Simulation of Several Round-Robin Variants in $M/G/1$ queues

Feng Zhang and Lester Lipsky
Department of Computer Science and Engineering
371 Fairfield Rd, Unit 2155
University of Connecticut, Storrs, CT 06269-2155
{Fzhang,Lester}@engr.uconn.edu

Keywords: Processor sharing, round-robin, shortest remaining processing time (SRPT), coefficient of variation, Pollaczek-Khinchin formula

Abstract
The concept of processor sharing introduces a heuristics to prevent long jobs from blocking short jobs in queue. In practice, round-robin has to be used. In the literature, it is generally assumed that newly arrived jobs are put at the end of the queue. However, for highly varying job demands, such an approach may perform poorly. By favoring newly arrived jobs, which are more likely to be short, the performance of round-robin can be improved considerably. In this paper, we study the performance of several round-robin variants as a function of time slice values for different classes of service time distributions.

1 INTRODUCTION
In the last ten years or so, it has become clear that the service times of many types of jobs (e.g., running times of optimization programs and durations of Internet sessions) have high variance (i.e., coefficient of variation $C_v^2 = \sigma^2/\bar{x}^2 \gg 1$, where $\bar{x}$ is mean service time and $\sigma$ is standard deviation) [1]. In other words, while the service times of most jobs are relatively short, there is a non-negligible probability that the service times of some jobs can be quite long. To handle highly varying job demands effectively, it is important for a queueing discipline or scheduling strategy to favor short jobs. For example, the well-known shortest remaining processing time (SRPT) strategy [2, 3], which was shown to be optimal, always preempts the current job in execution if the newly arrived job has less service time requirement. Similarly, the concept of processor sharing introduces a heuristics to prevent long jobs from blocking short jobs in queue. In 1970s, it was shown that an $M/G/1$ queue using processor sharing has the same mean system time (or response time) as an $M/M/1$ queue, which considerably outperforms an $M/G/1$ queue using FCFS if $C_v^2 \gg 1$.

In practice, processor sharing cannot be implemented due to its unrealistic assumption of serving each job at $1/k$-th the speed of a processor if there are $k$ jobs sharing the processor. Instead, one has to do round-robin (or sequential time slicing) to serve each job. Specifically, a job is chosen from the front of the queue and is served for at most a time slice $\Delta$. If it completes its service time requirement before the time slice, it releases the processor and leaves the system right away. Otherwise, it is preempted after the allocated time slice and put back at the end of the queue, awaiting for further service. Various round-robin strategies, such as prioritized round-robin [4], cycled round-robin [5], deficit round-robin [6], and group round-robin [7], have been developed for scheduling jobs in computer and communication systems. In studying these strategies, fairness is often considered as a major goal. However, an important parameter of round-robin (as well as its variants) is time slice $\Delta$, which could influence the performance of round-robin considerably. As far as we know, round-robin has not been well studied as a function of $\Delta$ for different classes of distributions. While analytical solutions of mean system time are known for the case of $\Delta \to 0$ (i.e., processor sharing) and the case of $\Delta \to \infty$ (i.e., FCFS), it is not clear how round-robin performs for the intermediate case of choosing $\Delta$. We have found that, for distributions with high variance (i.e., $C_v^2 > 1$), there actually exist some $\Delta$ values that outperform the case of $\Delta$ being zero. In this paper, we attempt to examine this issue. Certainly, there are no analytical results on time slice values. So everything will be based on extensive simulation.

In the rest of the paper, we first review the related work. Next, we summarize the known results of processor sharing and round-robin and describe several round-robin variants. The results are presented in Section 5.
2 RELATED WORK

Many researchers studied processor sharing, round-robin, and their variants. Because of their analytic tractability, \( M/M/1 \) queues [4, 8, 9, 10, 11] have been extensively studied and analytical solutions have been obtained. In particular, expressions of expected response time of a job conditioned on its service time were given for round-robin, processor sharing (with or without priority), \( FB_N \) (\( N \)-level foreground-background queue with finite \( \Delta \), favoring jobs with less attained service times) and \( FB_\infty \) (\( \Delta \rightarrow 0 \), FCFS, and so on. For \( M/M/1 \) queues, it is clear that mean system time of using round-robin is the same as that of using FCFS under the assumption of negligible job switching overhead. Therefore, if job switching overhead cannot be ignored, round-robin will be inferior to FCFS for \( M/M/1 \) queues. Some work has focused on quantifying the influence of overhead. For example, a constant overhead per time slice was assumed in [11, 12] for studying several time-sharing algorithms.

