

Report

Zongyi Li, Zhiquan Zhang

1. Introduction and Goal

This assignment is about a simulation for a queueing network. For example, a fast food restaurant can be regarded as a queueing network, s.t, people will wait in a line first to get a tray and then go to a service station for the purpose of getting some kinds of food. After getting the first kind of food, people can randomly go to other service station to get what they need. The whole system can be seen as a queueing network. And this assignment is to develop a model like such systems to help analyze the useful information, which can be helpful for the improvement of the service.

Our target is to 1. use C language to develop a library, which contains many functions that can help build a queueing network. 2. Use C language to develop a configuration file, which contains the structure information of the system like the number and type of different components. And finally, we can let our program check the configuration file and run those files of correct format.

2. Library and Configuration

In this project, we didn't use some other APIs or existing data structures/ libraries to implement the library. We define all the data structures in application and engine files.

Implementation of the library:

The library can be divided into two parts, one is engine and another is application. In files engine.h and engine.c, we simply used the file provided by instructor, the functions included are as below.

Function	Functionality
RunSim()	Used to remove the event with the highest timestamp in the priority queue, and input it into EventHandler() function in application files.
Schedule()	Used to insert the new event from application files into the priority queue.
CurrentTime()	Used to return the system current time, the default value is 0.0.
*Remove()	Used to remove the element from priority queue.
PrintList()	Print the list of event to the screen to monitor the process

In files application.h and application.c, the main struct we define are as below.

Struct	Properties	Description
Customer	CreationTime	Record the time customer is generated

	ArrTime	Record the time customer arrive at station, and being added into a queue
	totalWait	Total waiting time in the system
CustomerList	amount	The total number of customers in the list, which is used to record the information of all the customers
	*head	The head of the list
EventData	EventType	Type of event(GENERATE, ARRIVAL, DEPARTURE)
	*Cust	Arriving or departing customers; unused for GENERATE events
	CompID	ID of component where customer created, arrives or departs
FIFOQueue	*first	Pointer to first customer in queue
	*last	Pointer to last customer in queue
Generator	IntArrTime	Mean interarrival time for generated components
	DestComp	ID of next component customers are sent to
Queue_Station	*Q	A FIFO queue in station
	AvgWaitTime	Average total waiting time for all customers in the station
	totalWait	Total waiting time in the station
	inLine	Number of customers in line
	DestComp	ID of next component customers are sent to
Fork	num	number of ports it has
	*probability	probabilities for each port
	*DestComp	D of next component customers are sent to
Exit	Count	The number of customers that exited at this component
	totalTime	System total service time and wait time for all customers
	maxTime	The maximum time a customer stayed in the system
	minTime	The minimum time a customer stayed in the system
Component[]	ComponentType	GENERATOR, QUEUE_STATION, FORK, EXIT
	*Comp	Pointer to information on component (Generator, Exit struct, etc.)

In application.c, we also defined global variables to record the customer information.

Global variable	Description
*CustList	Used a linked list to record all customer information

totalExits	Total number of customers that exited from the system
numComponents	Number of components in the system

In implementation. Firstly we initialize the Customer List, and make all the components according to the input file. Initialize all the components in each function. While making the Generator component, we schedule the first event and put it into the engine. Then we iteratively remove the event from the priority queue and use EventHandler to handle each different type of event.

If the event type is GENERATOR, we make a new customer, and add it to the Customer List. Then we schedule ARRIVAL event at the component it connected. Then we schedule next GENERATION event with timestamp generated from an exponential distribution with mean U, where U is the mean interarrival time of generation.

If the event type is ARRIVAL, then we need to check which component it arrives. If the customer arrives at Queue Station component, if the queue in this station is empty, we schedule DEPARTURE event for this customer with timestamp at current time. The component ID is still the same as this component; If the queue is not empty, we add the customer in to queue, and record its arrive time. If the customer arrives at Fork component, we firstly random select a port with their probabilities. Then we schedule an ARRIVAL event at the port we choose. If the customer arrives at Exit component, we simply record its leaving time, and calculate its total time in the system.

