# Is Sharing with Retransmissions Causing Instabilities?

Predrag R. Jelenković Evangelia D. Skiani
Department of Electrical Engineering
Columbia University, New York, NY 10027
{predrag, valia}@ee.columbia.edu

## ABSTRACT

Retransmissions represent a primary failure recovery mechanism on all layers of communication network architecture. Similarly, fair sharing, e.g. processor sharing (PS), is a widely accepted approach to resource allocation among multiple users. Recent work has shown that retransmissions in failure-prone, e.g. wireless ad hoc, networks can cause heavy tails and long delays. In this paper, we discover a new phenomenon showing that PS-based scheduling induces complete instability in the presence of retransmissions, regardless of how low the traffic load may be. This phenomenon occurs even when the job sizes are bounded/fragmented, e.g. deterministic. Our analytical results are further validated via simulation experiments. Moreover, our work demonstrates that scheduling one job at a time, such as first-come-first-serve, achieves stability and should be preferred in these systems.

## Categories and Subject Descriptors

G.3 [**Probability And Statistics**]: Probabilistic algorithms; C.4 [**Performance of Systems**]: Performance Attributes; H.4 [**Information Systems Applications**]: Miscellaneous

## General Terms

Theory, Performance

## Keywords

Retransmissions; fair sharing; instabilities; failure-prone systems; Processor Sharing; Discriminatory Processor Sharing; FCFS; scheduling; M/G/1 queue

## 1. INTRODUCTION

High variability and frequent failures characterize the majority of large-scale systems, e.g. infrastructure-less wireless networks, cloud/parallel computing systems, etc. The nature of these systems imposes the employment of failure recovery mechanisms to guarantee their good performance. One of the most straightforward and widely used recovery mechanism is to simply restart all of the interrupted jobs from the beginning after a failure occurs. In communication systems, restart mechanisms lie at the core of the network architecture where retransmissions are used on all protocol layers to guarantee data delivery in the presence of channel failures e.g. Automatic Repeat reQuest (ARQ) protocol [3], contention based ALOHA type protocols in the medium access control (MAC) layer, end-to-end acknowledgements in the transport layer, HTTP downloading scheme in the application layer, and others.

Furthermore, sharing is a primary approach to fair scheduling and efficient management of the available resources. Fair allocation of the network resources among different users can be highly beneficial for increasing throughput and utilization. For instance, CDMA is a multiple access method used in communication networks, where several users can transmit information simultaneously over a single channel via sharing the available bandwidth. Another example is Processor Sharing (PS) scheduling [19] where the capacity is equally shared between multiple classes of customers. In *Generalized* PS (GPS) [14, 15], service allocation is done according to some fixed weights. The related *Discriminatory* PS (DPS) [1, 5, 13] is used in computing to model the Weighted Round Robin (WRR) scheduling, while it is also used in communications, as a flow level model of heterogenous TCP connections. Similarly, fair queuing (FQ) is a scheduling algorithm where the link capacity is fairly shared among active network flows; in weighted fair queuing (WFQ), which is the discretized version of GPS, different scheduling priorities are assigned to each flow.

In general, PS-based scheduling disciplines have been widely used in modeling computer and communication networks. Early investigations of PS queues were motivated by applications in multiuser computer systems [4]. The M/G/1 PS queue has been studied extensively in the literature [18]. In the case of the M/M/1 PS system, the conditional Laplace transform of the waiting time was derived in [4]. The importance of scheduling in the presence of heavy tails was first recognized in [2], and later, in [7], the M/G/1 PS queue was studied assuming subexponential job sizes; see also [7] for additional references.

In [17], it was proven that, although there are policies known to optimize the sojourn time tail under a large class of heavy-tailed job sizes (e.g. PS and SRPT) and there

This work is supported by the National Science Foundation grant number 0915784.


*SIGMETRICS'14,* June 16–20, 2014, Austin, Texas, USA.

http://dx.doi.org/10.1145/2591971.2592001.

are policies known to optimize the sojourn time tail in the case of light-tailed job sizes, e.g. FCFS, no policies are known to optimize the sojourn time tail across both light and heavy-tailed job size distributions. Indeed, such policies must "learn" the job size distribution in order to optimize the sojourn time tail. In the heavy-tailed scenarios, any scheduling policy that assigns the server exclusively to a very large job, e.g. FCFS, may induce long delays, in which case, sharing guarantees better performance.

In this paper, we study the effects of sharing on the system performance when restarts are employed in the presence of failures. We revisit the well-studied M/G/1 queue with failures and restarts and focus on the PS scheduling policy. We use the following generic model, which was first introduced in [6] in the application context of computing. The system dynamics is described as a process $(A, \{A_n\}_{n\geqslant 1})$, where $A_n$ correspond to the periods when the system is available. $(A, \{A_n\}_{n\geqslant 1})$ is a sequence of i.i.d random variables, independent of the job sizes. In each period of time that the system is available, say $A_n$, we attempt to execute a job of random size $B$. If $A_n > B$, we say that the job is successfully completed; otherwise, we restart the job from the beginning in the following period $A_{n+1}$ when the channel is available.

With regard to retransmissions, it was first recognized in [6, 16] that restart mechanisms may result in heavy-tailed (power law) delays even if the job sizes and failure rates are light-tailed. In [11], it was shown that the power law delays arise whenever the hazard functions of the data and failure distributions are proportional. In the practically important case of bounded data units, a uniform characterization of the entire body of the retransmission distribution was derived in [10, 8], which allows for determining the optimal size of data units/fragments in order to alleviate the power law effect. Later, these results were extended to the case where the channel is highly correlated [9], i.e. switches between states with different characteristics, and was proved that the delays are insensitive to the channel correlations and are determined by the 'best' channel state.

In this paper, our main contributions are the following. First, we prove that the M/G/1 PS queue is always unstable, regardless of how light the load is and how small the job sizes may be, see Theorems 3.1 and 3.2 in Section 3. This is a new phenomenon, since, contrary to the conventional belief, sharing the service even between deterministic jobs can render the system completely unstable when retransmissions/restarts are employed. The intuition is the following. If a large number of jobs arrives in a short period of time, then under the elongated service time distribution induced by retransmissions, the queue will keep accumulating jobs that will equally share the capacity, which further exacerbates the problem. Every time a failure occurs, the system resets and the service requirement for each job elongates as the queue size increases. The expected delay until the system clears becomes increasingly long and, consequently, the queue will continue to grow leading to instability. This result also applies to the *Discriminatory* PS (DPS) queue, where the service is not shared equally but according to some fixed weights.

We would also like to emphasize that job fragmentation cannot stabilize the system regardless of how small the fragments are made, since Theorem 3.1 shows instability for any minimum job size $\beta > 0$. In our experimental results, we make an interesting observation on the system behavior before it saturates. There exists a transient period, during which the queue appears as if it were stable. Although it may occasionally accumulate a substantial number of jobs, it returns to zero and starts afresh. However, there exists a time when the queue reaches a critical size after which the service rate of the jobs reduces so much that neither of them can depart. Hence, as the queue continues to increase in size, the system becomes unstable.

Next, in order to gain further insight into the system, we focus on its transient behavior and study the properties of the completion time of a finite number of jobs with no future arrivals. Specifically, we compare two work-conserving policies: scheduling one job at a time, e.g. FCFS, and PS. Overall, we discover that serving one job at a time exhibits uniformly better performance than PS; compare Theorems 4.2 and 4.3, respectively. Furthermore, under more technical assumptions, and for light-tailed job/failure distributions, we show that PS performs distinctly worse compared to the heavy-tailed ones.

From an engineering perspective, our results indicate that traditional approaches in existing systems may be inadequate in the presence of failures. This new phenomenon demonstrates the need of revisiting existing techniques to large-scale failure-prone systems, where PS-based scheduling may perform poorly. For example, since PS is unstable even for deterministic jobs, packet fragmentation, which is widely used in communications, cannot alleviate instabilities. Indeed, fragmentation can only postpone the time when the instability occurs, but cannot eliminate the phenomenon; see Example 1 in Section 5. Therefore, serving one job at a time, e.g. FCFS, is highly advisable in such systems.

The paper is organized as follows. In Section 2, we introduce the model along with the necessary definitions and notation. Next, in Section 3, we present our main results on the M/G/1 queue. Later, in Section 4, we analyze a static system under two different scheduling policies, e.g. serving the jobs one at a time and PS. Last, Section 5 presents our simulation experiments that validate our main theoretical findings, while Section 6 concludes the paper.

## 2. DEFINITIONS AND NOTATION

First, we provide the necessary definitions and notation assuming that the jobs are served individually. Consider a generic job of random size $B$ requesting service in a failure-prone system. Without loss of generality, we assume that the system is of unit capacity. Its dynamics is described as a process $(A, \{A_n\}_{n\geqslant 1})$ of availability periods, where at the end of each period $A_n$, the system experiences a failure, as shown in Figure 1.

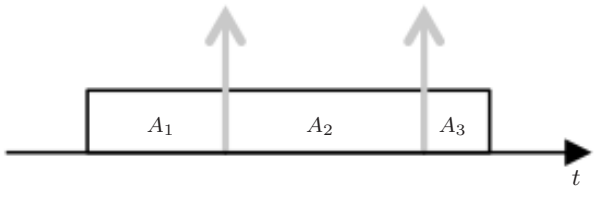


**Figure 1: System with failures.**

At each period of time that the system becomes available, say $A_n$, we attempt to process a generic job of size $B$. If $A_n > B$, we say that the job is completed successfully; otherwise, we wait until the next period $A_{n+1}$ when the channel is available and restart the job. A sketch of the model depicting the system is drawn in Figure 2.

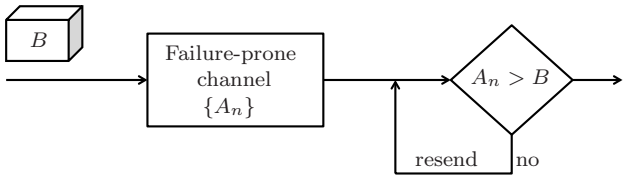


**Figure 2: Jobs executed in system with failures.**

**Definition 2.1** *The number of restarts for a generic job of size $B$ is defined as*

$$N \triangleq \inf\{n : A_n > B\}.$$

We are interested in computing the total service time $S$ until $B$ is successfully completed, which is formally defined as follows.

