Transform Analysis of Preemption Overhead in the M/G/1

Shefali Ramakrishna
Cornell University
sr899@cornell.edu

Ziv Scully
Cornell University
zivscully@cornell.edu

1. INTRODUCTION

Preemptive scheduling policies, which allow pausing jobs mid-service, are ubiquitous because they allow important jobs to receive service ahead of unimportant jobs that would otherwise delay their completion. The canonical example is Shortest Remaining Processing Time (SRPT), which preemptively serves the job with least remaining work at every moment in time [9]. There is a robust literature analyzing *response time* (elapsed time between a job's arrival and completion) in the M/G/1 queue under many preemptive policies [6, 10, 11], shedding light on questions such as how preemption affects the mean and tail of response time, and whether preemption is unfair towards low-priority jobs.

In practice, there is a cost to preempting individual jobs, known as *preemption overhead*. For instance, in computer systems, overhead can take the form of context switch, reloading from disk into memory, or reloading from memory into cache. However, with very few exceptions (§ 1.2), the queueing theory literature does not account for preemption overhead. It is thus unclear how overhead affects preemption tradeoffs.

In light of overhead, one might hope to alter preemptive policies to reduce the amount of preemption, and thus overhead, that occurs. An example is making the job in service "resistant" to preemption by increasing its priority. But with one exception (§ 1.2), policies with such resistance have not been analyzed, even without overhead. We thus lack the tools needed to theoretically understand preemption resistance.

1.1 Contributions and techniques

We give the first transform response time analysis of the M/G/1 queue under preemptive priority scheduling with preemption overhead. Our overhead model is flexible: overhead amounts are stochastic and can occur whenever a job is paused, resumed, or both. We derive recursive formulas for the Laplace transform (hereafter simply "transform") of each priority class's response time distribution, which yield closed-form formulas for all moments of response time. Our techniques further extend to analyze "preemption resistance".

At the heart of our analysis is a new queueing model we call the *arrival-sensitive M/G/1* (M/G/1/AS). The name refers to a job's service time being affected by arrivals during its service. One of our key insights is that under preemptive priority scheduling, preemption overhead can be viewed as a type of arrival sensitivity. That is, when a high-priority job J_1 arrives during service of a low-priority job J_2, we view J_1's arrival as increasing J_2's service time.

This work was supported by the NSF under grant no. CMMI-2307008.

Among the obstacles we overcome is analyzing *busy periods* in the M/G/1/AS. Recall that a busy period started by a set of jobs is the total service time of the initial set of "level 0" jobs, "level 1" arrivals that occur during service of level 0 jobs, "level 2" arrivals during level 1 jobs, etc. Busy period transforms play a critical role in many prior M/G/1 response time transform analyses [6, 9, 11]. However, arrival-sensitivity complicates busy period analysis, because it introduces a new dependency between a job's service time and the number of arrivals that occur during it. We develop a new toolbox for analyzing M/G/1/AS busy periods based on *multivariable* transforms that capture this new dependence. We then derive the required multivariable transforms for our preemption overhead model.

In this paper, we introduce the M/G/1/AS model and analyze its busy periods (§ 2), express a flexible model of preemptive priority with overhead and using the M/G/1/AS (§ 3), state our response time results for a system with preemption overhead (§ 4), and briefly describe how our results and techniques extend to handle resistance (§ 5). We view our work as a first step towards incorporating preemption overhead into SOAP [11] and other M/G/1 scheduling theory.

1.2 Prior work

Overhead from preempting individual jobs has not been studied extensively in queueing. There are two prior works considering preemption overhead. Goerg [5] studies SRPT with overhead and resistance, but only considers mean response time and has a less flexible preemption model than ours. Specifically, overhead is deterministic and only occurs when pausing jobs, not resuming them. Peng [8] studies preemptive priority with overhead under a more flexible preemption model, which we also adopt (§ 3), but only considers mean response time and does not consider resistance.