If the event type is DEPARTURE, we can only have a DEPARTURE event at Queue Station. We firstly schedule ARRIVAL event for the customer who is the first in queue at the component it connected, with timestamp generated from an exponential distribution with mean V, where V is the mean service time of this station, and remove this customer from the queue. Then if the queue in this station is not empty, we schedule DEPARTURE event for the next customer (who is now the first of the queue), with timestamp at current time. The component ID is still the same as this component.

After running RunSim() in the time we set, we can calculate different statistics for the whole system, and output them in files.

We wrote the random number generation functions in random.h and random.c and include them in application.c. There're three functions in these files.

Functions	Functionality
rand_init()	Initialize random function to set time seed
urand ()	Generate a random number uniformly distributed over the interval [0,1)
randexp()	Returns $-U * (\log(1 - \text{urand}()))$, where $\log()$ is the C function defined in $\langle \text{math.h} \rangle$, and U is the mean

Implementation of the configuration:

The configuration file only has a struct that help save the information of each read component.

Struct	Properties	Description
Component_read	ComponentType	The type of the component that's read

	Comp	The concrete component's pointer, which can be from generator, fork, queue or exit.
--	------	---

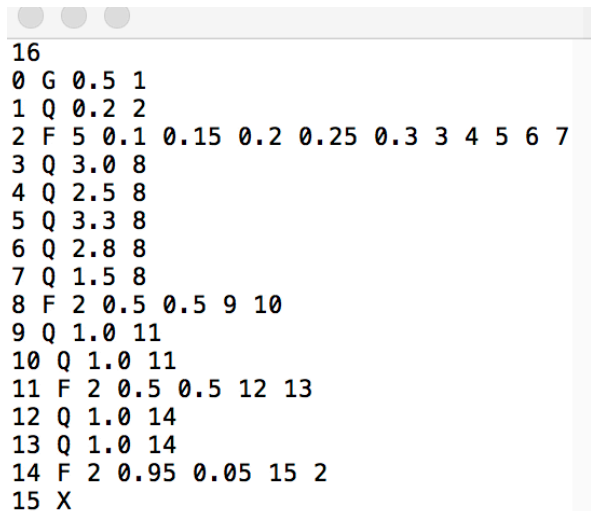
The below functions can be called directly by users to help set up a system.

Functions	Functionality
read_file()	It can read the component information and save them in the struct. It can also check the file's format and content's legality.
SetupSystem()	It can set up a system according to the content in the struct array that saves the information of a system.

In config.c file, there are some other functions that can check for the format or content errors for the configuration files but they are all hidden for users. For users, they can just use read_file() function to get and check a file's information and then call SetupSystem() function to set up a complete system. After establishing a system, users can directly run the system.

3. Test procedure and evidence of correctness of the code

First of all, based on the format of configuration file, writing a configuration program as the image below.



```

16
0 G 0.5 1
1 Q 0.2 2
2 F 5 0.1 0.15 0.2 0.25 0.3 3 4 5 6 7
3 Q 3.0 8
4 Q 2.5 8
5 Q 3.3 8
6 Q 2.8 8
7 Q 1.5 8
8 F 2 0.5 0.5 9 10
9 Q 1.0 11
10 Q 1.0 11
11 F 2 0.5 0.5 12 13
12 Q 1.0 14
13 Q 1.0 14
14 F 2 0.95 0.05 15 2
15 X

