Stochastic Petri Nets sensitivity to token scheduling policies

G. Balbo, M. Beccuti, M. De Pierro, and G. Franceschinis

Abstract Stochastic Petri Nets and their generalizations are powerful formalisms for the specification of stochastic processes. In their original definition they do not provide any specification for the token extraction order applied by a transition to its input places, however this aspect may be relevant if timed transitions are interpreted as servers and tokens as customers, so that the extraction order may be interpreted as a scheduling policy. In this paper we discuss how the token scheduling policies different from the Random Order one which is assumed by default, may be relevant for the computation of average performance indices.

1 Introduction

The original definition of Stochastic Petri Nets (SPN) [9] and of their later extensions (e.g. Generalized Stochastic Petri Nets [1] - GSPNs - and Stochastic Well-Formed Nets [5] - SWNs) do not provide any specification for the policy used to extract the tokens from the input place(s) of a transition upon its firing (queueing policy). This is due to the fact that when firing times have negative exponential distribution and tokens are indistinguishable, the specification of queueing policies become inessential as long as average performance indices are the objective of the analysis: hence tokens are picked from the input places in *Random Order* (RO).

When timed transitions are interpreted as servers, firing delays as activity durations, and tokens as customers that are *moved* from input to output places upon transition firings, response time *distributions* may become important and the need arises of specifying queueing policies for input places (an issue that has been studied in [2]) and of accounting for the relevant results of queueing theory that apply within this context. The impact that service policies have on the behavior of queueing models is well understood and average performance indices such as throughputs, utilizations, mean queue lengths, and mean response times are shown to be insensi-

G. Balbo · M. Beccuti · M. De Pierro
Dipartimento di Informatica, Università di Torino, e-mail: {balbo,beccuti,depierro}@di.unito.it

G. Franceschinis, Dipartimento di Informatica, Università del Piemonte Orientale "A. Avogadro", e-mail: giuliana.franceschinis@mfn.unipmn.it

tive to queueing policies, as long as they are work-conservative and the service time distribution is negative exponential [6]. Things start to be more complex if general service time distributions are considered in conjunction with queueing policies that exhibit preemptive or sharing features. Indeed, it is well known [6, 7, 3] that the average performance indices computed for an M/G/1 queue with Processor Sharing (PS) or Last-Come-First-Served Preemptive-Resume (LCFS-PR) queueing policies are identical to those of M/M/1 queues with same mean service times while they differ considerably when the queueing policy is First-Come-First-Served (FCFS): in this case the coefficient of variation (CV) of the service time distribution plays an important role. Hence, when generally distributed firing times are considered in SPNs (possibly represented through subnets of exponential transitions, in case of phase type distributions [1]) token queueing policies should not be overlooked.

Things become even more intriguing when we turn our attention towards High Level Stochastic Petri nets, such as SWNs, where tokens are *colored* and are thus distinct. In this case both the extraction criteria from the input place(s) and the service time may depend on the token *color*: this is implemented through arc functions and color dependent rates. A peculiar feature of SWNs is a carefully defined syntax, which allows to automatically exploit behavioral symmetries when performing state space based analysis. In case of completely symmetric SWNs and under some additional restrictions (on color synchronization), it may still happen that queueing policies do not influence the average performance indices, however if the color of tokens representing customers are used to model different behaviors (through *static subclasses*) or to synchronize tokens (as in the fork-join example of Sec. 2), the queueing policy may have an impact.

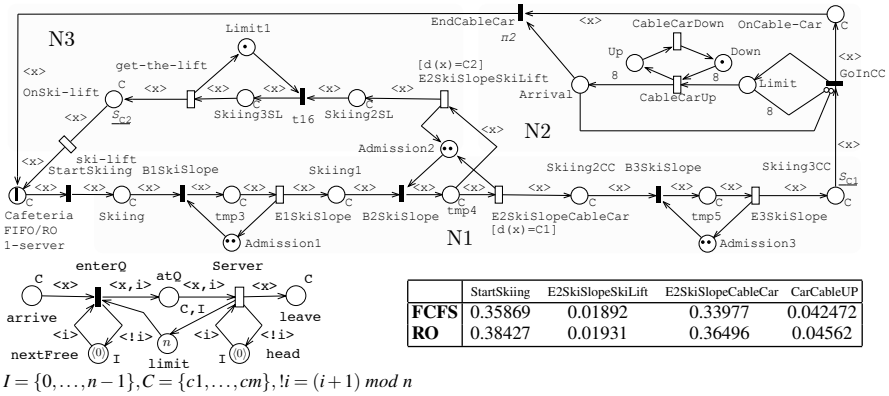
In this paper we address these last issues by presenting the results obtained for a set of SWN models which reflect some of these situations (Sec. 2). In the first example a case is discussed where the queueing policy at a service station with negative exponential service time and color independent rate is relevant, due to batch arrivals of tokens of a subclass. In the second example the effect of a hyper-exponential service time distribution is first considered, then the impact of different queueing policies on the branches of a fork-join structure is studied. Finally we consider a simplified version of a Queueing PN (QPN) model [4] proposed in [8], for which it was observed the importance of the queueing policy of a *buffer place* due to the presence of color dependent service rates.