Most literature on overheads in queues covers overheads from switching between *classes* of jobs [7, 12], or between *queues* of jobs in polling systems [1–3]. This work does not capture preemption overhead because almost all the policies studied never switch classes or queues during a job's service. To the best of our knowledge, the only exception is the work of Cao and Xie [3], but it studies a preempt-restart model, meaning that preempted jobs do not retain progress. In contrast, we study a preempt-resume model, meaning progress on a job is retained even after it is preempted.

2. THE ARRIVAL-SENSITIVE M/G/1

Like the ordinary M/G/1, the *arrival-sensitive M/G/1* is a single-server queueing system with Poisson job arrivals at rate λ. The main difference is the service dynamics, which build on the *Markov-process job model* of Scully [10] by adding arrival sensitivity.

Each job is an i.i.d. continuous-time absorbing Markov process, with absorption representing completion. A job's state evolves during service and stays static when waiting in the queue. This is the ordinary Markov-process job model. We add one more feature: whenever a job arrives, the job in service may undergo a state transition. This transition is specified by a transition kernel, i.e. the post-arrival state's distribution depends only on the pre-arrival state.

The M/G/1/AS is a highly general model because Markov processes are very flexible. A priori, one might worry that analyzing the M/G/1/AS requires discussing the specifics of the job Markov process. Our first key insight is that lots of information about the M/G/1/AS can be extracted from a *multidimensional* transform (§ 2.1). As such, we do not define notation for discussing the job Markov process directly.

Throughout, we denote the LST of V by $\widetilde{V}$, and we denote the *excess* of V [6] by $\mathcal{E}V$. In particular, $\widetilde{\mathcal{E}V}(\theta) = \frac{1-\widetilde{V}(\theta)}{\theta \mathbf{E}[V]}$.

2.1 Job transform

In the M/G/1/AS, each job has some service time, and some number of arrivals occur during that service. We denote by (R, A) the *joint* distribution of service time R and arrivals-during-service A. In the ordinary M/G/1, R is the same for all arrival rates λ, and we have $A \sim \text{Poisson}(\lambda R)$ conditional on R. But in the M/G/1/AS, both properties may fail.

Fortunately, one can still show that each job's service time and arrivals-during-service are drawn i.i.d. from (R, A). We omit the proof, which is a straightforward use of the homogeneity of Poisson arrivals, for lack of space. We therefore define the *job transform*

$$\mathcal{J}(\theta, z) = \mathbf{E}[e^{-\theta R} z^A].$$

Stability conditions can be directly extracted from $\mathcal{J}(\theta, z)$. By Wald's equation and Galton-Watson branching process theory, the system is stable if $\rho = \mathbf{E}[A] < 1$. In this situation, ρ is also the *load* (a.k.a. utilization) of the system, and a renewal-reward [6] argument shows

$$\rho = \lambda \mathbf{E}[R] = \mathbf{E}[A] \quad \text{and} \quad \rho = -\lambda \partial_\theta \mathcal{J}(0, 1) = \partial_z \mathcal{J}(0, 1).$$

We analyze M/G/1/AS in terms of $\mathcal{J}(\theta, z)$. The *busy period* started by a job, denoted $B(\mathcal{J})$, is the total service time of that "level 0" job, and all "level h" arrivals, where level $h \geq 1$ arrivals are those that occur during service of level $h-1$ jobs. The busy period started by an amount of work V, denoted $B(V)$, is defined similarly, except the level 0 job is replaced by a (non-arrival-sensitive) amount of work V.

Theorem 2.1. *The busy period transform in M/G/1/AS is*

$$\widetilde{B}(\mathcal{J})(\theta) = \mathcal{J}(\theta, \widetilde{B}(\mathcal{J})(\theta)),$$
$$\widetilde{B}(V)(\theta) = \widetilde{V}\big(\theta + \lambda(1 - \widetilde{B}(\mathcal{J})(\theta))\big).$$