```

Obviously, this file has the correct format of input. And it could be executed correctly. And the system can run the simulation.

```

customer has departed from station 10
Event List: 238.755134 239.218566 239.920919 240.058332 240.216747 240.262260 241.739942 242.644911
Processing Arrival event at time 238.755134, Component=11
The port number is: 12
customer will arrive at 12.
Event List: 238.755134 239.218566 239.920919 240.058332 240.216747 240.262260 241.739942 242.644911
Processing Arrival event at time 238.755134, Component=12
customer will depart from 12.
customer add in line.
Event List: 239.218566 239.637968 239.920919 240.058332 240.216747 240.262260 241.739942 242.644911
Processing Generate event at time 239.218566, Component=0
customer has been generated!
Event List: 239.218566 239.541213 239.637968 239.920919 240.058332 240.216747 240.262260 241.739942 242.644911
Processing Arrival event at time 239.218566, Component=1
customer add in line.
Event List: 239.541213 239.637968 239.920919 240.058332 240.216747 240.262260 241.739942 242.644911
Processing Generate event at time 239.541213, Component=0
customer has been generated!
Event List: 239.541213 239.637968 239.920919 239.984806 240.058332 240.216747 240.262260 241.739942 242.644911
Processing Arrival event at time 239.541213, Component=1
customer add in line.
Event List: 239.637968 239.920919 239.984806 240.058332 240.216747 240.262260 241.739942 242.644911
Processing Departure event at time 239.637968, Component=12
customer will arrive at 14.
customer has departed from station 12
Event List: 239.637968 239.920919 239.984806 240.058332 240.216747 240.262260 241.739942 242.644911
Processing Arrival event at time 239.637968, Component=14
The port number is: 15
customer will arrive at 15.
Event List: 239.637968 239.920919 239.984806 240.058332 240.216747 240.262260 241.739942 242.644911
Processing Arrival event at time 239.637968, Component=15
one customer has exited.
Event List: 239.920919 239.984806 240.058332 240.216747 240.262260 241.739942 242.644911
Processing Departure event at time 239.920919, Component=9
customer will arrive at 11.
customer will depart from 9.
customer has departed from station 9
Event List: 239.920919 239.984806 240.034173 240.058332 240.216747 240.262260 241.739942 242.644911
Processing Arrival event at time 239.920919, Component=11
The port number is: 13
customer will arrive at 13.
Event List: 239.920919 239.984806 240.034173 240.058332 240.216747 240.262260 241.739942 242.644911
Processing Arrival event at time 239.920919, Component=13
customer will depart from 13.
customer add in line.
Event List: 239.984806 240.034173 240.058332 240.216747 240.262260 241.075631 241.739942 242.644911
Processing Generate event at time 239.984806, Component=0

```

The events can happen in a time order and they will be scheduled by the system. So the validation of simulating a correct format of configuration file has been done. Then I want to verify the validation of the error checking.

Below is an image that contains the wrong format of configuration file, it doesn't contain the component Generator.

```

16
1 Q 0.2 2
2 F 5 0.1 0.15 0.2 0.25 0.3 3 4 5 6 7
3 Q 3.0 8
4 Q 2.5 8
5 Q 3.3 8
6 Q 2.8 8
7 Q 1.5 8
8 F 2 0.5 0.5 9 10
9 Q 1.0 11
10 Q 1.0 11
11 F 2 0.5 0.5 12 13
12 Q 1.0 14
13 Q 1.0 14
14 F 2 0.95 0.05 15 2
15 X

```

If we run the simulation, we can see that the program reminds that the number of generator or exit is 0, so it can not be executed.