The examples discussed in this paper are meant to provide evidence that disregarding the order of extraction of tokens from places may produce misleading results. As such, they represent the basis for the characterization of SWN model classes that are insensitive to the queueing policy, on which we are currently working. For the moment we emphasize the importance of adding some *syntactic sugar* to the SWN formalism thus providing some new primitives similar to those proposed in QPNs. In practice, this amounts to the development of compact submodels that can be automatically embedded in the SWN representation of a system: in Sec. 3 we discuss how this can be done without giving up the possibility of using the efficient Symbolic Reachability Graph (SRG) algorithm, that exploits model symmetries and considerably reduces its state space size.

2 Token queueing policies: some examples

Before discussing the SWN examples, let us introduce some notation used in the models; for more details on the formalism the reader can refer to [5]. The SWN formalism is a flavor of Colored Petri Nets with a well-defined syntax. Tokens in SWN can be distinguished through *colors*, i.e. they may carry some kind of information. In all the models discussed in this paper, we have only one color class: C in the models of Figs. 1 and 2, D in the model of Fig. 3. A class can be partitioned into static subclasses, e.g in Fig. 1, $C = C1 \cup C2$: intuitively colors in the same class have the same nature (e.g. skiers); while those in the same subclass share also the same potential behavior (e.g. skiers using the ski-lift). A color domain is associated with places and transitions. The colors of a place label the tokens that it contains (e.g place *Busy* in Fig. 2 has color domain C). Instead, the color domain of transitions define different *firing modes*. In order to specify the effect of each firing mode, a function is associated with each arc, specifying the colored tokens that are added to, or withdrawn from, the corresponding place when the transition fires in a given mode. A SWN color function is defined combining predefined basic functions: in all the examples only one function is used - $\langle x \rangle$ - which selects an item of a transition color tuple. Finally, transitions can have guards, specified through boolean expressions, whose terms are predefined atomic predicates like $[x = y]$. For example, the guard $[d(x) = C1]$ of transition *E2SkiSlopeCableCar* in Fig. 1 means that this transition may only fire for a color x belonging to static subclass $C1$. All timed transition in our examples will have as many modes enabled in parallel as the number of customers in their input place. In all models we use a notation which is not part of the SWN formalism, but is instead inherited from QPNs: queue places are drawn as circles with a vertical bar and represent a service center for which a queueing policy and a service time distribution are defined. Before proceeding with the solution, these places must be substituted with appropriate subnets, depending on the characteristics of the corresponding service center; subsequently the Continuous Time Markov Chain underlying the model is constructed and analyzed. Fig. 1 (lower) shows an example of subnet corresponding to a queue place with FCFS queueing policy, a single server and negative exponential service time. The queue is modeled by place atQ, with color domain $C \times I$, representing a circular array (I encodes the circularly ordered set of indices). Places *head* and *nextFree* represent the pointers to the first element in queue and to the first empty element, respectively. Transition *Server* has at most one enabled instance at a time, that upon firing removes from place atQ the token $< x, i >$ with i equal to the color of the (unique) token in place *head*, and updates the color of the token in *head* to the successor of i denoted $!i$.

