Available online at www.sciencedirect.com
SCIENCE DIRECT
Int. J. Electron. Commun. (AEÜ) 60 (2006) 131 – 135
International Journal of Electronics and Communications AEÜ
www.elsevier.de/aeue

Linking priority queues and tree-like processes

B. Van Houdt, C. Blondia*

Department of Maths and Computer Science, University of Antwerp, Middelheimlaan 1, B-2020 Atwerpen, Belgium

Abstract

This paper links the analysis of a general class of priority queues with three service classes with tree-like processes. This is realized by demonstrating that the operation of a 3-class priority queue can be mimicked by means of an alternate system that is composed of a single stack and queue. Some insights on how to reduce the evolution of this alternate system to a tree-like process are provided, allowing an efficient solution of the priority system through matrix analytic methods.
© 2005 Elsevier GmbH. All rights reserved.

Keywords: Priority queues; Tree-like processes; Matrix analytic methods

1. Introduction

It is a great pleasure to write a contribution for this special issue dedicated to Paul Kuehn on the occasion of his 65th birthday. I (the second author) met Paul in the late eighties when we were involved in a project of the European research programme RACE, entitled "Technology for ATD", also known as the ATM project. As workpackage leader, he knew how to motivate all researchers involved. His long experience in performance modelling ensured that all important design decisions were taken based on quantitative arguements that were obtained from analytical models or simulations.

An important issue in this project was the design of traffic management mechanisms that allow to differentiate various services (i.e. QoS support). In this context the introduction of priorities, be it in time (to differentiate with respect to delay) or in space (to differentiate with respect to loss probability) seems to be inevitable. Paul Kuehn has made crucial contributions in many areas of performance modeling, in particular also in the area of priority systems. The notion of conditional cycle time introduced in his paper on multiqueue systems with nonexhaustive cyclic service [1], is not only applicable for polling systems, but offers a new approach to the solution of priority systems.

Numerous methods to solve priority systems in an efficient way have been proposed. In this paper we discuss an interesting link between the analysis of priority queues with three classes of service and tree-like processes. The key observation of our approach lies in reformulating the traditional three queue problem into a combined queue and stack problem. Some indications that the behavior of the reformulated problem can often be captured by a Markov chain with a tree structured state space [2,3] are provided. Such Markov chains – that neither fit within the GI/M/1 or M/G/1 paradigm – can be reduced to binary tree-like processes [4,5], which are a special case of a tree structured QBD Markov chain [6]. Typical performance measures captured by this methodology include the queue length distribution and loss rate.

The arrival process to model the incoming jobs may be as general as the Markov arrival process with marked jobs, i.e. the MMAP[3] process [7,8]. The MMAP[3] is a set of arrival processes that allows correlated inter-arrival times and correlation between the classes of consecutive jobs. Moreover, the input traffic does *not* need to consist of three independent streams (one for each priority class). Furthermore, we allow that the amount of time needed to execute a job

* Corresponding author. Tel.: +32 3 2653903.
E-mail address: chris.blondia@ua.ac.be (C. Blondia).

1434-8411/$ - see front matter © 2005 Elsevier GmbH. All rights reserved.
doi:10.1016/j.aeue.2005.11.009

depends on its class, while the processing time of consecutive jobs of the same class is independent and identically distributed (iid). Processing times are typically phase-type distributed random variables. Thus, in general, one cannot simply lump the priority one and two traffic into a single class as this would create correlation into the service times of the lumped job class.

In a forthcoming paper we will demonstrate our approach in detail on a discrete-time preemptive priority queueing system without batch arrivals. As is apparent from this paper, the technique also applies to the continuous time setting or to nonpreemptive systems. The methodology is exact if the waiting rooms for the priority 1 and 3 jobs are both finite while the waiting room for the priority 2 jobs has an infinite capacity. The infinite nature of the priority 2 queue does not jeopardize the practical relevance of our model as priority queues are mainly effective if sufficient jobs have the lowest priority. Therefore, hardly any priority 1 or 2 jobs will leave the system without receiving full service; hence, whether the priority 2 queue is infinite or finite and sufficiently large makes little difference.

The study of priority queues has a long history that started in the 1950s (according to Miller [9]) and a plethora of applications in both communication and manufacturing systems has benefited from its development. Starting from the elementary M/M/1 priority queue, the complexity and generality of the models under study has grown ever since. Various, more recent studies have focused on priority queues with Markovian input traffic and phase-type (or general) service requirements [10–15].

