geq 0) \wedge (1 \leq m \leq M) \wedge \left(\sum_{m=1}^{M} n'_m = N\right)} \frac{u_1{}^{n'_1}}{A_1(n'_1)} \frac{u_2{}^{n'_2}}{A_2(n'_2)} \cdots \frac{u_M{}^{n'_M}}{A_M(n'_M)} \tag{7}$$

At this point, let's simplify the discussion and assume that there is only one queue, queue M, which has load-dependent service rates. Hence $\forall m, 1 \leq m < n : (A_m(n_m) = 1)$, and

$$\pi_{n_1 n_2 \ldots n_M} = \frac{u_1{}^{n_1} u_2{}^{n_2} \cdots u_{M-1}{}^{n_{M-1}} \frac{u_M{}^{n_M}}{A_M(n_M)}}{G(N)} \tag{8}$$

$$G(N) = \sum_{\text{all states} \in \left\{n'_1, n'_2, \ldots, n'_M\right\}, (n'_m \geq 0) \wedge (1 \leq m \leq M) \wedge \left(\sum_{m=1}^{M} n'_m = N\right)} u_1{}^{n'_1} u_2{}^{n'_2} \cdots u_{M-1}{}^{n'_{M-1}} \frac{u_M{}^{n'_M}}{A_M(n'_M)} \tag{9}$$

As before, we will define $g(n, m)$ to be the scaled normalization constant when there are n jobs and m queues in the network. Also as before, we will separate the terms in $g(n, m)$ into groups. Now, however, instead of separating them into two groups based on whether or not a term has a factor of u_m, we will separate them into $n + 1$ groups, one group for all terms with a factor $u_m{}^k$, $0 \leq k \leq n$.

For the group with factor $u_m{}^k$, we can factor out the $u_m{}^k$ term. For example,

$$\forall k, k = n : \left(\frac{u_m{}^n}{A_m(n)} 1\right)$$

$$\forall k, k = n - 1 : \left(\frac{u_m{}^{n-1}}{A_m(n-1)} (u_1 + u_2 + \cdots + u_{M-1})\right)$$

$$\forall k, k = n - 2 : \left(\frac{u_m{}^{n-2}}{A_m(n-2)} \left(u_1{}^2 + u_2{}^2 + \cdots + u_{M-1}{}^2 + u_1 u_2 + u_1 u_3 + \cdots + u_{M-2} u_{M-1}\right)\right)$$

For each value of k, the terms in parenthesis are just $g(n - k, m - 1)$. Hence,

$$g(n, m) = \sum_{k=0}^{n} \frac{u_m{}^k}{A_m(k)} g(n - k, m - 1) \tag{10}$$

With no jobs in the system, there is only one state, and $g(0, m) = 1$. if there is only one queue in the system, again there is only one state, and we have $g(n, 1) = \frac{u_1{}^n}{A_1(n)}$. From these starting conditions and the recurrence relation for $g(n, m)$ above, we can derive an algorithm similar to the basic Convolution Algorithm for single-server queues. However, instead of computing each term using the terms immediately above and to the left, we use all the terms in the column to the left, down to the same row.

Finally, suppose that the queue with load-dependent service rate is an IS queue. IS in the context of a closed network does not necessarily mean that there are an infinite number of servers, since there is only a finite number of jobs. What is important is the queue have a service rate that scales linearly with the number of jobs; i.e., $u_m(k) = ku_m(1)$. For a queue with this property, $A_m(k)$ is just $k!$, and

$$g(n,m) = \sum_{k=0}^{n} \frac{{u_m}^k}{k!} g(n-k, m-1) \tag{11}$$

The utilization of the one queue with a load-dependent service rate is the probability that the server is busy, which is 1 minus the probability that the queue is empty.

$$Pr[N_m = 0] = \sum_{\text{all states}\in\{n_1,n_2,\ldots,n_M\},(n_m\geq 0)\ \wedge\ (1\leq m<M)\ \wedge\ (n_M=0)\ \wedge\ \left(\sum_{m=1}^{M-1} n_m=N\right)} \frac{{\rho_1}^{n_1}{\rho_2}^{n_2}\cdots{\rho_{M-1}}^{n_{M-1}}}{C} = \sum_{\text{all states}\in\{n_1,n_2,\ldots,n_M\},(n_m\geq 0)\ \wedge\ (1\leq m<M)\ \wedge\ (n_M=0)\ \wedge\ \left(\sum_{m=1}^{M-1} n_m=N\right)} \frac{{u_1}^{n_1}{u_2}^{n_2}\cdots{u_{M-1}}^{n_{M-1}}}{G(N)} = \frac{g(N,M-1)}{g(N)} \tag{12}$$

$$\rho_M = 1 - \frac{g(N,M-1)}{g(N,M)}$$

1 Expected number of jobs in a load-dependent server queue:

For any queue m, the probability that it has at least n jobs in it is the sum of the state probabilities over all states for which $n_m \geq n$. For single-server queues with load-independent service rates, this sum was the same as the normalization constant for $N-n$ jobs, multiplied by ${u_m}^n$; i.e.,

$$Pr[N_m \geq n] = {u_m}^n \frac{G(N-n)}{G(N)}$$

However, this is not true for load-dependent servers, including infinite servers.

