NAUB JOURNAL OF SCIENCE AND TECHNOLOGY
(NAUBJOST)
e-ISSN:2811-2350

Email: jost@naub.edu.ng http://journal.naub.edu.ng/naubjournal/

APPLICATION OF NORMALIZATION CONSTANT COMPUTATION IN THE CLOSED QUEUING NETWORK

[1]Sunday Olanrewaju Agboola [2]Daniel Watifa
[1,2] Department of Mathematics, Faculty of Natural and Applied Science, Nigerian Army University Biu, PMB 1500 Biu, Borno State, Nigeria
E-mail: *agboolasunday70@gmail.com*
Tel.no: 08058222949

Abstract

This research affords the reader concept to properly model a basic queuing system using the multiple-node system in which a customer requires service at more than one node, which may be viewed as a network of nodes, and each node is a service center having storage room for queues to form and perhaps with multiple servers to handle customer requests. Normalization constant was computed through convolution algorithm and recursive process on Auxiliary function $g_m(n)$in closed queuing network where a fixed number of customers cycles among the service centers, never to depart. This is used to obtain one dimensional array prior to the computation of $g_m(n)$ taken load dependent nodes and load - independent nodes into consideration. A recursive formula is derived for all functions. An illustrative example is considered for given service rate μ_i, $i = 1,2,3,4$ on four nodes, such that $\mu_1 = 1.0$, $\mu_2 = 0.5$, $\mu_3 = 1.0$ and $\mu_4 = 2.0$ respectively, and the following results are obtained: The equilibrium distribution $f_i(n_i)$, $i = 1,2,3,4$ and $n_i = 0,1,2,3$, for load dependent node 1 customers is obtained as $(1,1,0.5,0.25)$. Also, for load-independent nodes $2,3,4$ as $(1,1,1)$, $(1,0.3,0.1)$, $(1,0.09,0.01)$, $(1,0.027,0.001)$ respectively; Auxiliary function $g_i(n)$, $i = 0,\cdots,4$, $n = 0,1,2,3$ is found for load dependent nodes$i = 1$ as $(1,1,0.5,\ 0.2)$, and for load- independent node $i = 2,3,4$ as $(1,2,2.5,\ 2.75)$, $(1,2.3,3.19,\ 3.707)$ and $(1,2.4,3.43,\ 4.05)$ respectively. Also, the probability of having all three customers at the central server is obtained as 0.0617.

Keywords: Auxiliary Function, Buzen's algorithm, Convolution Algorithm, Normalization Constant, Throughput and Visit Ratio.

Introduction

Queues form an indispensible part of our everyday activities (Wayne, 1991). Queuing theory is a branch of mathematics that studies and models the act of waiting in lines (Brockmeyer *et al.*, 1948). Queuing occur whenever there is competition for limited resources (Wayne, 1991). This article took a brief look into the formulation of queuing theory along with examples of models and applications of their use. The goal of the research is to provide the reader with concept to properly model a basic queuing system into one of the categories we looked at, which is the multiple-node system in which a customer requires service at more than one node, which may be viewed as a network of nodes, and each node is a service centre having storage room for queues to form and perhaps with multiple servers to handle customer requests. Also, this will enable the reader to begin to understand the basic ideas of how to determine useful information such as average waiting times from a particular queuing system.

Our quest is to compute the normalization constant through convolution algorithm and recursive process on auxiliary function $g_m(n)$ in closed queuing network where a fixed number of customers cycles among the service centres and never to depart. This is used to obtain one dimensional array prior to the computation of $g_m(n)$ taken load dependent nodes and load - independent nodes into consideration.

A single - node queuing systems is the type of queuing in which each customer arrives at a service center, receives a single service operation from this center, and then departs and never to return. A "multiple-node" system is one in which a customer requires service at more than one node. Such a system may be viewed as a network of nodes, in which each node is a service center having storage room for queues to form and perhaps with multiple servers to handle customer requests (Bacel and Muhammed, 2018).