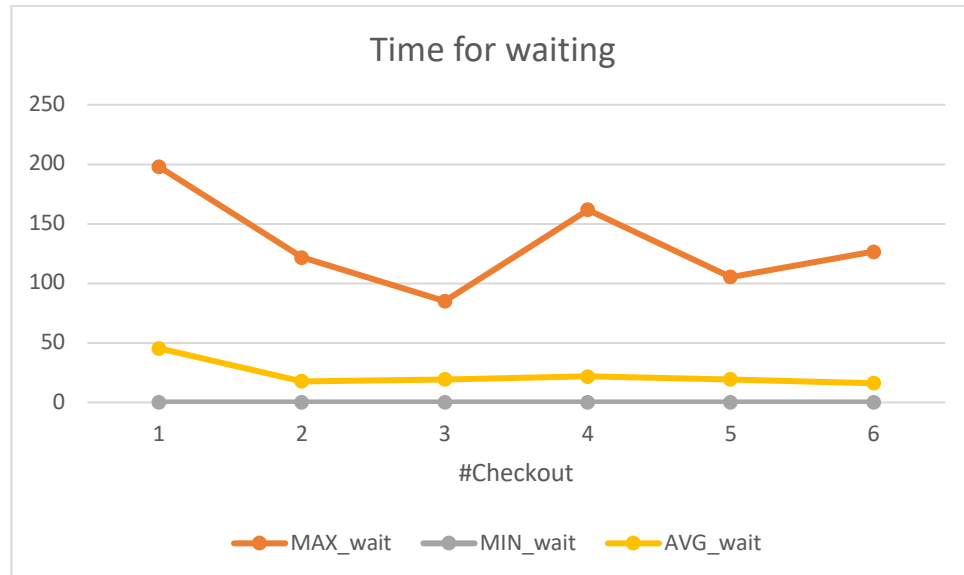
```
hed running PRO3 : PRO3
```

```
The number of generator or exit is 0!  
Program ended with exit code: 1
```

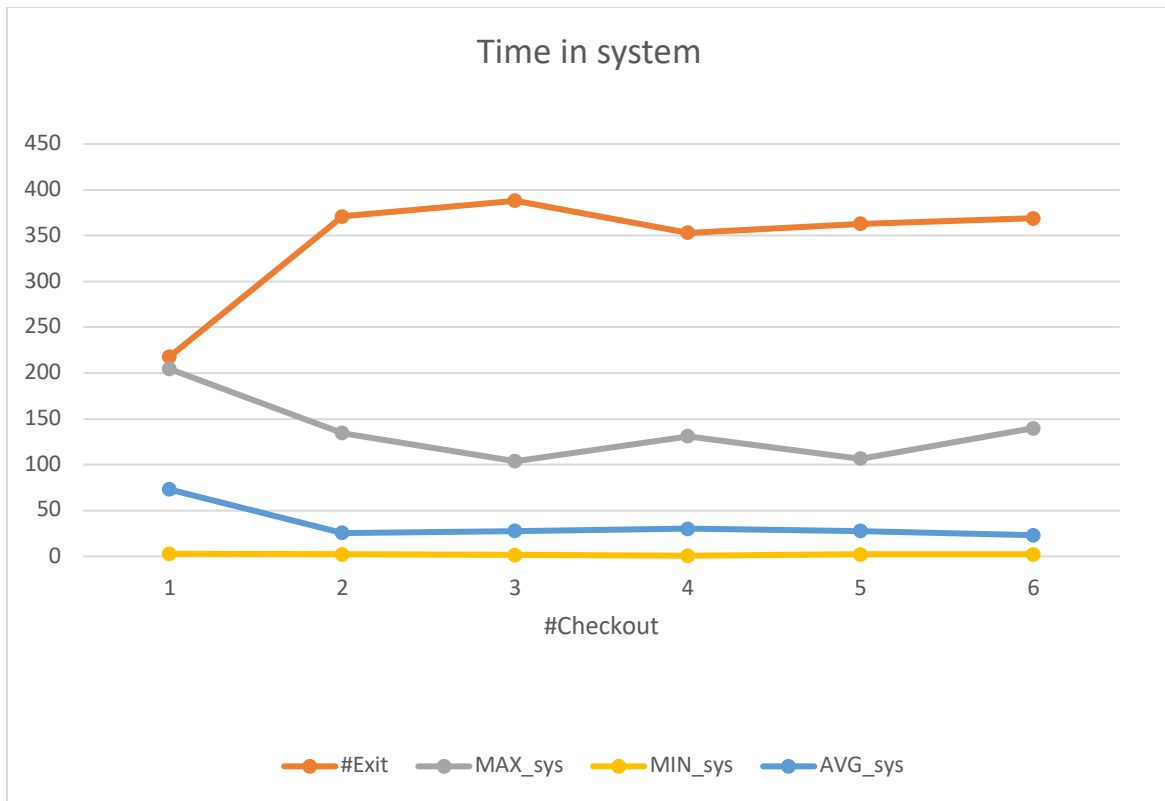
4. Implementation of simulation software.