**Definition 2.2** *The service time is the total time until a generic job of size $B$ is successfully completed and is denoted as*

$$S \triangleq \sum_{i=1}^{N-1} A_i + B.$$

We denote the complementary cumulative distribution functions for $A$ and $B$, respectively, as

$$\bar{G}(x) \triangleq \mathbb{P}(A > x) \qquad \text{and} \qquad \bar{F}(x) \triangleq \mathbb{P}(B > x).$$

Throughout the paper, we assume that the functions $\bar{G}(x)$ and $\bar{F}(x)$ are absolutely continuous for all $x \geqslant 0$. We also use the following standard notation. For any two real functions $f(x)$ and $g(x)$ and fixed $x_0 \in \mathbb{R} \cup \{\infty\}$, we say $f(x) \sim g(x)$ as $x \to x_0$, to denote $\lim_{x \to x_0} f(x)/g(x) = 1$.

# 3. M/G/1 QUEUE WITH RESTARTS

In this section, we discuss the stability of the M/G/1 queue under two scheduling disciplines: First Come First Serve (FCFS) and Processor Sharing (PS). In the following subsection, we derive the necessary and sufficient condition for the system to be stable when the jobs are processed one at a time and in the order they arrive in the system. Next, in subsection 3.2, we show in Theorems 3.1 and 3.2 that the M/G/1 PS queue is unstable.

## 3.1 First Come First Serve Queue

In the FCFS discipline, each job is processed one at a time and therefore the expected service time for a single job is given from Definition 2.2 as

$$\mathbb{E}[S] = \mathbb{E}\left[\sum_{i=1}^{N-1} A_i + B\right].$$

Note that $N \triangleq \inf\{n : A_n > B\}$ is a well defined stopping time for the process $(A, \{A_n\}_{n \geqslant 1})$, and thus the expected service time follows from Wald's identity as

$$\begin{aligned}\mathbb{E}[S] &= \mathbb{E}\left[\sum_{i=1}^{N} A_i - A_N + B\right] \\ &= \mathbb{E}[N]\mathbb{E}[A] - \mathbb{E}[A_N] + \mathbb{E}[B].\end{aligned}$$

Now, assuming that the availability periods $A$ are exponentially distributed with rate $\mu$ (Poisson failures), the expected service time is given by

$$\begin{aligned}\mathbb{E}[S] &= \mathbb{E}[N]\mathbb{E}[A] - (\mathbb{E}[A] + \mathbb{E}[B]) + \mathbb{E}[B] \\ &= (\mathbb{E}[N] - 1)\mathbb{E}[A], \qquad (3.1)\end{aligned}$$

since $\mathbb{E}[A_N] = \mathbb{E}\left[\mathbb{E}[A|A > B]\right] = \mathbb{E}[A+B] = \mathbb{E}[A]+\mathbb{E}[B]$, due to the memoryless property of the exponential distribution.

The necessary and sufficient condition for the stability of the M/G/1 FCFS queue is

$$\lambda \mathbb{E}[S] < 1.$$

Now, let the jobs be fixed and all equal to some positive constant $\beta > 0$. Since $A$ is exponentially distributed with rate $\mu$, then

$$\mathbb{P}[N > n] = \mathbb{P}(A \leqslant \beta)^n = G(\beta)^n,$$

and thus, the expected number of restarts is

$$\mathbb{E}[N] = \sum_{n=0}^{\infty} \mathbb{P}[N > n] = \sum_{n=0}^{\infty} G(\beta)^n = \bar{G}(\beta)^{-1} = e^{\mu\beta}. \quad (3.2)$$

Furthermore, for fixed jobs $B = \beta$, we can compute explicitly $\mathbb{E}[S]$ without the exponential assumption on $A$. To this end, note that

$$\begin{aligned}\mathbb{E}[S] &= \mathbb{E}\left[\sum_{i=1}^{N-1} A_i + \beta\right] = \mathbb{E}\left[\sum_{n=2}^{\infty} \mathbf{1}_{\{N=n\}} \sum_{i=1}^{n-1} A_i + \beta\right] \\ &= \mathbb{E}\left[\sum_{n=2}^{\infty} \mathbf{1}_{\{A_1<\beta, A_2<\beta, \ldots, A_{n-1}<\beta, A_n \geqslant \beta\}} \sum_{i=1}^{n-1} A_i\right] + \beta \\ &= \sum_{n=2}^{\infty} \mathbb{E}\left(\sum_{i=1}^{n-1} A_i \mathbf{1}_{\{A_1<\beta, A_2<\beta, \ldots, A_{n-1}<\beta, A_n \geqslant \beta\}}\right) + \beta,\end{aligned}$$

and since $(A, \{A_i\}_{i \geqslant 1})$ are i.i.d, we obtain

$$\begin{aligned}\mathbb{E}[S] &= \sum_{n=2}^{\infty} (n-1)\mathbb{E}\left[A\mathbf{1}_{\{A<\beta\}}\right] \mathbb{P}(A < \beta)^{n-2}\mathbb{P}(A \geqslant \beta) + \beta \\ &= \sum_{n=2}^{\infty} (n-1)\mathbb{E}\left[A\mathbf{1}_{\{A<\beta\}}\right] \mathbb{P}(N = n-1) + \beta,\end{aligned}$$

where we recall that $\mathbb{P}(N = n) = \mathbb{P}(A < \beta)^{n-1}\mathbb{P}(A \geqslant \beta)$, by definition, and thus,

$$\begin{aligned}\mathbb{E}[S] &= \mathbb{E}\left[A\mathbf{1}_{\{A<\beta\}}\right] \sum_{n=1}^{\infty} n\mathbb{P}(N = n) + \beta \\ &= \mathbb{E}\left[A\mathbf{1}_{\{A<\beta\}}\right] \mathbb{E}[N] + \beta. \qquad (3.3)\end{aligned}$$

Hence, for exponential $A$, the preceding expression (or (3.1)) yields

$$\mathbb{E}[S] = (e^{\mu\beta} - 1)\mu^{-1},$$

and the stability region reduces to

$$\lambda \mathbb{E}[S] = \lambda \mu^{-1}(e^{\mu\beta} - 1) < 1. \tag{3.4}$$

Note that $\mathbb{E}[S] \geqslant \mu^{-1}\mu\beta = \beta$, where $\lambda\beta < 1$ gives the stability region of the ordinary M/G/1 queue without failures. We observe that as the jobs grow in size, the stability region shrinks. In other words, the larger the $\beta$, the slower the arrival rate the queue can accommodate. For FCFS scheduling, if it is not possible to adjust the arrival rate, we could potentially decrease the job sizes, e.g. apply fragmentation techniques, in order to maintain a large stability region without generating too much overhead, resulting from dividing a single job into many smaller ones.

## 3.2 Instability of Processor Sharing Queue

Here, we show in Theorems 3.1 and 3.2 that the M/G/1 PS queue is unstable when jobs need to restart after failures. We prove that the queue size grows to infinity for general job size distributions, when the jobs have a minimum size. First, in Theorem 3.1, we assume that the job sizes are bounded from below by some positive constant $\beta$. This is a natural assumption for communication or computing applications where jobs, e.g. files, packets, threads, must have a header to contain the required information, such as destination address, thread id, etc. Hence, the job sizes, in practice, cannot be smaller than a positive constant. Finally, we drop this requirement and prove the instability result in general; see Theorem 3.2.

In this section, we will assume that $A$ has finite first moment, e.g. $\mathbb{E}A < \infty$. Here, we can assume that the failure process is in stationarity, or more generally, that the first failure occurs at time $0 \leqslant A_0 < \infty$ a.s. and that the time between subsequent failures $i$ and $i+1$, is $A_i$, where $\{A_i\}_{i\geqslant 1}$ are i.i.d., independent of $A_0$. If $A_0$ is the excess distribution of $A_1$, then the failure process is stationary. The following Theorem 3.1 shows that the queue is unstable, i.e. its size grows to infinity. Let $Q_t$ be the queue size at time $t$.

**Theorem 3.1** *If $\mathbb{E}A < \infty$ and $\mathbb{P}[B \geqslant \beta] = 1, \beta > 0$, then the M/G/1 PS queue is unstable, i.e., for any $Q_0 < \infty$, almost surely*

$$Q_t \to \infty \text{ as } t \to \infty.$$

**Remark 1** Although in many applications job arrival times are well modeled by Poisson processes, our result can be generalized to more general arrivals at the expense of additional technical complications. Here, we avoid this analysis for simplicity reasons, but defer it to the extended version of the paper.

**Remark 2** By carefully examining the proof, it can be seen that stronger statements about instability can be derived, i.e.

$$\lim_{t\to\infty} \mathbb{P}(\text{no job ever completes service after time } t) = 1.$$

This, along with other generalizations, will be developed in the journal version of the paper.

**Proof:** First, we begin with a preliminary result. We assume that there are at least $k$ jobs in the queue and there is a failure at time $t = 0$, and provide a lower bound on the probability that any job departs the system. Specifically, we show that, given $Q_0 \geqslant k$,

$$\mathbb{P}(\text{no job ever completes service}) \geqslant 1 - H(\mathbb{E}A\mathbf{1}(A \geqslant \beta k) + e^{-\theta k}), \tag{3.5}$$

for some $H, \theta > 0$ and all $k$ large.

Let $T_1 = \sum_{i=1}^{ck} A_i$ be the cumulative time that includes the first $ck$ failures; to simplify notation we write $\sum_x^y$ to denote $\sum_{\lceil x\rceil}^{\lfloor y\rfloor}$, where $\lceil x\rceil$ is the smallest integer $\geqslant x$ and $\lfloor y\rfloor$ is the largest integer $\leqslant y$. Now, define the event $\mathcal{A}_1 \equiv \mathcal{A}_1(k) \triangleq \{A_1 < \beta k, A_2 < \beta k, \ldots, A_{ck} < \beta k\}$. On this event, no job can leave the system since $Q_0 \geqslant k$ and all of them are at least of size $\beta$. Thus, if they were served in isolation, they could not have completed service in the first $ck$ attempts.

