Technical University of Cluj-Napoca

Automation and Computer Science

Year IV, Semester I

Distributed Control Systems Project

Traffic Intersection Controller

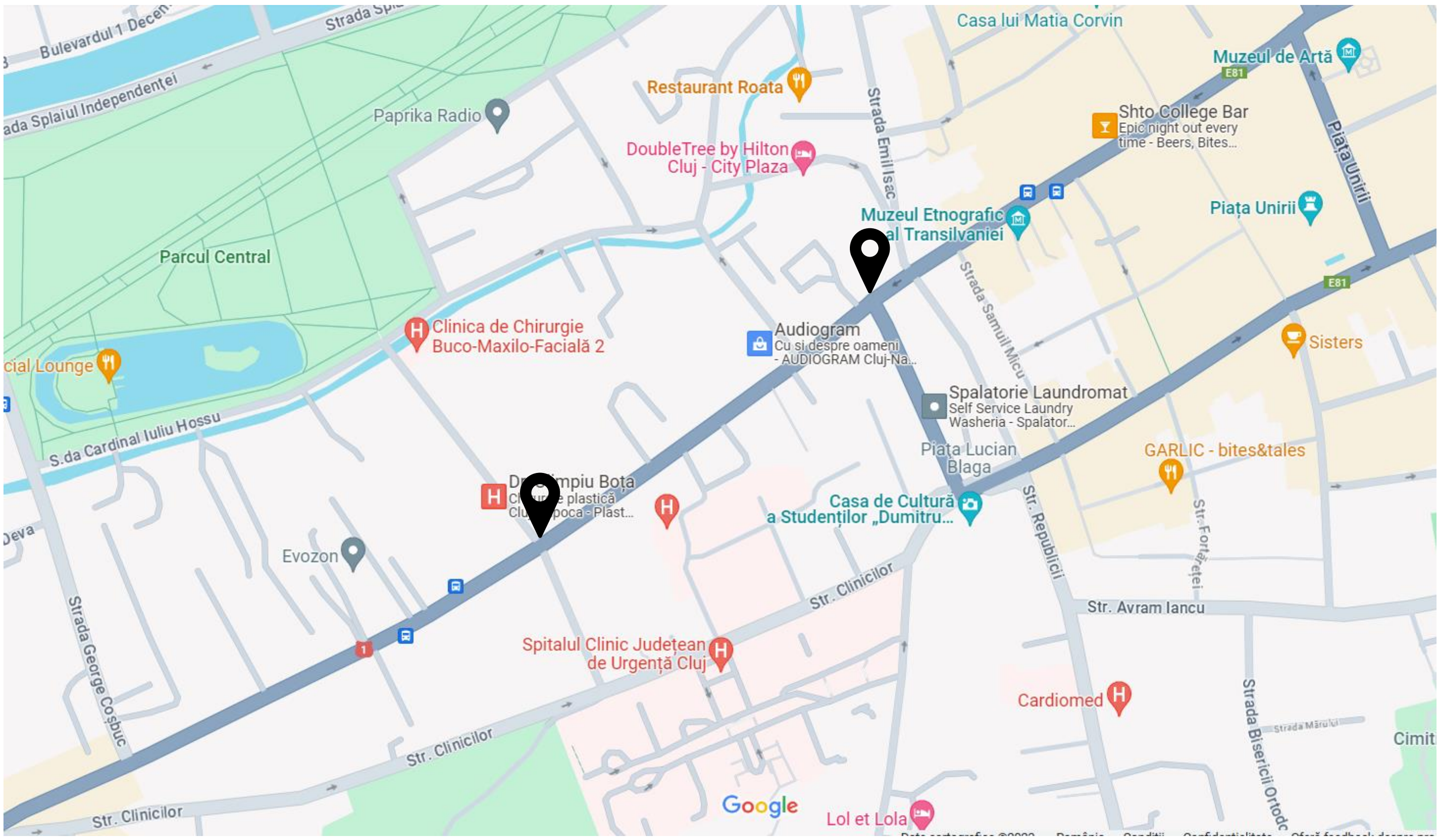
Team: Nagy Timea, Nemes Raluca, Oprea Florin-Octavian

