**Road Traffic Simulation for Analysing Travel Time for Synchronized and Unsynchronized Traffic Signals**

**Project Checkpoint**

**Team No. 24**

Chandra Khatri

Hidekazu Kose

Hao Luo

**1. Project Description**

The purpose of this project is to simulate any vehicle traveling along Peachtree Street within a given zone between 10th Street and 14th Street (Fig.1), obtain the distribution of travel time for all vehicles transversing the designated zone within a given time period and compare that for two scenarios. The first case assumes that all the traffic signals in the intersections are synchronized, meaning the phase among signals are fixed. The other case is an unsynchronized one, where the signals operates with a randomly assigned phase. Essentially, the former scenario is one special case of latter: the fixed phase can also be one of the random phases. Thus the synchronized phase can be chosen from unsynchronized pool. The one which outruns the rest in several unsynchronized simulations can be the choice for the synchronized simulation.

The static road network is obtained through third party information (Google Map). There are in total 5 intersections in the area of interest, of which 4 are signalized and 1 is unsignalized( intersection of 13th Street with Peachtree Street). The sequence for each signal follows green-yellow-red cycle, with the duration for the timing pattern of each signal fixed For both scenarios.The specific timing pattern for all the signals can be found in the Excel file signalTiming.xls. Along this part of Peachtree Street, there are 6 sections, with each section have different length and number of lanes. A general process for an individual vehicle is depicted as follows:

1) Enter the zone of interest: The vehicle can enter from North Bound(NB) or South Bound(SB) ;

2) Go through intersection or turn right or turn left if signal turns green;

3) Decide whether stop or continue its movement if signal turns yellow;

4) Stop if signal turns red and continue its movement as in 2 if signal turns green again;

5) Leave the zone of interest: as in 1, the vehicle can leave into four directions.

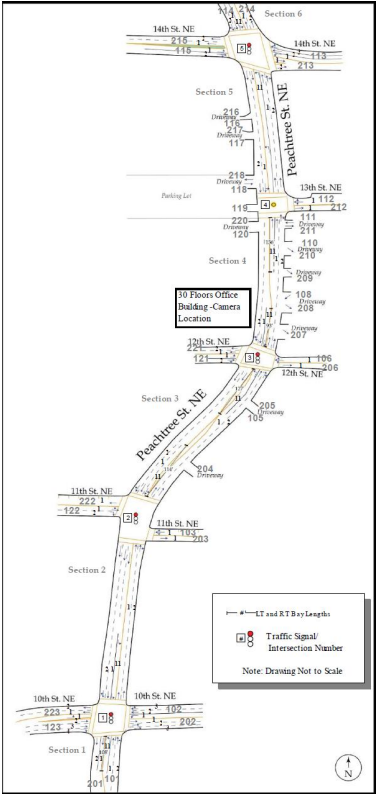


Fig. 1

**2. Conceptual Model**

**2.1 Assumptions(A) and simplifications(S):**

(A) Drivers don’t push the accelerator

-> (S) Velocity of a vehicle is always constant (terminal velocity = Vc). However, the velocity of each vehicle is drawn randomly.

-> (S) Vehicle reaches Vc immediately after it leaves from an intersection.

-> (S) Vehicle immediately stops in front of an intersection if its traffic signal is red or if there exist a queue. We might incorporate this in the model.

(S) Vehicle’s length is a constant;

(A) Driver ignores yellow signal **or** driver make a stops immediately after a traffic light turns yellow.

-> (S)Each phase of a yellow signal is regarded as a part of green signal phase **or** a part

of red signal phase.

(S) Distance between preceding vehicle and following vehicle is constant

(A) Drivers do not make lane change.

-> (S)Vehicle does not pass the car in front.

-> (S)Vehicle keeps the same lane position even if the neighboring lane is empty.

(S)All vehicles come from North(or South) to South (North). For now we assume that vehicles are only in North/South bound and the traffic coming/leaving from the East/West on the intersections or connecting lanes is negligible and does not impact the traffic along N/S bound.

**2.2 Input**

At time zero, an initial distribution of vehicles on the road is based on a random selection from the vehicle pool, which is constructed by all the vehicles at an arbitrary time frame denoted in NGSIM[1] data ( a comprehensive description of it can be found in final report). As time goes on, the entry of vehicles for each lane is modeled as Poisson distribution with lambda equal to 1 [2][3].

