OPERATING SYSTEMS

Assignment-4 (ES19BTECH11003)

REPORT

This Report contains the working details of algorithm,system model and analysis on graphs that are (N vs Avg Waiting Time) and (X vs Avg waiting time).

LOW LEVEL DESIGN AND WORKING DETAILS OF ALGORITHM

Initially we declare 2 semaphores MUTE and block globally and assign 1 and 0 respectively in the beginning of main.

We use rand function to generate random numbers and create those no of threads using create function and we also check that total no of threads will not exceed the total no of people(i.e; N) before the creation of threads.

We also calculate the eating time in the main function itself using exponential distribution and random_engine generator and we pass the thread index number and this time while calling the testCS function for each thread at time of creation.

And in testCS function the algorithm which is used to implement is similar to the algorithm that is posted in google classroom.

We define a double 2-D array ENTER_TIME which stores all the entry,access and exit times in microseconds and stores them in the same index of thread.

At the beginning of the testCS function we also use time_in_HH_MM_SS_MMM function to store the time strap and we store that time strap in the req_enter_time string array defined globally.

We use wait() for MUTE semaphore and we also declare a boolean named must_wait which helps us in denoting whether the table is completely filled or not.

Using the if condition it is correct if must_wait is true and also eating+1>X which also means table is completely filled then we increase waiting count and assign must_wait to true and assign MUTE.signal() and block.wait() so that all extra people will be blocked until table becomes empty.

In else we increment eating which indicates the exact count of eating people and also assign must_wait to true if waiting > 0 and eating==total no of seats which means the table is filled and forms a group.so we assign MUTE.signal() to make resource available again.

After else loop we again find the time and store them in respective arrays and later we use sleep(T) which indicates the eating time. And we use MUTE.wait() function so as to make newly entered people wait until the resource is available.

After completion of eating time we decrement the eating variable and now we check the eating value and if it is 0 which means table is empty so we assign k to min(no of seats,waiting people) and accordingly we increase eating variable and decrease waiting variable by k and now again we update the must_wait value basing on whether table is completely filled and waiting > 0 so that again a group is formed.

So now as k customers are moved to eating so we call block.signal() k times so as to make resource available for another k people and last we call mute.signal() so as to make resource available for new customer as this customer is leaving.here we also find the time straps and store in respective arrays.

Later using the print() function as we stored the no of microseconds in a 2-D array so by finding the minimum in that array each time and finding the index we got to know the ascending order of events that are taking place for a given thread.

SYSTEM MODEL

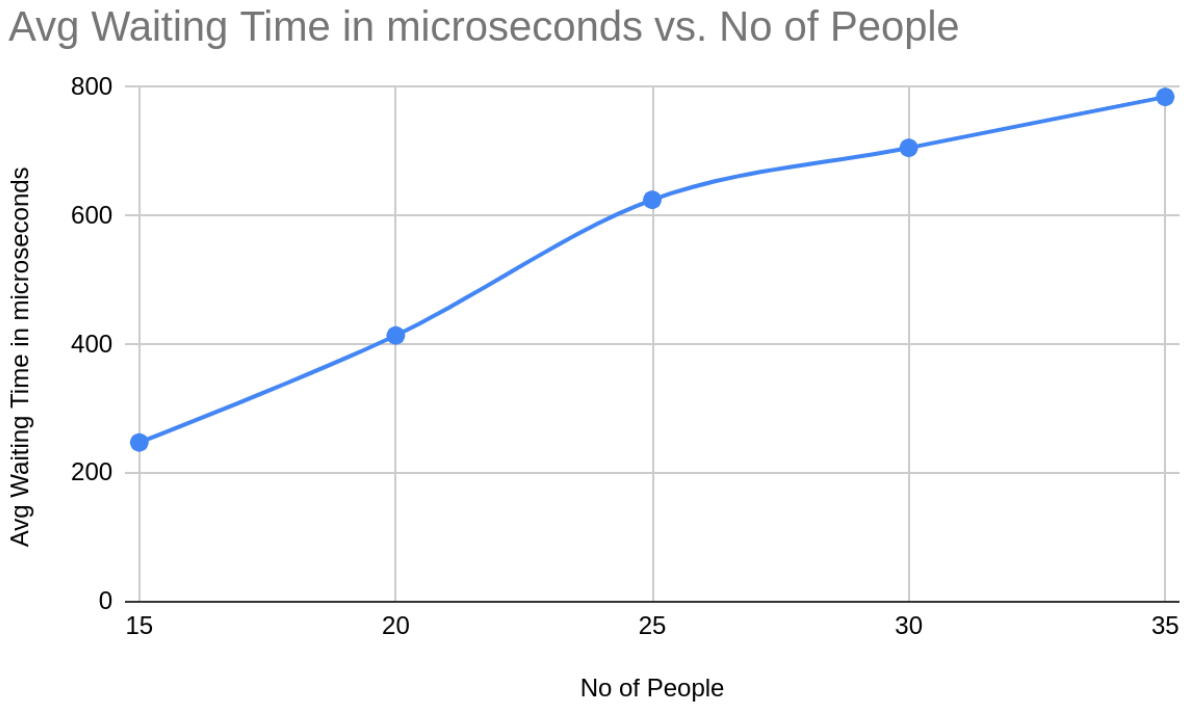
Here we follow the decentralised approach where all the customer threads are created in the main function itself and we don't create a master thread separately to make sure these threads execute correctly.We also see that during these each

customer thread creation using if condition we make sure that no of customer threads created is equal to total no of customers (i.e; N)

