CS 520 Homework 3 | CWID 10430147 | Divyendra Patil | Username: dpatil3  
Date: 09/10/2017

1] Since the time of drive of the Bus between the two adjacent stops and passenger boarding time are not changing, the distance between two buses also doesn’t change. The principal element having an effect on non-changing behavior in the distance is time shelled out by the bus on any given bus stop.

This time period spent on any stop is given by

**Time at stop for a bus = Avg board time (For passenger) × No of passengers (at one stop)**  
Since arrival time of passenger are not fixed / random, there can be any number of passenger arriving at stop. This will result in delay of bus leaving the stop if there are too many passengers to board. Or bus will leave early if there are too few passengers. It can also be the case that there won’t be any passenger on stop so bus need not stop/halt.

There are three solutions to keep uniform distance between buses:

1] Allow a certain number of passengers on each stop.   
Because boarding time for every/each person is persistent, this option will make identical number of passenger board on every stop & hence therefore making sure same time of bus on each stop.

2] By Making sure that every bus shows/tells every other bus that it is ready to leave the stop. Once all the buses are ready, each of them leave at the same time. The indication of the bus to leave should be only when all the passengers on the stop are on boarded.

3] The last point is that the bus should leave the stop only after a assured interval, that is at the same time regardless of the number of passengers boarded on the bus at the end of the time interval.

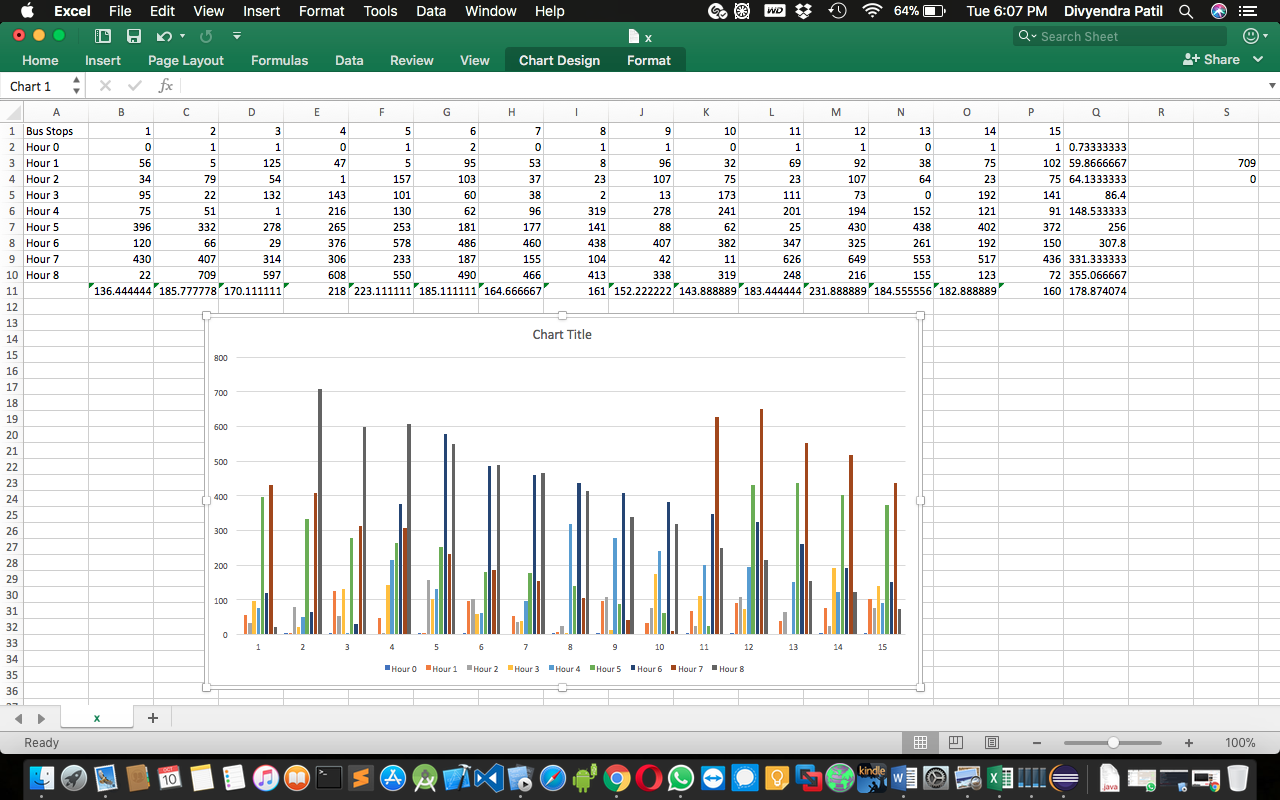
The point to note here is that in solutions 1 & 3 there are constant number of passengers to board the bus which makes sure that the time on the buses are equal. But the solutions are not perfect.