$$\begin{aligned} Pr[N_m \geq n] &= \sum_{\text{all states}\in\{n_1,\ldots,n_M\},(n_i\geq 0)\ \wedge\ (i\neq m)\ \wedge\ (n_m\geq n)\ \wedge\ \left(\sum_{j=1}^{M} n_j=N\right)} \pi_{n_1,\ldots,n_m,\ldots,n_M} \\ &= \sum_{k=n}^{N} \sum_{\text{all states}\in\{n_1,\ldots,n_M\},(n_i\geq 0)\ \wedge\ (i\neq m)\ \wedge\ (n_m=k)\ \wedge\ \left(\sum_{j=1}^{M} n_j=N\right)} \pi_{n_1,\ldots,n_m,\ldots,n_M} \end{aligned} \tag{13}$$

For each value of k in the second summation, each probability contains a factor $\frac{{u_m}^k}{A_m(k)}$, and

$$\frac{{u_m}^k}{A_m(k)} = \frac{{u_m}^n}{A_m(n)} \frac{{u_m}^{k-n}}{\frac{A_m(k)}{A_m(n)}} \neq \frac{{u_m}^n}{A_m(n)} \frac{{u_m}^{k-n}}{A_m(k-n)}$$

Hence,

$$Pr[N_m \geq n] \neq \frac{{u_m}^n}{A_m(n)} \frac{G(N-n)}{G(N)} \tag{14}$$

This means that in general there is no simple way to compute the expected number of jobs in a load-dependent server queue using the normalization constants. If we are interested in the expected number of jobs in the last queue and it is a load-dependent server, however, we can still avoid computing individual state probabilities, even if there are other load-dependent servers in the network.

$$\begin{aligned} Pr[N_M \geq n] &= \sum_{\text{all states}\in\{n_1,\ldots,n_M\},(n_i\geq 0)\ \wedge\ (i<M)\ \wedge\ (n_M\geq n)\ \wedge\ \left(\sum_{j=1}^{M} n_j=N\right)} \pi_{n_1,\ldots,n_m,\ldots,n_M} \\ &= \sum_{k=n}^{N} \sum_{\text{all states}\in\{n_1,\ldots,n_M\},(n_i\geq 0)\ \wedge\ (i<M)\ \wedge\ (n_M=k)\ \wedge\ \left(\sum_{j=1}^{M} n_j=N\right)} \pi_{n_1,\ldots,n_m,\ldots,n_M} \\ &= \sum_{k=n}^{N} \frac{{u_M}^k}{A_m(k)} \frac{g(N-k,M-1)}{g(N,M)} \end{aligned} \tag{15}$$

$$\begin{aligned} E\left[N_M\right] &= \sum_{n=1}^{N} Pr\left[N_M \geq n\right] \\ &= \sum_{n=1}^{N} \sum_{k=n}^{N} \frac{{u_M}^k}{A_M(k)} \frac{g(N-k,M-1)}{g(N,M)} \\ &= \sum_{k=1}^{N} k \frac{{u_M}^k}{A_M(k)} \frac{g(N-k,M-1)}{g(N,M)} \end{aligned} \tag{16}$$

If the load-dependent server is an IS queue, we can simplify this even further:

$$E\left[N_M\right] = \sum_{k=1}^{N} \frac{{u_M}^k}{(k-1)!} \frac{g\left(N-k, M-1\right)}{g\left(N, M\right)}$$

If there is only one load-dependent server queue in the network, the obvious approach is to make it the last queue and use the above method. What if there are $L > 1$ load-dependent server queues? We can use the convolution algorithm once for each load-dependent queue, putting a different load-dependent server queue last each time. This actually only requires $L - 1$ applications of the Convolution Algorithm. Compute the expected queue lengths in all of the load-independent server queues with one application of the Convolution Algorithm. Then use these values with those computed for $L - 1$ of the load-dependent server queues to find how many jobs are in the remaining queue.