Simulating and Measuring the Performance of Queues

Anthony Hung

2019-03-01

Prerequisites

This vignette continues from concepts covered in the vignette: "Introduction to Queueing Theory and Queueing Models".

Introduction

In addition to simply being able to represent waiting lines mathematically, queueing theory allows for the evaluation of the behavior and performance of queues. Being able to measure the performance of queues also allows us to determine how altering certain components of the queue will impact performance.

Simulating a M/M/1 queue

In simulating a M/M/1 queue, we want to keep track of three values of the queue over time.

1. Arrival times of customers
2. Departure times of customers
3. The number of customers in the system at every moment of arrivals or departures

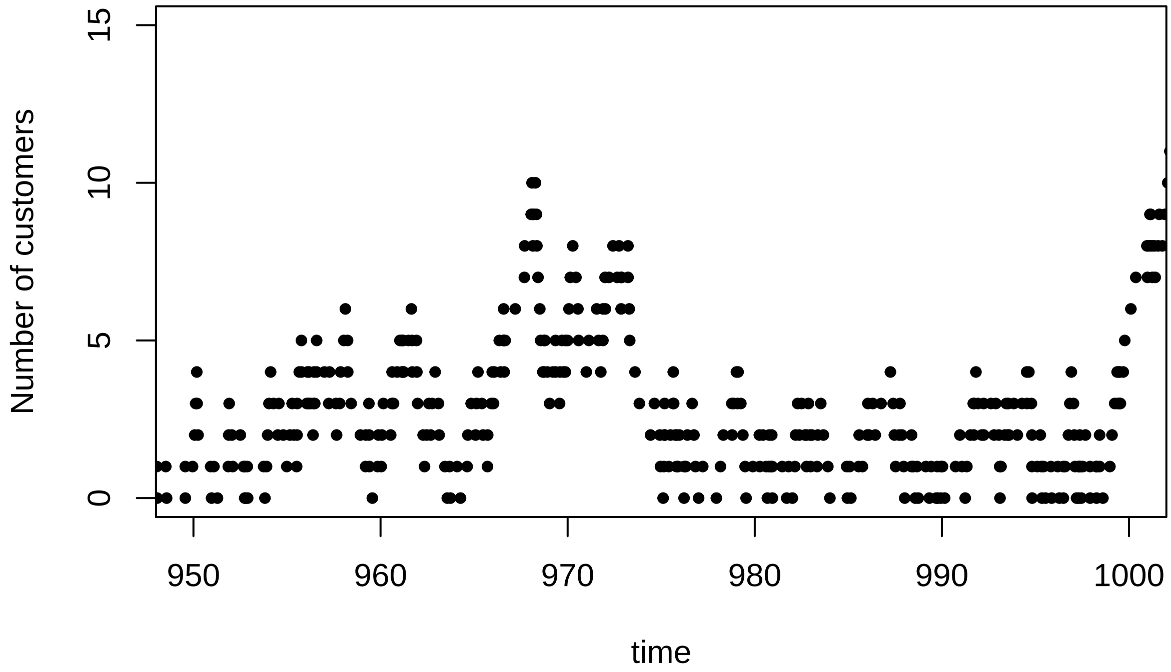
In simulating the queue behavior, we can take advantage of the superposition property of combined independent Poisson processes. Since arrivals and departures are indepdendent, the number of events in the combined process can be represented as a Poisson process with parameter $\lambda_{sum} = \lambda + \mu$. The probability of an event in this combined process being an arrival is $P_{arrival} = \frac{\lambda}{(\lambda+\mu)}$, and the probabilty of it being a departure is $P_{departure} = \frac{\mu}{(\lambda+\mu)}$.