Because the solutions don’t see the likelihood of passengers being absent on the stop.   
If there are no passengers at stop, all the solutions squander time of bus on stop.   
But also keep in mind that there is a wastage of time in solutions 2 & 3 at a bus stop where few or 0 passengers are present while passengers are allowed to board the bus at other stops.

2] My observations:

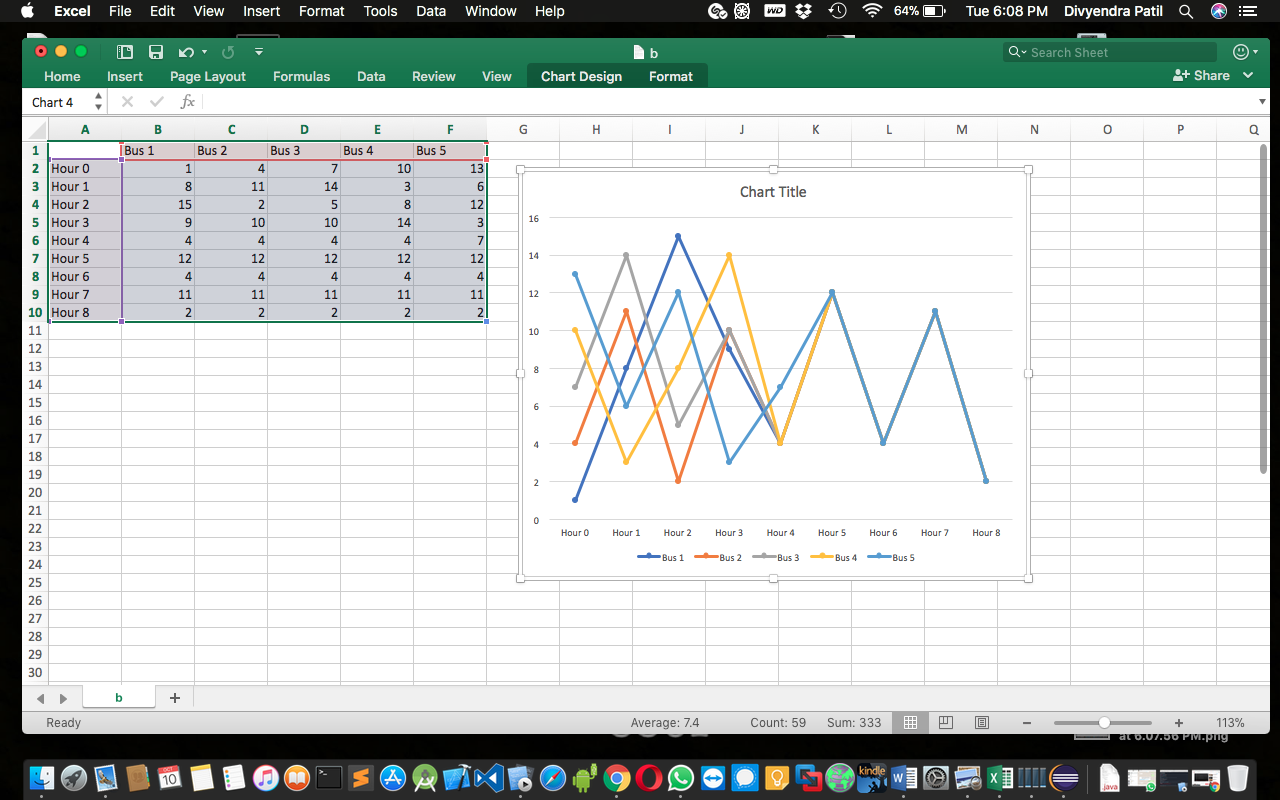
Average waiting queue size at each stop is 178 passengers per minute, maximum value is 709, minimum value is 0