Now, let $E_1$ denote the event that there is no departure in the first $ck$ attempts and there are at least $k$ arrivals in $(0, T_1]$; we use $Z_{(t_0,t_1]}$ to denote the number of Poisson arrivals in the interval $(t_0, t_1]$, whereas we simply write $Z_t$ for intervals $(0, t]$. Formally,

$$E_1 \supset \underline{E}_1 \triangleq \{Z_{T_1} \geqslant k, \mathcal{A}_1\},$$

on the set $\{Q_0 \geqslant k\}$. Now, observe that

$$\begin{aligned}\mathbb{P}(\underline{E}_1) &\geqslant \mathbb{P}(Z_{T_1} \geqslant k, T_1 \geqslant 2k/\lambda, \mathcal{A}_1)\\ &\geqslant \mathbb{P}(Z_{2k/\lambda} \geqslant k, T_1 \geqslant 2k/\lambda, \mathcal{A}_1)\\ &\geqslant \mathbb{P}(Z_{2k/\lambda} \geqslant k)\mathbb{P}(T_1 \geqslant 2k/\lambda, \mathcal{A}_1),\end{aligned}$$

since Poisson arrivals are independent of the failure process. Thus,

$$\mathbb{P}(\underline{E}_1) \geqslant \mathbb{P}(Z_{2k/\lambda} \geqslant k)\left(\mathbb{P}(\mathcal{A}_1) - \mathbb{P}(T_1 < 2k/\lambda)\right).$$

First, note that

$$\begin{aligned}\mathbb{P}(Z_{2k/\lambda} \geqslant k) &= 1 - \mathbb{P}(Z_{2k/\lambda} < k) = 1 - \mathbb{P}(2k - Z_{2k/\lambda} > k)\\ &\geqslant 1 - e^{-\theta k}\mathbb{E}e^{\theta(2k - Z_{2k/\lambda})} = 1 - e^{\theta k}\mathbb{E}e^{-\theta Z_{2k/\lambda}},\end{aligned}$$

by Cramer's bound for $\theta > 0$. Next, observe that $Z_{2k/\lambda}$ is Poisson with rate $2k$ and thus

$$\mathbb{P}(Z_{2k/\lambda} \geqslant k) \geqslant 1 - e^{\theta k}e^{2(e^{-\theta}-1)k} = 1 - e^{-\theta_1 k},$$

where $\theta_1 = 2(1 - e^{-\theta}) - \theta > 0$, for $\theta$ small.

Second, observe that

$$\begin{aligned}\mathbb{P}(T_1 < 2k/\lambda) &= \mathbb{P}\left(\sum_{i=1}^{ck} A_i < 2k/\lambda\right)\\ &= \mathbb{P}\left(\sum_{i=1}^{ck}(A_i - \mathbb{E}A) < 2k/\lambda - ck\mathbb{E}A\right)\\ &\leqslant \mathbb{P}\left(\sum_{i=1}^{3k/\lambda\mathbb{E}A}(\mathbb{E}A - A_i) > k/\lambda\right),\end{aligned}$$

by picking $c \triangleq 3/(\lambda\mathbb{E}A)$. Now, let $X_i \triangleq \mathbb{E}A - A_i$, which are bounded from above since $X_i \leqslant \mathbb{E}A < \infty$, from our main assumption. Therefore, Cramer's large deviation bound implies that

$$\mathbb{P}(T_1 < 2k/\lambda) \leqslant \mathbb{P}\left(\sum_{i=1}^{3k/\lambda\mathbb{E}A} X_i > k/\lambda\right) \leqslant H_2 e^{-\theta_2 k},$$

for some $H_2, \theta_2 > 0$.

Therefore,

$$\begin{aligned}\mathbb{P}(\underline{E}_1) &\geqslant (1-e^{-\theta_1 k})\left(\mathbb{P}(\mathcal{A}_1) - H_2 e^{-\theta_2 k}\right)\\ &\geqslant \mathbb{P}(A<\beta k)^{ck} - (e^{-\theta_1 k} + H_2 e^{-\theta_2 k} - H_2 e^{-(\theta_1+\theta_2)k})\\ &\geqslant (1-\mathbb{P}(A\geqslant \beta k))^{ck} - He^{-\theta k},\end{aligned}$$

where $\theta = \min(\theta_1, \theta_2)$ and $H > 0$ such that $H < (1 + H_2)$. Next, using $1 - x \geqslant e^{-2x}$ for small $x$, we have for all $k \geqslant k_0$

$$\begin{aligned}\mathbb{P}(\underline{E}_1) &\geqslant e^{-2ck\mathbb{P}(A\geqslant\beta k)} - He^{-\theta k}\\ &\geqslant 1 - 2ck\mathbb{P}(A\geqslant\beta k) - He^{-\theta k}\\ &\geqslant e^{-4ck\mathbb{P}(A\geqslant\beta k) - 2He^{-\theta k}}.\end{aligned}$$

Next, at time $\mathcal{T}_1 = T_1$, on event $\underline{E}_1$, the queue has at least $2k$ jobs, i.e. $Q_{\mathcal{T}_1} \geqslant 2k$, and no jobs have departed. Similarly as before, let $T_2 = \sum_{i=ck+1}^{3ck} A_i$ be the cumulative time that includes the next $2ck$ failures, and define $\mathcal{A}_2 \equiv \mathcal{A}_2(k) \triangleq \{A_{ck+1} < 2\beta k, A_{ck+2} < 2\beta k, \ldots, A_{3ck} < 2\beta k\}$. The probability that no job departs in $(0, \mathcal{T}_2]$, where $\mathcal{T}_2 = T_1 + T_2$, is lower bounded by

$$\begin{aligned}&\mathbb{P}(\text{no job departs in}(0,\mathcal{T}_2])\\ &\geqslant \mathbb{P}(Z_{T_1}\geqslant k, A_1, Q_{\mathcal{T}_1}\geqslant 2k, Z_{(\mathcal{T}_1,\mathcal{T}_2]}\geqslant 2k, A_2)\\ &\geqslant \mathbb{P}(Z_{T_1}\geqslant k, A_1, Z_{T_2}\geqslant 2k, A_2),\end{aligned} \tag{3.6}$$

since $\{Q_{\mathcal{T}_1} \geqslant 2k\} \supseteq \{Z_{T_1} \geqslant k, A_1\}$ on the set $\{Q_0 \geqslant k\}$.

Now, if $E_2$ is the event that there is no departure in the next $2ck$ attempts and there are at least $2k$ arrivals in $(\mathcal{T}_1, \mathcal{T}_2]$, then $E_2 \supset \underline{E}_2 \triangleq \{Z_{T_2} \geqslant 2k, \mathcal{A}_2\}$; note that $\underline{E}_2$ is independent of $\underline{E}_1$. Via identical arguments as before, we obtain

$$\begin{aligned}\mathbb{P}(\underline{E}_2) &\geqslant \mathbb{P}(Z_{T_2}\geqslant 2k, T_2\geqslant 4k/\lambda, \mathcal{A}_2)\\ &\geqslant \mathbb{P}(Z_{4k/\lambda}\geqslant 2k)\left(\mathbb{P}(\mathcal{A}_2) - \mathbb{P}(T_2 < 4k/\lambda)\right)\\ &\geqslant e^{-8ck\mathbb{P}(A\geqslant 2\beta k) - 2He^{-2\theta k}}.\end{aligned}$$

Therefore, at time $\mathcal{T}_2$, on event $\underline{E}_1 \cap \underline{E}_2$, there are at least $4k$ jobs.

In general, for any $n$, we can extend the reasoning from (3.6) to obtain

$$\begin{aligned}&\mathbb{P}(\text{no job departs in}(0,\mathcal{T}_n])\\ &\geqslant \mathbb{P}(Z_{T_1}\geqslant k, A_1, Z_{T_2}\geqslant 2k, A_2, \ldots, Z_{T_n}\geqslant 2^{n-1}k, A_n)\\ &= \mathbb{P}(\underline{E}_1\cap\underline{E}_2\cap\cdots\cap\underline{E}_n),\end{aligned}$$

where $\underline{E}_n = \{Z_{T_n} \geqslant 2^{n-1}k, A_n\}$ and $T_n = \sum_{i=(2^{n-1}-1)ck+1}^{(2^n-1)ck} A_i$. Similarly,

$$\mathbb{P}(\underline{E}_n) \geqslant e^{-2^{n+1}ck\mathbb{P}(A\geqslant 2^{n-1}\beta k) - 2He^{-\theta 2^{n-1}k}}.$$

Hence, we obtain

$$\begin{aligned}\mathbb{P}(E_1\cap E_2\cap\cdots\cap E_n) &\geqslant \mathbb{P}(\underline{E}_1\cap\underline{E}_2\cap\cdots\cap\underline{E}_n)\\ &= \mathbb{P}(\underline{E}_1)\mathbb{P}(\underline{E}_2)\cdots\mathbb{P}(\underline{E}_n),\end{aligned}$$

since the events $\underline{E}_i$'s are independent. Thus,

$$\begin{aligned}&\mathbb{P}(E_1\cap E_2\cap\cdots\cap E_n)\\ &\quad\geqslant \prod_{i=1}^{n} e^{-2^{i+1}ck\mathbb{P}(A\geqslant 2^{i-1}\beta k) - 2He^{-2^{i-1}\theta k}}\\ &\quad= e^{-4\sum_{i=0}^{n-1}2^i ck\mathbb{P}(A\geqslant 2^i\beta k) - 2H\sum_{i=0}^{n-1}e^{-2^i\theta k}}\\ &\quad\geqslant e^{-4\sum_{i=0}^{\infty}2^i ck\mathbb{P}(A\geqslant 2^i\beta k) - 2He^{-\theta k}\sum_{i=0}^{\infty}e^{-(2^i-1)\theta k}}.\end{aligned}$$

Now, observe that $\sum_{i=0}^{\infty} e^{-(2^i-1)\theta k} < \infty$, and thus we can pick $H$ such that

$$\mathbb{P}(E_1\cap E_2\cap\cdots\cap E_n) \geqslant e^{-4\sum_{i=0}^{\infty}2^i ck\mathbb{P}(A\geqslant 2^i\beta k) - He^{-\theta k}}.$$

