SOAP: One Clean Analysis of All Age-Based Scheduling Policies

Ziv Scully
Carnegie Mellon University
Computer Science Department
Pittsburgh, PA, USA
zscully@cs.cmu.edu

Mor Harchol-Balter
Carnegie Mellon University
Computer Science Department
Pittsburgh, PA, USA
harchol@cs.cmu.edu

Alan Scheller-Wolf
Carnegie Mellon University
Tepper School of Business
Pittsburgh, PA, USA
awolf@andrew.cmu.edu

ABSTRACT

We consider an extremely broad class of M/G/1 scheduling policies called SOAP: Schedule Ordered by Age-based Priority. The SOAP policies include almost all scheduling policies in the literature as well as an infinite number of variants which have never been analyzed, or maybe not even conceived. SOAP policies range from classic policies, like first-come, first-serve (FCFS), foreground-background (FB), class-based priority, and shortest remaining processing time (SRPT); to much more complicated scheduling rules, such as the famously complex Gittins index policy and other policies in which a job's priority changes arbitrarily with its age. While the response time of policies in the former category is well understood, policies in the latter category have resisted response time analysis. We present a universal analysis of all SOAP policies, deriving the mean and Laplace-Stieltjes transform of response time.

The full version of this work appears in POMACS [6].

CCS CONCEPTS

• **General and reference** → **Performance**; • **Mathematics of computing** → **Queueing theory**; • **Software and its engineering** → **Scheduling**; • **Computing methodologies** → *Model development and analysis*; • **Theory of computation** → Scheduling algorithms;

KEYWORDS

M/G/1; exact response time analysis; Gittins index; shortest expected remaining processing time (SERPT)

ACM Reference Format:
Ziv Scully, Mor Harchol-Balter, and Alan Scheller-Wolf. 2018. SOAP: One Clean Analysis of All Age-Based Scheduling Policies. In *SIGMETRICS '18 Abstracts: ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems Abstracts, June 18–22, 2018, Irvine, CA, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3219617.3219632

1 INTRODUCTION

Analyzing the response time of scheduling policies in the M/G/1 setting has been the focus of countless papers over the past half century. Although there has been much success in analyzing specific scheduling policies, such analyses are *limited to relatively simple scheduling policies*, such as first-come, first-served (FCFS), shortest remaining processing time (SRPT), and foreground-background (FB). Analyzing variants of these policies, let alone fundamentally different policies, is an open problem. For instance, none of the following scenarios have been analyzed before.

- We may have jobs that are neither fully preemptible nor fully nonpreemptible, but instead are preemptible only at specific "checkpoint" ages. We run a preemptive policy, say SRPT or FB, but only preempt jobs when they reach checkpoints.
- The *shortest expected remaining time policy* (SERPT) is a natural alternative to SRPT in scenarios when job sizes are not known. Aside from specific cases where SERPT is equivalent to a simpler policy, SERPT has not been analyzed before.
- The *Gittins index policy* [1, 3] has long known to be optimal for minimizing mean response time in the M/G/1 queue. In general, the Gittins index policy can have a complex priority scheme [2] which, while known to perform optimally, has only been analyzed in certain special cases [4, 5].

Approaching the above examples with state-of-the-art techniques, if possible at all, would require an ad-hoc analysis for each scenario. We seek *general principles and techniques* for response time analysis that apply to not just the above examples but to as many scheduling policies as possible, even those not yet imagined.

1.1 Contributions

We introduce *SOAP*, a *universal framework* for defining and analyzing M/G/1 scheduling policies. The SOAP framework can analyze *any SOAP scheduling policy*, which includes nearly any policy where a job's priority depends on its own characteristics: class, size, age, and so on. Specifically, we make the following contributions.

- We *define* the class of SOAP policies, a broad class of policies that includes the unsolved examples above as well as many other policies, from classic policies to those not yet imagined.
- We give a *universal response time analysis* that works for any SOAP policy, obtaining closed forms for the mean and Laplace-Stieltjes transform. We apply our results to analyze *previously intractable policies* like the Gittins index policy.