While closed-form solutions have been obtained for round-robin in the context of \( M/M/1 \) queues, no such closed-form results have been derived in the context of \( M/G/1 \) queues. Although there are a couple results on applying Laplace-Stieltjes transforms (LST) to derive condition response time [13], it seems difficult to apply the results to compute mean system time.

3 KNOWN RESULTS OF PROCESSOR SHARING AND ROUND-ROBIN

What we consider is single server queues (i.e., \( M/G/1 \) queues), in which job arrivals are Poisson and job service times are represented by some distribution \( G \). On one hand, if jobs are treated in FCFS fashion, the mean system time (or response time) \( \bar{T} \) is proportional to \( C_v^2 \) according to the Pollaczek-Khinchin formula:

\[
\bar{T} = \frac{\bar{x}}{1 - \rho} + \frac{\bar{x} \rho}{1 - \rho} \frac{C_v^2 - 1}{2} \quad (1)
\]

In Eq. (1), \( \rho (0 < \rho < 1) \), the utilization parameter, is \( \lambda \bar{x} \) with \( \lambda \) being mean job arrival rate. On the other hand, if processor sharing is employed, \( \bar{T} \) is the same as that for the \( M/M/1 \) queues (i.e., Eq. (1) with \( C_v^2 = 1 \)) [14]. Clearly, if \( C_v^2 < 1 \), FCFS should be used. However, for many applications, \( C_v^2 \) is actually large (with \( C_v^2 > 100 \) being not unusual) such that \( \bar{T} \) will become quite large even under moderate loads (e.g., \( \rho \approx 0.8 \)), which means that FCFS cannot handle highly varying demands well.

In this case, processor sharing is preferred since it does not depend on \( C_v^2 \) any more.

As mentioned in introduction, processor sharing is not realistic, and in practice, round-robin has to be used. While the performance of round-robin can be analyzed using processor sharing as \( \Delta \rightarrow 0 \) or using FCFS as \( \Delta \rightarrow \infty \), the case of a finite \( \Delta \) value is more common but mostly unknown. Although it turns out that \( \Delta \) has no influence on \( M/M/1 \) queues (with the assumption of negligible job switching overhead), it is not so simple for arbitrary \( M/G/1 \) queues. Apparently, no closed-form solutions have been obtained for computing \( \bar{T} \).

4 SEVERAL ROUND-ROBIN VARIANTS

In implementing our simulator, we come up with a new question: where should we put a newly arrived job in the queue? One approach is to append it at the end of the waiting queue [4, 5]. The second approach is to insert it at the front of the queue. This is equivalent to a two-level queueing discipline: the first level is LCFS with maximum service time being \( \Delta \) and the second level is round-robin with \( \Delta \). The first level holds newly arrived jobs. After getting a quantum, a job either leaves the system upon completion or joins the second level at the end of queue. In other words, newly arrived jobs are partially favored. This makes sense when the sizes of newly arrived jobs are typically small (as in the cases of \( C_v^2 > 1.0 \)), keeping in mind that, for the exponential distribution, it does not matter where to insert the new jobs due to its memoryless feature. Another approach is to insert it randomly in the queue. We found that the results are quite different for different approaches.

5 EXPERIMENTAL RESULTS

In this section, we study the performance of the above round-robin variants as a function of \( \Delta \) values for different classes of distributions. It is assumed that the job-switching overhead is negligible. We consider both service time distributions with large \( C_v^2 \) (> 1) and those with small \( C_v^2 \) (< 1). We find that the first variant is either the worst (if \( C_v^2 > 1 \)) or the best (if \( C_v^2 < 1 \)) among the three variants as \( \Delta \) increases, while the reverse is true for the second round-robin variant. The second variant is also compared to SRPT, an optimal strategy if the service times of jobs are known upon arrival. It is important to point out that we are not claiming the use of round-robin in the case of \( C_v^2 < 1 \). Clearly, FCFS is much more effective. Our purpose is to illustrate the characteristics of these round-robin variants in different settings.

Four classical service time distributions, Erlangian
more, although the three non-exponential distributions have more intermediate-size jobs than uniform. As a result (and shown in Figure 1.(a)), the descending order of reliability function values is \(H_2\), \(H_5\), \(H_E\) for moderately short jobs. In the intermediate range, the three distributions cross each other and the order is reversed to \(H_E\), \(H_2\), and \(PT_5\). Finally, as shown in Figure 1.(b), they cross again for large \(x\) values and the order is \(PT_5\), \(H_2\), \(H_E\). As will become clear, the final order seems to be the main factor of affecting the performance of round-robin.

