



MIDDLE EAST TECHNICAL UNIVERSITY

NORTHERN CYPRUS CAMPUS

Computer Engineering Department

CNG 1531
DISTRIBUTED INTERACTIVE
SIMULATION

PROJECT PROGRESS REPORT

Traffic Monitoring System

Waqar Haider 2409365

Date of Submission: November 18, 2019

Abstract

Modeling and Simulation of a Traffic Management System

Waqar Haider

2019, 36 Pages

The Proposed system will simulate 4-way traffic. Traffic Control signals installed at each intersection controls the traffic flow. The green light allows traffic to proceed. The yellow light warns that the signal is about to change to red. The red signal prohibits any traffic from proceeding. At one time only 1 road will be open to traffic. And at the time of emergencies any 1 road can be open. The reason for the simulation is to get the best timing results of each light duration during rush hours and automatically change the lights when some emergency occurs.

Contents

1	Introduction	4
2	4-Way Traffic Process Model	4
3	Federation Design	6
3.1	Federation Structure	6
3.2	Object Model	6
3.3	Publish/Subscribe Diagrams	7
4	Federate Design	8
4.1	TrafficLight Federate	9
4.1.1	Federate Architecture	9
4.1.2	Behavioral Specification	10
4.1.3	Federate Internal Behavior	10
4.2	Vehicle Federate	11
4.2.1	Federate Architecture	11
4.2.2	Behavioral Specification	12
4.2.3	Federate Internal Behavior	12
5	Federate Implementation	12
5.1	Traffic Light Federate	13
5.2	Vehicle Federate	14
6	References	15

List of Figures

1	Basic Lollipop Diagram of the System	4
2	Vehicle Process Model	5
3	Traffic Light Change Process	5
4	Federation structure for Traffic Monitoring System (TMS)	6
5	Object Classes	7
6	Interaction Classes	7
7	Vehicle Object Class Publish/Subscribe diagram	8
8	TrafficLight Object Class P/S diagram	8
9	LightMessage Interaction Class P/S diagram	8
10	VehicleMessage Interaction Class P/S diagram	8
11	layers in a federate application	9
12	TrafficLight Federate presentation, simulation and communica- tion layers	9
13	TrafficLight internal behaviour	11
14	Vehicle Federate presentation, simulation and communication layers	12
15	Vehicle Internal Model	13
16	trafficLightFed Directory Structure	33
17	Federation Creation	33
18	Federation Synchronization	34
19	Synchronization Point Announced	34
20	VehicleFed Directory Structure	35
21	VehicleFed Interaction Received Handler	36

1 Introduction

The basic idea of presenting a simulation of a 4-way traffic intersection is to see what the best time duration for each traffic signals is. Mostly we see that traffic lights in a 4-way intersection are manually set to a specific time duration for each side of the road regardless of the traffic scenario. Also, during emergencies like ambulance or police vehicles arrived at the time when the traffic light is still red, which does not allow for traffic passages. In that case, a traffic warden must manually set the traffic in way to give permission to that side of the road, which is not very reliable. Our idea is to set a simulation system, that can automatically set the signal to the desired one when needed. Like when an ambulance comes and road is blocked or about to block, the signal should be able to recognize when is it coming from and block other roads accordingly to allow the traffic flow on that road only. This will minimize the chance of waiting for patient or any other emergency. Secondly, we can also analyze the idea of how much time each of traffic light should be given. We can analyze the traffic on each road for a while and if there are no vehicles coming on one side of the road, there is no need to jam the signal for that side for the same amount as for the one with huge traffic.



Figure 1: Basic Lollipop Diagram of the System

2 4-Way Traffic Process Model

There are two scenarios in this simulation of 4-way traffic system. One according to a vehicle and second according to the traffic lights. From the prospective of a vehicle, say a Car, it can see the traffic light when it arrives at the intersection. It sees a red light, so it stops, and If a green light, it moves on. And according to the traffic light, generally it turns red after a specific amount of time, say 30 seconds. And then turn yellow and green again. But in our case, it will

be a little different. Each Traffic lights receive a signal from one of them and according to that signal or sign all the other traffic lights react.

