QUANTILE ESTIMATION FOR QUEUEING SYSTEMS

STAGE II

Boyu Yang
The Chinese University of Hong Kong, Shenzhen
boyuyang@link.cuhk.edu.cn

May 23, 2022

ABSTRACT

The performance of 5G network is highly sensitive to delays. The design of 5G network requires more accurate evaluation on delay distribution. This research aims to accurately measure expected steady-state waiting time and percentile delays of the queueing networks with routing schedules of different levels of complexity. Specifically, non-preemptive priority queue, weighted fair queueing system, and deficit round robin will be simulated using adaptive cross-entropy and importance sampling techniques. Numerical experiments show that our algorithm outperforms benchmark simulation methods. The results mainly contribute to the literature on rare event simulation for queueing systems.

Keywords Communication Network · Regenerative Method · Adaptive Cross-Entropy · Non-preemptive Priority Simulator · Weighted Fair Queueing · Deficit Round Robin · Stochastic Simulation

1 Introduction

The performance of 5G network is highly sensitive to delays. The design of 5G network requires more accurate evaluation on delay distribution. The project aims to develop fast simulation algorithms to evaluate tail probability and percentile of delay, which are known as service level agreement (SLA) in communication networks. The simulator is constructed using queueing theory and algorithm is designed using importance sampling techniques.

In the following simulation, random seed is set to be 43 to ensure the results are reproduceable.

2 Notation

The notations that will be used is summarized in the table.

Attribute	Description
W_n	Waiting time of the nth customer
V_n	Service time of the nth customer
T_n	Interarrival time of the nth customer
σ	Cycle length of the regenerative cycle
τ	The first time that W_n exceeds a threshold γ

Table 1: A table of notations description

3 M/M/1 First-in-First-out Queue

For the simplest M/M/1 queue, the expected waiting time and tail probability have analytical solutions. With independent poisson arrival (rate λ) and exponential service time (rate μ) assumption, the expected waiting time is given by $\mathrm{E}[W_\infty] = \rho/(\mu - \lambda)$ and the tail probability is given by $P(W_\infty > \gamma) = \rho \times e^{-(\mu-\lambda)\gamma}$. We consider a single case of $\lambda = 1.5, \mu = 2$ for the first two methods and two cases of light and heavy traffic for the change of measure method.

3.1 Direct Simulation by Lindley Recursion

Consider a single-server queue possessing an infinite capacity waiting room and processing customers according to "first-in-first-out" routing schedule. Let W_n, V_n, T_n be the waiting time, service time, and interarrival time for the nth customer to enter the queue. By the *Lindley Recursion* formula [1], we have

$$W_{n+1} = [W_n + V_n - T_n]^+ \tag{1}$$

In an M/M/1 queue, T_n, V_n follow i.i.d exponential distributions with rate λ and μ respectively. Hence we can directly simulate the dynamics of the M/M/1 queue using random number generator in Python. The estimation for the waiting time expectation is calculated by Crude Monte Carlo (CMC). The 95% normal confidence interval is given by $\hat{\mathrm{E}}[W_n] \pm 1.96 \times \sigma_{\hat{\mathrm{E}}[W_n]}/n^{1/2}$. We get the simulation mean and 95% confidence intervals for $E[W_n]$ with $n = 1, 10, 100, 1000, 10000$ where $\lambda = 1.5$ and $\mu = 2$ in table 2.

n	$\hat{\mathrm{E}}[W_n]$	95% CI
1	0.000000	[0.000000, 0.000000]
10	0.854324	[0.844877, 0.863772]
100	1.483175	[1.466490, 1.499860]
1000	1.504443	[1.487524, 1.521363]
10000	1.509648	[1.492566, 1.526730]

Table 2: Simulation by Lindley recursion

Since the theoretical result of the steady state mean waiting time is $\rho/(\mu - \lambda)$, in which case is 1.5 for the accurate value, we can see from the simulation result that when the system is runned for a long time, the expected mean waiting time is becoming closer to the true value.

We generate 50000 sample paths for each case to ensure the width of confidence intervals to be smaller than 0.05. For all the cases above, the 95% confidence intervals with width smaller than 0.05 cover the true value of 1.5. However, for other random seeds tried, the result and the confidence intervals are fluctuating, sometimes cover the true value but sometimes may not. This indicates simulating the steady-state distribution simply by running the system for a long time may be inaccurate since the relative error is large. Also the simulation takes plenty of time in order to stabilize the performance of CMC method especially when n is large.