**2.3 Design**

We build Process-oriented simulation program.

**2.4 Content**

State variables( Dynamic information stored for each vehicle, signal and queue)

1) For vehicles: (Attributes of vehicles. We should not put Vehicle’s variables as Global State variables)

-> local position( i.e. which lane, which section) (We actually are not using this. Since, we are modeling process based. We just create a Future Event List for each vehicle and then process the events)

-> movement( i.e. turning or going straight or stop) (We are treating each direction differently and maintaining different queues and Future Event List for each direction differently)

-> preceding vehicle;

->state: moving or stopped;

-> Arrival Time;

-> Wait time (queues, signals);

->service\_time (when the vehicle is actually travelling with some velocity);

-> travel\_time = service\_time + wait\_time.

2) For signals: (Yes this is part of state Variables)

->light state(i.e. green, yellow, red and corresponding left time)

-> We maintain a boolean list of state of each signal. So, size of list is 4 (4 signals). Value 1 for a signal implies it is green value 0 implies it is in red state.

3)For queues:

->each lane in each section is labeled as a queue;

->vehicles IDs;

->state of signal that is right before this queue;

->queue state: empty, or partially occupied or fully occupied. (Fully occupied meaning that the length of the queue is limited by the length of the segment, so if the length of the segment is 200 m and the length of each vehicle is 10 m (7 m length + 3 m gap between each vehicle). So, the max length of this queue = 200/10 i.e. 20.

For each individual vehicle, the process can be summarized as below. If this vehicle firstly enter the zone of interest, the vehicle should wait until the queue it enters is not fully occupied. If the queue is empty, then the vehicle would be the first vehicle in this queue. If queue is only partially occupied, then the last vehicle in the queue is the latest vehicle which just arrived. A Future Event List (arrival times of each vehicle) is created for both the directions (North and South Bound). Once the time exceeds the simulation time then we perform the processing of vehicles, hence Process-oriented simulation. The new vehicle is added onto the queue, and the vehicle updates its position and its preceding vehicle. Check if queue is full or not and change its state if necessary. Then if the vehicle has already been in the zone of interest, then we check which queue this vehicle is in. In the next step, check the state of the signal right before the queue. If it is red, the vehicle waits until the signal turns green and the state of the vehicle should be ‘ stopped’ and we schedule an event in the future for the vehicle when the signal turns green/yellow. If ‘signal is green/yellow’ is true, wait until the state of preceding vehicle is ‘moving’. Then the state of vehicle can be changed to ‘moving’ and position is updated with time. Check the distance and signal timing if the left time of green/yellow signal is enough for the vehicle to reach the front of the queue. If this is true, delete the preceding vehicle and check the destination of the vehicle. If destination is reached, then delete this vehicle from the system and save the total time it has taken to reach the other end/exit. This is the travel time for this vehicle. If not, implement the same procedure as if the vehicle firstly enter the next queue. If the next queue is not full, then delete the vehicle from current queue and add it to the next queue. But be noted that if the next queue is full, then the vehicle state will change from ‘moving’ to ‘stopped’. After there is empty space in the next queue, the state will change to ‘moving’ in spite of the signal state for the next queue. After become one part of the next queue, repeat the procedures as before. Continue this process for all the vehicles while the current time in the simulation reaches the simulation time.

**2.5 Output**

The output would be a distribution of reduced travel time for all the vehicles during the simulation. The reduced travel time is defined as the travel time over travel distance.

**3. Simulation Process**

In this project, we will build simulation program based on the Process-oriented world view. Therefore we need to identify the behavior of individual vehicles and define components which interact with vehicle during simulation process. Graphical modeling language such as SysML(Systems Modeling Language) allows a better understanding of our simulation model[4]. Hence we draw block definition diagram and sequence diagram in SysML to represent the behavior and components of the model.

Fig. 2 shows block definition diagram of a components of the simulation program (Note: Each block in Fig.2 is not necessarily implemented as a class in source code). Each rectangle(block) represents a component of the simulation model. Value properties and operations of components are shown in compartments of the block. Arrows with filled diamond head express composition a whole-part relationship and the multiplicity of the part end is illustrated near the arrow heads. As shown, our simulation model is composed of a number of vehicles, five roads, four intersections and one event scheduler.