Figure 2 shows the reliability functions of two distributions with \(C_v^2 = 1/3\), i.e., Erlangian-3 (\(E_3\)), and uniform. The exponential reliability function is plotted as a reference. It can be seen that Erlangian-3 and uniform cross twice. Since the density function of \(E_3\) takes zero for service time \(x = 0\), \(E_3\) has less number of very short jobs than uniform does. Hence, the initial descending order of reliability function values is \(E_3\), uniform. In the intermediate range, the order becomes uniform, \(E_3\) in that \(E_3\) has more intermediate-size jobs than uniform. As the maximum service time of uniform is 2.0, its reliability function becomes zero at service time being 2.0. There is no such restriction for \(E_3\) such that a few jobs can be long. In comparison to \(E_3\) and uniform, exponential has more short jobs as well as long jobs.

In the following, we first show how the round-robin variants perform for distributions with large \(C_v^2\). A finite set of \(\Delta\) values across a large range are selected to show the behavior of the round-robin variants near origin (i.e., for relatively small \(\Delta\) values) as well as their trends with respect to the increase of \(\Delta\). For each chosen \(\Delta\) and a given distribution, ten simulation runs are conducted and the average of the ten mean system time values is computed. In each simulation run, two million job samples are drawn from the given service time distribution. All the curves are drawn pairwise linearly. Figure 3 shows the results of simulating the first variant for three distributions with \(C_v^2 = 10.0\). We plotted \((1 - \rho)T\) instead of just \(T\) for the purpose of comparing with \(M/M/1\), which of course is the same as pure processor sharing. Note that \((1 - \rho)T\) is really the ratio of mean system time for a round-robin strategy to that for \(M/M/1\). It is reasonably concluded that the performance of the first variant of round-robin is monotonically decreasing. For small \(\Delta\) values, its performance is reflected by the processor sharing model. As \(\Delta\) increases, \((1 - \rho)T\) increases gradually and approaches to that of

\[
\begin{align*}
(C_v^2 < 1), \text{ uniform } (C_v^2 < 1), \text{ hyper-exponential } (C_v^2 > 1), \text{ and hyper-Erlangian } (C_v^2 > 1), \text{ are chosen because of their simplicity. In addition, a truncated-power-tail } (TPT) \text{ distribution } (C_v^2 > 1) \text{ is included as many practical applications have the job service times to be power tails [1]. We assume that the mean service time of each distribution is 1.0 without loss of generality. We consider Erlangian-3, which has } C_v^2 = 1/3, \text{ and set the uniform distribution to be in the range from 0 to 2, which also has } C_v^2 = 1/3. \text{ For the three classes of distributions with large variance, we set their } C_v^2 \text{ values to be 10.0 such that we can examine how different classes of distributions affect the performance of round-robin.}
\]

Figure 1 shows the reliability functions of three distributions with \(C_v^2 = 10.0\), i.e., hyper-exponential-2 (\(H_2\)), hyper-Erlangian-2 (\(H_E\)), and five-phase TPT (\(PT_5\)). As a comparison, the exponential reliability function is plotted as well. It can be seen that the distributions with large \(C_v^2\) all cross the exponential distribution (not necessarily at the same place). Specifically, there are not only less number of short jobs, but also less number of long jobs for the exponential case, while for the other cases, it is more likely to have short and long jobs. Furthermore, although the three non-exponential distributions have the same value of \(C_v^2\), their reliability functions are different. Since the density function of \(H_E\) takes zero for service time \(x = 0\), it is less likely to have very short jobs for \(H_E\) than for \(H_2\) and \(PT_5\). As a result (and shown in Figure 1.(a)), the descending order of reliability function values is \(H_E\), \(PT_5\), \(H_2\) initially (not including service time \(x = 0\)). Then as \(x\) increases slightly, \(H_E\) crosses \(H_2\) and \(PT_5\) such that the order becomes \(PT_5\), \(H_2\), \(H_E\) for moderately short jobs. In the intermediate range, the three distributions cross each other and the order is reversed to \(H_E\), \(H_2\), and \(PT_5\). Finally, as shown in Figure 1.(b), they cross again for large \(x\) values and the order is \(PT_5\), \(H_2\), \(H_E\). As will become clear, the final order seems to be the main factor of affecting the performance of round-robin.