The function "simulate_MM1" simulates the number of customers in a M/M/1 queue over time given values for lambda, mu, and N_0 from T_0 to T_{max}. It also keeps track of when events (arrivals or departures) occur during the time periods and what type of event occurs at each of those moments.

```
lambda <- 4
mu <- 5

simulate_MM1 <- function(lambda=lambda, mu=mu, N0=0, Tmax=2000){
  #Initialize vectors to store each of the values of interest throughout the simulation
  events <- 0 #stores the type of event (1 for arrival, -1 for departure)
  Times <- 0 #times of events
  customers <- N0 #number of customers at each time in Times

  while(tail(Times,1) < Tmax){ #keep simulating until you have an event \\
      #at a time greater than Tmax

    if(tail(customers,1)==0){ #separate behavior occurs if system currently has 0 customers
      tau <- rexp(1, rate=lambda) #interarrival intervals are exponentially distributed
      event <- 1 #only an arrival can occur if thre are 0 customers
    } else {
      tau <- rexp(1, rate=lambda+mu) #inter-event intervals are exponentially distributed
      if(runif(1,0,1) < lambda/(lambda+mu)){ #if runif is less than P(event = arrival)...
        event <- 1 #call the event an arrival
      } else{
        event <- -1 #otherwise, call the event a departure
      }
    }

    #now that we have simulatd one event, we need to do some accounting
    customers <- c(customers, tail(customers,1)+event)
    Times <- c(Times, tail(Times,1)+tau)
    events <- c(events, event)
  }

  #we need to toss out the information from the last event (it occured after Tmax)
  events <- head(events, -1)
  Times <- head(Times, -1)
  customers <- head(customers, -1)

  #we will also toss out the first 200 events to allow for burn-in
  events <- tail(events, -200)
  Times <- tail(Times, -200)
  customers <- tail(customers, -200)

  return(list(events,Times,customers))
}
```

After simulating the number of customers in the queue over one run of the simulation, we can plot the number of customers vs time. Here, I truncate the time axis so we can better appreciate how the number of customers in the system changes over an interval of the simulation.

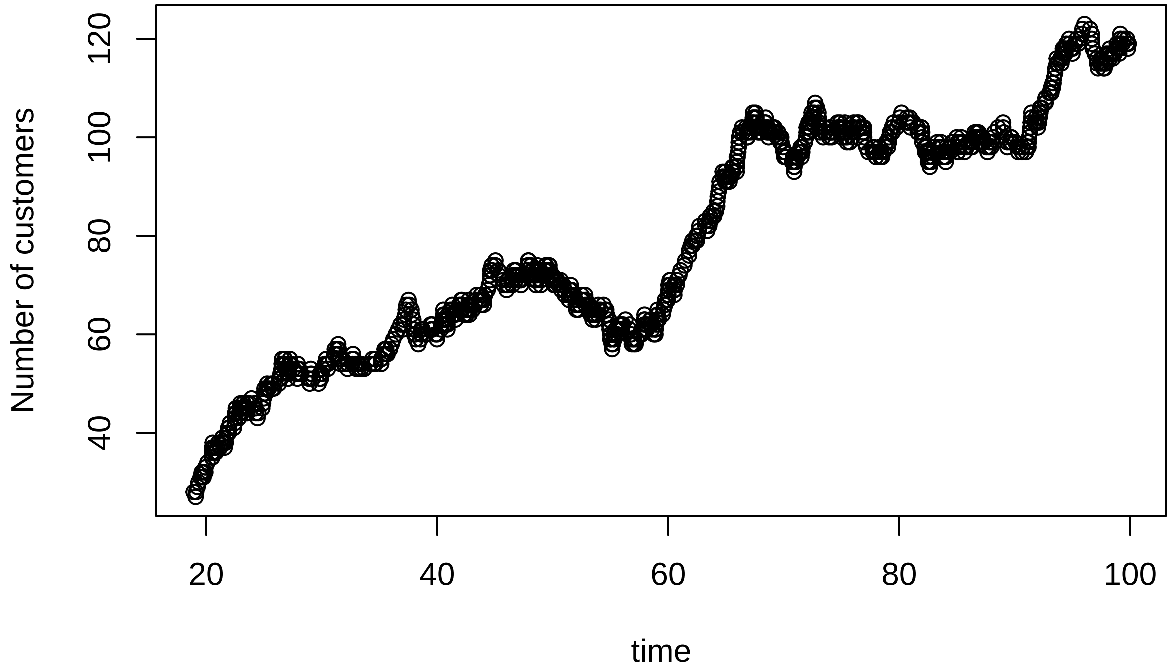
```
sim <- simulate_MM1(lambda = 4, mu=5, N0=0, Tmax=10000)
plot(x=sim[[2]], y=sim[[3]], xlim=c(950,1000), ylim=c(0,15),
     pch = 20, xlab="time", ylab="Number of customers")
```



```
#number of customers vs time
```

Notice below that if $\lambda > \mu$, the customer number explodes and will never reach a steady state.

```
sim2 <- simulate_MM1(lambda = 6, mu=5, N0=0, Tmax=100)
plot(x=sim2[[2]], y=sim2[[3]], xlab="time", ylab="Number of customers")
```



```
#number of customers vs time
```

Measuring the performance of queues

There are several formal quantities used to measure the performance of a queueing system (with c servers).