**3.1 Vehicle block**

Vehicle block represents a vehicle which pass through the zone of interest. The block has value properties such as vehicleID, entryPoint, exitPoint, velocity, length and arrivalTime. Each vehicle is distinguished by unique vehicleID. A location where the vehicles enter and exit the zone of interest is expressed as entryPoint and exitPoint, and passway of the vehicle can be obtained from them. Time intervals during which the vehicles arrive at and leave each intersection are stored in timeStamp value property.

**3.2 Road block**

Road block represents roads between intersections or between intersection and entry/exit point. As noted in model assumptions and simplifications part, velocity of a vehicle is constant throughout its travel, however it is different for different vehicles and is drawn randomly given some range of velocity. This means that traveling time of a vehicle on the roads/lane from one intersection to next intersection can be derived as a division of segmentLength by vehicle velocity (it is the time on road between two intersections hence does not include the waiting time at signal and queue). The method named getTravelTime calculate the total traveling time for a vehicle given its arrival time, direction, start and exit location and the global state variables (signalStates, Queues)

**3.3 Intersection block**

Each intersection has Northbound and Southbound TrafficLane and TrafficLight blocks.

**3.3.1 TrafficLane block**

TrafficLane block has listOfWaitingVehicle property to which vehicles waiting in the traffic lane are added. While a vehicle pass through an intersection, stateOfTrafficLane value property comes into occupied state. After the vehicle leaves the intersection, stateOfTrafficLane gets to empty state. The operation named getTimeToPass returns time interval that the vehicle pass through the intersection with considering the preceding vehicles.

**3.3.2 TrafficLight block**

TrafficLight block has signal\_color value property which represents color of traffic light (it is a boolean and 1 represents green/yellow while 0 represents red). Time interval of green and red light is defined by cycle\_timing\_list value property. The operation named getTimeToGreen returns time interval during which the vehicle of interest has to wait for green signal.

**3.4 Event handler**

Event handler manages events of vehicle instances. Fig. 3 shows a sequence diagram which illustrates a part of the simulation process. In Fig. 3, instances of the blocks defined in Fig. 3 are illustrated as rectangles and interactions among them are represented by solid or dashed arrows. Time flows from top to bottom of the diagram.

First, the event handler creates an instance of the vehicle(createVehicleInstance). Second, the vehicle instance calls getTimeToGreen and getTimeToPass operation to obtain time intervals during which the vehicle waits for a green signal and all preceding vehicles pass through the intersection. The vehicle instance also calls getTravelTime operation to get time to travel one intersection to the next one. These operations are repeatedly called until the vehicle leaves the zone of interest. After the vehicle leaves the zone, it calls getTravelTime operation to record the obtained time intervals.

Event Handler pops the entities/vehicles from the Future Event List and decides, when next vehicle enters the zone of interest and calls advanceTime operation to advance the simulation time by the time obtained in the previous operation. Then it creates another instance of vehicle(vehicle2) and continues the same processes until the simulation time reaches a finishing time.

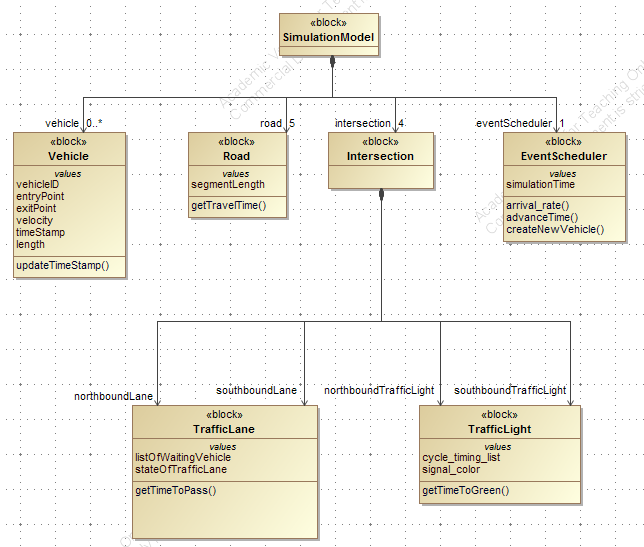


Fig. 2 Block Definition Diagram of the simulation model.