3.2 Simulation Using Regenerative Method

A general M/G/1 queue possesses the behavior that at each time point when a customer arrives at an empty system, the process probabilistically restart itself [5]. Let σ_i be the i.i.d length of the ith cycle of the regenerative process X_t.

$$\sigma = \inf\,[n > 1 | W_n = 0] - 1 \tag{2}$$

Suppose we want to estimate the steady-state expectation $E[W(X)]$, the regenerative method gives a way to approximate using the following formula.

$$R_i = \sum_{t=T_{i-1}}^{T_i - 1} W(X_t) \tag{3}$$

$$l = \mathrm{E}[W(X)] = \frac{\mathrm{E}[R]}{\mathrm{E}[\sigma]} \tag{4}$$

Then we can generate N regenerative cycles, calculate the i.i.d sequence of two-dimensional random vectors (R_i, σ_i), $i = 1, 2, \ldots, N$, and then use CMC to obtain a point estimate:

$$\hat{l} = \frac{\hat{R}}{\hat{\sigma}} \tag{5}$$

where $\hat{R} = N^{-1}\sum_{i=1}^{N} R_i$ and $\hat{\sigma} = N^{-1}\sum_{i=1}^{N} \sigma_i$. The regenerative estimator is biased, namely, $E[\hat{l}] \neq l$. However $\hat{l}$ is strongly consistent by applying the law of large numbers to the numerator and denominator respectively. A 95% confidence interval for the point estimator is given by

$$(\hat{l} \pm 1.96 \times \frac{S}{\hat{\sigma}N^{1/2}}), \tag{6}$$

where

$$S^2 = \frac{1}{N-1}\sum_{i=1}^{N}(R_i - \hat{R})^2 - 2 \times \hat{l} \times \frac{1}{N-1}\sum_{i=1}^{N}(R_i - \hat{R})(\sigma_i - \hat{\sigma}) + \hat{l}^2 \times \frac{1}{N-1}\sum_{i=1}^{N}(\sigma_i - \hat{\sigma})^2 \tag{7}$$

Using the above formula, we can design the simulation algorithm to get the estimation result for the expected waiting time in an M/M/1 queue in table 3.

N	$\hat{\mathrm{E}}[W_n]$	95% CI	Theory
1000000	1.498611	[1.485029, 1.512194]	1.50

Table 3: Simulation by regenerative method

We use a large number of cycles to ensure the consistency property of the regenerative method estimator is utilized to the utmost. The result tends to be much more stable and accurate than the previous. The 95% confidence interval covers the true value 1.5 and the estimated expectation is very close to the true value. Though the result is much more accurate, we still need to simulate for a large number of cycles to utilize the consistency property. If the cycle number generated is small (e.g. less than 10000), the estimation is not stable as well and the relative error is still very large.

3.3 Simulation Using Change of Measure

Substituting the estimation target of expected steady-state waiting time by the steady-state tail probability, namely, $P(W_\infty > \gamma)$, where γ is a threshold that is much larger than the expected value $E[W_n]$. This tail probability has practical intuitions in the 5G network. The 5G communication network requires high efficiency to deal with tasks of different packet sizes. If the waiting time of a single packet exceeds a threshold, the whole communication network may become jammed, which may cause devastating effects to the whole system. Hence we need more fast and low error estimation methods to accurately measure the tail probability.

By the previous regenerative method, the tail probability satisfies

$$P(W_\infty > \gamma) = \frac{\mathrm{E}[\sum_{n=1}^{\sigma} I(W_n \geq \gamma)]}{\mathrm{E}[\sigma]} \tag{8}$$

Hence the tail probability can be estimated directly using direct CMC method. However, due to the large relative error, we seek to improve the accuracy and lower down the estimation variance using importance sampling [3].