Group: 30343

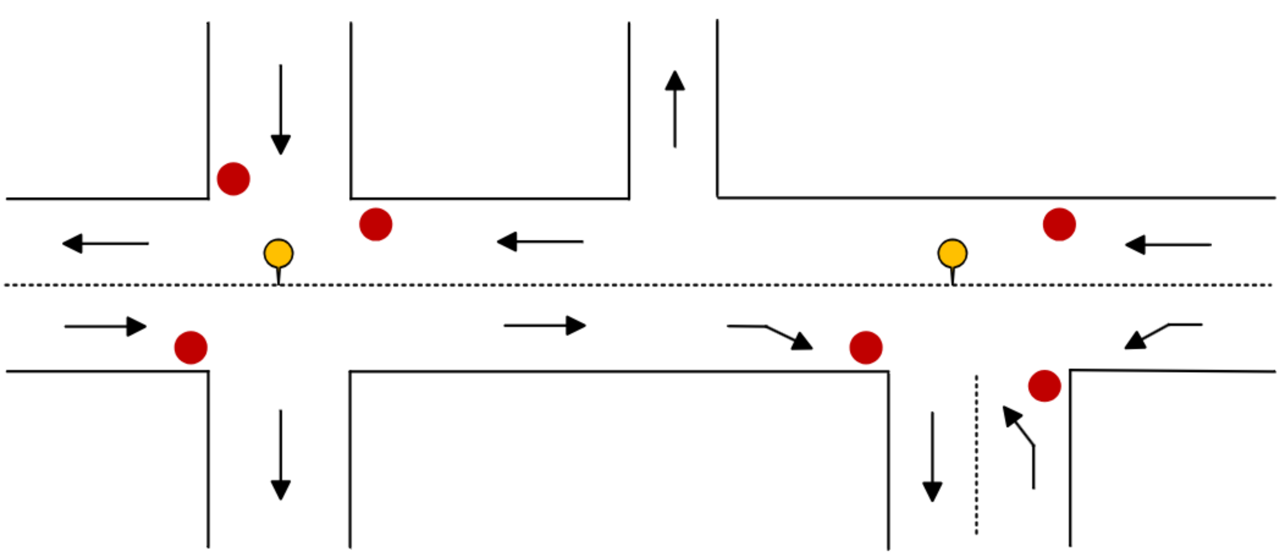
Specifications

According to the map given to each team, develop a controller for each intersection (plant), that controller is a closed-loop one (with the $in_{(1..n)}$ input channels that is connected to its intersection's output channels $op_{(1..n)}$ and an Intersections (with the OPs output channels). The controller must have dynamic delays feature to extend the time of the green light in case of a traffic jam.