Figure 2 shows the reliability functions of two distributions with \(C_v^2 = 1/3\), i.e., Erlangian-3 (\(E_3\)), and uniform. The exponential reliability function is plotted as a reference. It can be seen that Erlangian-3 and uniform cross twice. Since the density function of \(E_3\) takes zero for service time \(x = 0\), \(E_3\) has less number of very short jobs than uniform does. Hence, the initial descending order of reliability function values is \(E_3\), uniform. In the intermediate range, the order becomes uniform, \(E_3\) in that \(E_3\) has more intermediate-size jobs than uniform. As the maximum service time of uniform is 2.0, its reliability function becomes zero at service time being 2.0. There is no such restriction for \(E_3\) such that a few jobs can be long. In comparison to \(E_3\) and uniform, exponential has more short jobs as well as long jobs.

In the following, we first show how the round-robin variants perform for distributions with large \(C_v^2\). A finite set of \(\Delta\) values across a large range are selected to show the behavior of the round-robin variants near origin (i.e., for relatively small \(\Delta\) values) as well as their trends with respect to the increase of \(\Delta\). For each chosen \(\Delta\) and a given distribution, ten simulation runs are conducted and the average of the ten mean system time values is computed. In each simulation run, two million job samples are drawn from the given service time distribution. All the curves are drawn pairwise linearly. Figure 3 shows the results of simulating the first variant for three distributions with \(C_v^2 = 10.0\). We plotted \((1 - \rho)T\) instead of just \(T\) for the purpose of comparing with \(M/M/1\), which of course is the same as pure processor sharing. Note that \((1 - \rho)T\) is really the ratio of mean system time for a round-robin strategy to that for \(M/M/1\). It is reasonably concluded that the performance of the first variant of round-robin is monotonically decreasing. For small \(\Delta\) values, its performance is reflected by the processor sharing model. As \(\Delta\) increases, \((1 - \rho)T\) increases gradually and approaches to that of

\[
\begin{align*}
\text{Figure 1: Illustration of the reliability functions of three distributions with } C_v^2 = 10.0. \text{ The graphs in (b) are plotted in semi-log scale on y-axis to show the tail behavior.}
\end{align*}
\]
other at least twice and the final order is mentioned above, their reliability functions cross each other at least twice and the final order is \( PT_5 \), \( H_2 \), \( HE_2 \). If the service time distribution is \( PT_5 \), the improvement is more significant. As mentioned above, their reliability functions cross each other at least twice and the final order is \( PT_5 \), \( H_2 \), \( HE_2 \). In other words, \( PT_5 \) results in more long jobs than \( H_2 \), which again results in more long jobs than \( HE_2 \). Therefore, a large \( \Delta \) is more helpful in the case of \( PT_5 \) than in the case of \( H_2 \) as well as in the case of \( HE_2 \).

Unlike the first variant, the second variant of round-robin favors newly arrived jobs. While it is a small change, the second variant turns out to outperform the first one considerably. As shown in Figure 4, the performance of the second variant degrades more slowly with the increase of \( \Delta \) than that of the first variant. Using a large \( \Delta \) (e.g., 40) may still help greatly in comparison to \( FCFS \) (or \( LCFS \)). Furthermore, the performance of the second variant is not monotonically decreasing any more. On the contrary, as illustrated in Figure 5, there actually exist some \( \Delta \) values that outperform the case of \( \Delta \) being zero. Our explanation is that a \( \Delta \) value of the order of 1 allows short (and therefore comparable size) jobs to be \( LCFS \), but allows processor sharing for mixtures of lots of short jobs with 1 or 2 long ones. Notice that the system does not know which is which until they finish. While the improvement is small under light load conditions (as shown in Figure 5.(a,b)), it can be significant under moderately-heavy load conditions (as shown in Figure 5.(c,d)). For example, a 27% improvement in mean system time over pure processor sharing can be expected for \( PT_5 \) by judiciously choosing \( \Delta \), which is in the magnitude of seconds. This is important in practice as well. Specifically, we do not need to use a small \( \Delta \), which is preferred if the first variant is used. Since a small \( \Delta \) typically leads to non-negligible job switching overhead, the performance of the first variant is even worse in practice, while the second variant has no such drawback. For the same reason as mentioned above, i.e., \( PT_5 \) is more likely to have long jobs than \( H_2 \) and \( HE_2 \), a larger improvement is made for \( PT_5 \) than for \( H_2 \) and \( HE_2 \). While it is not shown here, the performance of the third round-robin variant is between that of the first two round-robin variants. This is expected since the performance of each variant is proportional to the degree of favoring the newly arrived jobs.