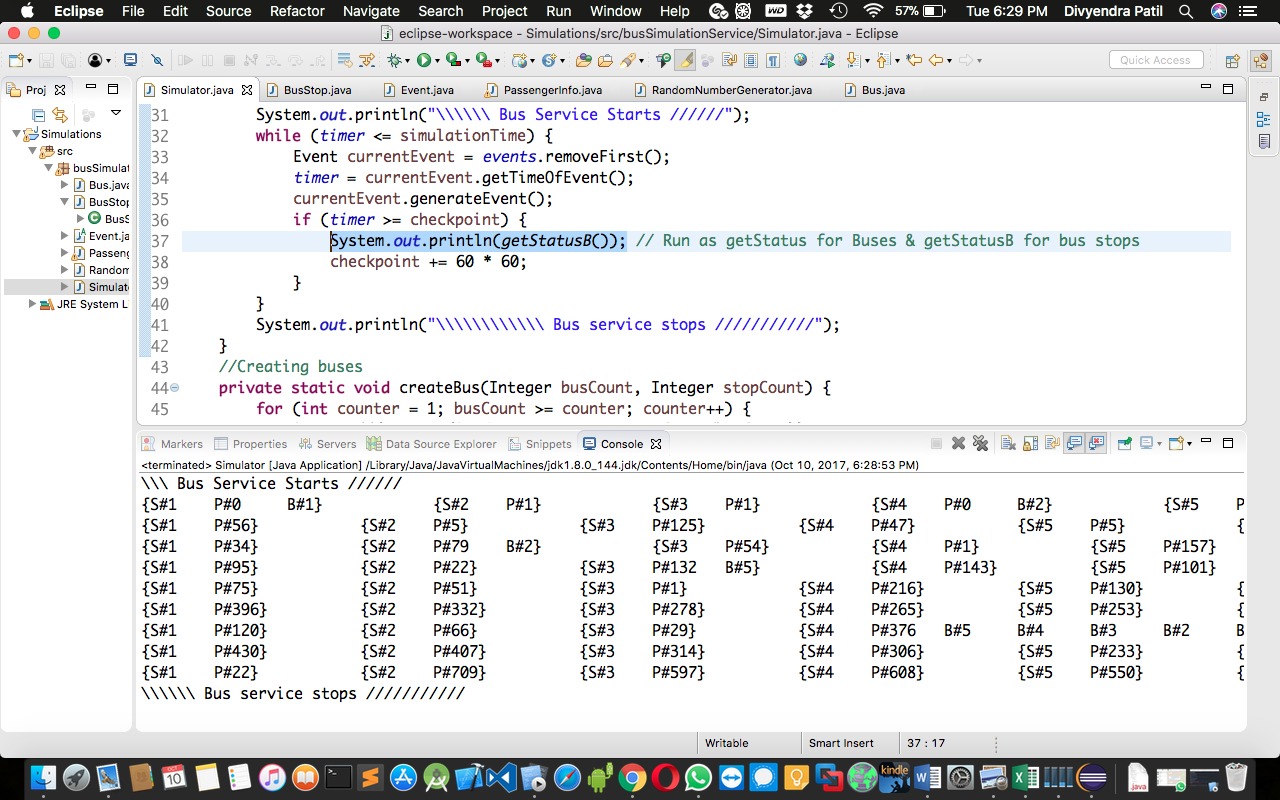
X Denotes Bus-stop numbers and Y denotes number of people in the queue. For eg Hour 3 has 100 people waiting on First Bus Stop.