The intersections



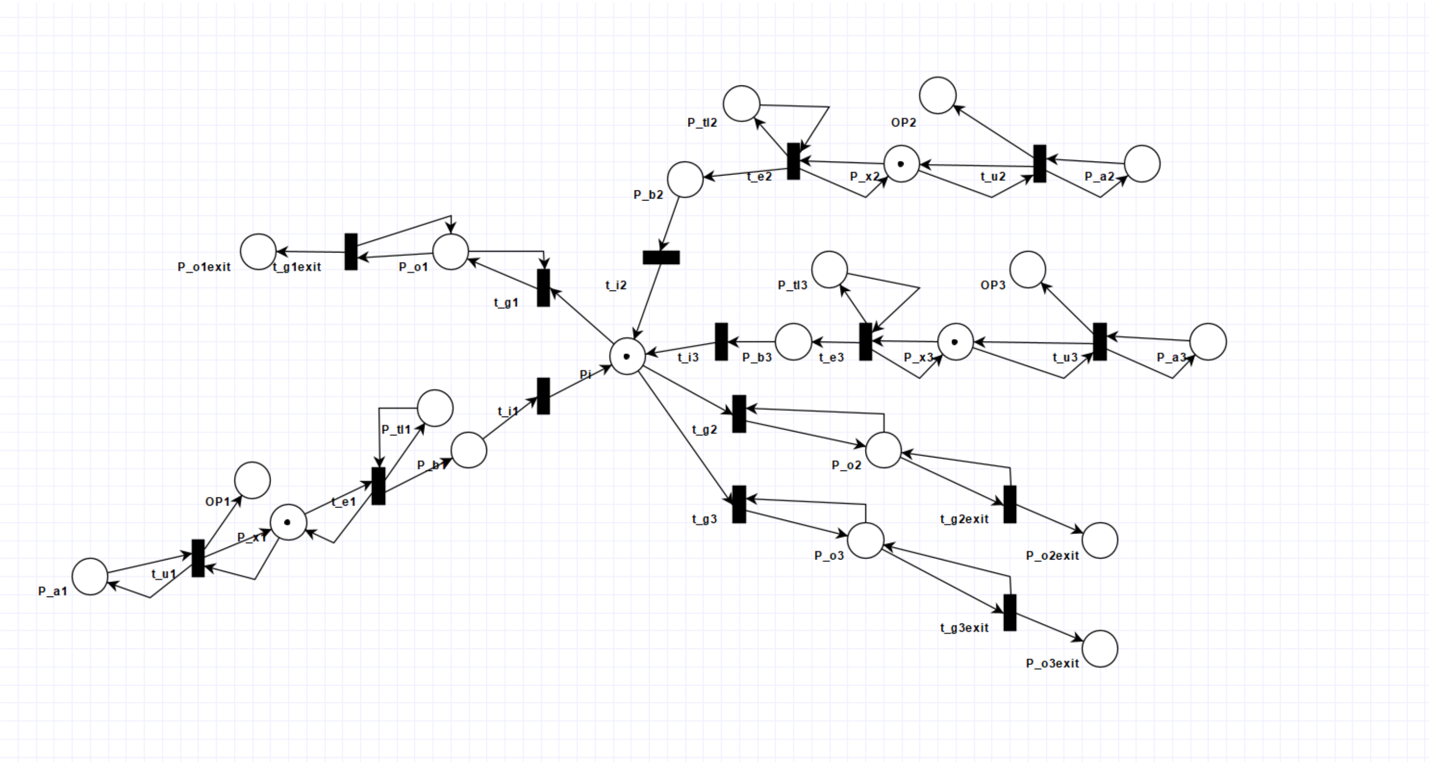
Simplified Model



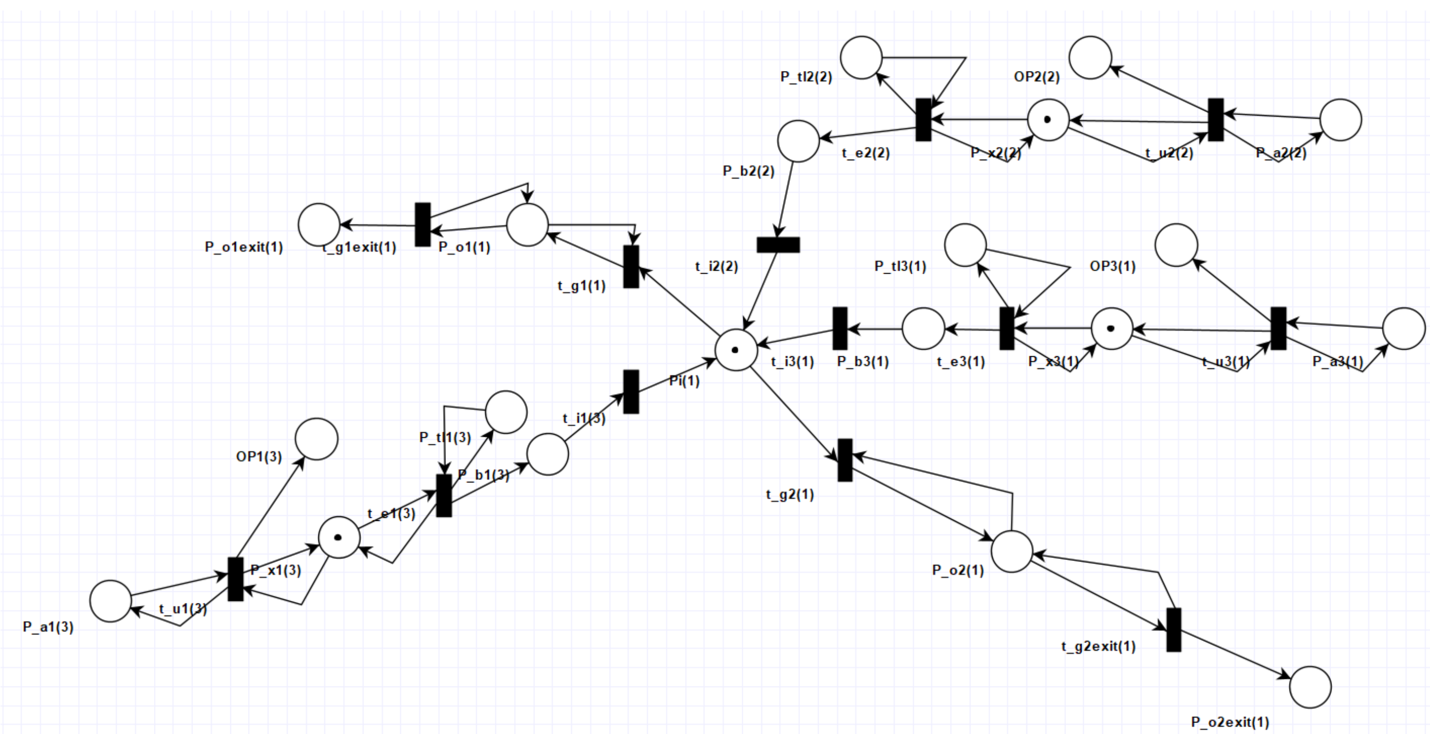
Design

OETPN Model for Plant

First Intersection



Second Intersection



Guards & Mappings for Plant

First Intersection – for the second intersection is the same but with one more input lane.

Input lanes: → P_a1 ; P_b1 ; P_a2 ; P_b2 ; P_a3,
→ P_b3 ; → Data Car
→ P_x1 ; P_x2 ; P_x3 → Data CarQueue
→ P_te1 ; P_te2 ; P_te3 → Data String
→ OP1 ; OP2 ; OP3 → Data Transfer

Output lanes: → Po1 , P-o2 → Data CarQueue
→ P-o1exit , P-o2 exit → Data Car
→ P_o2 Data Transfer
→ P_i Data CarQueue

GRD & MAP

t_u1: P_a1 != NULL && P_x1 add car
P_x1.add(P_a1)
P_a1 != NULL && P_x1 can not add cars
O_P1.send("Full")

! Same for t_u2 & t_u3

t_e1: (P_x1 haveCar && P_te1 == "green")
P_x1.popElementWithoutTarget(p_b1)
P_te1 = P_te1

! Same for t_e2 & t_e3

t_i1 P_b1 !=NULL && P_i can add cars
P_i.addElement(P_b1)

! Same: t_i2 & t_i3

t_g1: P_i.HaveCarToTle && P_o1 CanAddCars
P_i.PopElementwithTargetToQueue(P_o1)

! Same t_g3

t_out: P_o3Exit != NULL
P_o3Exit.Send
OverNetwor(Po3)