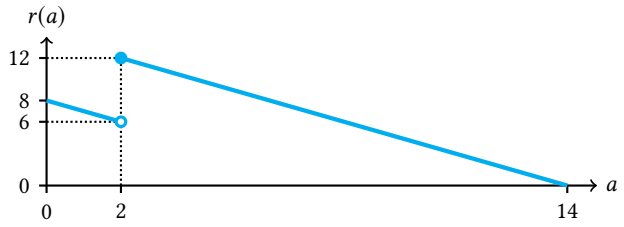
In defining and analyzing SOAP policies, there are two major technical challenges. The first challenge is to *express all such policies within a single framework*. The SOAP framework encodes a scheduling policy as a *rank function*, which maps each job to a priority level, or *rank*. All SOAP policies are based on a single rule.

Always serve the job of *minimum rank*.

We break ties with a first-come, first-served (FCFS) tiebreaking rule.

For example, in a preemptive class-based priority system, a job's rank is its class, whereas in SRPT, a job's rank is its remaining

SIGMETRICS '18 Abstracts, June 18–22, 2018, Irvine, CA, USA



The rank function for SERPT when all jobs have size either 2 or 14, each with probability 1/2. The rank is the expected remaining size of a job given it has reached its age a. A job's initial expected size is 8, but if it does not finish at age 2, it must be size 14, so its expected remaining size jumps to 12.

Figure 1.1: Rank Function for SERPT (Example 2.5)

service time. Rank functions can express a huge variety of policies, from virtually all classic policies to complex policies which have never been analyzed before. A notable exception is processor sharing (PS), which does not fit into the SOAP framework.

The second major challenge is to *analyze policies with arbitrary rank functions*. Nearly all previously analyzed scheduling policies, when expressed as SOAP policies, have rank functions that are *monotonic* in age. For example, under SRPT, a job's rank decreases with age, making it less and less likely to be preempted by another job, while under FB, a job's rank increases with age, making it more and more likely to be preempted by another job. Unfortunately, the techniques used in the past to analyze policies with monotonic rank functions *break down for arbitrary nonmonotonic rank functions*. For instance, SERPT can have a nonmonotonic rank function even for very simple job size distributions, as shown in Figure 1.1. We develop *new analytical tools* that work for arbitrary rank functions.

2 EXAMPLES OF SOAP POLICIES

2.1 Previously Analyzed SOAP Policies

Example 2.1. The *foreground-background* (FB) policy is a SOAP policy. FB always serves the job of least age, so it has rank function $r(a) = a$. It is likely that many jobs are tied for minimum rank under FB, but whichever job is served immediately loses minimum status, resulting in a processor-sharing effect.

There are always many rank functions that encode the same SOAP policy. For instance, any rank function monotonically increasing in age, such as $r(a) = a^2$, also describes FB.

Example 2.2. The *first-come, first-served* (FCFS) policy is a SOAP policy. FCFS is nonpreemptive, which is equivalent to always serving the job of maximal age, so it has rank function $r(a) = -a$. Crucially, FCFS tiebreaking breaks ties between jobs of age 0.

The next example illustrates a job's rank depending on *attributes other than age*, such as class or original size. These attributes can be anything as long as they are distributed i.i.d. for each arriving job.

Example 2.3. The *shortest remaining processing time* (SRPT) policy is a SOAP policy. A job of original size x has rank $r(x, a) = x - a$.

So far, a job's rank has been a single number. We can also assign jobs multipart ranks $r = \langle r_1, r_2 \rangle$, where r_1 is the *primary rank* and r_2 is the *secondary rank*. We sort ranks lexicographically.

Example 2.4. Consider a system with classes $\{1, \dots, n\}$ where jobs within each class are served in FCFS order but class 1 has highest priority, class 2 is next-highest, and so on. The *nonpreemptive priority* and *preemptive priority* policies are SOAP policies.

- Nonpreemptive: a job of class k has rank $r(k, a) = \langle -a, k \rangle$. The primary rank prevents preemption, and the secondary rank prioritizes the classes when starting a new job.
- Preemptive: a job of class k has rank $r(k, a) = \langle k, -a \rangle$. Because k is the primary rank, jobs from high-priority classes preempt those in low priority classes.

2.2 Newly Analyzed SOAP Policies

Example 2.5. The *shortest expected remaining processing time* (SERPT) policy is a SOAP policy. Its rank function is

$$r(a) = \mathbf{E}[X - a \mid X > a],$$

where X is the job size distribution. For example, consider a system where all jobs have size either 2 or 14, each with probability 1/2. The resulting rank function for SERPT, shown in Figure 1.1, is *nonmonotonic*. In contrast, *every rank function in Section 2.1* is monotonic. This nonmonotonicity has prevented previous techniques from analyzing SERPT in full generality.