2.2 Multiclass M/G/1/AS

In the M/G/1/AS, we also want to be able to distinguish between different classes of arrivals, as each class of arrival may affect service time differently. For instance, in the preemption overhead setting, only arrivals of a lower class will cause a preemption and thus induce overhead. Then R_k, the service time of a class k job, has a different dependence on the number of each class i of arrival $A_{k,i}$. The job transform $\mathcal{J}$ for a class k job, denoted $\mathcal{J}_k$, is defined below, taking a *vector* $\mathbf{z} = [z_i]_{i=1}^n$ instead of a scalar z:

$$\mathcal{J}_k(\theta, \mathbf{z}) = \mathbf{E}\Big[e^{-\theta R_k} \prod_{i=1}^n z_i^{A_{k,i}}\Big].$$

Analogous to Theorem 2.1, we can derive the transform of busy periods from $\mathcal{J}_k$. More generally, we can analyze *class $< l$ busy period*, denoted $B_{<l}(\cdot)$. These are defined analogously to busy periods $B(\cdot)$ in § 2.1, except for all levels $h \geq 1$, we only consider a job to be "level h" if it is also class $< l$. They play an important role in many response time analyses [5, 6, 8–11], including ours (§ 4).

Theorem 2.2.

$$\widetilde{B}_{<l}(\mathcal{J}_k)(\theta) = \mathcal{J}_k\big(\theta, \big[\mathbb{1}(i < l)\, \widetilde{B}_{<l}(\mathcal{J}_i)(\theta) + \mathbb{1}(i \geq l)\big]_{i=1}^n\big),$$
$$\widetilde{B}_{<l}(V)(\theta) = \widetilde{V}\big(\theta + \lambda_{<l}(1 - \widetilde{B}_{<l}(\mathcal{J}_{<l})(\theta))\big).$$

Above and throughout, we use the following notation conventions. For arrival rates and loads, subscript $< k$ denotes *sum*, e.g. $\lambda {<} k = \sum_{i=1}^{k-1} \lambda_i$. For busy periods, subscript $< k$ denotes a class $< k$ busy period (§ 2.2). For other distributions, subscript $< k$ denotes *mixture*, e.g. $\mathcal{J}_{<k}(\theta, z) = \sum_{i=1}^{k-1} \frac{\lambda_i}{\lambda_{<k}} \mathcal{J}_i(\theta, z)$. Subscripts $\leq k$ and $> k$ are analogous.

3. PREEMPTION OVERHEAD

Having obtained the transform for arrival-sensitive busy periods, we will apply it to a system with preemption overhead. We first describe our model and introduce some notation.

We are considering a system with a single server, Poisson arrivals, and *static priority*, meaning each arriving job has a class $k \in \{1, \ldots, n\}$ that remains unchanged during its time in the system. The scheduling policy is *preemptive priority*, meaning that at each point in time, the job with lowest class is being served. If a job of class i arrives during service of a class k job, and $i < k$, the class k job will be preempted to serve the class i job, incurring pause overhead when it is preempted and resume overhead when it resumes service.

We will need the following values for response time analysis:

- λ_k: the arrival rate of class k jobs.
- C_k: the distribution of a class k job's pause overhead.
- D_k: the distribution of a class k job's resume overhead.
- S_k: the distribution of a class k job's service time, *excluding* overhead.
- R_k: the distribution of a class k job's full service time, *including* overhead.
- γ_k: the load of class k jobs' pause overheads.
- δ_k: the load of class k jobs' resume overheads.
- $\sigma_k = \lambda_k \mathbf{E}[S_k]$: the load of class k jobs' service times, *excluding* overhead.
- $\rho_k = \sigma_k + \gamma_k + \delta_k$: the load of class k jobs' service times, *including* overhead.
- $B_{<k}(\mathcal{J}_i)$: a class $< k$ busy period started by a class i job, *including* overhead.
- $B_{<k}(C_i)$ and $B_{<k}(D_i)$: a class $< k$ busy period started by a class i pause or resume, respectively.

As a first step to analyzing preemption overhead, we compute the job transform induced by our overhead model.