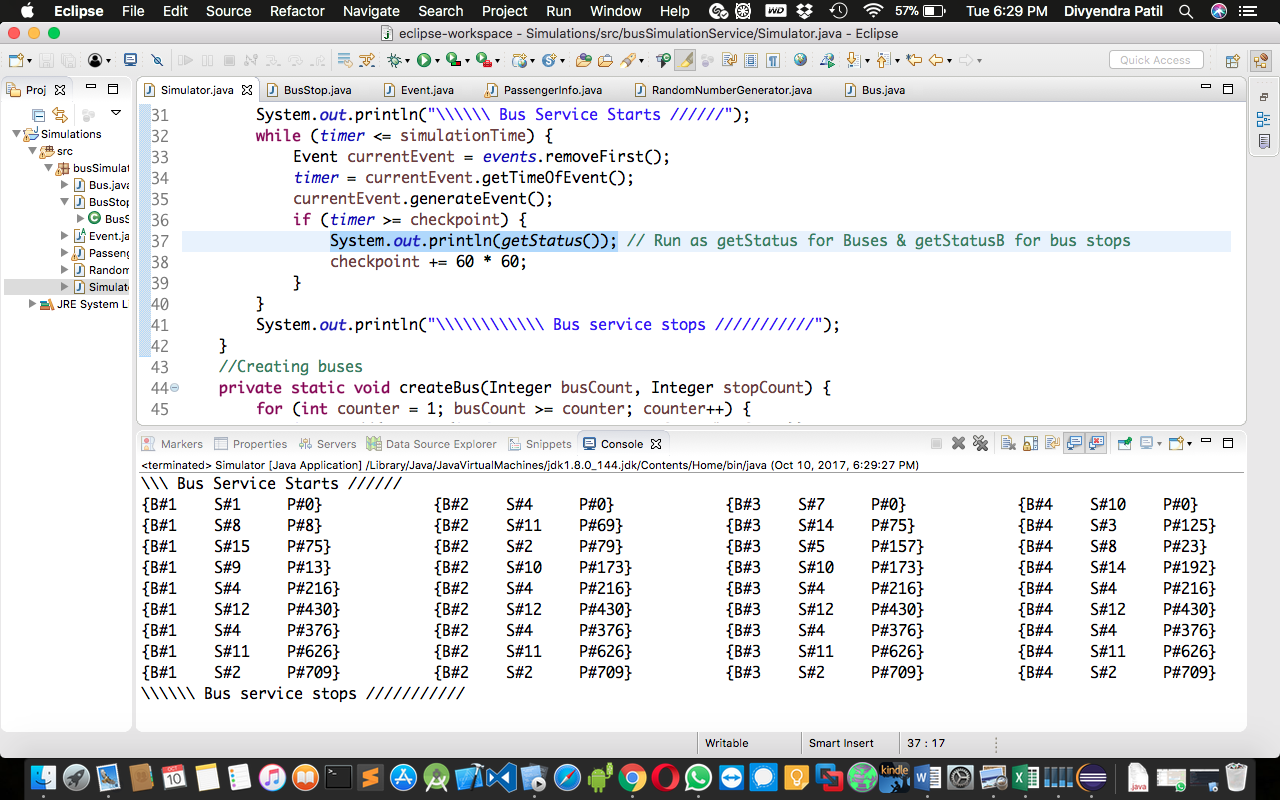


Plotting of buses with respect to time.

The colors gives bus number and X denotes the hour function (The position at that hour) as per Y. & Y denotes bus stop number.







What can be seen and concluded from this experiment:

1] If we change the mean arrival rate of passengers, we can see that more passengers arrive after decreasing the mean arrival rate which means that Length of waiting queue is inversely proportional to mean interval arrival time &

**Length of waiting queue = Time spent by bus on stop**

2] If we change the change the board time of every person then the boarding time was reduced when the number of passengers on the bus was more.   
This can conclude that   
Time spent by bus on any stop was inversely proportional to boarding time of passenger.

3] If we change the time of driving time of bus we can perceive that as the buses arrive early at each stop, the wait queue at bus stops were shorter & when driving time was reduced, wait queue were longer.   
Hence we can conclude that   
The length of the wait queue is inversely proportional to bus drive time

&   
Time spent by a bus on stop = Waiting queue.

4] If we change the number of passenger randomly then the passenger queue waiting at bus stop are longer than when single passenger was arriving which basically means the time spent by bus on any stop is greater in this case but the point to be noted here is that the time which increased was exactly same on all the bus stops.