1. π_j := The stationary probability that there are j customers in the system
2. a := Offered load. The mean number of requests per service time.
3. ρ := Server utilization or traffic intensity. Offered load per server (a/c).
4. L := Mean number of customers in the system, including those in the buffer and at servers.
5. L_s := Mean number of customers being served.
6. L_q := Mean number of customers waiting in the buffer.
7. W := Mean length of time between a customer's arrival and the customer's departure from the system.
8. W_s := Mean length of time a customer spends at the server.
9. W_q := Mean length of time between a customer's arrival and when the customer's service starts.

Here are the equations for the performance metrics for a M/M/1 queue

$\underline{\pi_j}$

For the M/M/1 queue, we previously calculated the stationary probabilities:

$$\pi_j = (1 - \frac{\lambda}{\mu})(\frac{\lambda}{\mu})^j$$

$\underline{a,\ \rho}$

The offered load of a queue is given by the ratio of the arrival rate to the departure rate (i.e. the mean number of arrivals that occur during the mean service time). Server utilization or traffic intensity is the offered load per server. One can think about the server utilization as, on average, what proportion of the time each server in the system is occupied. For a single server queue, the offered load equals the server utilization.

$$a = \frac{\lambda}{\mu}$$

$$\rho = \frac{a}{c} = \frac{\frac{\lambda}{\mu}}{1} = \frac{\lambda}{\mu}$$

Using our solution for ρ above, the stationary probabilities of the M/M/1 queue are commonly represented as:

$$\pi_j = (1 - \frac{\lambda}{\mu})(\frac{\lambda}{\mu})^j = (1 - \rho)\rho^j$$

A brief interlude: Little's Law

Little's law states that the long-term average of the number of customers in any queue at stationarity is equal to the long-term average arrival rate λ multiplied by the average time that a customer spends in the system. Expressed algebraically using our defined variables:

$$L = \lambda W$$

Little's law also holds for the number of customers being served and for the number of customers waiting in the queue buffer:

$$L_s = \lambda W_s$$

$$L_q = \lambda W_q$$

Little's Law was originally presented by John Little in 1954 without proof, but proofs of the relationship have been published since (https://pubsonline.informs.org/doi/abs/10.1287/opre.20.6.1115). We can make use of the relationships stated in Little's Law in working with the last six performance measures.

L, W

The mean number of customers in the system at stationarity can be computed knowing the stationary probabilities of having j customers in the system:

$$L = \sum_{j=0}^{\infty} j\pi_j = \sum_{j=0}^{\infty} j(1-\rho)\rho^j$$

When $\rho < 1$, this expression reprsents the expectation of a geometric distribution with parameter $p = 1 - \rho$. Therefore,

$$L = \sum_{j=0}^{\infty} j\pi_j = \sum_{j=0}^{\infty} j(1-\rho)\rho^j = \frac{\rho}{1-\rho}$$

Using Little's Law, we can find W:

$$W = \frac{L}{\lambda} = \frac{\rho}{(1-\rho)\lambda}$$

L_s, W_s

The mean length of time a customer spends at the server is simply the mean service duration:

$$W_s = \frac{1}{\mu}$$

Using Little's Law, we can find L_s:

$$L_s = \lambda W_s = \lambda \frac{1}{\mu} = \frac{\lambda}{\mu} = \rho$$

L_q, W_q

The number of individuals in a queue system is equal to the sum of the number of individuals in the buffer and the number of individuals currently being served:

$$L_q = L - L_s = \frac{\rho}{1-\rho} - \rho = \frac{\rho - \rho(1-\rho)}{1-\rho} = \frac{\rho^2}{1-\rho}$$

Using Little's Law, we can find W_q:

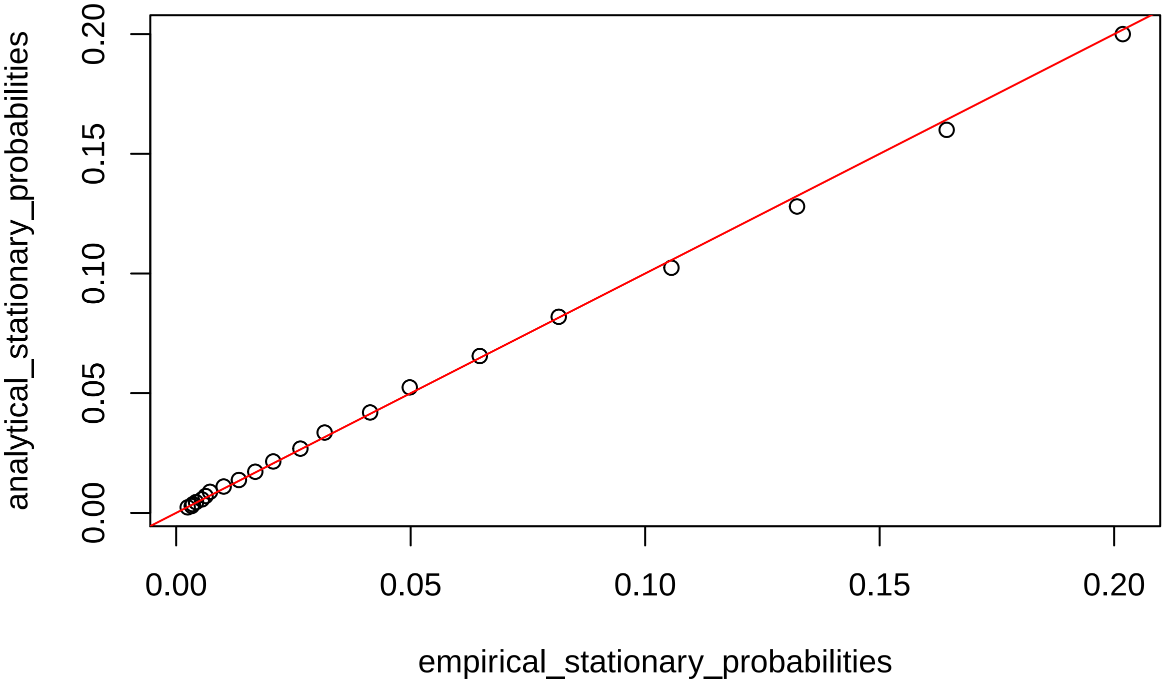
$$W_q = \frac{L_q}{\lambda} = \frac{\frac{\rho^2}{1-\rho}}{\lambda} = \frac{\rho^2}{(1-\rho)\lambda} = \frac{\rho\frac{\lambda}{\mu}}{(1-\rho)\lambda} = \frac{\rho}{(1-\rho)\mu}$$

Calculating performance measures empirically from simulated queue data

In addition to knowing the equations to compute these performance metrics for the M/M/1 queue, we can also compute them empirically for the data we simulated at the beginning of this vignette and compare them with the analytical results.

Stationary probabilities

```
###################### Working with the simulation data ######################
#this function takes as inputs the value for j and data simulated for a M/M/1 queue
#it outputs the stationary probability of having j customers in the queue
#the function divides the duration of time in which the simulated queue system has \\
  #j customers within it by the total duration of time that was simulated
calc_stationary_prob_empirical <- function(j, data){
  #calculate the time intervals between events in simulation data
  intervals <- diff(data[[2]]) #diff finds the intervals between every \\
    #two adjacent times in our simulation data
  num_customers <- head(data[[3]], -1) #for the number of customers at the time of \\
    #each event we toss out the last number because it does not correspond to a \\
    #time interval above
  stationary_prob <- sum(intervals[num_customers == j])/(max(data[[2]])-min(data[[2]]))
  return(stationary_prob)
}
empirical_stationary_probabilities <- lapply(seq(0,20,by=1),
                                             calc_stationary_prob_empirical, data=sim)
empirical_stationary_probabilities <- unlist(empirical_stationary_probabilities, use.names=F)

###################### Using analytical equations ######################
mu <- 5
lambda <- 4
rho <- lambda/mu
#This function calculates the stationary probability that jcustomers are in the system \\
  #using the analytical result
#stationary probability that j customers are in the system is given by (1-rho)*rho^j
calc_stationary_prob_analytical <- function(j, rho_){
  return((1-rho_)*rho_^j)
}
analytical_stationary_probabilities <- lapply(seq(0,20,by=1),
       calc_stationary_prob_analytical, rho_=rho)
analytical_stationary_probabilities <- unlist(analytical_stationary_probabilities, use.names=F)

#plotting the values for the stationary probabilities from the empirical vs analytical \\
  #methods for values of j from 0 to 20, we can see that they agree very well
plot(x=empirical_stationary_probabilities,
     y=analytical_stationary_probabilities)
abline(a=0, b=1, col="red")
```



```
#the MSE of empirical stationary probabilities from j=0 to 25 compared to analytical results
sum((empirical_stationary_probabilities - analytical_stationary_probabilities)^2)/
  length(empirical_stationary_probabilities)
```

```
## [1] 3.238479e-06
```