According to the requirement, we simulate the food court system in our program.

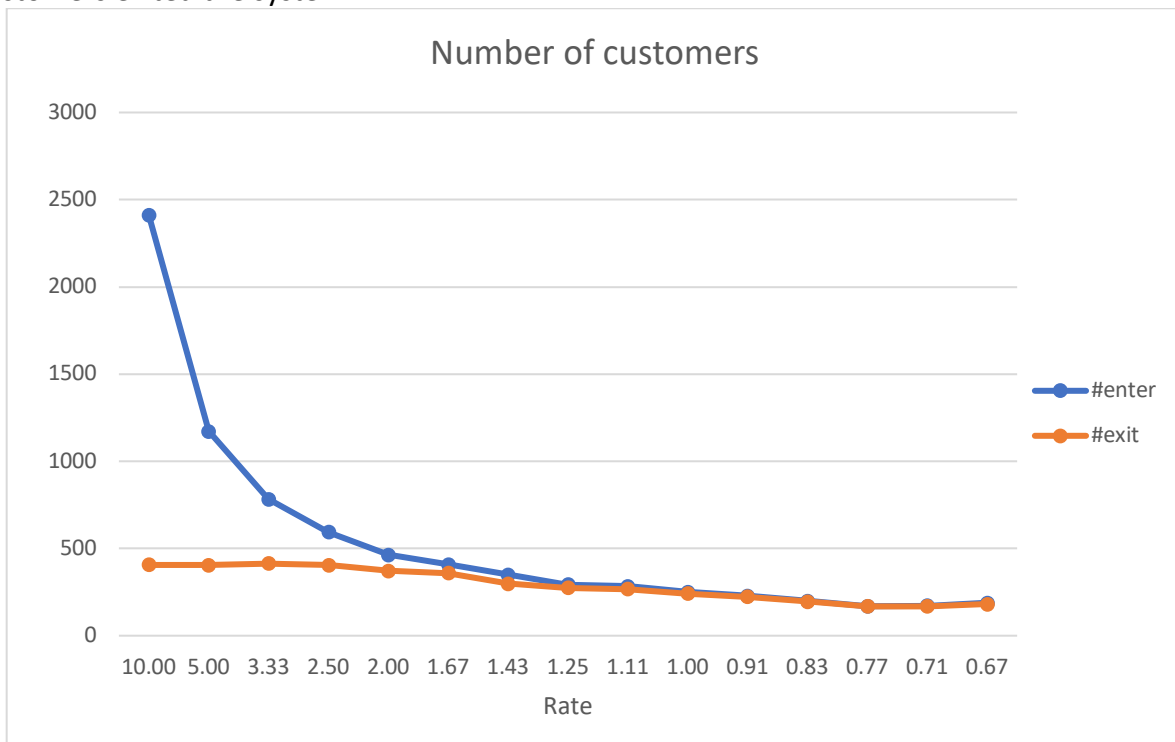
First of all, let's analyze the time data, varied based on the variation of number of check out station. It's not far to see that roughly with the increment of the number of check out station, the average waiting time decreased most apparently. As for the maximum and minimum, they don't change greatly.