The impact of batch arrivals: the skiers model. The customers in this example are skiers that get to the top of a mountain using either a cable-car (subnet $N2$) or a ski-lift (subnet $N3$): hence color class C representing customers, is partitioned in two static subclasses $C1$ and $C2$. When the skiers get to the top of the mountain, they all stop at the cafeteria for a cup of coffee (queue place *Cafeteria*). After drinking the coffee they start skiing down-hill along a common path with sections that, being very narrow, allow only two skiers to proceed "in parallel" (subnet $N1$). Then the



	StartSkiing	E2SkiSlopeSkiLift	E2SkiSlopeCableCar	CarCableUP
FCFS	0.35869	0.01892	0.33977	0.042472
RO	0.38427	0.01931	0.36496	0.04562

$I = \{0,\ldots,n-1\}, C = \{c1,\ldots,cm\}, !i = (i+1)\ mod\ n$

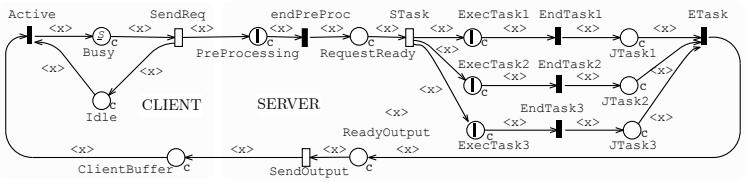
Fig. 1 SWN model for the skiers example (upper fig.); SWN submodel of FCFS policy (lower fig.); Table of the throughputs for RO or FCFS queueing policy at the cafeteria.

ski-lift skiers queue for the lift (place *Skiing2SL*), while the others proceed along the track until they get to the cable-car station (place *Skiing3CC*): the cable-car leaves when K skiers are ready to board.

Our experiments show that *Cafeteria* queueing policy (RO or FCFS) has an effect on the performance indices of the model, despite the service center behavior is completely symmetric w.r.t. the two types of customers. In the table of Fig. 1 we assume $K = 8$, $|C1| = 10$ and $|C2| = 2$ and we compare some throughput obtained with the two queueing policies. These results are due to the fact that the skiers belonging to $C1$ arrive in batch of size 8 to the cafeteria. The difference becomes smaller and smaller as the batch size (K) decreases.

The effect of service time distribution and of color-based synchronization: the Client-Server model. The model in Fig. 2 represents a client-server architecture where several similarly behaving clients, modeled by the color class C, can request a remote service from a server. When a client sends a request to the server, the message is queued in the server buffer for pre-processing (queue place *PreProcessing*) and next it is served by generating three threads that are executed in parallel (places $ExecTask_i$). The output is sent back to the client when all three threads end. The first experiment shows that depending on the CV of the pre-processing execution time, the queueing policy of queue place *PreProcessing* may have an impact on the performance indices. In Fig. 2, left table, the system throughput and the mean number of customers in the queueing place *PreProcessing* are compared when $|C| = 6$, the queueing policy of *PreProcessing* is either FCFS or PS, and the pre-processing execution time has a hyper-exponential distribution (rates $\lambda_1 = 1$ and $\lambda_2 = 0.0006622517$ are chosen with probability $p = 0.9998$ and $1-p$, respectively). The queueing policy of *PreProcessing* is irrelevant when the execution time has negative exponential distribution.

This model allows also to show that the fork/join construct, although completely symmetric, is sensitive to the queueing policies used in its branches. Experiments have been performed with PS queueing policy and negative exponential service time in the queue place *PreProcessing*, as well as queue places *ExecTaski* have either PS



PreProcessing	FCFS	PS
Throughput	0.54889	0.72409
Mean queue length	2.16360	0.71958

	3-PS	2-PS & 1-FCFS	1-PS & 2-FCFS	3-FCFS
System Throughput	0.657712	0.662527	0.669431	0.678263
$\sum_i E[Jtaski]$	3.75654	3.6294	3.4154	3.13833

Fig. 2 SWN model for the Client-Server example; Performance indices of the Client-Server model varying the queueing policy: (left) in PreProcessing with hyper exponential service time; (right) in the fork/join construct.

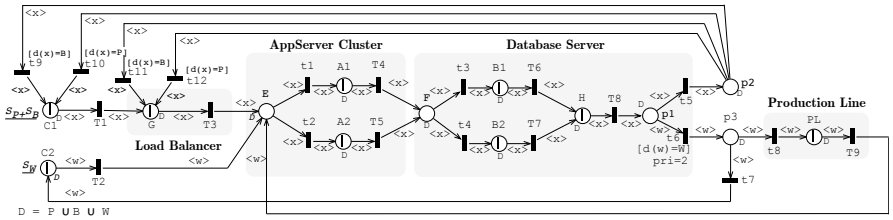
or FCFS queueing policy. In the right table of Fig. 2 we report the system throughput and the sum of the average numbers of tokens in places *JTaski* for these cases.

