Laboratoire des Systèmes Informatiques

جامعة هواري بومدين للعلوم والتكنولوجيا
USTHB

Rapport de Recherche

Analysis of Multi-Server Queueing Systems with Station and Server Vacation

Nawel GHARBI, Malika IOUALALEN

LSI – TR 0106 **Janvier 2006**

FACULTE ELECTRONIQUE & INFORMATIQUE
Département informatique
El Alia BP n°32 Bab Ezzouar 16111 Alger.
Tél / Fax : 213 (0) 21 24 79 17 - 24 76 07

Analysis of Multi-Server Queueing Systems with Station and Server Vacation

Nawel GHARBI Malika Ioualalen

LSI-Département Informatique, Faculté Electronique & Informatique, USTHB
El Alia BP n°32, Bab Ezzouar, Alger, Algérie.
Email: ngharbi@wissal.dz

Abstract. This paper deals with finite-population, single and multi-server queueing systems with two classes of vacation mechanisms : station vacation and server vacation. In the first one, all the servers take vacation simultaneously whenever the system becomes empty, and they also return to the system at the same time, i.e., station vacation is group vacation for all servers. For the second class of vacation, each server takes its own vacation whenever it completes a service and finds no customers waiting in the queue. We show a method for modeling theses queueing systems using Generalized Stochastic Petri Nets. We develop the general form of the underlying CTMCs, and we give two algorithms for computing the infinitesimal generator Q for systems with s servers ($s \geq 1$). Hence, several quantitative performance measures can be obtained for queueing systems with station or server vacation. this work.

Key-Words. Multi-server Queueing systems, Server vacation, Station vacation, Generalized Stochastic Petri nets, Performance Analysis.

1. Introduction

Queueing systems in which the server takes a vacation from a primary task either to attend to one or more secondary tasks or to rest, arise as models of many computer, communication, production and other stochastic systems. For example, in polling systems, such as the token-passing local networks, the server attends to a group of queues in a cyclic manner. From the point of view of a particular queue, the server goes on vacation each time it ceases to serve that queue. So, the server working on the secondary customers is equivalent to the server taking a vacation and not being available to the primary customers during this period. Similarly, processors in computer and communication systems do considerable testing and maintenance besides doing their primary functions (as processing telephone calls, processing interactive and batch jobs, receiving and transmitting data, etc). The testing and maintenance periods which are mainly doing to preserve the sanity of the system and to provide high reliability, may be regarded as server vacations. Hence, customers (messages for example) who arrive while the server is on break (vacation) have to wait until he is back. The machine breakdowns which may occur randomly, independent of the queue status, in manufacturing systems for example, can also be regarded as server vacations.

Thus, there is a natural interest in the study of queueing systems with server vacations.
This class of queues has been extensively studied by several researchers. The readers are referred to the surveys of (Doshi [2,3], Teghem [5]), the monograph of (Takagi [4]), as well as the references therein.

However, in almost all these papers, it is assumed that there is one single server in the system, and that the population is infinite. In this article, we consider queueing systems with finite population and multiple servers ($s \geq 1$).

Two vacation mechanisms are introduced : station vacation and server vacation. For the first one, all the servers take vacations simultaneously. That is, whenever the system is empty, all the servers leave the system for a vacation, and return to the system simultaneously when the vacation is completed. So, station vacation is a group vacation of all servers. This occurs when a system consists of several interconnected machines that are inseparable, or when all the machines are run by a single operator. In this case, if the system (or the operator who runs the system) is used for a secondary task when it becomes empty (or available), all the servers (the operator) will then be utilized to perform a secondary task. During this amount of time, the system is unavailable to further arrivals, and this is equivalent to taking a station vacation. The second vacation mechanism, server vacation, is encountered even more often in practice. In this case, each server is an independent working unit, and it can take its own vacation upon completing serving a customer and finding no customers waiting. This phenomenon occurs, for instance, in post offices where, when observing an empty queue, a clerk goes for other types of work (sorting, distributing, etc).