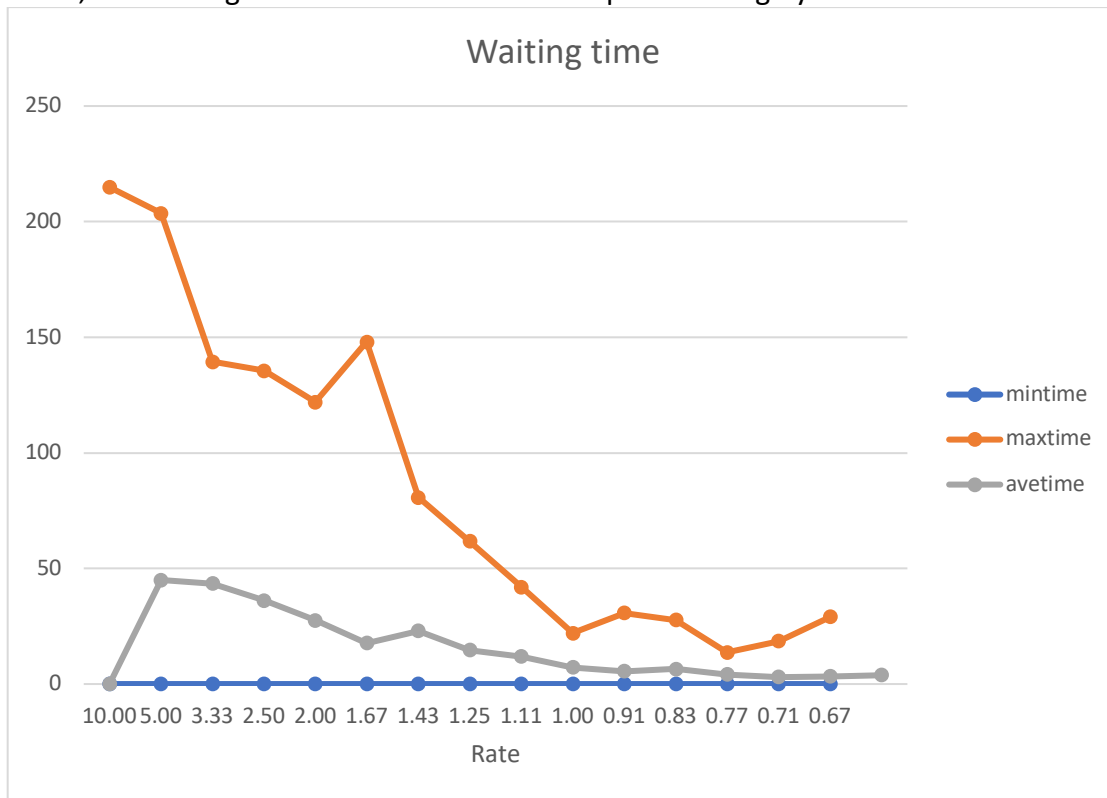
As for the time in system, the total time in a system of those who exited the system also decreased roughly with the increasement of the check out station. However, it's not very obvious because the number of check out stations will not influence the probability of exiting.