.

The Program­

Class: MainJavaSimulator

package busSimulationService;

import java.util.LinkedList;

/\*\*

 \* @author Lancelot

 \* Main Class from where the program runs

 \*/

public class MainJavaSimulator {

public static Double timer = 0.0; // Current time in the simulation

public static LinkedList<Event> events = new LinkedList<Event>(); // Current stack of the events to occur in future

private static LinkedList<BusStop> busStops = new LinkedList<BusStop>(); // List of current bus stops and their statuses

private static LinkedList<Bus> buses = new LinkedList<Bus>(); // List of buses and their statuses

public static void main(String[] args) {

//Initialization variables for Simulation

Double simulationTime = 8 \* 60 \* 60.0;

Integer noOfBuses = 5;

Bus.timeToTravelToNextStop = 5 \* 60.0;

Integer noOfBusStops = 15;

PassengerInfo.BoardingTime = 2.0;

BusStop.meanInterArrivalRate = 12.0; // Mean arrival time of passengers is 5 per minute, that is per passenger in 12 seconds.

// Creation of buses and their stops

createBusStops(noOfBusStops);

createBus(noOfBuses, noOfBusStops);

long checkpoint = 0;

System.out.println("\\\\\\ Bus Service Starts //////");

while (timer <= simulationTime) {

Event currentEvent = events.removeFirst();

timer = currentEvent.getTimeOfEvent();

currentEvent.generateEvent();

if (timer >= checkpoint) {

System.out.println(getStatus()); // Run as getStatus for Buses & getStatusB for bus stops

checkpoint += 60 \* 60;

}

}

System.out.println("\\\\\\\\\\\\ Bus service stops ///////////");

}

//Creating buses

private static void createBus(Integer busCount, Integer stopCount) {

for (int counter = 1; busCount >= counter; counter++) {

buses.add(new Bus(busStops, counter, stopCount/busCount));

buses.getLast().generateEvent();//Generating the first event for it

}

}

// Creating Bus Stops

private static void createBusStops(Integer count) {

for (int counter = 1; count >= counter; counter++) {

busStops.add(new BusStop(counter));

busStops.getLast().generateEvent(); // Generating first event for it i.e a passenger arrival

}

}

public static String getStatus() {

StringBuilder status = new StringBuilder();

buses.forEach((B) -> {status.append(B.getStatus() + "\t\t");});

return status.toString();

}

public static String getStatusB() {

StringBuilder status = new StringBuilder();

busStops.forEach((B) -> {status.append(B.getStatusB() + "\t\t");});

return status.toString();

}

}

Class: PassengerInfo

package busSimulationService;

/\*\*

 \* @author Lancelot

 \*

 \*/

public class PassengerInfo {

private Double Arrival\_Time;

public static Double BoardingTime = 0.0;

public PassengerInfo() {

Arrival\_Time = MainJavaSimulator.timer;

}

public PassengerInfo(double time\_of\_arrival) {

this();

this.Arrival\_Time = time\_of\_arrival;

}

}

Class: RandomNumberGenerator

package busSimulationService;

/\*\*

 \* @author Lancelot

 \* As per given in class from slides

 \*/

public class RandomNumberGenerator {

private static int seed = 1000;

public static Double random(){

Double x = - Math.log((seed + 1) / 65536.0);

seed = (25173 \* seed + 13849) % 65536;

return x;

}

}

Class: Event

package busSimulationService;

public abstract class Event {

private Double timeOfEvent;

public abstract void generateEvent();

private Integer position = -1;

protected void setTimeOfEvent(Double timeOfEvent) {

this.timeOfEvent = timeOfEvent;

}

public Double getTimeOfEvent() {

return timeOfEvent;

}

public void eventPerformed() {

position = 0;

}

public void PositionIncrementation() {

position++;

}

public Integer getPosition() {

return position;

}

// Adding event into the linked list by sorting on its time

public void addEvent(Event event) {

MainJavaSimulator.events.forEach((E) -> {if (E.getTimeOfEvent() <= event.getTimeOfEvent()) {

event.PositionIncrementation();