In this paper, we show a method for modeling and analyzing finite-population multi-server queueing systems with station and server vacations, using generalized stochastic Petri nets (GSPNs).

One of the advantages of using GSPNs to model queueing systems with vacations, is that we can incorporate features that may be somewhat difficult to model by the more conventional methods. For example, if the server is subject to breakdown in addition to his normal vacation, the two kinds of vacations can easily be modeled by GSPNs in a hierarchical manner.

On the other hand, GSPNs are a useful tool for the development of very compact and easy to understand models of parallel systems with interacting concurrent components.
This approach allows to obtain several benefits both for the qualitative and the quantitative analysis of these queueing systems. In particular, it offers the possibility of using results, methods and tools developed within the GSPNs framework, to obtain performance indices of multi-server queues with station or server vacations.

The paper is structured as follows : First, the basics of GSPNs are reviewed. In section 3, we give the GSPN describing multi-server queueing systems with server and station vacation mechanisms. Next, we develop the general form of the underlying continuous time Markov chains (CTMCs), and the algorithms for computing the infinitesimal generator Q, for a general number of servers s and a general finite-population capacity N. Finally, we give a conclusion.

2. An overview of Generalized Stochastic Petri nets

In the past decade, GSPNs have received much attention from researchers in the performance and reliability arena, and have been extensively used for analytical modeling in the context of independability, performance and performability of computer, communication, manufacturing and aerospace systems. Many structural and stochastic extensions have been proposed in recent years to increase their modeling power, and their capability to handle large systems. GSPNs are an important graphical and mathematical model appropriate for describing and analyzing systems that are characterized as being concurrent, asynchronous, distributed, non-deterministic and/or stochastic.

A GSPN is a directed bipartite graph that consists of two nodes called places (drawn as circles), and transitions that are partitioned into two different classes : timed transitions and immediate transitions (Ajmone et al. 1995)
Timed transitions describe the execution of time consuming activities. In this paper, we consider markovian timed transitions, that fire after an exponentially distributed firing time.
Immediate transitions have priority over timed transitions and fire in zero time once they are enabled. This class of transitions was added to Stochastic Petri nets because modeling real systems often involves the presence of activities or actions, whose duration is short, or even negligible, with respect to the time scale of the problem. Hence, it is desirable to associate an exponentially distributed firing time only with those transitions which are believed to have the largest impact on the system operation.
In the graphical representation of GSPNs, timed transitions are indicated by means of rectangles, and immediate transitions are drawn as thin bars.
Formally, a GSPN is a six-tuple $(P,T,I(.),O(.),W(.),M_0)$ where :
$P = \{ P_1 , P_2 , ... , P_n \}$ is the set of places,
$T = \{ t_1 , t_2 , ... , t_m \}$ is the set of transitions,
$I(.) : P \times T \rightarrow IN$ is the input function which provides the multiplicities of the input arcs from places to transitions.
$O(.) : T \times P \rightarrow IN$ is the output function which provides the multiplicities of the output arcs from transitions to places.
$M_0 : P \rightarrow IN$ is the initial marking which describes the initial state of the system
$W(.) : T \rightarrow IR^+$ is a function that associates rates of negative exponential distribution to timed transitions and weights to immediate transitions.

The system state is described by means of markings. The marking in general, is a mapping from P to IN which gives the number of tokens in each place after each transition firing.
Markings enabling no immediate transitions are called **tangible markings**. In this case, any timed transition can fire next (application of race policy commonly).
Marking enabling only immediate transitions are passed through in zero time and are called **vanishing markings**. In this case, only the enabled immediate transitions are allowed to fire because the lowest level is reserved for timed transitions.
Since the process spends zero time in the vanishing markings, they don't contribute to the dynamic behavior of the system. So, a procedure is envisaged to eliminate them from the reachability graph which contains all the reachable markings from initial one [1].
Given a tangible marking M_b directly reachable from a vanishing marking M_a, by the firing of an immediate transition, and S the set of tangible markings from which M_a is directly reachable. The vanishing marking M_a can be merged with M_b and so, eliminated by introducing arcs directly connecting M_c to M_b , $\forall M_c \in S$.