Furthermore, we observe that

$$\begin{aligned}\sum_{i=0}^{\infty} 2^i ck\mathbb{P}(A\geqslant 2^i\beta k) &\leqslant \frac{c}{\beta}\sum_{i=0}^{\infty}\beta k\int_{2^i}^{2^{i+1}}\mathbb{P}(A\geqslant x\beta k)dx\\ &\leqslant \frac{c}{\beta}\beta k\int_1^{\infty}\mathbb{P}(A\geqslant x\beta k)dx\\ &= \frac{c}{\beta}\int_{\beta k}^{\infty}\mathbb{P}(A\geqslant y)dy = \frac{c}{\beta}\mathbb{E}A\mathbf{1}(A\geqslant\beta k).\end{aligned}$$

and thus

$$\begin{aligned}\mathbb{P}(E_1\cap E_2\cap\cdots\cap E_n) &\geqslant e^{-4c\beta^{-1}\mathbb{E}A\mathbf{1}(A\geqslant\beta k) - He^{-\theta k}}\\ &\geqslant 1 - H(\mathbb{E}A\mathbf{1}(A\geqslant\beta k) + e^{-\theta k}),\end{aligned}$$

which further implies, by monotone convergence, that

$$\begin{aligned}\mathbb{P}(\cap_{i=1}^{\infty}E_i) &= \lim_{n\to\infty}\mathbb{P}(E_1\cap E_2\cap\cdots\cap E_n)\\ &\geqslant 1 - H(\mathbb{E}A\mathbf{1}(A\geqslant\beta k) + e^{-\theta k}).\end{aligned}$$

Hence, (3.5) follows directly since, by definition,

$$\mathbb{P}(\text{no job ever completes service}) \geqslant \mathbb{P}(\cap_{i=1}^{\infty}E_i).$$

Now, we are ready to finalize our proof. For any $k \geqslant 1$, let $T_k$ be the first time that there are $k$ jobs in the queue and a failure occurs. $T_k$ is almost surely finite since it is upper bounded by the time $\bar{T}_k$ that there are at least $k$ arrivals in an open interval of size $\beta$ just before a failure. The probability of this event is $\mathbb{P}(Z_\beta \geqslant k) > 0$.

Next, for any fixed time $t$, we obtain

$$\begin{aligned}&\mathbb{P}(\text{no job leaves after time } t)\\ &\geqslant \mathbb{P}(T_k\leqslant t, \text{no job leaves after } T_k)\\ &= \mathbb{E}[\mathbb{P}(T_k\leqslant t|Q_{T_k},\mathcal{B})\mathbb{P}(\text{no job leaves after } T_k|Q_{T_k},\mathcal{B})]\\ &\geqslant \mathbb{P}(T_k\leqslant t)(1-\epsilon),\end{aligned} \tag{3.7}$$

where $\mathcal{B} \triangleq \{B_1^{T_k}, \ldots, B_{Q_{T_k}}^{T_k}\}$ denotes the job sizes that are present in the queue at time $T_k$; the last inequality follows from (3.5) by letting $k \to \infty$ and observing that $\mathbb{E}A\mathbf{1}(A \geqslant \beta k) \to 0$; the equality follows from the fact the event {no job leaves after $T_k$} is independent of the past, i.e. $T_k \leqslant t$, given $Q_{T_k}, \mathcal{B}$. Next, recall that $T_k$ is almost surely finite, i.e. $\lim_{t\to\infty}\mathbb{P}(T_k \leqslant t) = 1$, and thus taking the limit as $t \to \infty$ and letting $\epsilon \downarrow 0$ in (3.7) yields

$$\lim_{t\to\infty}\mathbb{P}(\text{no job leaves after time } t) = 1.$$

Last, note that the number of arrivals $Z_t \to \infty$ as $t \to \infty$ a.s. and, without loss of generality, we can assume that

$Z_t(\omega) \to \infty$ as $t \to \infty$ for every $\omega$ (by excluding the set of zero probability). Then, for any $v > 0$,

$$\{\text{no job ever leaves after time } v\} \subset \{Q_t \to \infty \text{ as } t \to \infty\}.$$

Now, since there are no departures, the rate of increase of $Q_t$ is equal to the arrival rate, and thus $Q_t \to \infty$. Hence,

$$\mathbb{P}(Q_t \to \infty \text{ as } t \to \infty) \geqslant \mathbb{P}(\text{no job ever leaves after time } v)$$

and by passing $v \to \infty$, we obtain

$$\begin{aligned}\mathbb{P}(Q_t \to \infty \text{ as } t \to \infty) &\geqslant \lim_{v\to\infty} \mathbb{P}(\text{no job ever leaves after } v)\\ &= 1.\end{aligned}$$

□

Finally, we show instability, in general, without the condition $\mathbb{P}[B \geqslant \beta] = 1$.

**Theorem 3.2** *In the M/G/1 PS queue, if $\mathbb{E}A < \infty$ and $B > 0$ a.s., we have as $t \to \infty$,*

$$Q_t \to \infty \quad a.s.$$

**Remark 3** Note that $B > 0$ a.s. is just a non-triviality condition that excludes zero-sized jobs, i.e. non-existent ones.

**Proof:** First, by assumption, we can pick $\beta > 0$ such that $\mathbb{P}[B \geqslant \beta] > 0$. Then, for any time $t$, let $Q_t^\beta$ be the number of jobs whose size is at least $\beta$, i.e. they satisfy $\mathbb{P}[B \geqslant \beta] = 1$, and $q_t^\beta$ be the number of jobs that are smaller than $\beta$. Hence,

$$Q_t = Q_t^\beta + q_t^\beta \geqslant \underline{Q}_t^\beta,$$

where $\underline{Q}_t^\beta$ is the queue in a system with the same arrival process where jobs of size $B \geqslant \beta$ are served in isolation. By Theorem 3.1, $\underline{Q}_t^\beta \to \infty$ a.s., and, therefore, we obtain $Q_t \to \infty$ a.s. □

### 3.2.1 Extension to DPS

In modern system design, PS cannot capture the heterogeneity of users, which is associated with unequal sharing of resources. Hence, we discuss the *Discriminatory* Processor Sharing (DPS) queue which is a multi-class generalization of the PS queue: all jobs are served simultaneously at rates that are determined by a set of weights $w_i, i = 1, \ldots, K$. If there are $n_j$ jobs in class $j$, each class-$k$ job receives service at a rate $c_k = w_k / \sum_{j=1}^K w_j n_j$.

DPS has a broad range of applications. In computing, it is used to model Weighted-Round-Robin (WRR) scheduling. In communication networks, DPS is used for modeling heterogenous, e.g. with different round trip delays, TCP connections. Despite the fact that the PS queue is well understood, the analysis of DPS has proven to be very hard; yet, our previous result on PS is easily extended to DPS in the corollary below.

**Corollary 3.1** *Under the conditions in Theorem 3.2, the Discriminatory Processor Sharing (DPS) queue is also always unstable.*

**Proof:** Without loss of generality, assume that the set of weights is ordered such that $w_1 \leqslant w_2 \ldots \leqslant w_K$. In the M/G/1 DPS queue, the service allocation at any given time $t$ for a single customer in class $k$ is given by

$$c_k(t) = \frac{w_k}{\sum_{i=1}^K w_i n_i(t)} \leqslant \frac{w_k}{w_1 \sum_{i=1}^K n_i(t)} \leqslant \frac{w_K}{w_1 Q_t}.$$

Note that $c(t) = w_K/(w_1 Q_t)$ is the service rate in a PS queue with capacity $c = w_K/w_1 \geqslant 1$. Therefore, each class-$k$ job, $k = 1 \ldots K$, in the DPS queue is served at a lower rate than the rate $c$ of the PS queue. Hence,

$$Q_t^{DPS} \geqslant Q_t^{PS(c)},$$

and since, under the conditions in Theorem 3.2, the PS queue is always unstable, it follows that the DPS queue is also unstable. □

# 4. TRANSIENT BEHAVIOR - SCHEDULING A FINITE NUMBER OF JOBS

In the previous section, we focused on the steady state behavior of the M/G/1 queue and proved that PS is always unstable for failure distributions with finite first moment. In this section, in order to gain further insight into this system, we study its transient behavior. In this regard, we consider a queue with a finite number of jobs and no future arrivals and compute the total time until all jobs are completed. In Subsections 4.1 and 4.2, we analyze the system performance when the jobs are served one at a time and when Processor Sharing (PS) is used, respectively. More precisely, for a finite number of jobs with sizes $B_i, 1 \leqslant i \leqslant m$, and assuming no future arrivals, we study the completion time $\Theta_m$ until all $m$ jobs complete their service.

Note that in the case of traditional work conserving scheduling systems the completion time does not depend on the scheduling discipline and is always simply equal to $\sum_{i=1}^m B_i$. However, in channels with failures there can be a stark difference in the total completion time depending on the scheduling policy. This difference can be so large that in some systems the expected completion time can be infinite while in others finite, or even having many high moments.

Overall, we discover that, with respect to the distribution of the total completion time $\Theta_m$, serving one job at a time exhibits uniformly better performance than PS; see Theorems 4.2 and 4.3. Furthermore, when the hazard functions of the job and failure distributions are proportional, i.e. $\log \bar{F}(x) \sim \alpha \log \bar{G}(x)$, we show that PS performs distinctly worse for the light-tailed job/failure distributions as opposed to the heavy-tailed ones, see parts (i) and (ii) of Theorem 4.3.

Before presenting our results, we restate the following theorem on the logarithmic asymptotics of the service time $S$ (from Definition 2.2), which appears in [12].

**Theorem 4.1** *If $\log \bar{F}(x) \sim \alpha \log \bar{G}(x)$ for all $x \geqslant 0$ and $\alpha > 0$, and $\mathbb{E}[A^{1+\theta}] < \infty$ for some $\theta > 0$, then*

$$\lim_{t\to\infty} \frac{\log \mathbb{P}[S > t]}{\log t} = -\alpha \qquad \text{as } t \to \infty. \tag{4.1}$$

**Remark 4** By examining the proof of Theorem 4.1 in [12], it is easy to check that $\bar{S} = \sum_{i=1}^N A_i \geqslant S$, which includes the time from the job completion until the next failure, also satisfies (4.1).

## 4.1 Serving One Job at a Time

In this subsection, we consider the failure-prone system that was introduced in Section 2, with unit capacity. The jobs are served one at a time, e.g. First Come First Serve (FCFS). Herein, we analyze the performance of this system assuming that, initially, there are $m$ jobs in the queue and there are no future arrivals. Specifically, we study the total completion time, which is defined below.