PLOTTING AND ANALYSIS ON GRAPHS

GRAPH-1

No of people vs Avg waiting time by each customer.



The input file use for this graph is N changes from 15-35 with difference of 5
X =4
ƛ =0.2
r=1
Γ=5

Here the average waiting time is in microseconds.

As we can see as the no of customers increases (i.e; as N value increases) we can see increase in the value of average waiting time in somewhat linear manner but as

threads count increases the slope of the line is changing(slightly decreasing) which is leading to smaller increase in the average waiting time than the previous case.

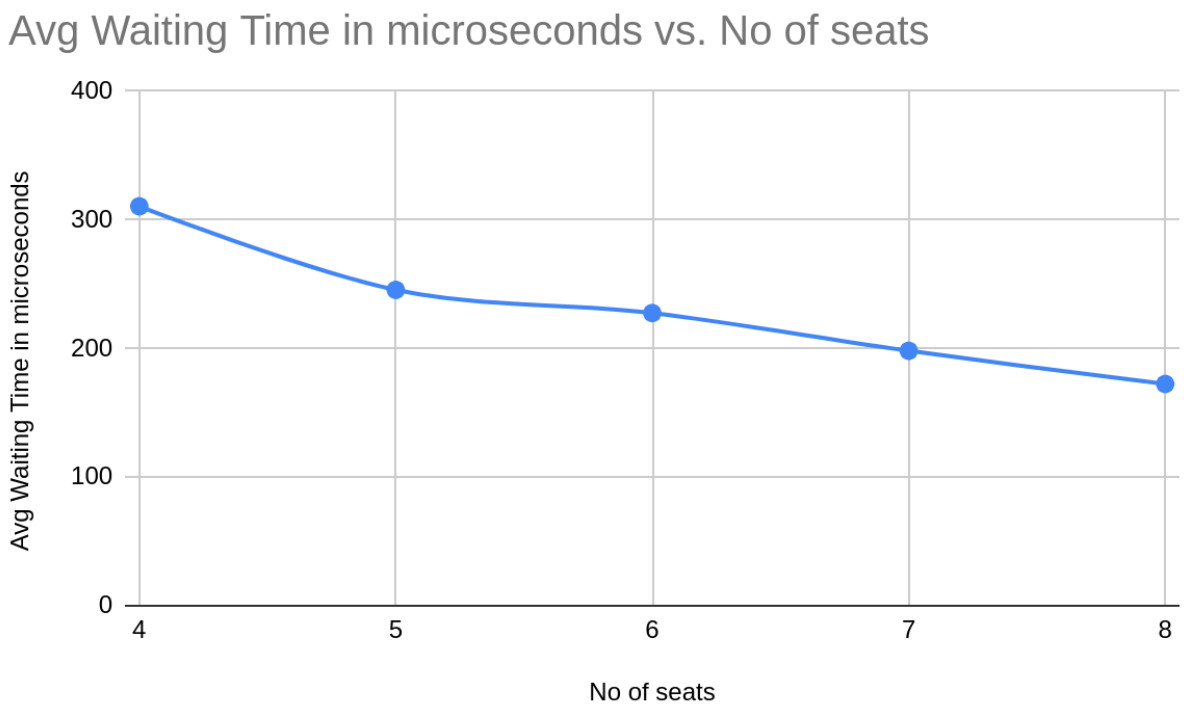
It is understandable because in a general manner as we know as no of people increases keeping the no of seats fixed.we obviously know that average waiting time must increase.

CONCLUSION

As N value increases the avg waiting time increases but rate of increase slightly decreases as N value increases gradually.

GRAPH-2

No of seats in the restaurant vs Avg waiting time by each customer.



The input file use for this graph is X changes from 4-8 with difference of 1

N =15

λ =0.2

r=1
Γ=5

Here the average waiting time is in microseconds.

As we can see that as X value increases we can see the slightly decrease in the average waiting time and it is not a linear decrease but slightly decreases as no of seats(i.e; X) increases.

And it is also quite understandable as keeping the no of people constant and if we increase the no of seats in the restaurant there should definitely be a decrease in average waiting time.

CONCLUSION

As X value increases so as no of seats increases so from graph average waiting time decreases.

So using of semaphores is very useful and helps in synchronization and also saves a lot of time making threads run parallely even in critical section.