This elimination of vanishing markings results in a reduced reachability graph which is isomorphic to a continuous time Markov chain (CTCM) due to the memoryless property of exponential distributions.
The states of the CTMC are the markings in the reduced reachability graph, and the state transition rates are the exponential firing rates of timed transitions in the GSPN.
The solution of this CTMC at steady-state is the probability distribution over the set of the states. The steady-state distribution $\pi = (\pi_1, \pi_2, \pi_3, \ldots.)$ with :

$$\pi_i = \lim_{t \to \infty} \text{Prob} \{ X(t) = i \} \qquad (1)$$

is computed as the solution of the linear system of equations :

$$\begin{cases} \pi . Q = 0 & (2) \\ \sum \pi_i = 1 \end{cases}$$

where Q is the generator matrix (transition matrix).
Having the steady-state probabilities vector π, we can easily compute several performance measures of the studied system.

3. GSPN models of queueing systems with vacation

We consider queueing systems in which customers arrive at the system according to Poisson process with rate λ. There are s parallel servers ($s \geq 1$), and the service times of the customers are independent and identically distributed exponential random variables with rate μ. The vacation times of all servers also are assumed to be independent and identically exponentially distributed with rate θ.
Any arriving customer who finds all servers busy upon arrival waits in the queue until a server will be idle. Every customer requires to be served by one and only one server, and leaves the system once the service is completed.

We consider multiple vacations politic (Doshi [2,3]). That is, the server or station, upon returning from a vacation shall leave immediately for another one if the system is empty at that moment. The process continues until the servers return and find any customer waiting.

The exhaustive service discipline (Doshi [2,3]) is assumed here. That is, upon completing a vacation, the server(s) return(s)to the system and start(s) to serve customers, if any, till the system becomes empty.

3.2. Queues with Server Vacation

This model is used for describing many practical problems where servers take individual vacations. This means, whenever a server completes service and there are no more customers waiting in the queue, it takes a vacation independently of others servers state.
A typical example is encountered in the post office. When a clerk completes services and finds no customers waiting, he might go to work on a secondary task, say sorting letters. This is what we refer to as the server vacation model.

Figure 1 shows the GSPN model describing the above system.

- The place P_a contains N tokens, which represents the condition that none of the N customers has arrived for service. N is the population capacity.
- The place P_B contains the customers waiting for service.
- The place P_s contains the number of customers in service.
- The place P_d represents the idle (available) servers.
- The place P_v contains the servers that are on vacation.

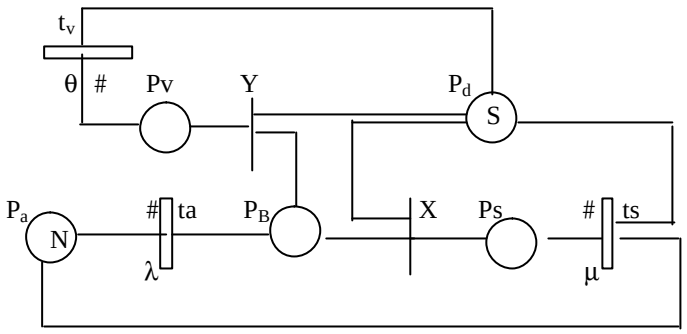

Fig. 1. GSPN for the M/M/s//N Queueing System with Server Vacations

Currently, the marking (initial marking) of the net is $M_o=\{M(P_a), M(P_B), M(P_s), M(P_d), M(P_v)\}=\{N,0,0,S,0\}$ which implies that none of the customers has arrived for service and that the S servers are idle.