Theorem 3.1. *The job transform for a class k job is*

$$\mathcal{J}_k(\theta, \mathbf{z}) = \mathcal{J}_k^*\bigg(\theta + \sum_{i=k}^n \lambda_i(1 - z_i), \sum_{i=1}^{k-1} \frac{\lambda_i}{\lambda_{<k}} z_i\bigg),$$

where

$$\mathcal{J}_k^*(\theta, z) = \widetilde{S}_k\big(\theta + \lambda_{<k}\big(1 - z\,\mathcal{O}_k(\theta, \lambda_{<k}(1-z))\big)\big),$$

$$\mathcal{O}_k(\theta, \eta) = \frac{\widetilde{C}_k(\theta+\eta)\,\widetilde{D}_k(\theta+\lambda_{<k})}{1 - \widetilde{C}_k(\theta+\eta)\big(\widetilde{D}_k(\theta+\eta) - \widetilde{D}_k(\theta+\lambda_{<k})\big)}.$$

From Theorem 3.1, we can compute γ_k and δ_k, obtaining

$$\gamma_k = \frac{\lambda_{<k}\sigma_k}{\widetilde{D}_k(\lambda_{<k})}\mathbf{E}[C_k] \qquad \delta_k = \frac{\lambda_{<k}\sigma_k}{\widetilde{D}_k(\lambda_{<k})}\mathbf{E}[D_k].$$

4. RESPONSE TIME ANALYSIS

We use this joint transform $\mathcal{J}$ to analyze the transform of response time for a preemptive-priority scheduling policy in a system with preemption overhead. With the busy period transforms from Theorem 2.2 in hand, we can use a "tagged-job" style analysis [6, 9, 11].

A class k job must wait behind all class $< k$ jobs present in the system at the time of its arrival, as well as all class $< k$ jobs that arrive while it is waiting in the system. To analyze a class k job's response time, we make the following key observation: the class k job will not leave until the entire class $< k$ busy period started by it completes.

This means we can nearly view a class k job's response time as the response time of M/G/1 with arrival rate λ_k and job size distribution $B_{<k}(\mathcal{J}_k)$. Specifically, the job sizes in the modified system correspond to class k *residence times* in the original system. A job's residence time is the amount of time between when it starts service and when it completes [6]. However, there are some delays not yet accounted for. Specifically, a class k job experiences additional delay if it

(a) arrives during a class $> k$ job's resume,
(b) arrives during a class $> k$ job's pause,
(c) arrives during a class $> k$ job's service and causes a preemption, or
(d) arrives during a class $< k$ job's service that is not yet accounted for as part of a class k residence time.

We account for these extra delays by adding a type of *generalized vacations* to the modified M/G/1, in the sense of Fuhrmann and Cooper [4]. Applying their decomposition theorem reduces the response time analysis to determining X_k, the response time distribution of class k jobs that do *not* arrive during another class k job's residence time.

Theorem 4.1. *The transform for the response time of a class k job in a system with preemption overhead is*

$$\frac{(1-\rho_{<k}-\rho_k)\widetilde{B}_{<k}(\mathcal{J}_k)(\theta)}{1-\rho_{<k}-\rho_k\widetilde{\mathcal{E}B}_{<k}(\mathcal{J}_k)(\theta)}\widetilde{X}_k(\theta),$$

where

$$\begin{aligned}\widetilde{X}_k(\theta) = &\sum_{j=k+1}^{n}\frac{\lambda_j}{\lambda_{>k}}\frac{\delta_{>k}}{1-\rho_{\le k}}\widetilde{\mathcal{E}B}_{<k}(D_j)(\theta)\,\widetilde{B}_{<k}(C_j)(\theta)\\ &+\sum_{j=k+1}^{n}\sum_{i=1}^{j-1}\frac{\lambda_i}{\lambda_{<j}}\frac{\gamma_j}{1-\rho_{\le k}}\widetilde{\mathcal{E}B}_{<k}(C_j)(\theta)\big(\widetilde{B}_{<k}(\mathcal{J}_i)(\theta)\big)^{\mathbb{1}(i<k)}\\ &+\frac{(1-\rho_{<k})\sigma_{>k}}{1-\rho_{\le k}}\widetilde{B}_{<k}(C_{>k})(\theta)\\ &+\frac{(1-\rho+\sigma_{>k})\rho_{<k}}{1-\rho_{\le k}}\widetilde{\mathcal{E}B}_{<k}(\mathcal{J}_{<k})(\theta).\end{aligned}$$

The main challenge is computing $\widetilde{X}_k(\theta)$. Roughly speaking, items (a)–(c) above correspond to the first three terms in $\widetilde{X}_k(\theta)$, respectively, and item (d) corresponds to the fourth term and the class $< k$ busy periods throughout.