In [10,11] Alfa et al. studied priority queues with $C \geqslant 2$ classes, MMAP[C] input and phase-type services by setting up a QBD Markov chain [16] that generalizes Miller's result [17]. Although the queues considered in [10,11] are infinite, a similar approach can be used to solve the system when some of the queues are finite. In case of 3 service classes and a finite capacity queue of size K for each of the class-1 and class-3 jobs, the resulting QBD has size $O(K^2)$ block matrices (the structure of which can be exploited to compute the rate matrix R). Our approach has the advantage that the blocks characterizing the binary tree-like process are of size $O(K)$ only.

Takine and Hasegawa [14] analyzed the preemptive priority queue with $C \geqslant 2$ service classes, C independent MAP arrival streams and state-dependent service times, using the workload process of the queue. They generalize the model presented by Machihara in [12] that relied on the diagonalization of some matrices. In our setting we do not require independent MAP arrivals streams and focus on the queue length distributions, while the approach in [14] can deal with more general service requirements and is especially effective to compute the waiting time distributions. Takine [13,15] also studied the nonpreemptive priority MAP/G/1 queue and derived various formulas for the generating function of the queue length and Laplace–Stieltjes transform of the waiting time for each job class. The remainder of the paper is organized as follows. The stack and queue model reformulation of the 3-class priority queue is given in Section 2. Section 3 gives some guidelines on how to construct a tree-like process to analyze the stack and queue problem and discusses the nature of the arrival and service process needed.

2. Analogy of the 3-class priority queue and a combined stack and queue model

Consider a preemptive priority queueing system with three service classes consisting of a single server and three waiting rooms, one for each service class. Assume that class-1 jobs have the highest priority, followed by the class-2 jobs and finally the class-3 jobs. The processing times of consecutive jobs belonging to the same class are iid. Let the capacity of both[1] the class-1 and class-3 waiting rooms be finite and of size $K > 0$, while the class-2 waiting room is considered infinite in size. Jobs that find their corresponding waiting rooms full, leave the system without being executed. A class-3 job that is pushed out of the service facility will (re)occupy the leading position in the class-3 waiting room, possibly pushing out the youngest waiting class-3 job. We refer to this traditional model as the three queue model (3Q). We now replace the class-2 and class-3 waiting rooms by a single, infinite size stack, to obtain a new model termed the stack & queue (S&Q) model. We will show that the state of the 3Q model is uniquely identified by the current state of the S&Q model.

The evolution of the S&Q model goes as follows. An example to further clarify the relationship between the two systems is depicted in Fig. 1. During the description of the S&Q model, we will refer to this figure and indicate the time instants at which such an event occurs. As long as there is at least one class-1 job in the system (meaning a class-1 job is being processed), we use the queue to store priority 1 jobs (time 1–3, 6–9 and 15), while all arriving class-2 or class-3 jobs are simply pushed on the stack (time 2, 3 and 8). If there are no other class-1 jobs in the system when a priority 1 job arrives, it will occupy the server. The possible class-2 or class-3 job that was in service will be pushed on the stack together with the entire contents of the queue, making the queue available again for class-1 jobs (time 1, 6 and 15). Hence, as far as the class-1 jobs are concerned there is no difference between the 3Q model or the S&Q model. During periods when there is no class-1 traffic present, we utilize the queue for other means (time 0, 4–5, 10–14 and 16–18). Note the number of priority 3 jobs on the stack might become larger than K, the capacity of its waiting room in the 3Q model, as we simply push all class-3 arrivals on the stack when a class-1 job is in service (time 5–13). Further on, it will become apparent that the number of class-3 jobs waiting for service in the 3Q model is identical to the minimum of K

[1] There is no need to select the same capacity $K > 0$ for the class-1 and class-3 waiting rooms. It only simplifies the presentation of the model.