- When the transition t_a fires, one token is taken from P_a and deposited in P_B. The firing of t_a indicates the arrival of a customer. This firing is said to be "marking dependent"; that is, the firing rate of t_a depends on the number of tokens in P_a. Thus, if there are k tokens in P_a, then the firing rate is $k\lambda_a$, and if there is only one token, the firing rate is λ_a. The condition of marking dependent firing is represented by the symbol # placed next to the transition. The timed transitions t_S and t_v are also "marking dependent".
- The immediate transition X is enabled when the place P_B contains at least 1 token, which represents a waiting customer, and the place P_d at least one idle server. Hence, one token is taken from P_B and from P_d and deposited in P_S . This token represents a customer in service.
- The input arc from P_B to the immediate transition Y is an inhibitor arc. The firing of Y represents the event that a server is commencing a vacation since there is no customer left to be served.
- When the timed transition t_v fires, which represents the end of the vacation time, one token is deposited in P_d which represents the fact that a server is returned to the system.
- When the timed transition t_s fires, one token is deposited in P_a, which represents the condition that the customer has returned to be idle or thinking state and one token is deposited in P_d, which represents the condition that this server is ready to serve another customer.

It is assumed that $S \leq N$; that is, the servers number cannot exceed the population size of the system.

Note : For our model, tokens still accounting in the input places of transition, so much his firing not finished.

Considering all possible values for the population size N and the servers number s, one obtains the CTMC for this model as follows :

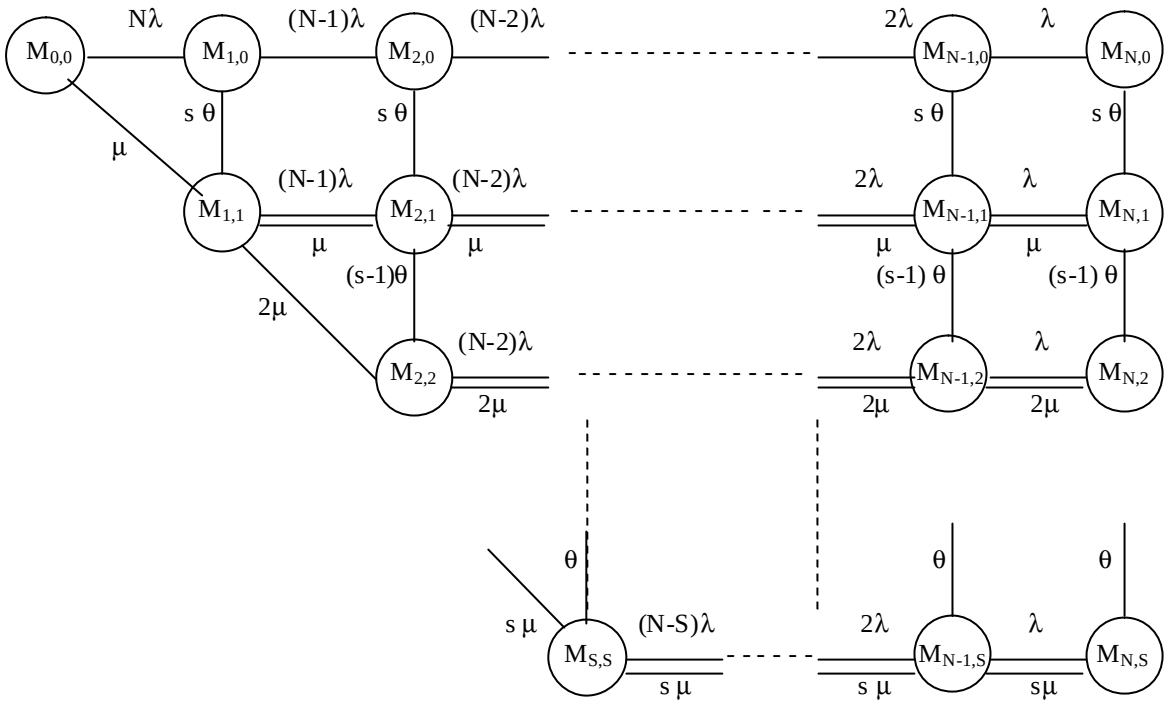

Fig. 2. The CTMC of Multi-Server Queues with Server Vacation

Where $M_{i,x}$ is the marking i in the level x, describing the system state at a given moment.
The general form of these CTMC states are as follows :