The study of closed queuing system with exponential servers was first introduced by Gordon and Newell (1967), and this was extended to computational algorithms for closed queuing networks with exponential servers (Buzen, 1973). Chang and Lavenberg (1974) introduced the work rate in closed queuing networks with general independent servers, while the Baskett *et al.* (1975) discussed open, closed and mixed networks with different classes of customers. The application of queue model to computer system with general service time distribution was shown by Schumi (1976). The arrival theorem was proved independently by (Lavenberg 1980; Sevcik 1981) to arrive at useful mean values equations for closed queuing network. Knuth (1998) studied the art of computer programming in queuing network while Haverkort (1998) analysed the performance of computer communication systems. Law and Kelton (2000) presented the simulation modeling analysis of solving a queue networks while Rose (2003) extended the simulation to solved queue networks and this was investigated and built upon by Stewart (2009) into Probability, Markov Chain, Queues and queuing networks, while Agboola (2016) extended the application of queuing network to solved machine repair problems with multiple servers. Quan-lin *(*2017) discussed new invention in queuing models, application and network, while Ronald (2018) applied queuing network model to the analysis of manufacturing systems and Barcel (2018) derived an exact solution for the steady state system size probabilities.

Nomenclature

n_i	Number of customers at node i
N	Total customer population
M	Number of nodes
μ_i	The mean service rate at node i
p_{ij}	The fraction of departures from node i that go next to node j
λ_i	The overall arrival rate to node i
$G(N)$	Normalization constant
$S(N, M)$	The set of all possible states
V_i	Visit ratio at node i
$X_i(N)$	The throughput of node i in a closed network with N customers
$g_i(n)$	Auxiliary functionof node i in a closed network with n customers
$p_i(n, N)$	Marginal queue length distribution
$f_i(n_i)$	Equilibrium distribution for load dependent node i of number customers n_i
$Y_i^{n_i}$	Equilibrium distribution for load independent node i of number customers n_i

Methodology

When we visualize a system of queues and servers, we typically think of customers who arrive from the exterior, spend some time moving among the various service centers and eventually depart altogether. However, in a closed queuing network, a fixed number of customers forever cycles among the service centers, never to depart. Although this may seem rather strange, it does have important applications. For example, it has been used in modelling a multiprogramming computer system into which only a fixed number of processes can be handled at any one time. If the processes are identical from a stochastic point of view, then the departure of one is immediately countered by the arrival of a new, stochastically identical, process. This new arrival is modelled by routing the exiting process to the entry point of the new process. Similar situations in other contexts are readily envisaged. Closed Jackson queuing networks are more frequently called Gordon and Newell networks, after the names of the pioneers of closed queuing networks.

Analysis of Mean Values in Closed Queuing Networks

Consider a closed single-class queuing network consisting of M nodes. Let n_i be the number of customers at node i. Then the total customer population is given by

$$N = n_1 + n_2 + \cdots + n_M \qquad (1)$$

A state of the network is defined as follows:

$$n = (n_1,\ n_2, \cdots, n_M), n_i \geq 0, \textstyle\sum_{i=1}^{M} n_i = N.$$

The total number of states is given by the binomial coefficient

$$\binom{n+M-1}{M-1}.$$

This is the number of ways in which N customers can be placed among the M nodes. The queuing discipline at each node is first come first served (FCFS) and the service time distributions are exponential. Let μ_i be the mean service rate at node i. When the service rate is load dependent, we shall denote the service rate by $\mu_i(k)$.

Let $p_{ij,}$ be the fraction of departures from node i that moves next to node j and let $\lambda_i$be the overall arrival rate to node i. Since the rate of departure is equal to the rate of arrival, when the network is in steady state, we have

$$\lambda_i = \textstyle\sum_{j=1}^{M} \lambda_j P_{ji,}\ \ i = 1, 2, \cdots, M \qquad (2)$$

$$\lambda = \lambda P \quad \text{or} (P^T - 1)\lambda^T = 0 \qquad (3)$$

The linear equations defined by (3) are called the traffic equations. This system of equations does not have a unique solution. Observe that P is a stochastic matrix and this can be solved numerically: the $\lambda_i$can be computed only to a multiplicative constant. We now introduceV_i, thevisit ratio at node i. This gives the mean number of visits to node i relative to a specified node, which in our case will always be node 1. Thus $V_1 = 1$and V_i is the mean number of visits to node i between two consecutive visits to node 1. Notice that V_i may be greater (or smaller) than 1. If $X_i(N)$ is the throughput of node i in a closed network with N customers, then we must have$V_i = \frac{X_i(N)}{X_1(N)}$. At steady state, the rate of arrival to a node is equal to the rate of departure (throughput) so that we may write$V_i = \frac{\lambda_i}{\lambda_1}$ which means that the $V_i$uniquely satisfy the system of equations