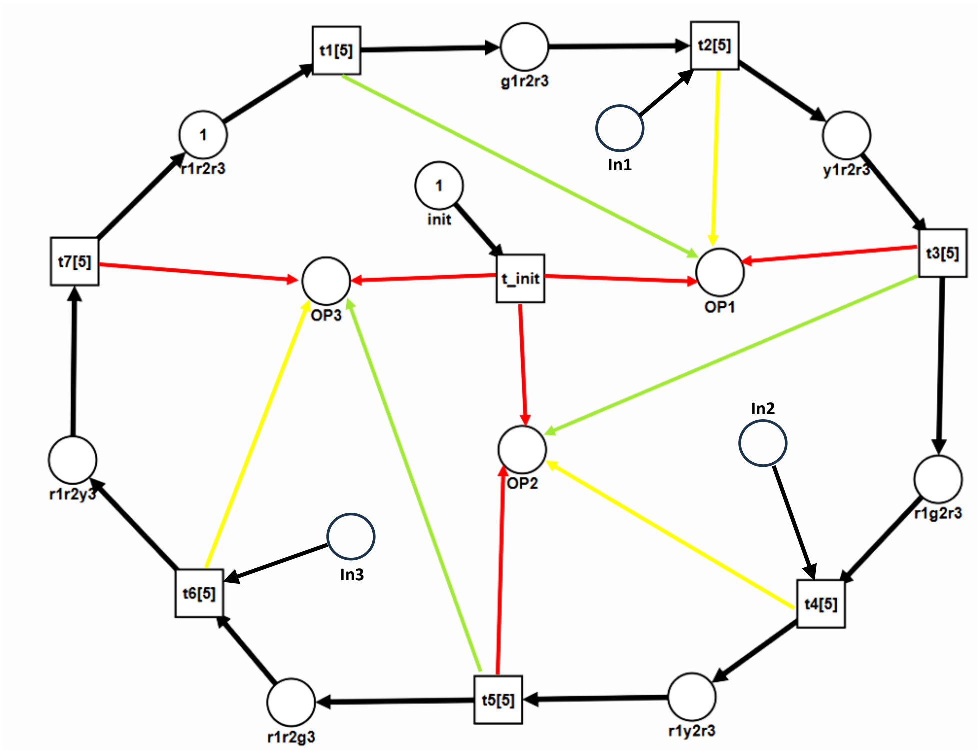
t_g1 exit P-o1.HaveCar
P-o1.PopElement(P_o1Exit)