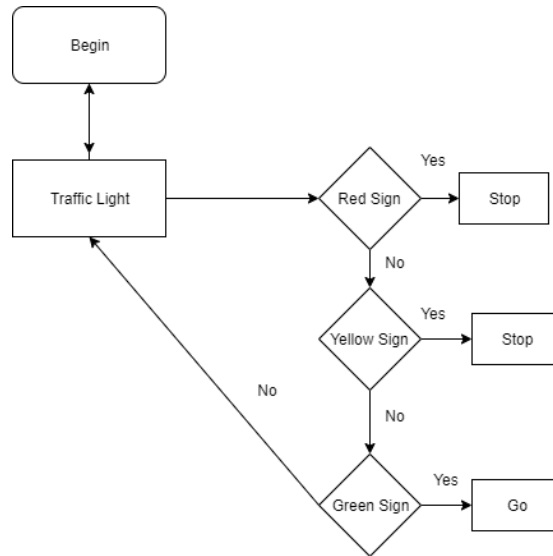


Figure 2: Vehicle Process Model

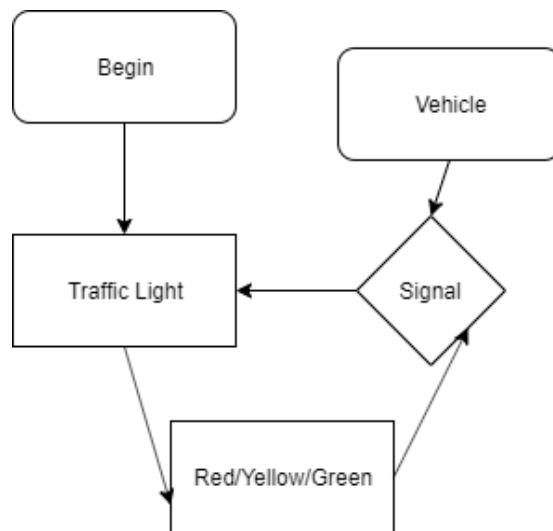


Figure 3: Traffic Light Change Process

3 Federation Design

An overall lollipop diagram of the federation can be seen in above figure 1. We discuss the details of the structure next.

3.1 Federation Structure

The structure of our federation is demonstrated in figure 4. Our federation execution consists of 2 Federate applications. The TrafficLightFed and VehicleFed. As we are simulating 4-way traffic on a road there can be 4 traffic lights at each end of the road so our federation execution will contain 4 TrafficLightFed. Each road can have 0 or many vehicles at the same time, so our VehicleFed has 0 or many relations, meaning our federation execution can contain 0 or many Vehicle federates at the same time.

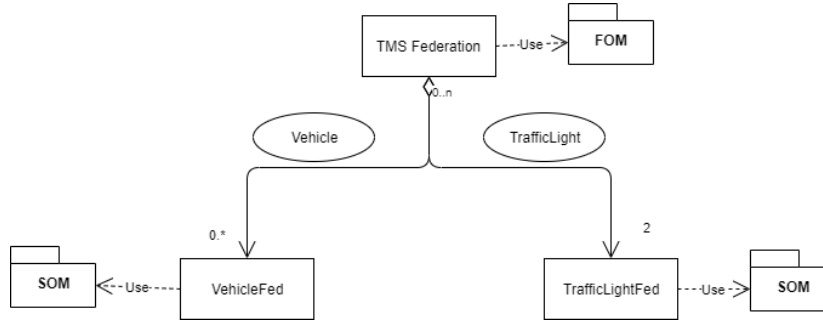


Figure 4: Federation structure for Traffic Monitoring System (TMS)

3.2 Object Model

The Federation Object Model for Traffic Monitoring System Consists of 2 Object Classes and 2 Interaction classes. The 2 Object classes namely TrafficLight Vehicle, are inherited from HLAObjectRoot. The Interaction classes are inherited from HLAInteractionRoot. They are LightMessage VehicleMessage.[1] The TrafficLight Object class has 3 attributes namely Position, Sign, Duration Vehicle also has 3 attributes of VehicleName, Status, Direction. The Interaction classes also has parameters. LightMessage has 3 parameters of Sign, Duration, DateTime while VehicleMessage has Type parameter only.