Next, we examine how the round-robin variants perform for distributions with small \( C_2 \). Figures 6 and 7 show the results of using the first two variants as a func-

\[
\begin{align*}
\text{PDF} & \quad \text{CDF} \\
\text{Exp} & \quad \text{Uniform} \\
\text{Normal} & \quad \text{Erlang} \\
\end{align*}
\]

\[
R(x) = \frac{\exp(-x)}{x^2} \\
R(2x) = \frac{\exp(-2x)}{2x^2} \\
R(3x) = \frac{\exp(-3x)}{3x^2} \\
\]

\[
\begin{align*}
\text{Figure 2: Illustration of the reliability functions of two distributions with } C_2 = 1/3.
\end{align*}
\]

\[
\begin{align*}
\text{Figure 3: The performance of the first round-robin variant as a function of } \Delta \text{ for three distributions with } C_2 = 10.0: (a) } \rho = 0.3; (b) \rho = 0.5; (c) \rho = 0.7; (d) \rho = 0.9. \text{ Here, } \rho \text{ and } \bar{T} \text{ are utilization parameter and mean system time, respectively. Note that } (1 - \rho)\bar{T} \text{ is really the ratio of mean system time for the round-robin strategy to that for } M/M/1 \text{ (or pure processor sharing). The graphs are plotted in semi-log scale on } x \text{-axis to show how the strategy performs near } \Delta = 1 \text{ as well as gets to its asymptotic value.}
\end{align*}
\]
Figure 4: The performance of the second round-robin variant as a function of $\Delta$ for three distributions with $C_2 = 1$ and $\rho = 0.9$. The graphs are plotted in semi-log scale on the x-axis to show the behavior near $\Delta = 1$, and how long it takes to get to its asymptotic value for large $\Delta$. It can be seen that this variant performs better than pure processor sharing for some values of $\Delta$.

Figure 5: The behavior of the second round-robin variant for time slice values less than five times the mean service time (i.e., $0 < \Delta < 5$) for three distributions with $C_2 = 10$. (a) $\rho = 0.3$; (b) $\rho = 0.5$; (c) $\rho = 0.7$; (d) $\rho = 0.9$. All the graphs are plotted in semi-log scale on the x-axis to show the behavior near $\Delta = 1$, and how long it takes to get to its asymptotic value for large $\Delta$. It can be seen that this variant performs better than pure processor sharing for some values of $\Delta$.

It is important to note that, by judiciously choosing $\Delta$, one can get considerable savings in mean system time over pure processor sharing.
tion. On the other hand, SRPT becomes more effective for the three distributions with \( C_v^2 = \frac{1}{3} \). Since \( PT_5 \) is more likely to have long jobs than \( H_2 \) and \( HE_2 \), SRPT has the best performance for \( PT_5 \). It is important to note that the performance of SRPT may not be achievable as the service times of jobs are typically unknown before completion. However, by using SRPT as a reference, we may still be able to examine how well a queuing discipline without requiring service times of individual jobs performs. Figure 8.(b) compares the performance of the second round-robin variant to that of SRPT for the three distributions with \( C_v^2 = \frac{1}{3} \). Since the performance of SRPT may not be achievable in practice, it is more important to find such variants.

6 CONCLUSIONS

In the view that round-robin has not been well studied as a function of time slice values for different classes of distributions, we conduct a simulations study of several round-robin variants. It is demonstrated that the variants have quite different performance. In particular, for highly varying job demands (i.e., \( C_v^2 > 1 \)), the variant of favoring newly arrived jobs outperforms others consistently. Furthermore, one can get considerable improvement in mean system time over pure processor sharing by judiciously choosing time slice value. Since the optimum time slice value is large in comparison to mean service time, such a variant can be used in practice without incurring much job switching overhead.

In this paper, all the results are based on extensive simulation. As a future work, we would like to come up with some empirical formulas that capture the performance of the round-robin variants as a function of time.
Figure 8: (a): The performance of SRPT as a function of $\rho$ values for different distributions; (b): Comparison of SRPT with the second round-robin variant.

slice values. Such formulas could be used to choose an appropriate time slice value for practical use. Another future work is to investigate the performance of other round-robin variants.

REFERENCES