me t_g3exit

OETPN Model for Controllers

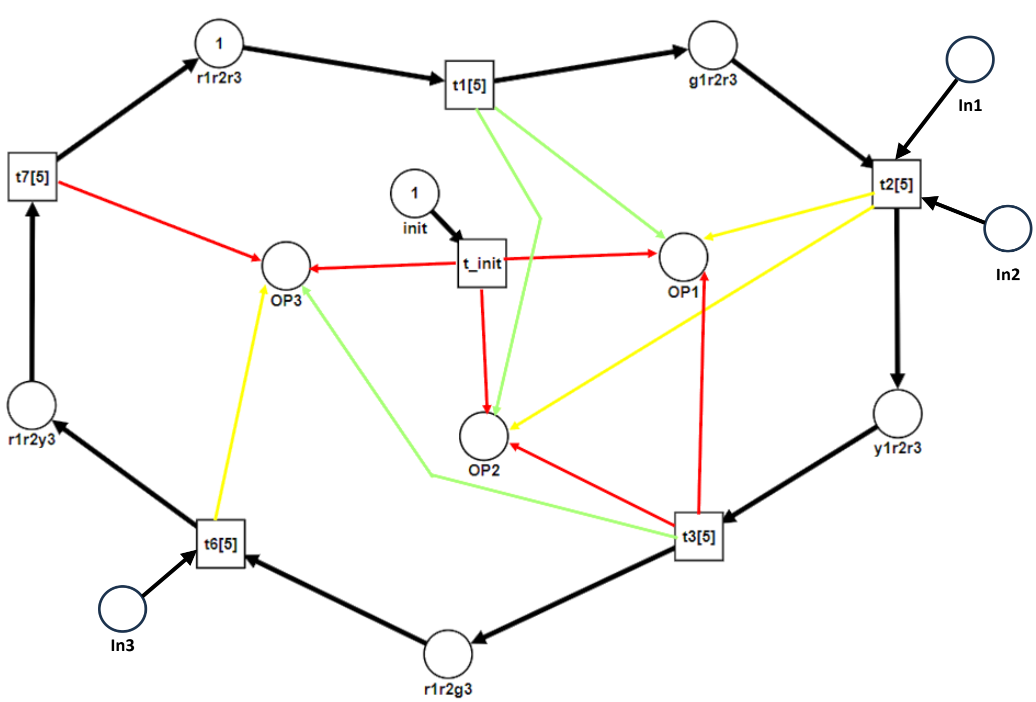
First Intersection

- The 3 traffic lights are green one at a time.



Second Intersection

- OP3 traffic light is green: the other 2 must be red.
- OP3 TL is red: the other 2 can be green simultaneously.



Guards & Mappings for Controllers

PLACE TYPES

in1 : DataString   r1r2r3, g1r2r3, y1r2r3,
in2 : DataString   r1g2r3, r1y2r3, r1r2g3,
in3 : DataString   r1r2y3, : DataString
OP1, OP2, OP3   DataTransfer
t_init   DataString

Guards & Mappings

t_ini : {
init != 0
OP1. SendOverNetwork (init)
OP2. SendOverNetwork (init)
OP3. SendOverNetwork (init)
init . MakeNull

t1 : {
r1r2r3 != 0
OP1. SendOverNetwork ("green")
g1r2r3 = r1r2r3

t2 : {
g1r2r3 != 0 && in1 == 0
OP1. SendOverNetwork ("yellow")
y1r2r3 = g1r2r3
DynamicDelay ("five")

{
g1r2r3 != 0 && in1 != 0
OP1. SendOverNetwork ("yellow")
y1r2r3 = g1r2r3
DynamicDelay ("ten")

t3 : {
y1r2r3 != 0
r1g2r3 = y1r2r3
OP1. SendOverNetwork ("red")
OP2. SendOverNetwork ("green")

t4 : {
r1g2r3 != 0 && in2 == 0
OP2. SendOverNetwork ("yellow")
r1y2r3 = r1g2r3
Dynamic Delay ("five")

t4 : {
r1g2r3 != 0 && in2 != 0
OP2. SendOverNetwork ("yellow")
r1y2r3 = r1g2r3
DynamicDelay ("ten")

t5 : {
r1y2r3 != 0
r1r2g3 = r1y2r3
OP2. SendOverNetwork ("red")
OP3. SendOverNetwork ("green")