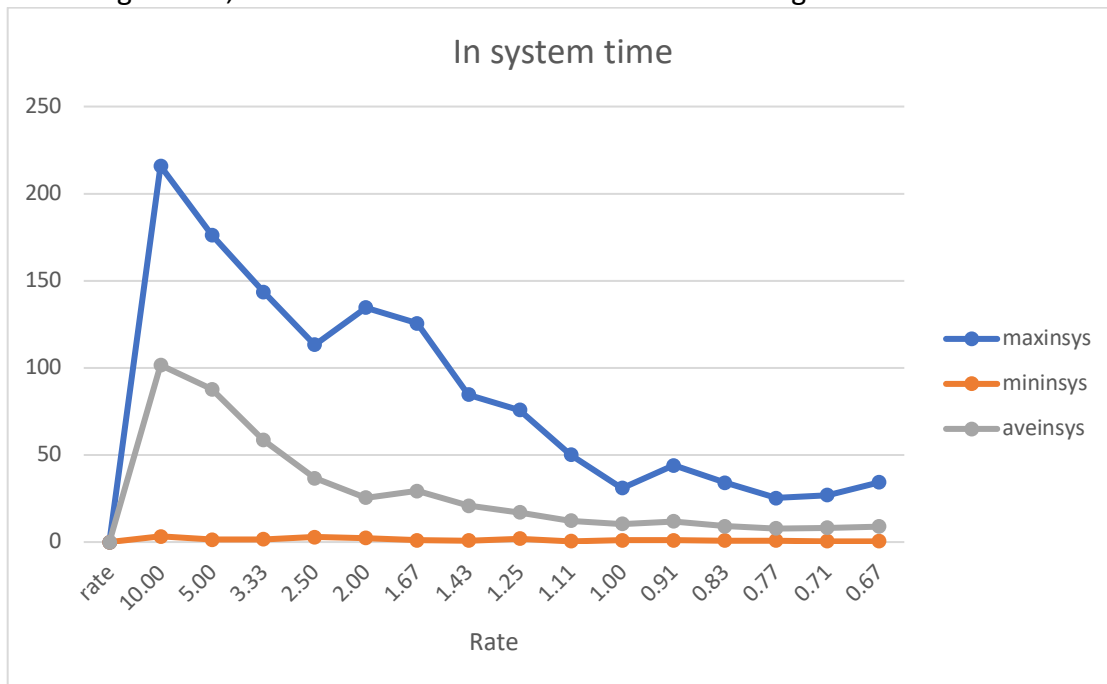
Then let's observe the variation of the rate. It's very obvious that with the decreasing of rate, the number of customers entering the system drop down and approximates the number of customers exited the system.



When talking about the waiting time's variation, it's not far to see that with the decreasing of the rate, the waiting time for customers also drop down roughly.



As for the time the costumers in tje system, it's also obvious that they all drop down with the decreasing of rate, no matter maximum or minimum or average.



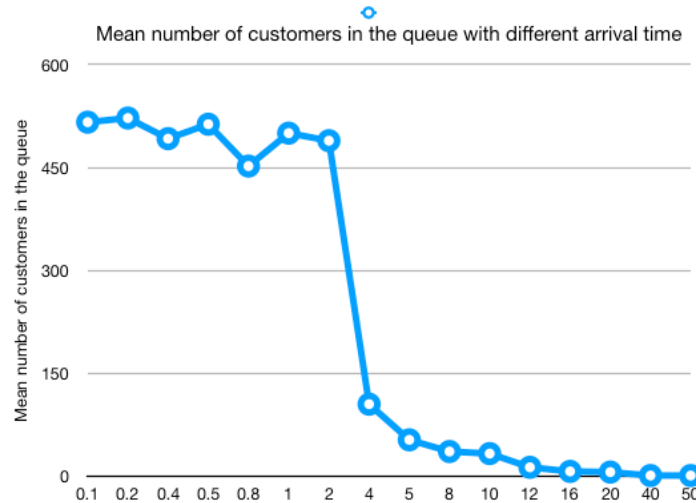
5. M/M/1 result.

M/M/1 Queueing System is a very popular model and some scientists have mentioned^[1] some different solutions to different M/M/1 model.

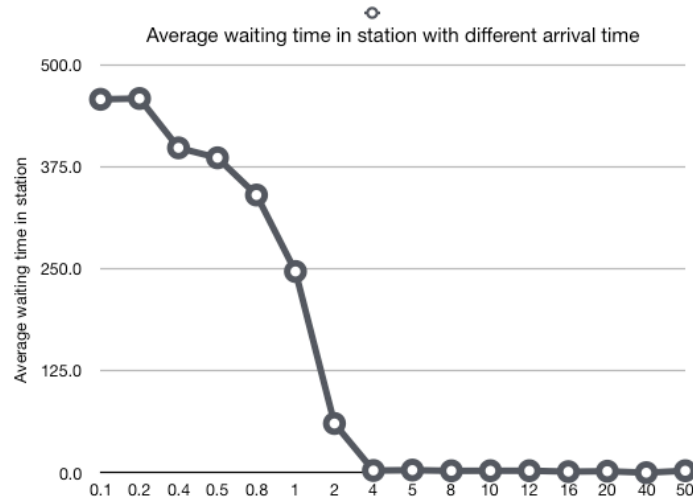
In general M/M/1 Queueing System, we fixed the mean service time of station $\frac{1}{\mu}$ as 2 min, and varies the mean interval arrival time of generator $\frac{1}{\lambda}$, where λ is the average arrival rate, and μ is the average service rate. And we define $p = \frac{\lambda}{\mu}$ as the traffic intensity (sometimes called occupancy). We test different p as below. We set the total simulation time as 1000 min.