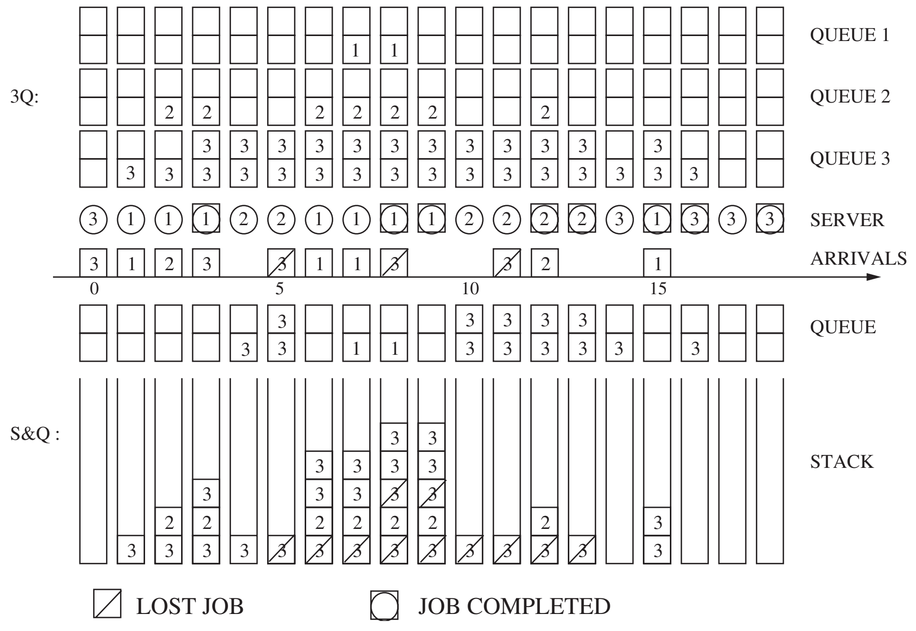

Fig. 1. System dynamics during a busy period for the 3Q and S&Q system.

and the number of waiting class-3 jobs in the S&Q model. There is no need to keep track of the correct number of class-3 jobs at all times, it suffices that the number is identical to the one of the 3Q system when the server becomes available for the class-3 jobs. Thus, the number of class-3 job losses is identical in both models, however, the times at which the these losses occur need not coincide.

Assume that a class-1 job is completed and there are no other class-1 jobs ready to start service, that is, the server is now available for class-2 or class-3 traffic (time 4, 10 and 16). Moreover, the queue used to store the class-1 jobs is now empty and can be utilized for other purposes as long as there is no class-1 traffic in the system. In such a case we start looking for a class-2 job in the stack, by popping jobs from the stack until we encounter a class-2 job. This class-2 job is placed in the service facility. As the service times within a class are iid, the identity of and order in which the jobs of a specific class are served is irrelevant, as long as an interrupted job is continued when the server becomes available again to this job's priority class. This is realized by assuming that the class-2 jobs mutually switch positions in the stack such that the oldest ones are on top.

The class-3 jobs that are removed from the stack while searching the stack for a class-2 job, are stored in the queue (time 4, 10, 14 and 16). Thus, as long as there are no class-1 jobs, we use the queue (temporarily) to store class-3 jobs. If the queue is full (with class-3 jobs) when popping a class-3 job from the stack, the job leaves the system without being performed (time 10 and 14). As with the class-2 jobs, we assume that the class-3 jobs also mutually exchange positions such that the youngest jobs get dropped first, while the elder receive service before the younger ones (that is why the lost jobs in the 3Q model immediately move to the lowest class-3 positions on the stack at time 5 and 8 in the S&Q model). If a class-2 job is completed (and there is no class-1 arrival) we start to pop jobs from the stack in search of another class-2 job (time 14). If no such job is encountered, meaning that the stack is empty and all class-3 jobs are now present in the queue, we start executing the priority 3 jobs.

Finally, if a class-2 job arrives while a class-3 job is being performed, it will interrupt its service and the class-3 job returns to the queue. Hence, we do not push the entire contents of the queue on the stack in such case. Class-3 arrivals that occur while a class-2 or class-3 job is being executed are always directly stored in the queue (time 5), or lost if the queue is full, while class-2 arrivals are pushed onto the stack in such cases. The execution of class-3 jobs can only start when the stack is empty (time 0, 14 and 16–18), therefore, the number of lost jobs during a busy period is identical in both the 3Q and S&Q system.

3. Analyzing the S&Q model via tree-like processes

In this section we present some guidelines on how to model the S&Q model using a tree-like process. From the discussion in the previous section it should be clear that the analogy between the 3Q and S&Q model applies to both continuous and discrete time systems, as well as preemptive and nonpreemptive priority queues. To assess the performance of the S&Q model via a tree-like process some restrictions on the arrival and service time process seem necessary.