**Definition 4.1** *The total completion time is defined as the total time until all the jobs are successfully completed and is denoted as*

$$\Theta_m \triangleq \sum_{i=1}^{m} S_i,$$

*where $m$ is the total number of jobs in the system and $S_i$ is the service requirement for each job.*

Next, we define the forward recurrence time, i.e. the elapsed time between some fixed $t_0$ until the time that the next failure occurs after $t_0$.

**Definition 4.2** *Let $L_t$ be the number of failures in the interval $(0,t)$, i.e. the number of regenerative points of the renewal process $(A, \{A_n\}_{n\geqslant 1})$. The forward recurrence time, which corresponds to the elapsed time until the next failure after time $t$, is defined as*

$$\tau_t := \sum_{n=1}^{L_t+1} A_n - t. \tag{4.2}$$

In the following theorem, we prove that the tail asymptotics of the total completion time, from Definition 4.1, under this policy is a power law of the same index as the service time of a single job.

**Theorem 4.2** *If $\log \bar{F}(x) \sim \alpha \log \bar{G}(x)$ for all $x \geqslant 0$ and $\alpha > 0$, and $\mathbb{E}[A^{1+\theta}] < \infty$ for some $\theta > 0$, then*

$$\lim_{t\to\infty} \frac{\log \mathbb{P}[\Theta_m > t]}{\log t} = -\alpha.$$

**Proof:** Recall that the service requirement for a job $B_i$ was previously defined as $S_i = \sum_{j=1}^{N_i-1} A_j + B_i$.

For the *lower* bound, we observe that

$$\mathbb{P}[\Theta_m > t] \geqslant \mathbb{P}[S_1 > t],$$

since the total completion time is at least equal to the service time of a single job. By taking the logarithm and using Theorem 4.1, we have

$$\frac{\log \mathbb{P}[\Theta_m > t]}{\log t} \geqslant -(1+\epsilon)\alpha. \tag{4.3}$$

For the *upper* bound, we compare $\Theta_m$ with the completion time in a system where the server is kept idle between the completion time of the previous job and the next failure. Clearly,

$$\Theta_m \leqslant \bar{\Theta}_m \triangleq \sum_{i=1}^{m} \bar{S}_i, \tag{4.4}$$

where $\bar{S}_i \triangleq \sum_{j=1}^{N_i} A_j$ are the service times that include the remaining availability period $A_{N_i}$. We prove this intuitive claim more formally by induction in the appendix.

Then, we argue that

$$\mathbb{P}[\Theta_m > t] \leqslant \mathbb{P}\left[\sum_{i=1}^{m} \bar{S}_i > t\right] \leqslant m\mathbb{P}\left[\bar{S}_1 > \frac{t}{m}\right],$$

which follows from the union bound. By taking the logarithm and using Theorem 4.1, we have

$$\frac{\log \mathbb{P}[\Theta_m > t]}{\log t} \leqslant -\alpha(1-\epsilon) + \frac{\log m}{\log t} \leqslant -(1-2\epsilon)\alpha, \tag{4.5}$$

where we pick $t$ large enough such that $\log t \geqslant \log m/(\alpha\epsilon)$.

Letting $\epsilon \to 0$ in both (4.3) and (4.5) finishes the proof. □

## 4.2 Processor Sharing Discipline

In this subsection, we analyze the Processor Sharing discipline where $m$ jobs share the (unit) capacity of a single server. We present our main theorem on the logarithmic scale, which shows that the tail asymptotics of the total completion time is determined by the shortest job in the queue. In particular, under our main assumptions, this time is a power law, but it exhibits a different exponent depending on the job size distribution.

- If the jobs are subexponential (heavy-tailed) or exponential, the total delay is simply determined by the time required for any single job to complete its service, as if it was the only one present in the queue.

- If the jobs are superexponential (light-tailed), the total delay is determined by the service time of the *shortest* job. This job generates the heaviest asymptotics among all the rest.

Our main result, stated in Theorem 4.3 below, shows that on the logarithmic scale the distribution of the total completion time $\Theta_m^{PS}$ is heavier by a factor $m^{\gamma-1}$ for superexponential jobs relative to the subexponential or exponential case. Therefore, in systems with failures and restarts, sharing the capacity among light-tailed jobs induces long delays, whereas, for heavy-tailed ones, PS appears to perform as good as serving the jobs one at a time. Interestingly enough, this deterioration in performance is determined by the time it takes to serve the shortest job in the system.

Note that in a PS queue with no future arrivals, the shortest job will depart first. Immediately after this, the server will continue serving the remaining $m-1$ jobs, and, similarly, the shortest job, i.e. the second shortest among the original $m$ jobs, will depart before all the others. This pattern will continue until the departure of the largest job, which is served alone.

**Theorem 4.3** *Assume that the hazard function $-\log \bar{F}(x)$ is regularly varying with index $\gamma \geqslant 0$. If $\log \bar{F}(x) \sim \alpha \log \bar{G}(x)$ for all $x \geqslant 0$ and $\alpha > 0$, and $\mathbb{E}[A^{1+\theta}] < \infty$ for some $\theta > 0$, then*

*(i) if $\gamma \leqslant 1$, i.e. $B$ is subexponential or exponential, then*

$$\lim_{t\to\infty} \frac{-\log \mathbb{P}[\Theta_m^{PS} > t]}{\log t} = \alpha,$$

*(ii) if $\gamma > 1$, i.e. $B$ is superexponential, then*

$$\lim_{t\to\infty} \frac{-\log \mathbb{P}[\Theta_m^{PS} > t]}{\log t} = \frac{\alpha}{m^{\gamma-1}} < \alpha.$$

**Remark 5** When $\alpha > 1$, we easily verify that $\mathbb{E}[\Theta_m^{PS}] < \infty$ in case $(i)$, and the system is stable; if the jobs are superexponential, e.g. case $(ii)$, then $\mathbb{E}[\Theta_m^{PS}] = \infty$ if $\alpha < m^{\gamma-1}$.

**Proof:** Let $B^{(1)} \leqslant B^{(2)} \leqslant \ldots \leqslant B^{(m)}$ be the order statistics of the jobs $B_1, B_2, \ldots, B_m$.

The assumption that $-\log \bar{F}(x)$ is regularly varying with index $\gamma$ implies that

$$\log \bar{F}(\lambda x) \sim \lambda^\gamma \log \bar{F}(x), \tag{4.6}$$

for any $\lambda > 0$.

We begin with the *lower* bound.

*(i) Subexponential or exponential jobs ($\gamma \leqslant 1$).*
The total completion time is lower bounded by the time required for a single job to depart when it is exclusively served, e.g. if the total capacity of the system is used. Hence, it follows that

$$\mathbb{P}[\Theta_m^{PS} > t] \geqslant \mathbb{P}[S_1 > t], \tag{4.7}$$

where $S_1$ is the service time of a single job of random size $B_1$, when there are no other jobs in the system. Now, recalling Theorem 4.1, it holds that

$$\lim_{t\to\infty} \frac{\log \mathbb{P}[S_1 > t]}{\log t} = -\alpha.$$

By taking the logarithm in (4.7), the lower bound follows immediately.

*(ii) Superexponential jobs ($\gamma > 1$).*
The total completion time is lower bounded by the delay experienced by the shortest job, and hence,

$$\mathbb{P}[\Theta_m^{PS} > t] \geqslant \mathbb{P}[S_1^{PS} > t], \tag{4.8}$$

where $S_1^{PS}$ is the service time of the smallest job $B^{(1)}$. First, note that the distribution of $B^{(1)}$ is given by

$$\begin{aligned}\mathbb{P}(B^{(1)} > x) &= \mathbb{P}(B_1 > x, B_2 > x, \ldots, B_m > x)\\ &= \mathbb{P}(B_1 > x)\mathbb{P}(B_2 > x)\cdots\mathbb{P}(B_m > x)\\ &= \mathbb{P}(B_1 > x)^m = \bar{F}(x)^m,\end{aligned} \tag{4.9}$$

since $B_i, i = 1, \ldots, m$, are independent and identically distributed. Now, the service time $S_1^{PS}$ is determined by the number of failures this job has experienced, i.e.

$$\mathbb{P}[N_1 > n] = \mathbb{E}\left[\mathbb{P}\left(B^{(1)} > \frac{A}{m}\right)\right]^n = \mathbb{E}\left(1 - \bar{G}(mB^{(1)})\right)^n,$$

and, using (4.9) and (4.6), together with our main assumption, we observe that

$$\begin{aligned}\log \mathbb{P}(mB^{(1)} > x) &= m \log \bar{F}\left(\frac{x}{m}\right)\\ &\sim m^{1-\gamma} \log \bar{F}(x) \sim \alpha m^{1-\gamma} \log \bar{G}(x).\end{aligned}$$

Then, Theorem 4.1 applies with $\alpha/m^{\gamma-1} \leqslant \alpha$, i.e.

$$\lim_{t\to\infty} \frac{\log \mathbb{P}[S_1^{PS} > t]}{\log t} = -\frac{\alpha}{m^{\gamma-1}}.$$

Next, we derive the *upper* bound. To this end, we consider a system where the server is kept idle after the completion of each job until the next failure occurs. At this time, all the remaining jobs are served under PS until the next shortest one departs. If there are more than one jobs of the same size, only one of these departs. Under this policy, it clearly holds that

$$\Theta_m^{PS} \leqslant \sum_{i=1}^m \bar{S}_i^{PS},$$

where $\bar{S}_i^{PS}$ corresponds to the service time of the $i^{th}$ smallest job and includes the time until the next failure.