Distributed component-based model. This example is inspired by the QPN model proposed in [8]; it represents a distributed component-based system consisting of four parts: Load Balancer (LB), Application Server Cluster (APC), Database, and Production Line Station interacting to manage dealer, order entry, and manufacturing applications. The corresponding SWN model is shown in Fig. 3; where the system applications are modeled by color class D which is partitioned in three static subclasses: B dealer applications, P order entry applications and W manufacturing applications. The LB distributes the incoming B and P requests across the nodes in the APC depending on their loads. All service times depend on the type (static subclass) of application. The APC subnet models a cluster which processes the incoming B, P, and W applications. Each node implements a PS queueing policy (places *A1* and *A2*). Processed tasks are submitted to the Database subnet for their execution. Each task execution involves a CPU burst, followed by an I/O burst. The CPU is assigned to each task using a PS policy. Then, a disk access is required for each task according to a FCFS policy (place H). Finally, the last subnet represents the production line station: a w task in place PL models the execution of a manufacturing order by the production line stations.

Similarly to what was observed in [8], the queueing policy applied to the LB component (place G) influences the system performance. In our experiment we studied the effects of the FCFS and PS policies, assuming $|B| = 2$, $|P| = 2$ and $|W| = 3$. The table shown in Fig. 3 reports the throughputs and the average number of customers in some of the queue places.

3 Discussion and concluding remarks

The examples presented in the previous section show that when developing SWN models, place queueing policies may have an impact on average performance indices. At first sight this observation may appear obvious, however we believe it is important because the interplay between queueing policies and color-driven behavior, or non-exponential (possibly phase type) firing time distribution, may be overlooked when modelers are used to work with formalisms characterized by undistinguishable tokens and negative exponential firing delays. This suggests that pro-



	Mean queue lengths		Throughputs	
	FCFS	**PS**	**FCFS**	**PS**
Load Balancer	0.501757	0.60422	0.01575875 (67%P 33%B])	0.01482976 (60%P 40%B)
Production Line	1.85466	1.86103	0.00556954 (W)	0.00558867 (W)
Database Disk	0.155609	0.143083	0.02689785 (40%P 19%B 41%W)	0.02600706 (34%P 23%B 43%W)

Fig. 3 SWN model for the distributed component-based example; throughputs and average number of customers computed for different queueing policies at the LB.

viding the modeler with syntactic elements favoring the explicit specification of queueing policies as well as firing time distributions in a compact form, may help to build correct models. Moreover we are interested in embedding these new features in SWN models still preserving the most interesting feature of SWNs, that is their efficient state space analysis algorithms based on symmetries exploitation; the examples considered in this paper were solved by substituting the queue places with relatively simple subnets satisfying the SWN syntax constraints: this transformation could be automated, allowing to exploit the standard efficient analysis algorithms based on the SRG. For instance, in the Client-Server model, the state space reduction achieved by the SRG w.r.t. to the RG is ~ 275 when the queue places $ExecTask_i$ have PS queueing policy; while it is ~ 3000 when their queueing policy is FCFS.

References

1. M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. J. Wiley, New York, NY, USA, 1995.
2. G. Balbo, M. Beccuti, M. De Pierro, and G. Franceschinis. First passage time computation in Tagged GSPN with queue places. *Computer Journal*, 2010. Accepted for publication.
3. F. Baskett, K. M. Chandy, R. R. Muntz, and F. G. Palacios. Open, closed, and mixed networks of queues with different classes of customers. *J. ACM*, 22(2):248–260, 1975.
4. F. Bause and P.S. Kritzinger. *Stochastic Petri Nets - An Introduction to the Theory*. Friedr. Vieweg & Sohn Verlag, Braunschweig/Wiesbaden, Germany, Second Edition, 2002.
5. G Chiola, C. Dutheillet, G. Franceschinis, and S. Haddad. Stochastic well-formed coloured nets for symmetric modelling applications. *IEEE Trans. on Computers*, 42(11):1343–1360, nov 1993.
6. E. Gelenbe and I. Mitrani. *Analysis and Synthesis of Computer Systems*. Academic Press, London, 1980.
7. L. Kleinrock. *Queueing Systems. Volume 1: Theory*. J. Wiley, New York, NY, USA, 1975.
8. Samuel Kounev. Performance Modeling and Evaluation of Distributed Component-Based Systems Using Queueing Petri Nets. *IEEE Trans. Softw. Eng.*, 32(7):486–502, 2006.
9. M. K. Molloy. Performance analysis using Stochastic Petri Nets. *IEEE Tr. on Computers*, 31(9):913–917, 1982.