The empirical and analytical results for the stationary probability of our M/M/1 queue show very strong agreement.

Server utilization

The server utilization is the proportion of the time the server in the system is occupied, or $(1 - \pi_0)$

```
###################### Working with the simulation data ######################
1-calc_stationary_prob_empirical(j=0, data=sim)
```

```
## [1] 0.7981572
```

```
###################### Using analytical equations ######################
#server utilization is equal to rho
rho
```

```
## [1] 0.8
```

The empirical and analytical results for the server utilization of our M/M/1 queue are very close.

Average number of customers in the system and average duration of time spent by customers in the system

The empirical average number of customers in the system can be found by multiplying the number of customers at every point during the simulation by the duration of time over which there were that many customers.

The empirical average amount of time customers spend in the system can be found by subtracting each customer's arrival time from its departure time and taking the average of those values. In order to calculate these intervals, I write a function that records the number of customers in the system at the moment a customer arrives. The function then counts the number of departures from that moment in time until the number of departures matches the number of customers that were in the system at the time of the arrival (which includes the newly arrived customer). It then reports the time of that nth departure as the moment the customer departs. For example, if there are 5 customers in the system at the moment when a customer arrives, then after 5 departures have occured since that moment in time, the newly arrived customer has departed.

```
###################### Working with the simulation data ######################
### Empirical average number of customers in the system
intervals <- diff(sim[[2]]) #calculate length of time intervals between events in the simulation
n_customers <- head(sim[[3]], -1) #toss out the last recorded # customers as \\
  #it does not correspond to a time interval
#divide the sum of the time intervals * # customers in each interval by \\
  #the total amount of time that was simulated in our simulation
mean_customers_system <- sum(intervals*n_customers)/(max(sim[[2]])-min(sim[[2]]))
mean_customers_system
```

```
## [1] 3.849757
```

```
### Empirical average time a customer spends in the system
#step 1: extract all the arrival times and departure times from our simulation
times_of_arrivals <- sim[[2]][sim[[1]]==1]
times_of_departures <- sim[[2]][sim[[1]]==-1]
#step 2: find the number of customers in the system at the moment of each arrival
n_customers_at_arrivals <- sim[[3]][sim[[1]]==1]
#this function takes as input a single time of arrival for an individual customer, \\
  #the number of customers in the system at the moment of the customer's arrival, \\
  #and the vector of all departure times in our simulation
#the function then counts forward in time from the moment of the customer's arrival \\
  #until the number of customers that have departed equals the number of customers who \\
  #were in the system at the moment the customer first arrived
calc_duration_in_system <- function(time_of_arrival, n_customers_at_arrival,
                                    times_of_departures_){
  #subset the departure times to focus on times that are greater than the time that the \\
    #customer first arrives into the system
  subset_times_of_departures <- times_of_departures_[times_of_departures_ > time_of_arrival]
  #The time of departure is the moment that the number of departures that occur after the \\
    #moment of arrival equals the number of customers who were in the system at the moment \\
    #the customer first arrived
  time_of_departure <- subset_times_of_departures[n_customers_at_arrival]
  return(time_of_departure-time_of_arrival)
}
#use the above function to store the time spent in the system for all customers in our simulation
durations_in_system <- c()
for(i in 1:length(times_of_arrivals)){
  durations_in_system <- c(durations_in_system,
                           calc_duration_in_system(time_of_arrival=times_of_arrivals[i],
                                                   n_customers_at_arrival=n_customers_at_arrivals[i],
                                                   times_of_departures_=times_of_departures))
}
#in calculating the mean, we must remove NA values, which represent customers\\
  # who still have not departed from the system by the time the simulation ends\\
mean_duration_system <- mean(durations_in_system, na.rm=TRUE)
mean_duration_system
```

```
## [1] 0.9656773
```

```
###################### Using analytical equations ######################
### Analytical average number of customers in the system
#L = rho/(1-rho)
rho/(1-rho)
```

```
## [1] 4
```

```
### Analytical average time a customer spends in the system
#W = rho/((1-rho)*lambda)
rho/((1-rho)*lambda)
```

```
## [1] 1
```

The empirical and analytical results for L and W are very close.