$$M_{i,x} = [\ N-i\ ,\ i-x\ ,\ x\ ,\ 0\ ,\ S-x\ ] \qquad (3)$$

where $0 \le x \le S$ and $x \le i \le N$

The infinitesimal generator of this CTMC is a IK*IK matrix Q where IK, the markings number of all CTMC equals : $\Sigma(N-i+2)$ for $1 \le i \le S+1$
This transition matrix Q can be constructed as follows :

$$Q[(i,x),(j,y)] = \begin{cases} \lambda[(i,x),(j,y)] \ , \ (i,x) \neq (j,y) \\ -\sum_{\substack{(k,z)=(0,0) \\ (k,z)\neq(i,x)}}^{(N,S)} \lambda[(i,x),(k,z)] \ , \ (i,x)=(j,y) \end{cases} \qquad (4)$$

where $\lambda[(i,x),(j,y)]$ is the transition rate from marking M(i,x) to marking M(j,y) that can be calculated by applying the following algorithm :

Algorithm 1 :

```
BEGIN
For x := 0 To S
     Do For i := x To N-1 Do
          Begin
            λ[(i,x),(i+1,x)] := (N-i)λ ;
```

```
                If x>0 Then λ[(i+1,x),(i,x)] := x μ ;

             End;
For i := 1 To S

      Do For x := 0 To i-1

                Do λ[(i,x),(i,x+1)] := (s-x) θ ;

For i := S+1 To N

      Do For x := 0 To S-1 Do λ[(i,x),(i,x+1)] := (s-x) θ ;

   For x := 0 To S-1 Do λ[(x+1,x+1),(x,x)] := (x+1) μ ;
END
```

Once the infinitesimal generator Q is obtained, the vector of steady-state probabilities $\pi=(\pi_1,\pi_2,\pi_3, \ldots.)$ can be computed by solving the linear system of
equations $\pi.Q = 0$ with the normalization condition $\sum \pi_i = 1$
Having the steady-state probabilities vector π, we can easily compute several performance measures of the studied system.

3.2. Queues with Station Vacation

In this model, as soon as the system is empty (all servers become idle), the station takes a vacation. As one may expect, this situation appears to be more complicated that the previous one. In fact, it is more simple, because all servers take a vacation simultaneously and return to the system at the same time also. Hence, the GSPN modeling this system with multiple station vacations, is the same model as the one given in Figure 1 in which the multiplicity of the arc connecting the place P_d to transition Y and transition t_v to place P_d equals S (rather than 1), because the S servers of the station take a vacation together. So, if the place Pd contains s idle servers, the immediate transition Y fires, which represents the begin of the station vacation time. At the end of this period (after a mean delay equals $1/\theta$), s tokens corresponding to the s servers will be deposited in P_d. On the other hand, the transition t_v is not "marking dependent".
Considering all possible values for the population size N and the servers number s, the CTMC for the model with station vacation is given below :

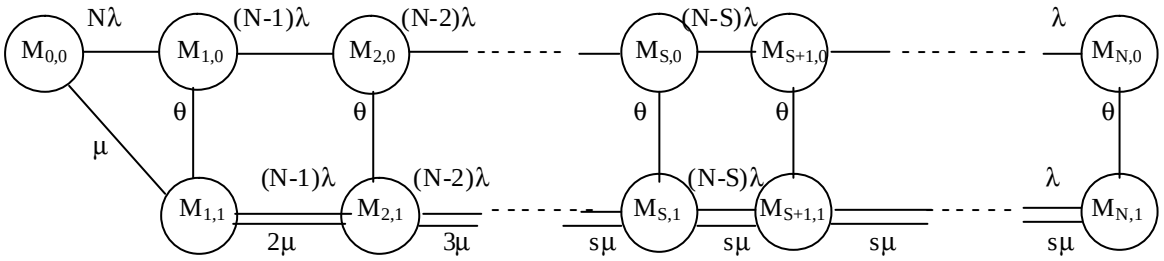

Fig. 3. The CTMC of Multi-Server Queues with Station Vacation

Where :

$$\begin{cases} M_{i,0} = [N-i\ ,\ i\ ,\ 0\ ,\ 0\ ,\ 1\ ]\ ,\ 0 \le i \le N \\ M_{i,1} = [N-i\ ,\ 0\ ,\ i\ ,\ S-i\ ,\ 0\ ]\ ,\ 1 \le i \le S \end{cases} \qquad (5)$$