$$V_1 = 1 \quad \text{and} \quad V_i = \textstyle\sum_{j=1}^{M} V_j P_{ji,}\ \ i = 1, 2, \cdots, M$$

$$P(n) = p(n_1,\ n_2, \cdots, n_M) = \frac{1}{G(N)} \prod_{i=1}^{M} f_i(n_i) \tag{4}$$

Where

$$f_i(n_i) = \frac{{V_i}^{n_i}}{\prod_{k=1}^{n_i} \mu_i(k)} \text{if node } i \text{ is load dependent} \tag{5}$$

$$f_i(n_i) = \left(\frac{V_i}{\mu_i}\right)^{n_i} = Y_i^{n_i} \text{ if node } i \text{ is load - independent} \tag{6}$$

The constant $G(N)$ is called the normalization constant. It ensures that the sum of the probabilities is equal to one. It is apparent from Equation (4) that once $G(N)$ is known the stationary distribution of any state can readily be computed using only $G(N)$ and easily computable network parameters. To characterize the normalization constant we proceed as follows:

Let the set of all possible states be denoted by $S(N, M)$. Then

$$S(N, M) = \left\{ (n_1, n_2, \cdots, n_M) \sum_{i=1}^{M} n_i = N, \quad n_i > 0, \quad i = 1, 2, \cdots, M \right\}$$

And

$$1 = \sum_{S(N,M)} P(n) = \frac{1}{G(N)} \sum_{S(N,M)} \prod_{i=1}^{M} f_i(n_i)$$

Which imply that

$$G(N) = \sum_{S(N,M)} \left[\prod_{i=1}^{M} f_i(n_i) \right]$$

Where $G(N)$ is written as a function of N only and also a function of M, $p_{ij,}V_i$ and so on.

Computation of Normalization Constant

Let the auxiliary function be defined as

$$g_M(n) = \sum_{S(N,M)} \prod_{i=1}^{M} f_i(n_i)$$

Such that

$$G(N) = g_M(N)$$

Also,

$$g_m(n) = G(n), \qquad n = 0, 1, 2, \cdots, N$$

$$g_m(n) = \sum_{S(n,m)} \prod_{i=1}^{M} f_i(n_i) = \sum_{k=0}^{n} \left[\sum_{\substack{S(n,m) \\ n_m=k}} \prod_{i=1}^{M} f_i(n_i) \right]$$

$$g_m(n) = \sum_{k=0}^{n} f_m(k) \left[\sum_{S(n-k,m-1)} \prod_{i=1}^{M} f_i(n_i) \right]$$

This gives a recursive formula

$$g_m(n) = \textstyle\sum_{k=0}^{n} f_m(k)\, g_{m-1}(n-k) \qquad (7)$$

Some simplifications are possible when node i is load independent. In this case, and using

$$f_m(k) = Y_m^K = Y_m f_m(k-1)$$

We find

$$g_m(n) = \sum_{k=0}^{n} Y_m^K\, g_{m-1}(n-k) = Y_m^0 g_{m-1}(n) + \sum_{i=1}^{n} Y_m f_m(k-1)\, g_{m-1}(n-k)$$

$$= g_{m-1}(n) + Y_m \sum_{i=0}^{n-1} f_m(I) g_{m-1}(n-1-I)$$

Which gives the result

$$g_m(n) = g_{m-1}(n) + Y_m g_m(n-1) \qquad (8)$$

While the initial recursive condition is

$$g_1(n) = \textstyle\sum_{S(n,1)} \prod_{i=1}^{M} f_i(n_i) = \sum_{|(n)|} f_1(n), \;\; n = 0,1,2,\cdots,N \qquad (9)$$

And

$$g_m(0) = 1, \quad m = 0,1,2,\cdots,M$$

This later is obtained from

$$g_m(0) = \sum_{S(0,m)} f_1(n_1),\;\; f_2(n_2),\;\; \cdots,\qquad f_m(n_m)$$

Which for $n = (0,0,0,\cdots,0)$ is just

$$g_m(0) = f_1(0) f_2(0) \cdots f_m(0)$$

In the load independent case, this becomes

$$g_m(0) = Y_1^0 Y_2^0 \cdots\; Y_m^0 = 1$$

Equations (7) and (8) are then used in computing the normalization constant.