Using the union bound, we obtain

$$\mathbb{P}[\Theta_m^{PS} > t] \leqslant \mathbb{P}\left[\sum_{i=1}^m \bar{S}_i^{PS} > t\right] \leqslant (1+\epsilon)\sum_{i=1}^m \mathbb{P}\left(\bar{S}_i^{PS} > \frac{t}{m}\right). \tag{4.10}$$

It is easy to see that the service time of the $i^{th}$ smallest job $B^{(i)}$ depends on the number of jobs that share the server, i.e. $m - i + 1$, since $m - i$ jobs have remained in the queue. Now, the distribution of the $i^{th}$ shortest job is derived as

$$\begin{aligned}\mathbb{P}(B^{(i)} > x) &= \sum_{k=0}^{i-1} \binom{m}{k} \mathbb{P}(B_1 \leqslant x)^k \mathbb{P}(B_1 > x)^{m-k}\\ &\sim \binom{m}{i-1} \mathbb{P}(B_1 > x)^{m-i+1} \sim \bar{F}(x)^{m-i+1}.\end{aligned} \tag{4.11}$$

The number of restarts for the $i^{th}$ smallest job, $N_i$, is computed as

$$\begin{aligned}\mathbb{P}[N_i > n] &= \mathbb{E}\left[\mathbb{P}\left(B^{(i)} > \frac{A}{m-i}\right)\right]^n\\ &= \mathbb{E}\left(1 - \bar{G}((m-i+1)B^{(i)})\right)^n.\end{aligned}$$

Next, starting from (4.11), it easily follows that

$$\begin{aligned}\log \mathbb{P}\left((m-i+1)B^{(i)} > x\right) &\sim \log \bar{F}\left(\frac{x}{m-i+1}\right)^{m-i+1}\\ &\sim (m-i+1)^{1-\gamma} \log \bar{F}(x)\\ &\sim \alpha(m-i+1)^{1-\gamma} \log \bar{G}(x),\end{aligned}$$

where we use (4.6) and our main assumption and define $\alpha_i \triangleq \alpha/(m-i)^{\gamma-1}$.

Now, recalling Theorem 4.1 and Remark 4 after it, we have

$$\frac{\log \mathbb{P}[\bar{S}_i^{PS} > t]}{\log t} \to \alpha_i \text{ as } t \to \infty,$$

and thus (4.10) yields

$$\frac{\log \mathbb{P}[\Theta_m^{PS} > t]}{\log t} \leqslant -(1-\epsilon) \min_{i=1\ldots m} \alpha_i,$$

for all $t \geqslant t_0$.

*(i) Subexponential or exponential jobs ($\gamma \leqslant 1$).*
Observe that $\min_{i=1\ldots m} \alpha_i = \alpha$, and thus

$$\frac{\log \mathbb{P}[\Theta_m^{PS} > t]}{\log t} \leqslant -(1-\epsilon)\alpha. \tag{4.12}$$

*(ii) Superexponential jobs ($\gamma > 1$).*
In this case, $\min_{i=1\ldots m} \alpha_i = \alpha/m^{\gamma-1}$, and thus

$$\frac{\log \mathbb{P}[\Theta_m^{PS} > t]}{\log t} \leqslant -(1-\epsilon)\frac{\alpha}{m^{\gamma-1}}. \qquad (4.13)$$

Letting $\epsilon \to 0$ in (4.12) and (4.13), we obtain the upper bound. □

# 5. SIMULATION

In this section, we present our simulation experiments in order to demonstrate our theoretical findings. All the experiments result from $N = 10^8$ (or more) samples of each simulated scenario; this guarantees the existence of at least 100 occurrences in the lightest end of the tail that is presented in the figures. First, we illustrate the instability result from Theorem 3.1 of Section 3.

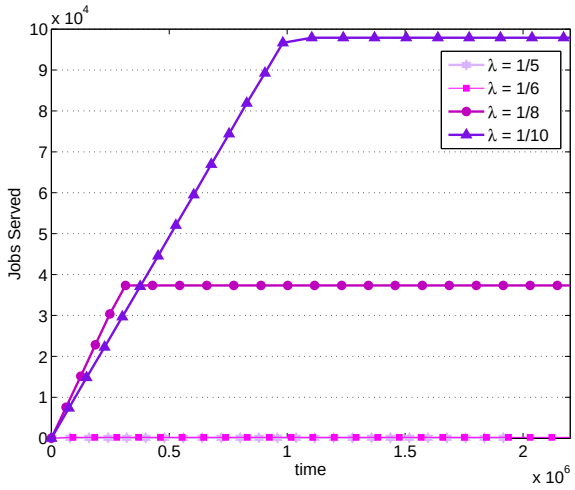


**Figure 3: Example 1. Jobs served over time.**

**Example 1.** *M/G/1 PS is unstable.* In this example, we show that the PS queue becomes unstable by simulating the M/G/1 PS queue for different arrival rates $\lambda > 0$, which all satisfy the stability condition for the M/G/1 FCFS queue, when jobs are served one at a time. In this regard, we assume constant job sizes $\beta = 1$ and Poisson failures of rate $\mu = 1/20$. Therefore, by evaluating (3.4), we obtain

$$\lambda\mathbb{E}[S] = \lambda\mu^{-1}(e^{\mu} - 1) = 20(e^{0.05} - 1)\lambda = 1.025\lambda < 1,$$

or equivalently the stability region for the FCFS queue is given by $\Lambda = \{\lambda \leqslant 0 : \lambda < 0.9752\}$. Hence, in this example, we use $\lambda$ from the FCFS stability region, $\lambda \in \Lambda$.

In Fig. 3, we plot the number of jobs that have received service up to time $t$. We observe that the cumulative number of served jobs always converges to a fixed number and does not increase any further. This happens after some critical time when the queue starts to grow continuously until it becomes unable to drain. For larger values of $\lambda$, the system saturates faster meaning that the cumulative throughput at the saturated state is lower.

Furthermore, we observe from the simulation that the system behaves as if it were stable until some critical time or queue size after which it is unable to drain. From Fig. 3, we can see that the case $\lambda = 10^{-1}$ saturates at time $t = 10^6$ and the total number of served jobs reaches $10^5$. Hence, the departure rate until saturation time is $10^5/10^6 = 10^{-1}$, which is exactly equal to the arrival rate $\lambda = 10^{-1}$, corresponding to the departure rate of a stable queue. This further emphasizes the importance of studying the stability of these systems since, at first glance, they may appear stable.

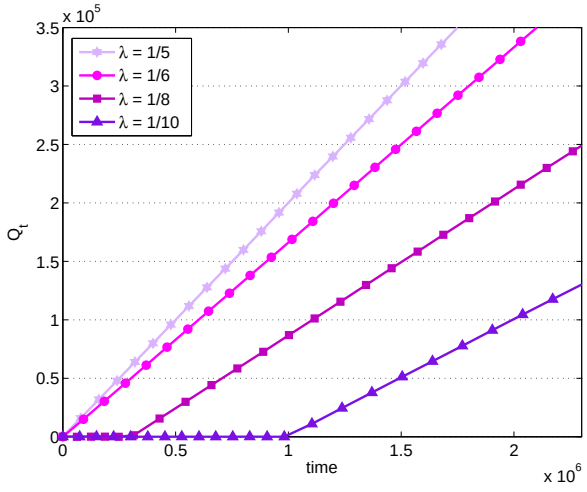


**Figure 4: Example 1. Queue size over time.**

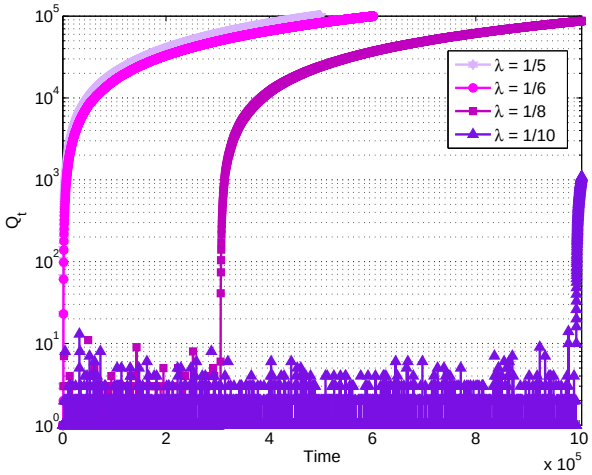


**Figure 5: Example 1. Queue size for small $t$. It zooms in the time range $[0, 10^6]$ of Fig. 4; $Q_t$ ($y$-axis) is shown on the logarithmic scale.**

Fig. 4 demonstrates the queue size evolution over time. Similarly as in Fig. 3, we observe that for any arrival rate $\lambda$, there is a critical time after which the queue continues to grow and never empties. This time varies depending on the simulation experiment; yet, on average, we observe that the queue remains stable for longer time when $\lambda$ is smaller. Now, we zoom in on the queue evolution on the logarithmic scale in Fig. 5. Again, we observe that the queue looks stable until some critical time/queue size.

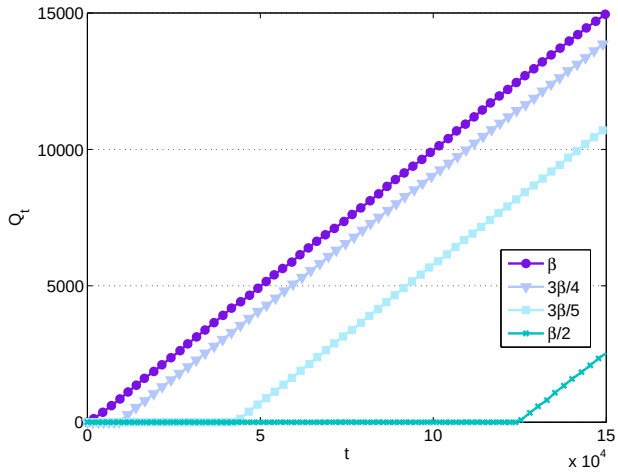


**Figure 6: Example 1. Queue size over time parameterized by fragment length; $\beta = 2, \lambda = 0.1$.**

Last, in Fig. 6, we plot the queue evolution for different job sizes, namely $\beta = 1, 1.2, 1.5$ and 2. We observe that larger job fragments cause instability much faster than the smaller units. For example, $\beta = 2$ leads to instability almost immediately, while $\beta = 1.5$ renders the queue unstable after $10^4$ time units. Similarly, reducing the fragment size by 60% delays the process by an additional $3 \times 10^4$ units. Last, cutting the jobs in half causes instability after approximately $13 \times 10^4$ time units. This implies that one should apply fragmentation with caution in order to select the appropriate fragment size that will maintain good system performance for the longest time.