The job arrival process may be as general as the MMAP[3] process with m phases. It is characterized by a set of four matrices D_0, D_1, D_2 and D_3. The (i, j)th entry of D_0 represents the probability that the phase changes from i (at time t) to j (at time $t+1$), while no arrivals occur (at time t). Similarly, D_i gives the probability for the same event, but with a class-i job arrival (at time t), for $i = 1, 2$ and 3. Define θ as the stochastic vector that satisfies $\theta(D_0 + D_1 + D_2 + D_3) = \theta$. The arrival rate of the class-i jobs λ_i can be computed as $\lambda_i = \theta D_i e_m$ (where e_m is a column vector of size m with all its entries equal to one). To make the model tractable the processing time of a class-i job has to follow a phase-type distribution (α_i, T_i), where the jth entry of the stochastic $1 \times m_i$ vector α_i provides the probability that a class-i job starts its service in phase j. The (j, j')th entry of T_i, on the other hand, is the probability that such a job continues its service in phase j' at the next time instant provided that it is in phase j at the current time instant. The mean processing time of a class-i job equals $\mu_i = \alpha_i(I - T_i)^{-1}e_{m_i}$.

Tree-like processes are very useful to analyze systems with an underlying stack structure. A tree-like process is a bivariate Markov chain $\{(X_t, N_t), t \geqslant 0\}$ in which the values of X_t are represented by nodes of a $(d+1)$ary tree, for $d \geqslant 0$, and where N_t takes integer values between 1 and h. The root node of the $(d+1)$ary tree is denoted as $\emptyset$ and the remaining nodes are denoted as strings of integers, where each integer takes a value between 0 and d. For instance, the kth child of the root node is represented by k, the lth child of the node k by kl, and so on. We use the '+' to denote the concatenation on the right and '−' to represent the deletion from the right. For example, if $J = k_1k_2\ldots k_n$, then $J + k = k_1k_2\ldots k_nk$. Let $f(J, 1)$, for $J \neq \emptyset$, denote the rightmost element of the string J, then $J - f(J, 1)$ represents the parent node of J.

The following restrictions need to apply for a Markov chain (X_t, N_t) to be a tree-like process. At each step the chain can only make a transition to its parent (i.e. $X_{t+1} = X_t - f(X_t, 1)$, for $X_t \neq \emptyset$), to itself ($X_{t+1} = X_t$), or to one of its own children ($X_{t+1} = X_t + s$ for some $0 \leqslant s \leqslant d$). Moreover, the chain's state at time $t+1$ is determined as follows:

$$P[(X_{t+1}, N_{t+1}) = (J', j) | (X_t, N_t) = (J, i)]$$
$$= \begin{cases} f^{i,j} & J' = J = \emptyset, \\ b^{i,j} & J' = J \neq \emptyset, \\ d_k^{i,j} & J \neq \emptyset,\ f(J,1) = k,\ J' = J - f(J,1), \\ u_s^{i,j} & J' = J + s, s = 0, \ldots, d, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Note the transition probability between two nodes depends only on the spacial relationship between the two nodes and not on their specific values. Various numerical methods exist to examine the stability of a tree-like process and to determine its stationary behavior [4].

For the S&Q model we set d equal to 1 and let X_t represent the contents of the stack, that is, a class-2 job is represented by a 0, while a class-3 job is represented by a 1. The auxiliary variable N_t will keep track of the queue length, the state of the arrival process, as well as of the necessary information of the service process to make the system Markovian. For instance, if the job arrival process is a MMAP[3] and the class-i service times are geometric with a mean p_i, it suffices to define $N_t = (S_t, M_t, Q_t)$. M_t denotes the phase of the MMAP[3] job arrival process at time t. While $S_t = i$, for $i = 1, 2$ and 3, if a class-i job is executed at time t and $S_t = 0$ if there are no jobs in the system. Define Q_t as the queue contents at time n (where Q_t takes values in the range 0 to K).