The Convolution Algorithm: Implementation Details

For $m = 1, 2, \cdots, M$

$$g_m(n) = g_{m-1}(n) + Y_m g_m(n-1), \;\; n = 1, 2, \cdots, N$$

$$g_m(n) = \sum_{k=0}^{n} f_m(k)\, g_{m-1}(n-k)$$

The recursion proceeds along two directions, n and m, with n assuming values from 0 through N and m assuming value from 1 through M. The first of these corresponds to increasing numbers of customers in the network, and the second to a progression through the nodes of the network. All the values of $g_m(n)$ may be visualized as the elements in a two-dimensional table having N +1 rows and M columns

TABLE 1:Tabular Layout for Computation of Normalization

N	F (N)						G(n)
	1	2	….	m - 1	m	….	M
0	1	1	⋯	1	1	⋯	1
1	$f_1(1)$						
2	$f_1(2)$						
⋮	⋮				$g_m(n-1)$		
n	$f_1(n)$			$g_{m-1}(n)$	$g_m(n)$		
⋮	⋮						$G(n-1)$

The elements may be filled in one after the other, column by column until the table is completed and the required element, $G_M(N)$, is the final value in the last column. The first row and column are obtained from initial conditions: the first row contains all ones (since $g_m(0) = 1, m = 1,2,\ldots, M$) and the first column is obtained as $g_1(n) = f_1(n)$, $n = 0,1,\ldots, N$. The implementation proceeds by considering the nodes of the network one after the other. The actual order in which they are taken is not an immediate concern. As m takes on values from 1 through M, in other words, as we proceed through all the nodes in the network, we use one or the other of the Equations (7) and (8), depending on whether the current value of m corresponds to a load independent or a load dependent node.

A small disadvantage of this visualization offered by Table 1 is that it may lead one to think that a two-dimensional array is required to implement Buzen's algorithm. This is not the case. As we now show, a single column, a vector of length $N + 1$, suffices. Since each of these equations involves g_{m-1} as well as g_m, it is apparent that at least some of the values computed for each preceding node will be needed in the evaluation of the g values for the current node. Let us first consider the case when node m is a load-independent node. We assume that we have already processed nodes 1 through $m - 1$ and in particular, we have all the values $g_{m-1}(n)$ for $n = 0,1,\ldots, N$ and that these values are in consecutive locations in a one-dimensional array. Equation (7) is the appropriate equation in the load-independent case: $g_m(n) = g_{m-1}(n) + Y_m g_{m-1}(n)$, $n = 0,1,2,\ldots, N$. The first element is the same for all values of m: from the initial conditions we have, for all m, $g_m(0) = 1$. As successive elements are computed, i.e., as the value of n in $g_m(n)$ increases from 0 to 1 to 2 and so on, the newly computed value of $g_m(n)$ overwrites the previous value of $g_{m-1}(n)$ in the array. This is illustrated graphically in Table 2. The figure shows the situation immediately before and immediately after the element $g_m(n)$ is computed from Equation (7). In this figure, the current contents of the array are contained within the bold lines. Overwritten values of $g_{m-1}(n)$are

TABLE 2: Array Status for Load Independent Nodes: One Dimensional Array Prior to the Computation of $g_m(n)$

Previous value of $g_{m-1}(j)$ now overwritten with the new value of $g_m(j),\ j = 0, 1,\ 2, \cdots, n-1$	$g_{m-1}(0)$	$g_m(0)$
	$g_{m-1}(1)$	$g_m(1)$
	$g_{m-1}(2)$	$g_m(2)$
	$\vdots$	$\vdots$
	$g_{m-1}(n-2)$	$g_m(n-2)$
	$g_{m-1}(n-1)$	$g_m(n-1)$
Next element to be overwritten	$g_{m-1}(n)$	$g_m(n) = g_{m-1}(n) + g_m(n) * Y$
	$g_{m-1}(n)$	
	$\vdots$	$\vdots$
	$g_{m-1}(N)$	