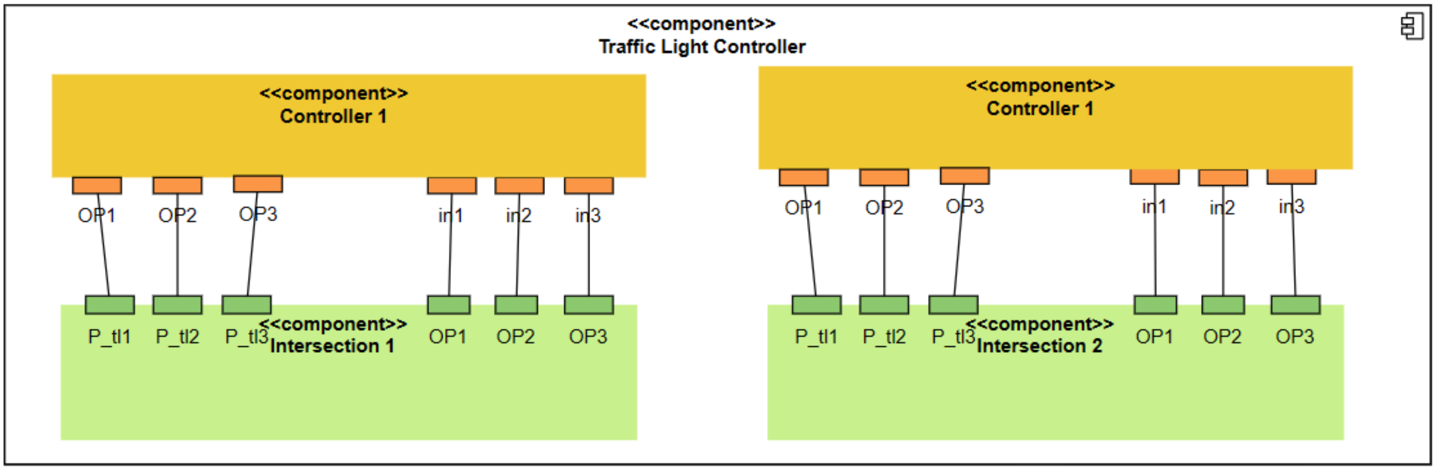
t6 : {
r1r2g3 != 0 && in3 == 0
OP3. Send Over Network ("yellow")
r1r2y3 = r1r2g3
Dynamic Delay ("five")

t6 : {
r1r2g3 != 0 && in3 != 0
OP3. Send OverNetwork ("yellow")
r1r2y3 = r1r2g3
Dynamic Delay ("ten")

t7 : {
r1r2y3 != 0
OP3. Send OverNetwork ("red")
r1r2r3 = r1r2y3

The same place types, guards and mappings apply for controller 2 too :)

Component Diagram



Implementation

The entire system (plants + controllers) is implemented in Java using OERTPN Framework. The repository can be found on GitHub:

`github.com/NTimea302/Traffic_Intersection_Controller`

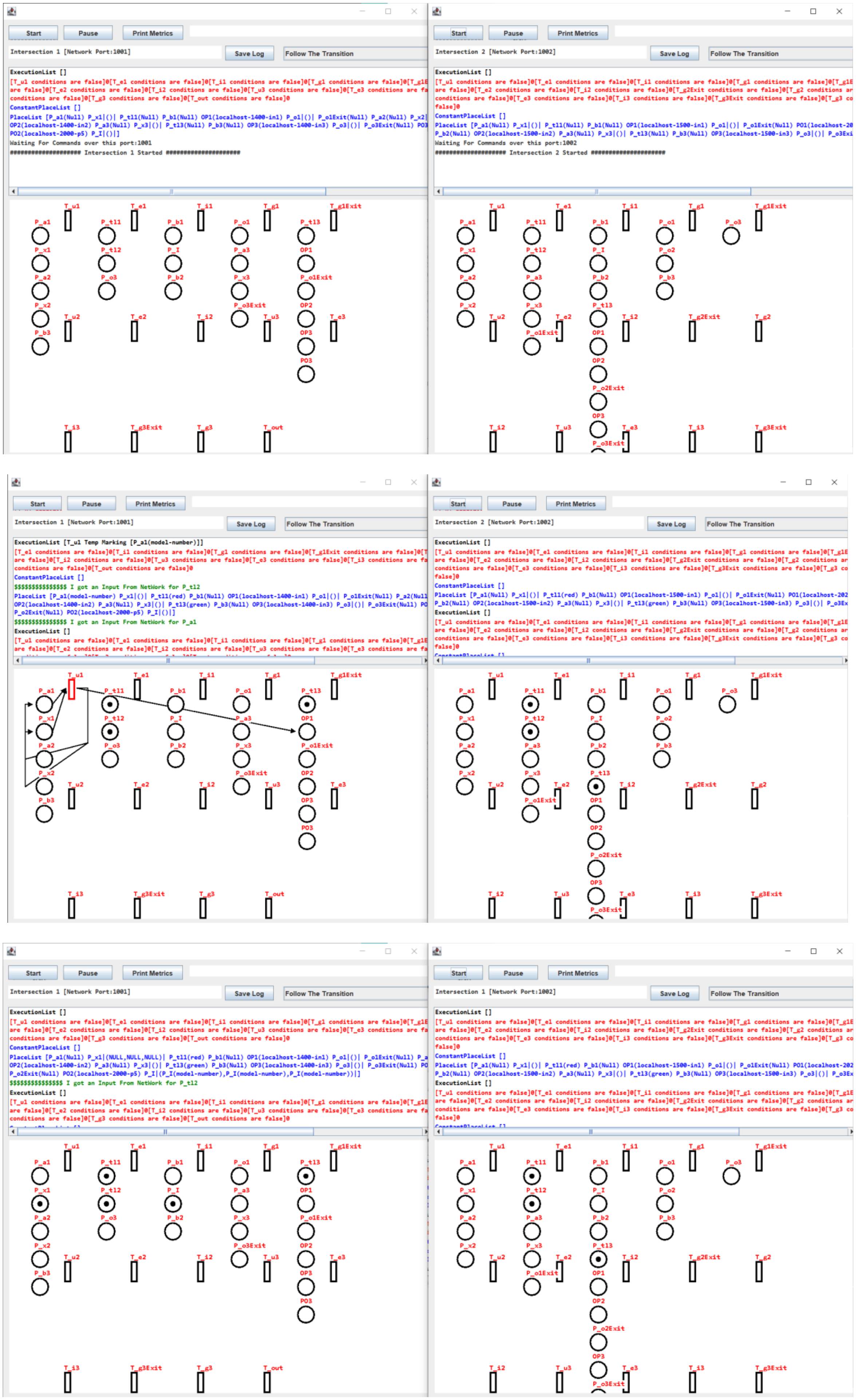
Testing

We tested the application for the following use cases:

1. Send a car from the 1st intersection, that should go through the middle street and exit from one of the exit lanes from the 2nd intersection.

2. Traffic jam: for each intersection, create a traffic jam case by sending the maximum number of cars to the input lane of the intersection, start the controller, then send the last car. The controller should receive a signal from the plant (intersection) and the transition that is responsible for sending a yellow light to that lane where you input the cars to, should have changed the delay to 10 sec. Let the controller OETPN run until it reaches the same transition (2 loops) to show that the delay is changed back to 5 sec.

Screen shots from the execution


Start
Pause
Print Metrics
Intersection 1 [Network Port:1001]
Save Log
Follow The Transition
ExecutionList []
ConstantPlaceList []
Waiting For Commands over this port:1001
#################### Intersection 1 Started ####################
Intersection 2 [Network Port:1002]
Waiting For Commands over this port:1002
#################### Intersection 2 Started ####################
ExecutionList [T_u1 Temp Marking [P_a1(model-number)]]
$$$$$$$$$$$$$$$ I got an Input From Network for P_t12
$$$$$$$$$$$$$$$ I got an Input From Network for P_a1
T_u1
T_e1
T_i1
T_g1
T_g1Exit
P_a1
P_t11
P_b1
P_o1
P_t13
P_x1
P_t12
P_I
P_a3
OP1
P_a2
P_o3
P_b2
P_x3
P_o1Exit
P_x2
T_u2
T_e2
T_i2
P_o3Exit
T_u3
OP2
T_e3
P_b3
OP3
PO3
T_i3
T_g3Exit
T_g3
T_out
P_o2
P_b3
T_g2Exit
T_g2
P_o2Exit
T_e2
T_g3Exit