5. EXTENSION: ADDING "RESISTANCE"

A scheduling policy that potentially balances the costs and benefits of preemption is what we call *preemptive priority with resistance.* This is like preemptive priority, except a class k job is treated as class $r(k) \le k$ while in service, and thus is only preemptible by jobs of class $< r(k)$.

The same approach used to derive Theorem 4.1 can also handle preemption resistance, but there are two new challenges to solve. First, X_k becomes much more complicated. This is because we have to account for class k jobs that arrive during class l with $1 < r(l) \le k < l$. Such jobs resist preemption by class k but might be preempted by other classes. We thus have to account for class k jobs being delayed by "partial" class l jobs. Second, class k residence times become more complicated. Without resistance, a class k job's residence time is the class $< k$ busy period the job starts. But if the job resists preemption, then some jobs from its class $< k$ busy period may be served after it departs. Accounting for this involves reasoning about partial class k jobs.

We have seen that both of the main challenges involve reasoning about partial jobs. We overcome these challenges by defining and computing additional multivariable transforms. For example, to compute a class k job's residence time, we derive the joint transform of a class k job's total service time and the "unpreempted last part" of its service time.

References

[1] Marko A. A. Boon, Rob D. van der Mei, and Erik M. M. Winands. 2011. Applications of Polling Systems. *Surv. Oper. Res. Manag. Sci.* 16, 2, 67–82.

[2] Sem C. Borst and Onno J. Boxma. 2018. Polling: Past, Present, and Perspective. *TOP* 26, 3, 335–369.

[3] Jianyu Cao and Weixin Xie. 2017. Stability of a Two-Queue Cyclic Polling System with BMAPs under Gated Service and State-Dependent Time-Limited Service Disciplines. *Queueing Syst.* 85, 1, 117–147.

[4] S. W. Fuhrmann and Robert B. Cooper. 1985. Stochastic Decompositions in the $M/G/1$ Queue with Generalized Vacations. *Oper. Res.* 33, 5, 1117–1129.

[5] Carmelita Goerg. 1986. Evaluation of the Optimal SRPT Strategy with Overhead. *IEEE Trans. Commun.* 34, 4, 338–344.

[6] Mor Harchol-Balter. 2013. *Performance Modeling and Design of Computer Systems: Queueing Theory in Action.* Cambridge University Press, Cambridge, UK.

[7] Marcel F. Neuts. 1977. The $M/G/1$ Queue with Several Types of Customers and Change-over Times. *Adv. Appl. Probab.* 9, 3, 604–644.

[8] Edwin Peng. 2022. Exact Response Time Analysis of Preemptive Priority Scheduling with Switching Overhead. *SIGMETRICS Perform. Eval. Rev.* 49, 2, 72–74.

[9] Linus E. Schrage and Louis W. Miller. 1966. The Queue $M/G/1$ with the Shortest Remaining Processing Time Discipline. *Oper. Res.* 14, 4, 670–684.

[10] Ziv Scully. 2022. *A New Toolbox for Scheduling Theory.* Ph. D. Dissertation. Carnegie Mellon University, Pittsburgh, PA.

[11] Ziv Scully, Mor Harchol-Balter, and Alan Scheller-Wolf. 2018. SOAP: One Clean Analysis of All Age-Based Scheduling Policies. *Proc. ACM Meas. Anal. Comput. Syst.* 2, 1, Article 16, 30 pages.

[12] Mark P. Van Oyen, Dimitrios G. Pandelis, and Demosthenis Teneketzis. 1992. Optimality of Index Policies for Stochastic Scheduling with Switching Penalties. *J. Appl. Probab.* 29, 4, 957–966.