TABLE 3: Array Status for Load Independent Nodes: One Dimensional Array After the Computation of $g_m(n)$

Previous value of $g_{m-1}(j)$ now overwritten with the new value of $g_m(j),\ j = 0, 1,\ 2, \cdots, n-1$	$g_{m-1}(0)$	$g_m(0)$
	$g_{m-1}(1)$	$g_m(1)$
	$g_{m-1}(2)$	$g_m(2)$
	$\vdots$	$\vdots$
	$g_{m-1}(n-2)$	$g_m(n-2)$
	$g_{m-1}(n-1)$	$g_m(n-1)$
Next element to be overwritten	$g_{m-1}(n)$	$g_m(n)$
	$g_{m-1}(n)$	
	$\vdots$	$\vdots$
	$g_{m-1}(N)$	

To consider the case when node m is a load-dependent node, and as before, we assume that all nodes prior to node m, whether load dependent or load independent, have already been handled. This time, Equation (8) is the appropriate equation to use:

$$g_m(n) = \sum_{k=0}^{n} f_m(k)\, g_{m-1}(n-k), \qquad n = 0, 1, 2, \cdots, N$$

Observe that, whereas in the load-independent case, we require only a single value from the previous node, $g_{m-1}(n)$, this time we need $(n+1)$ values, namely,$g_{m-1}(j)$ for $j = n$, $n-1, \ldots, 0$. It is instructive

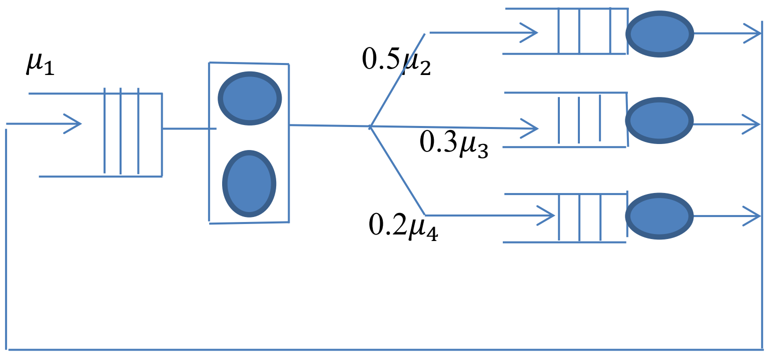
to write Equation (8) as shown below. Notice that we begin with$g_m(1)$, since we know that $g_m(0) = 1$ for all m:

$$g_m(1) = f_m(0)g_{m-1}(1)+ f_m(1)g_{m-1}(0),$$
$$g_m(2) = f_m(0)g_{m-1}(2)+ f_m(1)g_{m-1}(1)+ f_m(2)g_{m-1}(0)$$
$$g_m(3) = f_m(0)g_{m-1}(3)+ f_m(1)g_{m-1}(2)+ f_m(2)g_{m-1}(1)+f_m(3)g_{m-1}(0)$$
$$\vdots$$
$$g_m(N-1) = f_m(0)g_{m-1}(N-1) + \cdots + f_m(N-1)g_{m-1}(0),$$
$$g_m(N) = f_m(0)g_{m-1}(N) + f_m(1)g_{m-1}(N-1) + \cdots + f_m(N)g_{m-1}(0),$$

This shows that $g_{m-1}(1)$ is needed in computing all $g_m(n)$ for $n = 1,2,\ldots,N$ and cannot be overwritten until all these values have been computed. Likewise, $g_{m-1}(2)$ is needed in computing all $g_m(n)$ for $n = 2,3,\ldots,N$ and cannot be overwritten before then. On the other hand, $g_{m-1}(N)$ is used only once, in computing$g_m(N)$. Once $g_m(N)$ has been computed it may be placed into thearray location previously occupied by $g_{m-1}(N)$. Also, $g_{m-1}(N-1)$ is used only twice, in computing $g_m(N-1)$ and $g_m(N)$. Its position in the array may be overwritten when it is no longer needed, i.e., once both $g_m(N)$ and $g_m(N-1)$ have been computed. In this way, we may proceed from the last element of the array to the first, overwriting elements $g_{m-1}(j)$with$g_m(j)$ from$j = N, N-1,\ldots,1$. There is one instance in which this more complicated approach for handing load dependent nodes can be avoided—by designating the load dependent node as node number 1. Then the column (column number 1) is formed from the initial conditions rather than from the convolution formula. Of course, if there is more than one load dependent node, only one of them can be treated in this fashion. One final point concerning the convolution algorithm is worth noting. The coefficients $f_m(n)$may be computed from Equations (5) and (6) and incorporated directly into the scheme just described for updating the one-dimensional array. In particular, it is not necessary to generate the $f_m(n)$ in advance nor to store them in a two-dimensional array.