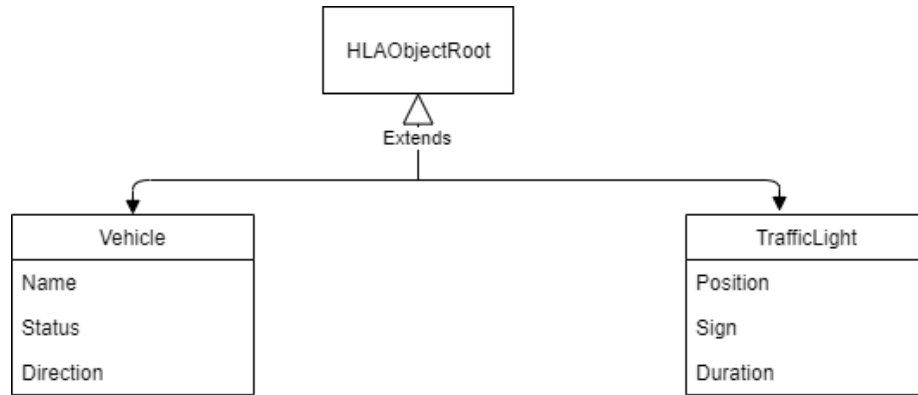


Figure 5: Object Classes

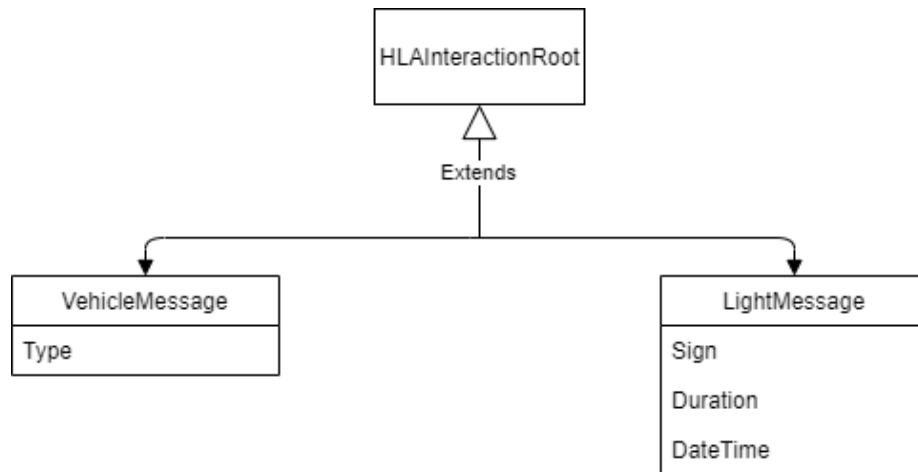


Figure 6: Interaction Classes

3.3 Publish/Subscribe Diagrams

The publish/subscribe diagram shows the basic communication structure of the simulation. It shows whether a federation can publish or subscribe an Object or Interaction. In our case, **VehicleFed** will both publish and subscribe to the **Vehicle** Object class. While **TrafficLight** will be subscribed to this Object. On the other hand **TrafficLightFed** will publish the **TrafficLight** Object and also subscribe to it. **VehicleFed** will just be a subscribed to this Object.