3.3.1 Importance Sampling

Now we present the method of importance sampling in M/G/1 network [4]. Denote the first time that W_n exceeds level γ to be τ, then

$$\tau = \inf\,[n > 1, W_n \geq \gamma] \tag{9}$$

Consider the following switching change of measure

$$T_n \sim f^T \rightarrow \tilde{f}^T \quad \text{and} \quad V_n \sim g^V \rightarrow \tilde{g}^V, \quad \text{for } n = 1, 2, \ldots min(\tau, \sigma). \tag{10}$$

Initially we use the IS densities for V_n and T_n until the process W_n exceeds level γ, after which we switch back to the original densities. By switching like this the whole process returns to the regenerative state. The likelihood ratio satisfies

$$L_n = \begin{cases} L_{n-1}\frac{f^T(T_n)g^V(V_n)}{\tilde{f}^T(T_n)\tilde{g}^V(V_n)} & n \leq min(\tau, \sigma) \\ L_\tau & n \geq min(\tau, \sigma) \end{cases} \tag{11}$$

Therefore we can obtain the new formula for estimating the tail probability

$$\ell = P(W_\infty > \gamma) = \frac{\mathrm{E_w}[\sum_{n=1}^{\sigma} I(W_n \geq \gamma)]}{\mathrm{E}[\sigma]} = \frac{\mathrm{E_{w'}}[\sum_{n=1}^{\sigma} I(W_n \geq \gamma) L_n]}{\mathrm{E}[\sigma]} \tag{12}$$

where the notation w' represents the switched distribution under change of measure.

3.3.2 M/M/1 Queue Results

In M/M/1 queue, we switch the rate of interarrival time from λ to $\lambda + \theta$ and the rate of service time from μ to $\mu - \theta$. The optimal parameter value occurs when $\theta^* = \mu - \lambda$. We try both the optimal θ and other values of θ that are near the optimal value to obtain more insights.

By using Monte Carlo estimation separately for numerator and denominator, we obtain the estimation results as the following table. It should be noted that the 95% confidence interval for the numerator is the usual normal confidence interval, and the 95% confidence interval for the whole probability is calculated using the same formula as equation (6). The relative error is calculated using σ_{IS}/l where l is the estimation expectation. The number of cycles is fixed to be 10^5 and the time for simulation is measured.

θ	$\frac{\lambda}{\lambda+\theta}$ $\frac{\mu}{\mu-\theta}$	γ	Theory	Num Mean	Num 95% CI	Num_RE	Prob Mean	Prob 95% CI	Prob_RE	Cycle Num	Time
1.6348	0.9	3.2273	1e-3	0.001498	[0.001481, 0.001516]	0.00587	0.001002	[0.000989, 0.001015]	0.00667	1e5	9.6s
1.8	1	3.2273	1e-3	0.001495	[0.001478, 0.001510]	0.005498	0.00099967	[0.000987, 0.001012]	0.006355	1e5	9.7s
1.91534	1.1	3.2273	1e-3	0.001496	[0.001478, 0.001513]	0.005899	0.00099945	[0.000986, 0.001012]	0.006701	1e5	9.7s
1.6348	0.9	5.78573	1e-5	1.5076e-05	[1.4893e-05, 1.5260e-05]	0.006210	1.00413e-05	[9.9039e-06, 1.0179e-05]	0.006979	1e5	14.9s
1.8	1	5.78573	1e-5	1.5066e-05	[1.4905e-05, 1.5227e-05]	0.005440	1.00067e-05	[9.8831e-06, 1.0130e-05]	0.006306	1e5	14.8s
1.91534	1.1	5.78573	1e-5	1.5015e-05	[1.4835e-05, 1.5195e-05]	0.006112	9.9763e-06	[9.8418e-06, 1.0111e-05]	0.006878	1e5	14.5s
1.6348	0.9	12.1818	1e-10	1.4980e-10	[1.4767e-10, 1.5193e-10]	0.007255	9.95643e-11	[9.8020e-11, 1.0111e-10]	0.007913	1e5	28.7s
1.8	1	12.1818	1e-10	1.4821e-10	[1.4663e-10, 1.4979e-10]	0.005448	9.85027e-11	[9.7288e-11, 9.9718e-11]	0.006293	1e5	27.8s
1.91534	1.1	12.1818	1e-10	1.5049e-10	[1.4844e-10, 1.5255e-10]	0.006999	1.00299e-10	[9.8788e-11, 1.0181e-10]	0.007683	1e5	26.1s

Table 4: Simulation by Change of Measure: Light Traffic ($\lambda = 0.9, \mu = 2.7$)