Results and Discussion

As an example we shall apply Buzen's algorithm to compute the normalization constant for the central server model shown in Figure 1 below. In this model, the central server (node 1 in the diagram) possesses two exponential servers each having rate μ = 1.0. The three nodes that feed the central server (nodes 2, 3, and 4 in the diagram) each possess a single server providing exponential service at rates 0.5, 1.0, and 2.0 respectively. On exiting the central server, customers go to node i = 2,3,4 with probabilities 0.5, 0.3, and 0.2, respectively. We shall assume that N = 3 customers circulate in this network.



$$\mu_1 = 1.0\mu_2 = 0.5\mu_3 = 1.0\mu_4 = 2.0$$

Figure 1: Central server queuing model

The first step is to construct the routing matrix and then solve the traffic equations to obtain the $v's$. From the diagram, we see that the routing matrix is given by

$$\begin{pmatrix} 0 & .5 & .3 & .2 \\ 1 & 0 & 0 & 0 \\ 1 & & & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

The traffic equation are given by

$$(v_1, v_2, v_3, v_4) = (v_1, v_2, v_3, v_4)\begin{pmatrix} 0 & .5 & .3 & .2 \\ 1 & 0 & 0 & 0 \\ 1 & & & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

which may be written more conveniently as

$$v_1 = v_2 + v_3 + v_4,$$

$$v_2 = .5v_1$$

$$v_3 = 0.3v_1,$$

$$v_4 = .2v_1$$

By setting $v_1 = 1$ these equations yield

$$v_1 = 1, v_2 = .5,\ v_3 = .3, v_4 = .2.$$

Node 1 is a load-dependent node and hence the values of $f_m(n)$ are computed from Equation (5). Rewriting this equation again, we have

$$f_i(n_i) = \frac{V_i^{n_i}}{\prod_{k=1}^{n_i} \mu_i(k)}$$

with i taking the value 1 (node 1) and n_1 taking all possible values from 0 through 3 (all possible number of customers present at node 1). This gives

$$f_i(0) = \frac{V_i^{0}}{\prod_{k=1}^{0} \mu_i(k)} = 1$$

$$f_1(1) = \frac{V_1^{1}}{\mu_1(1)} = \frac{1}{1} = 1$$

$$f_1(2) = \frac{V_1^{2}}{\mu_1(1) * \mu_1(2)} = \frac{1}{1 * 2} = 0.5$$

$$f_1(3) = \frac{V_1^{3}}{\mu_1(1) * \mu_1(2) * \mu_1(3)} = \frac{1}{1 * 2 * 2} = 0.25$$

Nodes 2, 3, and 4 are all load-independent nodes and so we use Equation (6)

$$f_i(n_i) = \left(\frac{V_i}{\mu_i}\right)^{n_i} = \left(\frac{V_i}{s_i}\right)^{n_i}$$

We obtain $f_2(0) = 1, f_3(0) = 1, f_4(0) = 1,$

$$f_2(1) = \frac{V_2}{\mu_2} = \frac{0.5}{0.5} = 1, \; f_3(1) = \frac{V_3}{\mu_3} = \frac{0.3}{1} = 0.3, \; f_4(1) = \frac{V_4}{\mu_4} = \frac{0.2}{2} = 0.1$$

$$f_2(2) = \left(\frac{V_2}{\mu_2}\right)^2 = \left(\frac{0.5}{0.5}\right)^2 = 1, \; f_3(2) = \left(\frac{V_3}{\mu_3}\right)^2 = (0.3)^2 = 0.09, f_4(2) = \left(\frac{V_4}{\mu_4}\right)^2 = (0.1)^2 = 0.01,$$

$$f_2(3) = \left(\frac{V_2}{\mu_2}\right)^3 = \left(\frac{0.5}{0.5}\right)^3 = 1, \; f_3(3) = \left(\frac{V_3}{\mu_3}\right)^3 = (0.3)^3 = 0.027, f_4(3) = \left(\frac{V_4}{\mu_4}\right)^3 = (0.1)^3 = 0.001.$$