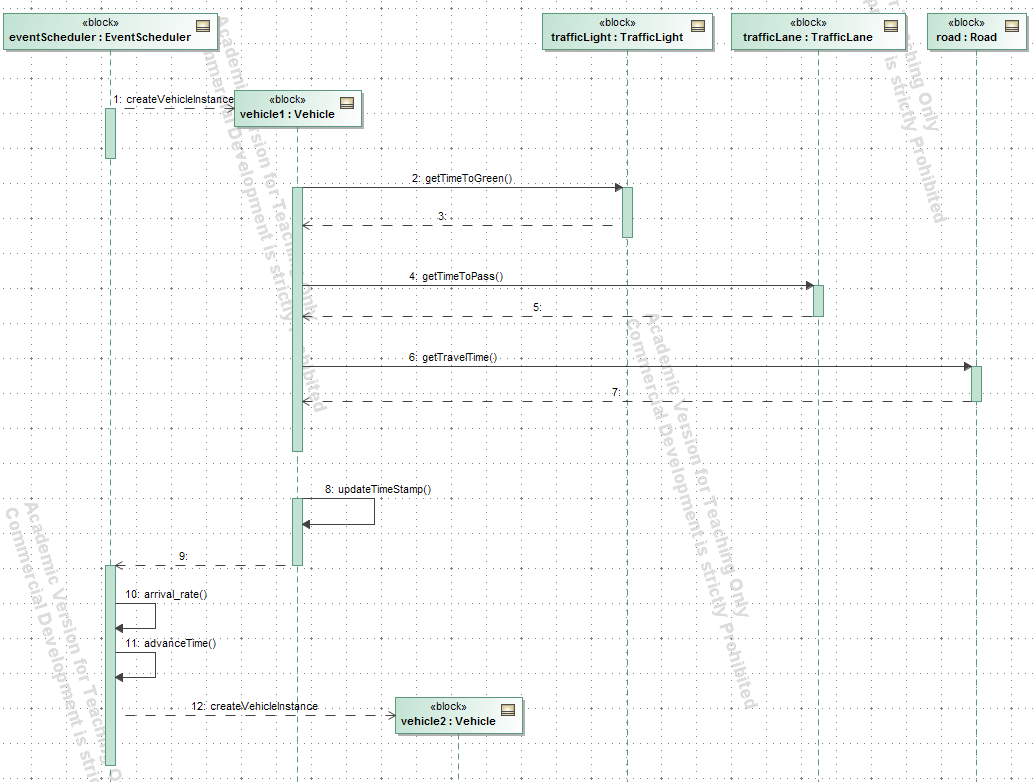


Fig.3 Sequence Diagram of the simulation model.

**4 Simulation Code**

**4.1 Tools**

Python is being used for modelling the entire Process-oriented simulation model. Till now the libraries/default function being used for simulation are: “random()”, for generating random vehicle velocity and “random.expovariate()” to generate the arrivals of vehicles.

**4.2 Code Overview**

The inter-arrival rate of 1 veh/s is assumed for now and based on Exponential distribution, arrivals are captured. A Future Event List is created. Currently, as soon as a vehicle arrives it is being sent to the Event handler, which is working parallely and serving the vehicles and the travel time for each vehicle is being captured. In the future and final implementation, a future event list will be created first and then the Event Handlers will process the threads completing the simulation model. Also, it is clearly visible that the Event Handler (TrafficSim method), completes one process and stores the travel times for all the vehicles. Once, the simulation time is reached then we finally calculate the mean of travel time, service time and waiting time.

Currently, we have modeled only for one direction from North to South. Implementation for both sides is yet to be done. Once, we incorporate all the important aspects of the simulation then we will extend it to both the directions and multiple lanes. Furthermore, we are not using the NGSIM data. Also, currently the units considered in the model are all SI units.

Assumptions: Length of the segments are defined in the segmentList

**5 References**

1. Feng, Y., Hourdos, J., Davis, G., Collins, M., “Vehicle Probe Based Real-Time Traffic Monitoring on Urban Roadway Networks”. Minnesota Traffic Observatory, Department of Civil Engineering University of Minnesota
2. Chandrasekaran, B., “Survey of Network Traffic Models”
3. Camps, F., Thibault, E., Harasse, S., “Statistical Distribution of Traffic Sources in Network Simulation Tools”. Proceedings of the World Congress on Engineering and Computer Science 2008 WCECS 2008, October 22 - 24, 2008, San Francisco, USA
4. S. Friedenthal , A. Moore and R. Steiner, “A Practical Guide to SysML, Second Edition: The Systems Modeling Language.” , Morgan Kaufmann/OMG, Elsevier