θ	$\frac{\lambda}{\lambda+\theta}$ $\frac{\mu}{\mu-\theta}$	γ	Theory	Num Mean	Num 95% CI	Num_RE	Prob Mean	Prob 95% CI	Prob_RE	Cycle Num	Time
0.28761	0.99	16.7115	1e-3	0.00493	[0.0047, 0.00510]	0.020451	0.00099	[0.00099, 0.00103]	0.020676	1e5	48.4s
0.4	1	16.7115	1e-3	0.00503	[0.0049, 0.00515]	0.012286	0.0009975	[0.00097, 0.001022]	0.012635	1e5	59.7s
0.7752	1.1	16.7115	1e-3	0.00396	[0.00326, 0.00465]	0.089492	0.000795	[0.00065, 0.00093]	0.089551	1e5	1m25.3s
0.28761	0.99	28.2244	1e-5	5.00243e-05	[4.7239e-05, 5.2809e-05]	0.028402	9.96397e-06	[9.4056e-06, 1.0522e-05]	0.028587	1e5	1m18.2s
0.4	1	28.2244	1e-5	4.96383e-05	[4.8433e-05, 5.0843e-05]	0.012381	9.92358e-06	[9.6746e-06, 1.0172e-05]	0.012796	1e5	1m39.4s
0.7752	1.1	28.2244	1e-5	3.50894e-05	[1.32093e-05, 5.6969e-05]	0.318137	7.02506e-06	[2.64424e-06, 1.1406e-05]	0.318162	1e5	2m17.7s
0.28761	0.99	57.0067	1e-10	4.6518e-10	[4.2200e-10, 5.0836e-10]	0.047354	9.3716e-11	[8.49982e-11, 1.024e-10]	0.047462	1e5	2m55.4s
0.4	1	57.0067	1e-10	5.0849e-10	[4.9623e-10, 5.2074e-10]	0.012294	1.02568e-10	[1.0001e-10, 1.0512e-10]	0.012703	1e5	3m44.9s
0.7752	1.1	57.0067	1e-10	2.65308e-10	[8.6503e-12, 5.2190e-10]	0.493568	5.31031e-11	[1.7280e-12, 1.0447e-10]	0.493593	1e5	5m39.5s

Table 5: Simulation by Change of Measure: Heavy Traffic ($\lambda = 1.6, \mu = 2$)

It can be easily seen that in most cases, the 95% confidence interval covers the theoretical value, and the estimated mean is close to the true value. However, in the heavy traffic cases, some choices of θ fail to give an accurate estimate and their confidence intervals do not cover the true value. We leave it in the discussion below.

3.4 Conclusion

In the above sections, different simulation techniques have been used to simulate the expected waiting time and tail probability of an M/M/1 queue. We summarize some important insights below.

Lindley Recursion Lindley recursion gives a direct recursive way to simulate the waiting time sequence. However, directly running simulations to obtain point estimates for $\mathrm{E}[W_n]$ for a large value of n to approximate the steady-state performance is inaccurate. The results and confidence intervals fluctuate with different random seeds and are not stable due to a large relative error. It also takes plenty of time to generate i.i.d samples of W_n in order to use the Monte Carlo method.

Regenerative Method Regenerative method obtains a consistent estimator for the evaluated expectation. When the number of cycles generated is large, the consistency principle can be utilized and the estimated waiting time expectation is very close to the actual value. The simulation takes much less time to finish the estimation, but the high relative error problem still exists. This means if the number of cycles generated is relatively small, the obtained point estimate will have large variance and may be inaccurate.

Change of Measure By using proper change of measure, the relative error is reduced effectively. However, the performance among different changes of measure differ a lot. We discuss the results by comparing the performance in both the light traffic case and heavy traffic case below:

- In most of the cases, using change of measure presents better point estimate and confidence intervals. As shown in the table, the mean of probability estimate is close to the actual value and the 95% confidence interval covers the theoretical value in most cases.
- Different changes of measure obtain similar result of the numerator as well as the estimated probability. Indeed the numerator estimation should be unbiased for different changes of measure.
- When the threshold γ increases, the probability becomes harder to estimate and the algorithm takes longer time to finish. This makes sense since we will switch back to the original densities only when the "rare event" occurs. When γ becomes larger, the "rare event" occurs with much lower probability.
- The best change of measure always give the lowest relative error. Taking values of θ near the optimal value will all increase the relative error. This means even if in the current case (random seed is 43) the algorithm performs better than the optimal change of measure, it may underperform if the random seed is changed.
- Improper changes of measure hurt the performance of the algorithm, even worse than the simulation without change of measure. As shown in the table 5, in the heavy traffic case, when the second column is 1.1, the value of θ actually differ from the optimal parameter (0.4) a lot. It can be easily seen that the corresponding cases ($\theta = 0.7752$) have extremely large relative errors, even hundred times larger than the optimal cases. The estimated value is biased, and the confidence interval fails to cover the true probability.
- The heavy traffic case is far more difficult to estimate than the light traffic case. In the algorithm above, simulation for the heavy traffic cases take several times longer than the light traffic cases. Also the tail probability estimation for the light traffic case is more accurate than that of the heavy traffic case. The heavy traffic cases take much longer time and have larger relative error.