To find the auxiliary functions and the computation of $G(N)$. We shall compute the elements of the array G node by node. All nodes other than the first are load independent and hence successive updates for$m = 2, 3,$ and 4 are obtained from

$$g_m(n) = g_{m-1}(n) + Y_m g_m(n-1), \; n = 1, 2, \cdots, N$$

With

$$Y_2 = \frac{0.5}{0.5} = 1, \; Y_3 = \frac{0.3}{1} = 0.3, \quad Y_4 = \frac{0.2}{2} = 0.1.$$

The values for node 1 are obtained from the initial conditions, Equation (9) having designated this single load-dependent node as node number 1, precisely to avoid the use of the convolution formula. We have

$$g_i(n) = f_i(n), \quad n = 0, 1, 2, \cdots, N$$

and so the initial values in G are

$$g_1(0) = 1, \; g_1(1) = 1, \; g_1(2) = 0.5, \quad g_1(3) = 0.25,$$

Given these initial values, we successively obtain

$$g_2(0) = 1, \quad g_3(0) = 1, \quad g_4(0) = 1$$

$$g_2(1) = 1 + (1 \times 1) = 2, \; g_3(1) = 2 + (0.3 \times 1) = 2.3, \; g_4(1) = 2.3 + (0.1 \times 1) = 2.4$$

$$g_2(2) = 0.5 + (1 \times 2) = 2.5, g_3(2) = 2.5 + (0.3 \times 2.3) = 3.19, g_4(2) = 3.19 + (0.1 \times 2.4) = 3.43$$

$$g_2(3) = 0.25 + (1 \times 2.5) = 2.75, g_3(3) = 2.75 + (0.3 \times 3.19) = 3.707, g_4(3) = 3.707 + (0.1 \times 3.43) = 4.05.$$

To compute the stationary distribution of any state of this queuing network. For example, the probability of having all three customers at the central server is given by

$$p(n) = p(3,0,0,0) = \frac{1}{G(N)}\prod_{i=1}^{4} f_i(n_i) = \frac{1}{4.05} f_1(3) \times f_2(0) \times f_3(0) \times f_4(0) = \frac{0.25}{4.05}$$
$$= 0.617$$

It could be observed that the equilibrium distribution for load dependent node 1 for $n = 0,1,2,3$ customers is obtained as $(1,1,0.5,0.25)$. Also, for load - independent nodes $2,3,4$ as $(1,1,1)$, $(1,0.3,0.1)$, $(1,0.09,0.01)$, $(1,0.027,0.001)$ for $n = 0,1,2,3$ respectively.

Auxiliary functions $g_i(n)$ for load - dependent node $i = 1$ is obtained as $(1,1,0.5,\ 0.2)$, and for load- independent node $i = 2,3,4$ as $(1,\ 2,\ 2.5,\ 2.75)$, $(1,\ 2.3,\ 3.19,\ 3.707)$ and $(1,2.4,\ 3.43,\ 4.05)$ respectively with probability of having three customers obtained as 0.0617