The process $\{(X_t, N_t), t \geqslant 0\}$ is however not yet a tree-like process as its transitions are a lot more general than those presented in Eq. (1). For instance, on some occasions we need to push the entire contents of the queue on the stack at once, or we may need to pop a series of class-3 jobs from the stack to retrieve a class-2 job. Nevertheless, we can often reduce the Markov chain $\{(X_t, N_t), t \geqslant 0\}$ to a tree-like process. The technique proposed to construct an expanded Markov chain $\{(\mathscr{X}_t, \mathscr{N}_t), t \geqslant 0\}$ is similar to Ramaswami's [18] to reduce a classic M/G/1-type Markov chain to a QBD Markov chain or to the approach taken in Van Houdt et al. [19] to construct a tree structured QBD. The key idea behind this expanded Markov chain is that whenever a transition occurs that adds or removes a string of length r to the stack, we split this transition into $r+1$ transitions that each add or remove one integer to the node variable. Whenever we need to add or remove such a string, it will consist entirely of ones (class-3 jobs), except for its first element that might equal 0 (a class-2 job). This property makes it possible to create an expanded Markov chain $\{(\mathscr{X}_t, \mathscr{N}_t), t \geqslant 0\}$, where the range of its auxiliary variable $\mathscr{N}_t$ is only marginally larger than that of N_t. Details of this procedure will be provided in a forthcoming paper.

4. Conclusion

In this paper we have presented a link between the analysis of a general class of priority queues with three service classes and tree-like processes. A detailed discussion on how to transform the priority model to a S&Q problem was included. We also provided future guidelines on how to exploit this transformation to set up a tree-like process that enables us to assess performance measures of priority queues.

Acknowledgement

B. Van Houdt is a post-doctoral fellow of the FWO-Flanders.

References

[1] Kuehn PJ. Multiqueue systems with nonexhaustive cyclic service. Bell Syst Tech J 1979;58(3):671–98.

[2] Takine T, Sengupta B, Yeung RW. A generalization of the matrix M/G/1 paradigm for Markov chains with a tree structure. Stoch Models 1995;11(3):411–21.

[3] Yeung RW, Sengupta B. Matrix product-form solutions for markov chains with a tree structure. Adv Appl Probab 1994;26:965–87.

[4] Bini DA, Latouche G, Meini B. Solving nonlinear matrix equations arising in tree-like stochastic processes. Linear Algebra Appl 2003;366:39–64.

[5] Van Houdt B, Blondia C. Tree structured QBD markov chains and tree-like QBD processes. Stoch Models 2003;19(4): 467–82.

[6] Yeung RW, Alfa AS. The quasi-birth-death type markov chain with a tree structure. Stoch Models 1999;15(4):639–59.

[7] He Q, Neuts MF. Markov chains with marked transitions. Stoch Process Appl 1998;74:37–52.

[8] He Q. Queues with marked customers. Adv Appl Probab 1996;28:567–87.

[9] Miller R. Priority queues. Ann Math Stat 1960;31:86–103.

[10] Alfa AS. Matrix-geometric solution of discrete time MAP/PH/1 priority queue. Nav Res Logist 1998;45:23–50.

[11] Alfa AS, Liu B, He QM. Discrete-time analysis of MAP/PH/1 multiclass general preemptive priority queue. Nav Res Logist 2003;50:662–82.

[12] Machihara F. On the queue with PH-markov renewal preemption. J Oper Res Soc Jpn 1993;36:13–28.

[13] Takine T. A nonpreemptive priority MAP/G/1 queue with two classes of customers. J Oper Res Soc Jpn 1996;39(2): 266–90.

[14] Takine T, Hasegawa T. The workload in a MAP/G/1 queue with state-dependent services: its applications to a queue with preemptive resume priority. Stoch Models 1994;10(1): 183–204.

[15] Takine T, Hasegawa T. The nonpreemptive priority MAP/G/1 queue. Oper Res 1999;47(6):917–27.

[16] Latouche G, Ramaswami V. Introduction to matrix analytic methods and stochastic modeling. Philadelphia, PA: SIAM; 1999.

[17] Miller DG. Computation of steady-state probabilities for M/M/1 priority queues. Oper Res 1981;29(5):945–58.

[18] Ramaswami V. The generality of QBD processes, In Advances in matrix analytic methods for stochastic models. Neshanic Station, NJ: Notable Publications Inc.; 1998. p. 93–113.

[19] Van Houdt B, Blondia C. Throughput of Q-ary splitting algorithms for contention resolution in communication networks. Commun Inform Syst 2005;4(2):135–64.