}});

MainJavaSimulator.events.add(event.getPosition(), event);

}

}

Class: Bus Stop

package busSimulationService;

import java.util.LinkedList;

/\*\*

 \* @author Lancelot

 \*

 \*/

public class BusStop extends Event {

// waiting queue of passengers for buses at the stop

private LinkedList<PassengerInfo> passengers = new LinkedList<PassengerInfo>();

public static Double meanInterArrivalRate = 0.0;

private Integer busStopNo;

// List of buses at current stop by their order of arrival

private LinkedList<Bus> busAtStop = new LinkedList<Bus>();

private BusStop() { }

public BusStop(Integer busStopNo) {

this();

this.busStopNo = busStopNo;

}

public String getStatus() {

return "S#" + busStopNo + "\tP#" + passengers.size();

}

public StringBuilder getStatusB() {

StringBuilder sBuilder = new StringBuilder();

sBuilder.append("{S#" + busStopNo + "\tP#" + passengers.size());

busAtStop.forEach((B)->{sBuilder.append(B.getStatusB());});

sBuilder.append("}");

return sBuilder;

}

public Integer getPassengersCount(){

return passengers.size();

}

// Handling event generating i.e arrival of passengers & putting new event in the queue for future

public void generateEvent() {

passengers.add(new PassengerInfo()); // Adding New passenger

eventPerformed();

// Setting the time for next passenger to arrive

setTimeOfEvent(MainJavaSimulator.timer + meanInterArrivalRate \* RandomNumberGenerator.random());

// Adding the event into the event list..

addEvent(this);

}

public void passengerBoardBus() {

passengers.removeFirst();

}

public boolean passengersAvailable() {

if (passengers.size() > 0) {

return true;

}

return false;

}

public void busArrived(Bus bus) {

busAtStop.add(bus);

}

// Buses will depart as per their arrival order

public Bus busDeparting(Bus bus) {

if (busAtStop.peekFirst().equals(bus)){

busAtStop.remove(bus);

return null;

}

return busAtStop.get(busAtStop.indexOf(bus) - 1);

}

public boolean isThisBusHere(Bus bus) {

return busAtStop.contains(bus);

}

}

Class: Bus

package busSimulationService;

import java.util.LinkedList;

/\*\*

 \* @author Lancelot

 \*

 \*/

public class Bus extends Event{

// Order of the bus stop the bus will arrive at.

private LinkedList<BusStop> busStops = new LinkedList<BusStop>();

public static Double timeToTravelToNextStop = 0.0;

private Integer busNo;

private Bus() {}

public Integer getBusNo() {

return busNo;

}

public Bus(LinkedList<BusStop> busStops, Integer busNo, Integer interval) {

this();

this.busNo = busNo;

this.busStops.addAll(busStops);

// Initializing buses at bus stops at equal distances from each other.

for (int count = 0; count < (busNo - 1) \* interval; count++) {

this.busStops.addLast(this.busStops.removeFirst());

}

}

public String getStatus() {

return "{B#" + busNo + "\t" + busStops.getFirst().getStatus() + "}";

}

public String getStatusB() {

return "\tB#" + busNo;

}

/\*

\* Handling event for bus where in (if there are passengers on stop bus will wait for

\* another boarding time of passengers) else bus will move to next stop.

\* \*/

public void generateEvent() {

BusStop currentStop = busStops.getFirst(); //checking current stop where the bus is..

// Was the bus already at this stop

if (!currentStop.isThisBusHere(this)) {

currentStop.busArrived(this);

}

// Are the passangers on stop

if (currentStop.passengersAvailable()) {

currentStop.passengerBoardBus();

setTimeOfEvent(MainJavaSimulator.timer + PassengerInfo.BoardingTime);

} else {

// No passengers on stop, thus departing for the next stop

Bus busAhead = currentStop.busDeparting(this);

if (busAhead == null) {

setTimeOfEvent(MainJavaSimulator.timer + timeToTravelToNextStop);

busStops.removeFirst();

busStops.addLast(currentStop);

} else {

setTimeOfEvent(busAhead.getTimeOfEvent());

}

}

eventPerformed();

addEvent(this);

}

}