Example 2.6. The *Gittins index* of a job with age a is [1, 3]

$$G(a) = \sup_{\Delta > 0} \frac{\mathbf{P}\{X_d - a \le \Delta \mid X_d > a\}}{\mathbf{E}[\min\{X_d - a, \Delta\} \mid X_d > a]},$$

where X is the job size distribution. The *Gittins index policy* is the scheduling policy that always serves the job of maximal Gittins index, meaning it has rank function $r(a) = 1/G(a)$. Although the Gittins index policy has long been known to minimize mean response time in the M/G/1 queue [3], only a few special cases have been analyzed in the past [4, 5]. Like SERPT, the Gittins index policy often has a nonmonotonic rank function, making it impossible to analyze in general using previous techniques.

Example 2.7. Consider a system in which jobs, rather than being completely nonpreemptible or preemptible, are *preemptible at specific checkpoints*, say every 1 time unit. The *discretized FB* policy is a variant of FB for jobs with checkpoints: when possible, it serves the job of minimal age, but it does not preempt jobs between checkpoints. Discretized FB has rank function

$$r(a) = \langle \lfloor a \rfloor - a, a \rangle.$$

Roughly speaking, the primary rank encodes the "discretized" aspect, preempting a job only at integer ages a when $\lfloor a \rfloor - a = 0$, and the secondary rank encodes the "FB" aspect.

We can "discretize" any other SOAP policy by using primary rank $\lfloor a \rfloor - a$. For instance, *discretized SRPT* has rank function $r(x, a) = \langle \lfloor a \rfloor - a, x - a \rangle$.

We have seen a variety of features that SOAP policies can model:

- jobs that are nonpreemptible, preemptible, or preemptible at checkpoints;
- jobs with known or unknown exact size;
- class-based priority in multiclass systems; and
- priority that changes nonmonotonically as a job ages.

As the following example shows, SOAP policies go even further: they can *combine many such features* as part of a single policy.

Example 2.8. Consider a system with two customer classes.

- Humans, unpredictable and easily offended, have unknown size, are nonpreemptible, and are served by FCFS relative to other humans.
- Robots, precise and ruthlessly efficient, have known size, are preemptible, and are served by SRPT relative to other robots.

A policy might have humans outrank most robots but let short robots, those with remaining size less than a threshold x_H, outrank humans that have not yet started service. This gives rank function

$$r(H, a) = \langle -a, x_H \rangle \quad r(R_x, a) = \langle 0, x - a \rangle,$$

where H and R_x denote humans and robots of size x, respectively.

3 MAIN RESULT

In this section we state the formula for the mean response time under *any* SOAP policy. Its proof and extension to Laplace-Stieltjes transforms are in the full version of this work [6].

The mean response time formula conditions on a job's *descriptor*, which captures all attributes the scheduler knows about the job upon arrival, and its *size*, which may or may not be known to the scheduler. We write λ for the overall job arrival rate and write X_d for the size distribution of jobs with descriptor d.

Definition 3.1. The *worst future rank* of a job with descriptor d, size x, and age a is[1]

$$r^{\text{worst}}_{d,x}(a) = \sup_{a \le b < x} r(d, b).$$

Definition 3.2. Let r be a rank. The *new r-work* is a random variable, written $X^{\text{new}}[r]$, representing how long a job that just arrived to the system is served until it completes or reaches rank at least r. Specifically, we define $X^{\text{new}}[r] = X^{\text{new}}_D[r]$, where D is the random descriptor assigned to a new job and, for any specific descriptor d,

$$c_d[r] = \inf\{a \ge 0 \mid r(d, a) \ge r\}$$
$$X^{\text{new}}_d[r] = \min\{X_d, c_d[r]\}.$$

That is, $c_d[r]$ is when a job of descriptor d reaches rank at least r.