4 M/M/1 Non-preemptive Priority Queue

Now we study the same M/M/1 queueing system, however with different routing principle, i.e. the priority queue. We study the nonpreemptive priority queue, meaning the ongoing service cannot be terminated even if a customer with higher priority comes along. Denote the arrival rates of the k types of customers to be $\lambda_1, \lambda_2, \ldots, \lambda_k$ and the service rates are $\mu_1, \mu_2, \ldots, \mu_k$ respectively. The target again is to estimate 1) expected waiting time in steady states, i.e. E$[W_\infty^{(k)}]$, and 2) tail probability $P(W_\infty^{(k)} > \gamma)$ for type k customers.

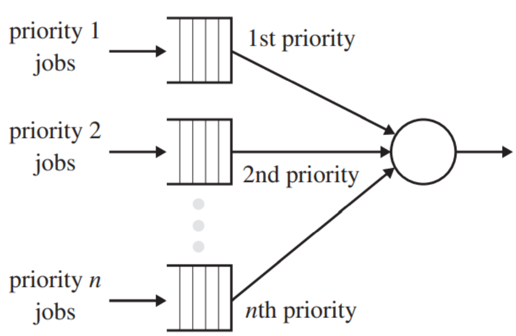


Figure 1: M/M/1 priority queue illustration

We use event-based simulation to construct the algorithm. In the first step direct simulation will be constructed to simulate finite-step waiting time expectation $\mathrm{E}[W_n^{(k)}]$, then regenerative method and possible changes of measure will be used to estimate the tail probability. It is worth mentioning that in the non-preemptive priority queue case, the expected waiting time has analytical solutions though the best change of measure does not.

4.1 Event-Based Simulation

Consider discrete event systems with finitely many components and each component has only finite number of states. Each event takes place at a particular time but has no duration. The system dynamics (or called environment) can be modeled as combinations of upcoming events, with their corresponding time and action. The components of the system interact through different types of events.

In the nonpreemptive priority queueing system, the components are coming customers and the server. Customers arrive, keep line in a waiting queue, get served, and leave after finishing their services. Hence, the components interact with the environment through three types of events including "arrival", "starting service", and "departure". If a customer arrives, the components will add an event of type "arrival" to the environment. The environment then checks whether the server is free to determine whether to send the instructions of starting service to the server. After starting service, the server state will be set busy and a "departure" type event will be sent to the environment after the service time. After the departure of the current customer, the server state is set free again to serve the next potential customer.

4.2 Direct Simulation of Expected Waiting Time

4.2.1 Theoretical Results

We first present the analytical solutions of the steady-state waiting time [2]. Assume customers of the first type have the highest priority. Let V_k be the service time of priority k customers. Assume the service utilization rate is smaller than 1, that is

$$\rho = \sum_{i=1}^{n} \rho_i = \sum_{i=1}^{n} \lambda_i \mathrm{E}[T_i] = \sum_{i=1}^{n} \frac{\lambda_i}{\mu_i} < 1 \tag{13}$$

Use $\mathrm{E}[V_e]$ to represent the expected remaining time of the currently served customer given that there is one customer in service. Then $\mathrm{E}[V_e]$ is given by

$$\mathrm{E}[V_e] = \frac{\mathrm{E}[V^2]}{2\mathrm{E}[V]} = \frac{\sum_{i=1}^{n} p_k \mathrm{E}[V_k^2]}{2\sum_{i=1}^{n} p_k \mathrm{E}[V_k]} = \frac{\sum_{i=1}^{n} \frac{\lambda_k}{\sum_i \lambda_i} \times \frac{2}{\mu_k^2}}{2\sum_{i=1}^{n} \frac{\lambda_k}{\sum_i \lambda_i} \times \frac{1}{\mu_k}} \tag{14}$$