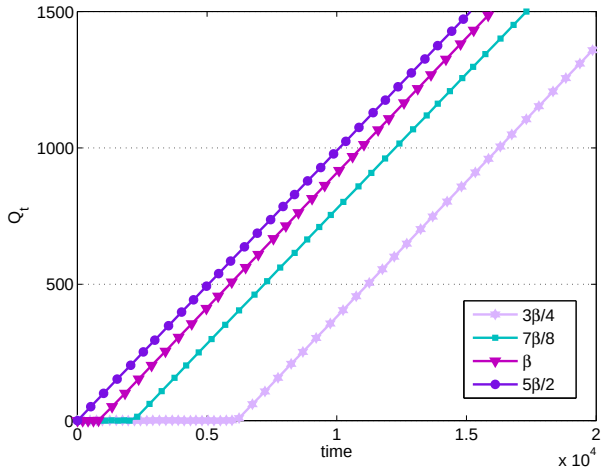


**Figure 7: Example 2. Queue size over time parameterized by job size; $\beta = 4$.**

**Example 2.** *General arrivals.* In this example, we consider non-Poisson arrivals. We assume that the failure distribution is exponential with mean $\mathbb{E}A = 10$ and that jobs interarrival times follow the Pareto distribution with $\alpha = 2$ and mean $\mathbb{E}A = 10.1$. Similarly as in the previous example, Fig. 7 shows the queue evolution with time for different job sizes $\beta$.

Next, we validate the results on the transient analysis from Section 4.

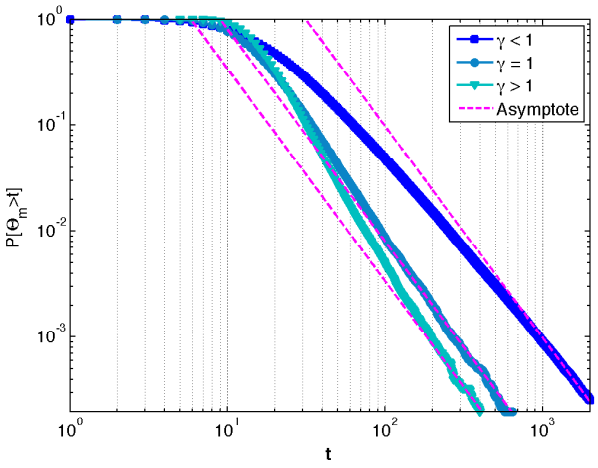


**Figure 8: Example 3. FCFS: Logarithmic asymptotics when $\alpha = 2$ for exponential, superexponential ($\gamma > 1$) and subexponential ($\gamma < 1$) distributions.**

**Example 3.** *Serving one job at a time/FCFS: Always the same index $\alpha$.* In this example, we consider a queue of $m = 10$ jobs, which are served First Come First Serve (FCFS), i.e. one at a time. The logarithmic asymptotics from Theorem 4.1 implies that the tail is always a power law of index $\alpha = 2$.

In Fig. 8, we plot the distribution of the total completion time in a queue with 10 jobs that are processed one at a time. On the same graph, we plot the logarithmic asymptotics (dotted lines) that correspond to a power law of index $\alpha = 2$. We consider the following three scenarios:

1. Weibull distributions with $\gamma = 2$. The failures $A$ are distributed according to $\bar{G}(x) = e^{-(x/\mu)^2}$ with mean $\mathbb{E}[A] = \mu\Gamma(1.5) = 1.5$, and jobs $B$ also follow Weibull distributions with $\bar{F}(x) = e^{-(x/\lambda)^2}, \lambda = \mu/\sqrt{2}$. In this case, it is easy to check that the main assumption of Theorem 4.1 is satisfied, i.e.
$$\log \bar{F}(x) = -(x/\lambda)^2 = \alpha \log \bar{G}(x), \alpha = (\mu/\lambda)^2.$$

2. Exponential distributions. $A$'s are exponential with $\mathbb{E}[A] = 2$, $\bar{G}(x) = e^{-x/2}$, and the jobs $B$ are also exponential of unit mean, i.e. $\bar{F}(x) = e^{-x}$. Then, trivially,
$$\log \bar{F}(x) = 2 \log \bar{G}(x).$$

3. Weibull distributions with $\gamma = 0.5$. $A$'s are Weibull with $\bar{G}(x) = e^{-\sqrt{x}/2}$, i.e $\mathbb{E}[A] = 8$. Also, we assume Weibull jobs $B$ with $\bar{F}(x) = e^{-\sqrt{x}}$. Thus,
$$\log \bar{F}(x) = -\sqrt{x} = 2 \log \bar{G}(x).$$

In all three cases, we obtain $\alpha = 2$. Yet, we observe that the tail asymptotics is the same regardless of the distribution of the job sizes. For the subexponential jobs (case 3: Weibull with $\gamma < 1$), the power law tail appears later compared

to the case of superexponential jobs. This is because the constant factor of the exact asymptotics is different for each case, and it depends on the mean size of $A$, $\mathbb{E}[A]$.

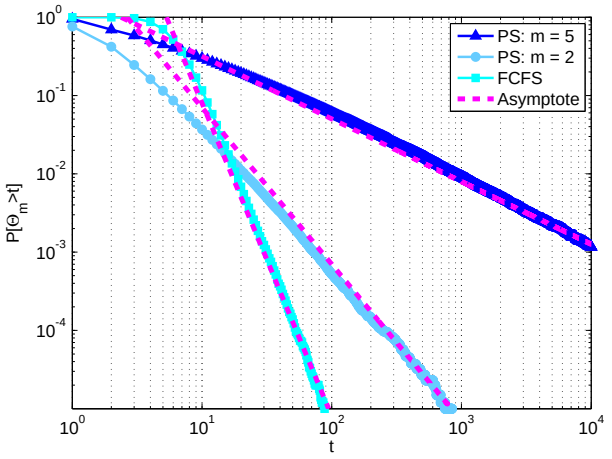


**Figure 9: Example 4. Logarithmic asymptotics for different number of superexponential jobs when $\alpha = 4$ under PS and FCFS discipline.**

**Example 4.** *PS: The effect of the number of jobs.* In this example, we consider a PS queue with $m = 5$ and $m = 2$ superexponential jobs, and compare it against a FCFS queue with $m = 5$ jobs. We assume superexponential job sizes $B$'s and $A$'s, namely Weibull with $\gamma = 2$; see case 1 of Example 3. Here $\alpha$ is taken equal to 4. The logarithmic asymptotics is given in Theorems 4.2 and 4.3.

In Fig. 9, we demonstrate the total completion time $\Theta_m^{PS}$, for different number of jobs, when $\gamma = 2$. Theorem 4.2(ii) states that $\alpha(m) = \alpha/m^{\gamma-1}$ and, thus, for $\gamma = 2$ we have $\alpha(m) = \alpha/m$, e.g. we expect power law asymptotes with index $\alpha/m$ for the different values of $m$. On the same figure, we also plot the FCFS completion time $\Theta_m$, which is always a power law of index $\alpha = 4$, as we previously observed in Example 3. It can be seen that PS generates heavier power laws, for superexponential jobs. In particular, PS with $m = 2$ results in power law asymptotics with $\alpha(2) = 2$, while PS with $m = 5$ jobs leads to system instability since $\alpha(5) = 4/5 < 1$.

**Example 5.** *PS: The effect of the distribution type.* In this example, for completeness, we evaluate the impact of the job distribution on the total completion time under both heavy and light-tailed job sizes. To this end, we consider the PS queue from Example 4, with $m = 5$ jobs, and compare it against FCFS. In Fig. 10, we re-plot the logarithmic asymptotics of the total completion time $\mathbb{P}(\Theta_m^{PS} > t)$, for different distribution types of the failures/jobs and index $\alpha = 4$, as before. In particular, we consider Weibull distributions as in Example 3 with $\gamma = 1/2 < 1$ and $\gamma = 2 > 1$ for the subexponential and superexponential cases, respectively.

On the same graph, we plot the distribution of the completion time $\Theta_m$ in FCFS, which is always a power law of the same index, as illustrated in Example 3. By fixing the number of jobs to be $m = 5$, Fig. 10 shows that when the jobs are superexponential, PS yields the heaviest asymptotics among all three scenarios; for subexponential jobs,

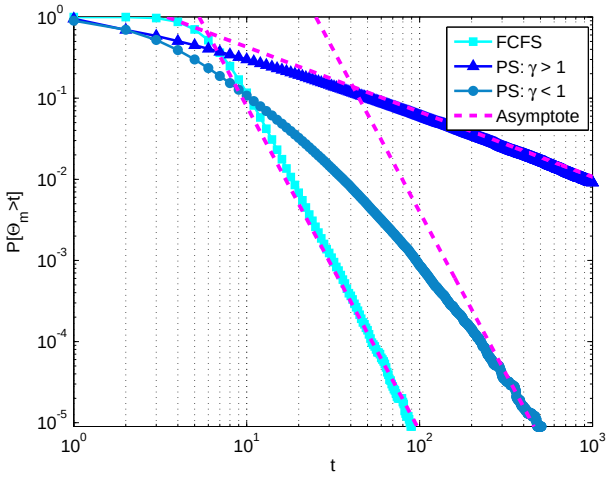


**Figure 10: Example 5. Logarithmic asymptotics under FCFS, PS with subexponential and superexponential jobs.**

PS generates asymptotics with the same power law index as in FCFS, albeit with a different constant factor.

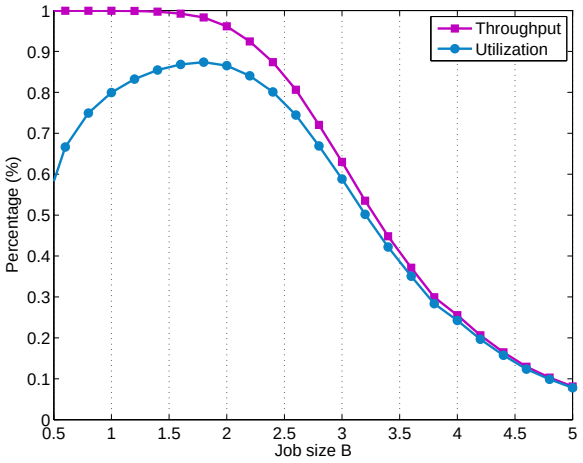


**Figure 11: Example 6. Throughput vs. utilization tradeoff.**

**Example 6.** *Limited queue: Throughput vs. overhead tradeoff.* In practice, job and buffer sizes are bounded and therefore the queue may never become unstable. However, our results indicate that the queue may lock itself in a 'nearly unstable' state, where it is at its maximum size and the throughput is very low. Here, we would like to emphasize that, unlike in the case of unlimited queue size, job fragmentation can be useful in increasing the throughput and the efficiency of the system. In this case, one has to be careful about the overhead cost of fragmentation. Basically, each fragment requires additional information, called the 'header' in the context of communications, which contains details on how it fits into the bigger job, e.g. destination/routing information in communication networks. Hence, if the fragments

are too small, there will be a lot of overhead and waste of resources. In view of this fact, one would like to optimize the fragment sizes by striking a balance between throughput and utilization.