$1/\lambda$	0.1	0.2	0.4	0.5	0.8	1	2	4	5	8	10	12	16	20	40	50
p	20	10	5	4	2.5	2	1	0.5	0.4	0.25	0.2	0.16	0.125	0.1	0.05	0.04

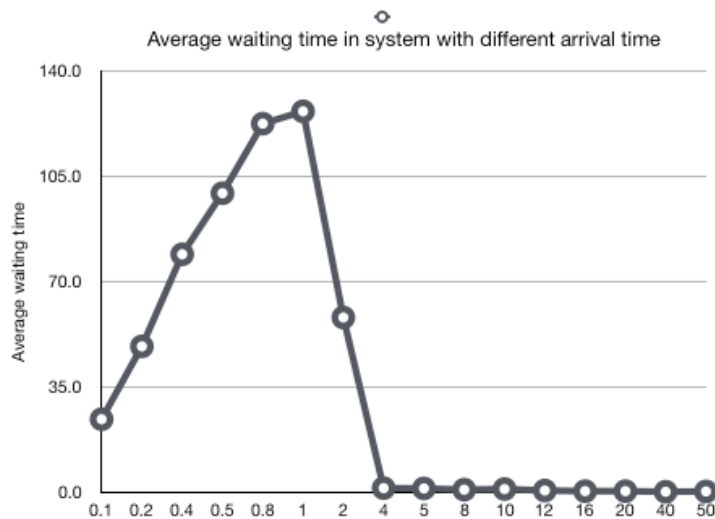
According to note^[2], we can define $Lq = \frac{p^2}{1-p}$ as the mean number of customers in the queue. The graph of Lq varies with different average arrival time is as below. We can see that as $1/\lambda$ varies from 0.1 to 50, Lq is high while $1/\lambda$ is less than 2, which means that p is small. However, as $1/\lambda$ increases and exceeds 2, Lq decreases rapidly to low values, which satisfies the function of Lq .



The mean waiting time in the queue is defined as $Wq = \frac{Lq}{\lambda} = \frac{p}{\mu(1-p)}$. The graph of average waiting time in station (queue) with different average arrival time is as below. Also, we can see from the graph that Wq drops slowly when $1/\lambda$ is less than 2, and decreases rapidly to some low values while $1/\lambda$ increases after 2. It shows that while p becomes larger than 1, Wq becomes very small. The result also satisfies the equation defined in the paper.



The graph of average waiting time in the system is defined as $W = W_q + \frac{1}{\mu} = \frac{1}{\mu(1-p)}$. Since μ is fixed, W only varies with p . We can also confirm from the graph below that while $1/\lambda$ increases from a small value (0.1) to 1, W increases. But when $1/\lambda$ reaches some threshold (like some value above 1 but below 2), W decreases rapidly to some very small values.



We can summarize from the graphs that when the average arrival rate starts approaching the average service rate, p approaches 1. In this situation, the server would always be busy hence leading to a queue build up (large number of customers in queue), and the average waiting time in system will become very large. However, if the arriving time (intervals) become larger than service time, the average total waiting time in system will drop rapidly.

Reference:

[1] Sherif I. Ammar. Transient solution of an M/M/1 vacation queue with a waiting server and impatient customers[J]. Journal of the Egyptian Mathematical Society, 25-3(2017): 337-342.

[2] Note Queuing Formulas. A. Gosavi, Department of Engineering Management and Systems Engineering, Mis- souri S & T.