Conclusion

In queuing network, Customers enter the system by arriving at one of the service centers, queue for and eventually receive service at this center, and upon departure either proceed to some other service center in the network to receive additional service, or else leave the network completely. A closed queuing network as one of the known queuing network where a fixed number of customers forever cycles among the service centers, never to depart has been investigated, and some insight were provided into the mean value analysis. Also, the normalization constant is computed through convolution algorithm and recursive process on Auxiliary function $g_m(n)$ to obtain one dimensional array.The following results are obtained from an illustrative example considered for given service rate μ_i, $i = 1,2,3,4$ on four nodes, where node 1 is load – dependent and nodes $2,3,4$ are load – independent, such that $\mu_1 = 1.0$, $\mu_2 = 0.5$, $\mu_3 = 1.0$ and $\mu_4 = 2.0$. The equilibrium distribution $f_i(n_i)$, $i = 1,2,3,4$ and $n_i = 0,1,2,3$, for load dependent node 1 customers is obtained as $(1,1,0.5,0.25)$. Also, for load independent nodes $2,3,4$ as $(1,1,1)$, $(1,0.3,0.1)$, $(1,0.09,0.01)$, $(1,0.027,0.001)$ respectively. Auxiliary function $g_i(n)$, $i = 0,\cdots,4$, $n = 0,1,2,3$ is found for load - dependent nodes$i = 1$ as $(1,1,0.5,\ 0.2)$, and for load- independent node $i = 2,3,4$ as $(1,\ 2,\ 2.5,\ 2.75)$, $(1,\ 2.3,\ 3.19,\ 3.707)$ and $(1,\ 2.4,\ 3.43,\ 4.05)$ respectively. Also,the probability of having all three customers at the central server is obtained as 0.0617.

References

Brockmeyer, E., Halstrom, H.L., & Jensen, A. (1948). The life and works of A.K Erlang. *Transaction of Danish Academic Journal of Science and Technology,* 2,227

Agboola, S. O.(2016) : Repairman problem with multiple batch deterministic repairs, Unpublished Ph.D. Thesis, Obafemi Awolowo University, Ile-Ife, Nigeria.

Bacel, M. and Muhammed, E. (2018) : Analysis of Two – Node Closed Queuing Cyclic Network with One Non – Exponential Node. *Elsevier, Computer and Industrial Engineering.* 110(1): 297 – 306.

Baskett, F. F., Chandy, K. M., Muntz, R. R. and Palacios, F. G. (1975): Open, Closed and Mixed Networks with Different Classes of Customers. *Communication Journal, ACM.* 22(2):248 – 260.

Buzen, J. P. (1973): Computational Algorithms for Closed Queuing Networks with Exponential Servers. *Communication Journal, ACM.* 16(9):527 – 531.

Chang, A. and Lavenberg, S. (1974): Work Rate in Closed Queuing Networks with General Independent servers. *Communication Journal, ACM.* 22(4): 838 – 847.

Gordon, W. J. and Newell, G. F. (1967): Closed Queuing System with Exponential Servers, *Operations Research* 15(1): 254 – 265.

Haverkort, B.R. (1998): Performance of Computer Communication Systems, John Wiley and Sons, Chichester, U.K. P 1 -12

Knuth, J. D. (1998): The Art of Computer Programming, Vol. 2: *Semi numerical Algorithms*. Third edition, Addison-Wesley, Reading, Mass., 2(3): 1 – 32.

Lavenberg, S.S. and Resier, M. (1980): Stationary State Probabilities at Arrival Instants for Closed Queuing Networks with Multiple Types of Customers. Journal of Applied Probability, 17(4): 1048–1061.

Law, A.M. and Kelton, W. D. (2000): Simulation Modeling and Analysis. Third edition, McGraw-Hill, New York. P 1-30

Quan-Lin, Li, Shunfu Jin, and Zhanyon Ma. (2017): Queuing Theory and Network Application: 12th International Conference, QTNA, Qiuhuangdao, China. August 21 -23

Reiser, M. and Kobayshi, H. (1975): Queuing Networks with Multiple Closed Chains, Theory and Computation Algorithms. *IBM Journal of Resource Development*. 19(1):283 – 294.

Ronald, G. A. and Girish, J. (2018): Queuing Network Models for Analysis of Non – stationary Manufacturing System. *International Journal of Production Research*. 56(1): 12 -21

Ross, M., Yoni, N. and Giang N. (2018): Queuing System, Theory and Application. Springer link. 96.

Schum, A. W. C. (1976): Queuing Models for Computer System with General Service time Distributions, PhD. Thesis, Div. Eng. And Applied Physics, Harvard University, Cambridge, Mass.

Sevcik, K.C. and Mitrani, I (1981): The Distribution of Queuing Network States at Input and Output Instants. *Journal of the ACM,* 28(2): 358–371.

Stewart, W. J. (2009): Probability, Markov Chain, Queues and Simulation, Princeton University Press, United Kingdom

Wayne L Winston (1991). Operations Research: Applications and Algorithms, 2nd edition, *PWS-Kent Publishing, Boston.*