$M_{i,1} = [N-i\ ,\ i-S\ ,\ S\ ,\ 0\ ,\ 0\ ]\ ,\ S+1 \le i \le N$

The infinitesimal generator of the CTMC of Fig. 3 is a IK*IK matrix Q, where IK=2N+1.
The components of Q are given by :

$$Q[(i,x),(j,y)] = \begin{cases} \lambda[(i,x),(j,y)]\ ,\ (i,x)\neq(j,y) \\ -\sum_{\substack{(k,z)=(0,0) \\ (k,z)\neq(i,x)}}^{(N,1)} \lambda[(i,x),(k,z)]\ ,\ (i,x)=(j,y) \end{cases} \quad (6)$$

The transition rates can be calculated by applying the following algorithm :

Algorithm 2 :

```
Begin

  For x := 0 To 1

      Do For i := x To N-1

             Do  λ[(i,x),(i+1,x)] := (N-i)λ ;

For i := 2 To s  Do λ[(i,1),(i-1,1)] := i μ  ;

For i := s+1 To N  Do λ[(i,1),(i-1,1)] := s μ;

For i := 1 To N Do λ[(i,0),(i,1)] := θ ;

End.
```

3.3. Single Server Queues with Vacation

The GSPN modeling the single server queue with station vacation is given in Figure 1 with S=1. Hence, the CTMC of the model is the same as the CTMC of Figure 3 where :

$$\begin{cases} M_{i,0} = [N-i\ ,\ i\ ,\ 0\ ,\ 0\ ,\ 1\ ]\ ,\ 0 \le i \le N \\ M_{i,1} = [N-i\ ,\ i-1\ ,\ 1\ ,\ 0\ ,\ 0\ ]\ ,\ 1 \le i \le N \end{cases} \quad (7)$$

In fact, the two models with server vacation and station vacation give the same results for S=1. So, the models coincide since there is only one server in the whole station. Hence, for calculating transition matrix we can use one of the two algorithms given above.

Conclusion

In this paper, we proposed a technique that allows to represent finite-population, single or multi-server queueing systems with server and station vacations using GSPNs.

The novelty of the investigation is the multiplicity of the servers, and also the introduction of server and station vacation mechanisms, which make the system rather complicated.
The flexibility of GSPN modeling approach allowed a simple construction of detailed and compact models for these queueing systems. We have also developed the general form of the underlying CTMCs, and two algorithms for computing the infinitesimal generator Q, for a general value of the servers number and a general finite-population capacity. Hence, several quantitative performance measures can be derived.

Finally, many vacation queueing problems and their solution can be simplified using the GSPN modeling approach with all the results developed within this framework.

Bibliography

[1] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli and G. Franceschinis, *Modelling with Generalized Stochastic Petri Nets*. John Wiley & Sons, New York,1995.

[2] B.T. Doshi, Queueing systems with vacations : a survey, *Queueing Systems*, 1, pp 29-66, 1986.

[3] B T. Doshi, Single server queues with vacations, *Stochastic Analysis of Computer and Communication Systems*, H. Takagi (editor), Elsevier science Publishers B.V., Amsterdam, 217-265, 1990.

[4] H. Takagi, Queueing Analysis, *A foundation of Performance Evaluation*, Vol. I, Elsevier Science Publishers B.V., 1991.

[5] J.Jr. Teghem, Control of the service process in a queueing system, *European Journal of Opeartions Research*, 23, pp 141-158, 1986.