In this example, we demonstrate the tradeoff between throughput and generated overhead, assuming limited queue size $\tilde{Q}$. If the newly arriving job does not fit in the queue, i.e. the number of jobs currently in the queue is equal to $\tilde{Q}$, it is discarded. We define throughput as the percentage of the jobs that complete service among all jobs that arrive at the M/G/1 PS queue. It basically corresponds to the total work that is carried out in the system. On the other hand, we define utilization as the useful work that is served over the aggregated load in the system. Specifically, we consider jobs that require a minimum size $\beta$, where $\beta$ represents the overhead, e.g. the packet header, thread id, etc. The remaining job size, $B-\beta$, represents the useful information.

We consider different job sizes $B$ from 0.4 up to 5 bytes, with overhead $\beta = 0.2$. We simulate the M/G/1 PS queue with maximum queue size $\tilde{Q} = 10$ jobs for a fixed time $T = 10^8$ time units. The arrivals are Poisson with rate 1/10 and the failures are exponential of the same rate.

In Fig. 11, we observe that for small job sizes, the throughput is 100% and it deteriorates as the job size $B$ increases. In particular, when the job size exceeds 1.5, the throughput drops exponentially. Utilization exhibits a different behavior; it is low when the job size is small, i.e. the useful job size is comparable to the overhead $\beta$, and reaches its peak at $B \approx 1.8$. After this, it starts decreasing following similar trend as the throughput. In this case, $B-\beta \approx 1.5$ appears to be the optimal size for the job fragments. This phenomenon of combining limited queue size with job fragmentation may require further investigation.

# 6. CONCLUDING REMARKS

Retransmissions/restarts represent a primary failure recovery mechanism in large-scale engineering systems, as it was argued in the introduction. In communication networks, retransmissions lie at the core of the network architecture, as they appear in all layers of the protocol stack. Similarly, PS/DPS based scheduling mechanisms, due to their inherent fairness, are commonly used in computing and communication systems. Such mechanisms allow for efficient and fair resource allocation, and thus they are preferred in engineering system design.

However, our results show that, under mild conditions, PS/DPS scheduling in systems with retransmissions is always unstable. Furthermore, this instability cannot be resolved by job fragmentation techniques. On the contrary, serving one job at a time, e.g. FCFS, can be stable and its performance can be further enhanced with fragmentation. Interestingly, systems where jobs are served one at a time can highly benefit from fragmentation and, in fact, their performance can approach closely the corresponding system without failures.

Overall, using PS in combination with retransmissions in the presence of failures deteriorates the system performance and induces instability. In addition, our findings suggest that further examination of existing techniques is necessary in the failure-prone environment with retransmission/restart failure recovery and sharing, e.g. see Example 6.

# 7. ACKNOWLEDGMENTS

The authors are grateful to the anonymous reviewers for their helpful comments.

# APPENDIX

**Proof:** [of (4.4) in Theorem 4.2]
We formally prove that $\Theta_m \leqslant \bar{\Theta}_m = \sum_{i=1}^{m} \bar{S}_i$. The proof follows by induction.

*n = 1*. Let $\tau_{\Theta_1}$ denote the time between $\Theta_1$ until the first failure occurs after $\Theta_1$, i.e. the time after the departure of the first job (see also Definition 4.2). Then the total service time for jobs $B_1$ and $B_2$ is

$$\Theta_2 = \begin{cases} \Theta_1 + \tau_{\Theta_1} + S_2, & \text{if } B_2 > \tau_{\Theta_1} \\ \Theta_1 + B_2, & \text{otherwise.} \end{cases}$$

If we idle the server after the successful completion of the first job until the next failure, the total completion time will be equal to $\bar{\Theta}_2 = \Theta_1 + \tau_{\Theta_1} + S_2$, since we discard the remaining interval $\tau_{\Theta_1}$ and start service at $\Theta_1 + \tau_{\Theta_1}$. Therefore, $\Theta_2 \leqslant \bar{\Theta}_2$ (see Figure 12 for an illustration).

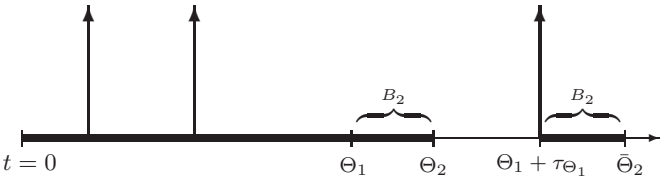


**Figure 12: Completion time in a failure-prone system: Assume that there are two jobs $B_1$ and $B_2$ and the first succeeds at time $\Theta_1$. In the original system, job $B_2$ starts service immediately and completes at time $\Theta_2$, before the next failure occurs after $\tau_{\Theta_1}$ time units. In the alternate system, $B_2$ will only start its service at time $\Theta_1 + \tau_{\Theta_1}$. If $B_2 < \tau_{\Theta_2}$, then $\bar{\Theta}_2 = \Theta_1 + \tau_{\Theta_1} + B_2$.**

*Induction step.* Assume $\Theta_n \leqslant \bar{\Theta}_n$ for $n < m$. If $\Theta_n$ is the time when the $n^{th}$ job is completed, then $\tau_{\Theta_n}$ is the time until the next failure after $\Theta_n$. Now, for the following job $B_{n+1}$, we have

$$\Theta_{n+1} = \begin{cases} \Theta_n + \tau_{\Theta_n} + S_{n+1}, & \text{if } B_{n+1} > \tau_{\Theta_n} \\ \Theta_n + B_{n+1}, & \text{otherwise.} \end{cases}$$

If $\Theta_n = \bar{\Theta}_n$, then clearly $\bar{\Theta}_{n+1} = \Theta_n + \tau_{\Theta_n} + S_{n+1} \geqslant \Theta_{n+1}$.

If $\bar{\Theta}_n > \Theta_n$ then, by construction, job $B_{n+1}$ can start its service after time $\Theta_n + \tau_{\Theta_n}$, i.e the time that the first failure occurs after $\Theta_n$. This implies that $\bar{\Theta}_n \geqslant \Theta_n + \tau_{\Theta_n}$. Now, if $B_{n+1}$ finishes before the failure occurs, then clearly $\bar{\Theta}_{n+1} \geqslant \Theta_{n+1}$. If not, it will either succeed during the period $(\Theta_n + \tau_{\Theta_n}, \bar{\Theta}_n)$ implying that $\Theta_{n+1} \leqslant \bar{\Theta}_{n+1}$, or it will synchronize with the other system and $\Theta_{n+1} = \bar{\Theta}_n + \tau_{\bar{\Theta}_n} + S_{n+1} \leqslant \bar{\Theta}_{n+1}$. □

# A. REFERENCES

[1] E. Altman, K. Avrachenkov, and U. Ayesta. A survey on discriminatory processor sharing. *Queueing Systems*, 53(1-2):53–63, June 2006.

[2] V. Anantharam. Scheduling strategies and long-range dependence. *Queueing Systems*, 33(1/3):73–89, March 1999.

[3] D. P. Bertsekas and R. Gallager. *Data Networks*. Prentice Hall, 2nd edition, 1992.

[4] E. G. Coffman, Jr., R. R. Muntz, and H. Trotter. Waiting time distributions for processor-sharing systems. *Journal of the ACM*, 17(1):123–130, Jan. 1970.

[5] G. Fayolle, I. Mitrani, and R. Iasnogorodski. Sharing a processor among many job classes. *Journal of the ACM*, 27(3):519–532, July 1980.

[6] P. M. Fiorini, R. Sheahan, and L. Lipsky. On unreliable computing systems when heavy-tails appear as a result of the recovery procedure. *SIGMETRICS Performance Evaluation Review*, 33(2):15–17, December 2005.

[7] P. Jelenković and P. Momčilović. Large deviation analysis of subexponential waiting times in a processor sharing queue. *Mathematics of Operations Research*, 28(3):587–608, 2003.

[8] P. R. Jelenković and E. D. Skiani. Distribution of the number of retransmissions of bounded documents. *Advances in Applied Probability (submitted)*, New York, 2012, eprint arXiv:1210.8421v1.

[9] P. R. Jelenković and E. D. Skiani. Retransmissions over correlated channels. *SIGMETRICS Performance Evaluation Review*, 41(2):15–25, August 2013.

[10] P. R. Jelenković and E. D. Skiani. Uniform approximation of the distribution for the number of retransmissions of bounded documents. In *Proceedings of the 12th ACM SIGMETRICS/PERFORMANCE joint international conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '12, pages 101–112, June 2012.

[11] P. R. Jelenković and J. Tan. Can retransmissions of superexponential documents cause subexponential delays? In *Proceedings of IEEE INFOCOM'07*, pages 892–900, May 2007.

[12] P. R. Jelenković and J. Tan. Characterizing heavy-tailed distributions induced by retransmissions. *Advances in Applied Probability*, 45(1): 106-138, 2013. (extended version) arXiv: 0709.1138v2.

[13] L. Kleinrock. Time-shared systems: A theoretical treatment. *Journal of the ACM*, 14(2):242–261, April 1967.

[14] A. K. Parekh and R. G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: The single-node case. *IEEE/ACM Transactions on Networking*, 1(3):344–357, June 1993.

[15] A. K. Parekh and R. G. Gallagher. A generalized processor sharing approach to flow control in integrated services networks: The multiple node case. *IEEE/ACM Transactions on Networking*, 2(2):137–150, April 1994.

[16] R. Sheahan, L. Lipsky, P. M. Fiorini, and S. Asmussen. On the completion time distribution for tasks that must restart from the beginning if a failure occurs. *SIGMETRICS Performance Evaluation Review*, 34(3):24–26, December 2006.

[17] A. Wierman and B. Zwart. Is tail-optimal scheduling possible? *Operations Research*, 60(5):1249–1257, September 2012.

[18] S. Yashkov. Mathematical problems in the theory of shared-processor systems. *Journal of Soviet Mathematics*, 58(2):101–147, 1992.

[19] S. Yashkov and A. Yashkova. Processor sharing: A survey of the mathematical theory. *Automation and Remote Control*, 68(9):1662–1731, 2007.