Then the expected steady-state waiting time for type-k customers is

$$\mathrm{E}[W_\infty^{(k)}] = \frac{\rho \mathrm{E}[V_e]}{(1 - \sum_{i=1}^{k} \rho_i)(1 - \sum_{i=1}^{k-1} \rho_i)} \tag{15}$$

4.2.2 Simulation Results

We start the simulation of the two-class case, where $\lambda_1 = 0.6, \lambda_2 = 0.2$ and $\mu_1 = 2, \mu_2 = 1$. Then by the previous formula, the theoretical results will be

$$\mathrm{E}[W_\infty^{(1)}] = \frac{\rho \mathrm{E}[V_e]}{1 - \rho_1} \tag{16}$$

$$\mathrm{E}[W_\infty^{(2)}] = \frac{\rho \mathrm{E}[V_e]}{(1 - \rho_1)(1 - \rho_1 - \rho_2)} \tag{17}$$

which is 0.5 for $\mathrm{E}[W_\infty^{(1)}]$ and 1 for $\mathrm{E}[W_\infty^{(2)}]$. The simulated mean and 95% normal confidence interval for fixed $n = 2, 10, 100, 1000, 10000$ are given in the table (we start with $n = 2$ to ensure neither of the waiting time is always 0). For every fixed n, We simulate more than 10000 distinctive paths to make sure the performance of CMC method is stable and the width of the confidence intervals is small enough.

Compared with the theoretical results (0.5 and 1), we can see the simulation results for both classes are very close to the true value. Except for the case when $n = 2$, all other scenarios have their 95% confidence intervals covered the true parameter value, though simulations with large value of n take extremely long time to finish. In later sections we will explore possible changes of measure and use regenerative method again to obtain more fast and accurate simulation algorithms.

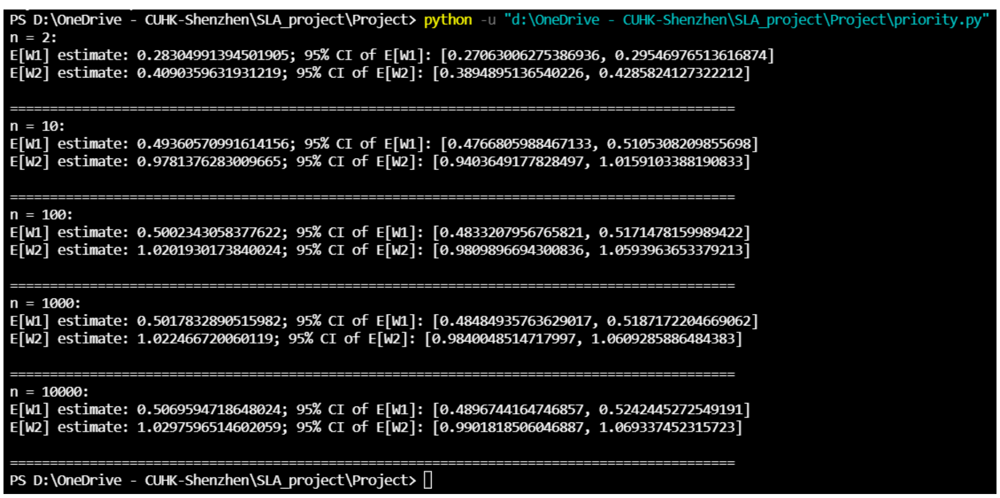
n	$\hat{\mathrm{E}}[W_n^{(1)}]$	Theory_1	95% CI of $\hat{\mathrm{E}}[W_n^{(1)}]$	$\hat{\mathrm{E}}[W_n^{(2)}]$	Theory_2	95% CI of $\hat{\mathrm{E}}[W_n^{(2)}]$
2	0.283049	0.50	[0.27063, 0.29546]	0.409035	1.00	[0.38948, 0.42858]
10	0.493606	0.50	[0.47668, 0.51053]	0.978137	1.00	[0.94036, 1.01591]
100	0.500234	0.50	[0.48332, 0.51714]	1.020193	1.00	[0.98099, 1.05939]
1000	0.501783	0.50	[0.48485, 0.51871]	1.022466	1.00	[0.98400, 1.06092]
10000	0.50695	0.50	[0.48967, 0.52424]	1.029759	1.00	[0.99018, 1.06933]

Table 6: Direct event-based simulation results

4.2.3 Some Insights

From the table, customers with higher priority have smaller expected waiting time than those with lower priority. It can be expected that if the priority queue is preemptive, then those customers with higher priority will have even smaller expected waiting time.