Average number of customers at the server and average service time

For a queue system with only one server, the number of customers being served can only be 0 or 1. Therefore, the average number of customers at the server is $1 - \pi_0$.

The empirical average service time can be found by dividing the total duration of time over which service is occuring in our simulation by the number of service events that occur in that duration of time. As no service occurs during intervals where there are 0 customers in the system, we must subtract these time intervals out from the total time simulated.

```
###################### Working with the simulation data ######################
### Empirical average number of customers at the server
#average number of customers at the server is 1-pi0
mean_customers_server <- 1-calc_stationary_prob_empirical(j=0, data=sim)
mean_customers_server
```

```
## [1] 0.7981572
```

```
### Empirical average service time
intervals <- diff(sim[[2]]) #calculate the intervals between events in our simulation
n_customers <- head(sim[[3]],-1) #drop the last n_customer since it does not \\
  #correspond to a time interval
total_time <- max(sim[[2]]) -min(sim[[2]]) #total time the simulation was run
non_zero_time <- total_time - sum(intervals[n_customers ==0]) #subtract the \\
  #intervals where the n_customers was 0 from the total time
n_departures <- sum(sim[[1]] == -1) #the number of departure events that occured \\
  #throughout the course of the simulation
mean_service_time <- non_zero_time/n_departures #divide the numbers to find the \\
  #mean service time
mean_service_time
```

```
## [1] 0.2002111
```

```
###################### Using analytical equations ######################
### Analytical average number of customers at the server
#Ls = rho
rho
```

```
## [1] 0.8
```

```
### Analytical average service time
#Ws = 1/mu
1/mu
```

```
## [1] 0.2
```

The empirical and analytical results for L_s and W_s are quite close.

Average number of customers in the queue buffer and average amount of time spent by customers in the queue buffer

The number of customers in the system equals the sum of the number of customers in the queue and number of customers at the server. Therefore,

$$L_q = L - L_s$$

.

The average amount of time a customer spends in the system equals the sum of the average amount of time a customer spends in the queue buffer and the average amount of time a customer spends at the server. Therefore,

$$W_q = W - W_s$$

.

```
###################### Working with the simulation data #####################
### Empirical average number of customers in the queue buffer
mean_customers_system-mean_customers_server
```

```
## [1] 3.0516
```

```
### Empirical average amount of time customers spend in the queue buffer
mean_duration_system-mean_service_time
```

```
## [1] 0.7654663
```

```
###################### Using analytical equations ######################
### Analytical average number of customers in the queue buffer
#Lq = rho^2/(1-rho)
rho^2/(1-rho)
```

```
## [1] 3.2
```

```
### Analytical average amount of time customers spend in the queue buffer
#Wq = rho/((1-rho)*mu)
rho/((1-rho)*mu)
```

```
## [1] 0.8
```

The empirical and analytical results for L_q and W_q are quite close.

Exercise: The M/M/c queue

The M/M/c queue has Poisson arrivals and c independent identical servers, each with exponentially distributed service times.

1. What is the stationary distribution for the number of customers in a M/M/c queue system?
2. Analytically compute the 9 performance measures above for a M/M/2 queue.

3. From the viewpoint of minimizing W, the average amount of time customers spend in the queue system, which performs better: a M/M/2 queue, or a M/M/1 queue with a server that is twice as efficient as either of the servers in the M/M/2 system?

Summary

1. With knowledge of certain parameters, the behavior of a queue can be simulated.
2. Performance measures of queues can be used to determine the impact of changing certain queue parameters or to compare the performance of different queue systems.

References:

Cooper, Robert B. (1981). *Introduction to Queueing Theory* New York, NY: North Holland.

https://www.edx.org/course/queuing-theory-from-markov-chains-to-multi-server-systems-0

Ross, Sheldon M. (2014). *Introduction to Probability Models*