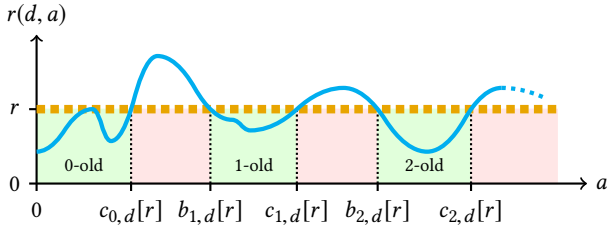
Definition 3.3. Let r be a rank and d be a descriptor. The *i-old r-interval* for descriptor d is the ith interval of ages during which a job of descriptor d has rank at most r. See Figure 3.1 for an illustration. Specifically, the interval is $[b_{i,d}[r], c_{i,d}[r]]$, where

$$b_{0,d}[r] = 0$$
$$c_{0,d}[r] = \inf\{a \ge 0 \mid r(d, a) > r\}$$
$$b_{i,d}[r] = \inf\{a > c_{i-1,d}[r] \mid r(d, a) \le r\}$$
$$c_{i,d}[r] = \inf\{a > b_{i,d}[r] \mid r(d, a) > r\}.$$

The *i-old r-work* is a random variable, written $X^{\text{old}}_i[r]$, representing how long a job will be served while its age is in its i-old r-interval. Specifically, we define $X^{\text{old}}_i[r] = X^{\text{old}}_{i,D}[r]$, where D is the random descriptor assigned to a new job and, for any specific descriptor d,

$$X^{\text{old}}_{i,d}[r] = \begin{cases} 0 & \text{if } X_d < b_{i,d}[r] \\ X_d - b_{i,d}[r] & \text{if } b_{i,d}[r] \le X_d < c_{i,d}[r] \\ c_{i,d}[r] - b_{i,d}[r] & \text{if } c_{i,d}[r] \le X_d. \end{cases}$$

[1] See the full version of this work [6] for discussion of corner cases.



We show the rank of a job with descriptor d in cyan. The i-old r-intervals, highlighted in green, are the intervals of ages during which the job's rank is at most r. The i-old r-work $X^{\text{old}}_{i,d}[r]$ is the amount of service the job requires while its age is in its i-old r-interval.

Figure 3.1: Illustration of Old Work (Definition 3.3)

If $b_{i,d}[r] = c_{i,d}[r] = \infty$, we define $X^{\text{old}}_{i,d}[r] = 0$.

Theorem 3.4 (SOAP Mean Response Time). *In an M/G/1 using any SOAP policy, the mean response time of jobs with descriptor d and size x is*

$$\mathbf{E}[T_{d,x}] = \frac{\lambda \sum_{i=0}^{\infty} \mathbf{E}[(X^{\text{old}}_i[w])^2]}{2(1 - \rho^{\text{old}}_0[w])(1 - \rho^{\text{new}}[w])} + \int_0^x \frac{1}{1 - \rho^{\text{new}}[w(a)]}\, da,$$

where λ is the overall arrival rate and

$$w(a) = r^{\text{worst}}_{d,x}(a) \quad \rho^{\text{new}}[r] = \lambda \mathbf{E}[X^{\text{new}}[r]]$$
$$w = r^{\text{worst}}_{d,x}(0) \quad \rho^{\text{old}}_i[r] = \lambda \mathbf{E}[X^{\text{old}}_i[r]].$$

Proof. See the full version of this work [6, Theorem 5.5].

ACKNOWLEDGMENTS

We thank Peter van de Ven and the anonymous referees for their helpful comments. Ziv Scully was supported by an ARCS Foundation scholarship and the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE-1745016. Mor Harchol-Balter was supported by NSF-CMMI-1538204. Mor Harchol-Balter and Ziv Scully were supported by NSF-XPS-1629444.

REFERENCES

[1] Samuli Aalto, Urtzi Ayesta, and Rhonda Righter. 2009. On the Gittins index in the M/G/1 queue. *Queueing Systems* 63, 1 (2009), 437–458.
[2] Samuli Aalto, Urtzi Ayesta, and Rhonda Righter. 2011. Properties of the Gittins index with application to optimal scheduling. *Probability in the Engineering and Informational Sciences* 25, 03 (2011), 269–288.
[3] John Gittins, Kevin Glazebrook, and Richard Weber. 2011. *Multi-armed Bandit Allocation Indices.* John Wiley & Sons.
[4] Esa Hyytiä, Samuli Aalto, and Aleksi Penttinen. 2012. Minimizing slowdown in heterogeneous size-aware dispatching systems. In *ACM SIGMETRICS Performance Evaluation Review*, Vol. 40. ACM, 29–40.
[5] Natalia Osipova, Urtzi Ayesta, and Konstantin Avrachenkov. 2009. Optimal policy for multi-class scheduling in a single server queue. In *Teletraffic Congress, 2009. ITC 21 2009. 21st International.* IEEE, 1–8.
[6] 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 (April 2018), 30 pages. https://doi.org/10.1145/3179419