Besides, the numerator of the expected waiting time formula is the same, regardless of which type the customer belongs to. This term is due to waiting for the customer that is currently served, which equals the probability that there is one customer in the service (ρ) multiply by the expected remaining time on that customer given that there is a customer in the service ($\mathrm{E}[V_e]$). The denominator, however, decreases as the priority decreases. The term $(1 - \sum_{i=1}^{k} \rho_i)$ represents the contribution due to waiting for customers that arrive earlier and have equal or higher priority. Therefore the summation is to number k. The term $(1 - \sum_{i=1}^{k-1} \rho_i)$ represents the contributions due to waiting for customers that arrive later but have strictly higher priority. Therefore the summation is to number $k-1$. This two types of wait make the system different from the FIFO case discussed before.



```
PS D:\OneDrive - CUHK-Shenzhen\SLA_project\Project> python -u "d:\OneDrive - CUHK-Shenzhen\SLA_project\Project\priority.py"
n = 2:
E[W1] estimate: 0.28304991394501905; 95% CI of E[W1]: [0.27063006275386936, 0.29546976513616874]
E[W2] estimate: 0.4090359631931219; 95% CI of E[W2]: [0.3894895136540226, 0.4285824127322212]

================================================================================
n = 10:
E[W1] estimate: 0.49360570991614156; 95% CI of E[W1]: [0.4766805988467133, 0.5105308209855698]
E[W2] estimate: 0.9781376283009665; 95% CI of E[W2]: [0.9403649177828497, 1.0159103388190833]

================================================================================
n = 100:
E[W1] estimate: 0.5002343058377622; 95% CI of E[W1]: [0.4833207956765821, 0.5171478159989422]
E[W2] estimate: 1.0201930173840024; 95% CI of E[W2]: [0.9809896694300836, 1.0593963653379213]

================================================================================
n = 1000:
E[W1] estimate: 0.5017832890515982; 95% CI of E[W1]: [0.48484935763629017, 0.5187172204669062]
E[W2] estimate: 1.022466720060119; 95% CI of E[W2]: [0.9840048514717997, 1.0609285886484383]

================================================================================
n = 10000:
E[W1] estimate: 0.5069594718648024; 95% CI of E[W1]: [0.4896744164746857, 0.5242445272549191]
E[W2] estimate: 1.0297596514602059; 95% CI of E[W2]: [0.9901818506046887, 1.069337452315723]

================================================================================
PS D:\OneDrive - CUHK-Shenzhen\SLA_project\Project>
```

Figure 2: Priority queue simulation results in python

In this setting of this two customer classes, type 1 customers have a service rate of 2 while type 2 customers have a service rate of 1. This indicates that type 1 customers generally take less time to finish their service. Indeed, it can be proved that in nonpreemptive M/G/1 systems, the **shortest-job-first (SJF)** routing principle can be proved to minimize the mean waiting time over all the customers [2].

Except for the long time it takes to estimate the waiting time expectation, the nonpreemptive policy, however, is still a poor choice even under the SJF principles. This is because the numerator contains second-moment term $\mathrm{E}[V^2]$, which results in huge variance especially under heavy-tail service distributions. Even for high priority customers, the mean waiting time is suffered due to the variance in the service distribution. This can be seen that a high priority customer may still be stuck behind possible ongoing low priority customers if they already start the service. Hence the ability to preempt ongoing service is important though in real world it can scarcely be satisfied.

References

[1] Søren Asmussen and Peter W Glynn. *Stochastic simulation: algorithms and analysis*, volume 57. Springer, 2007.

[2] Mor Harchol-Balter. *Performance modeling and design of computer systems: queueing theory in action*. Cambridge University Press, 2013.

[3] Sheldon M Ross. *Introduction to probability models*. Academic press, 2014.

[4] Reuven Y Rubinstein and Dirk P Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation, and machine learning*, volume 133. Springer, 2004.

[5] Reuven Y Rubinstein and Dirk P Kroese. *Simulation and the Monte Carlo method*. John Wiley & Sons, 2016.