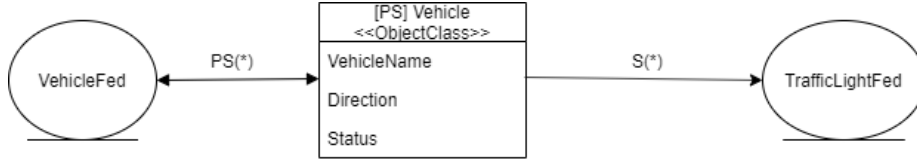


Figure 7: Vehicle Object Class Publish/Subscribe diagram

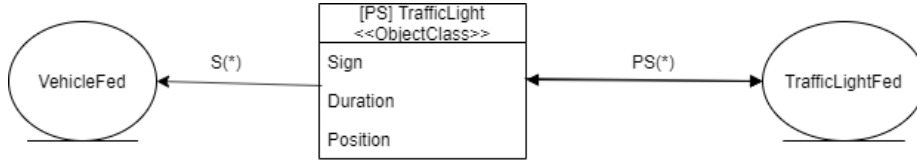


Figure 8: TrafficLight Object Class P/S diagram

Interaction Object classes are inherited From HlaInteraction. In our case both VehicleFed and TrafficLightFed are subscribed to LightMessage Interaction class, which will be published by TrafficLightFed Federation.

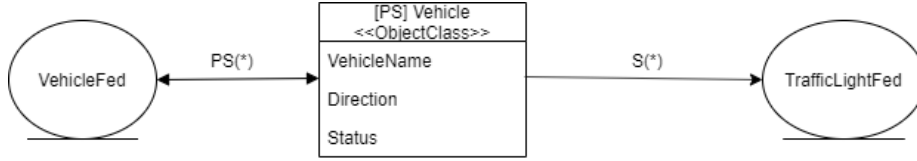


Figure 9: LightMessage Interaction Class P/S diagram

The VehicleMessage has only one parameter named Type. This Interaction is published by VehicleFed and TrafficLightFed is subscribed to that Interaction.

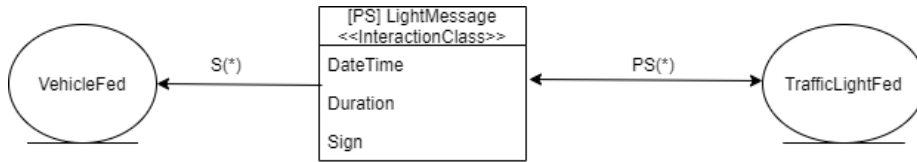


Figure 10: VehicleMessage Interaction Class P/S diagram

4 Federate Design

A Layered Architecture is the organization of the project structure into four main categories: presentation, application, domain, and infrastructure. Each of

the layers contains objects related to the particular concern it represents. We believe that the layered architectural style is a good choice for the design of a federate. Figure below is a simple representation of layered architecture.

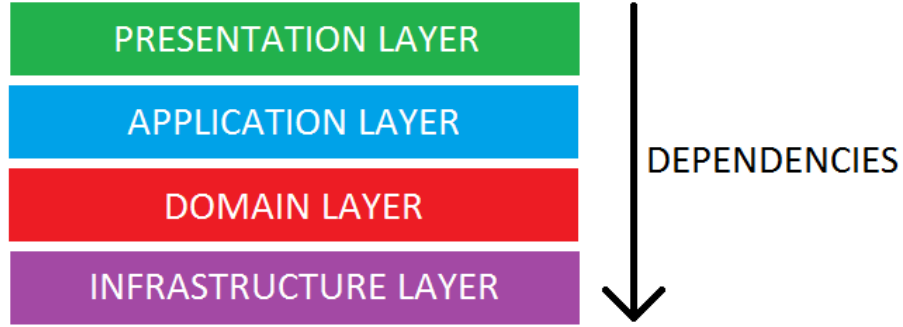


Figure 11: layers in a federate application

Our simulation is consisted of 2 federate applications, namely, TrafficLight Federate and Vehicle Federates. We discussed each of these federates in details below;

4.1 TrafficLight Federate

4.1.1 Federate Architecture

As we are using layered architecture, promoted in the book, bellow diagram shows the layers of our TrafficLight federate.

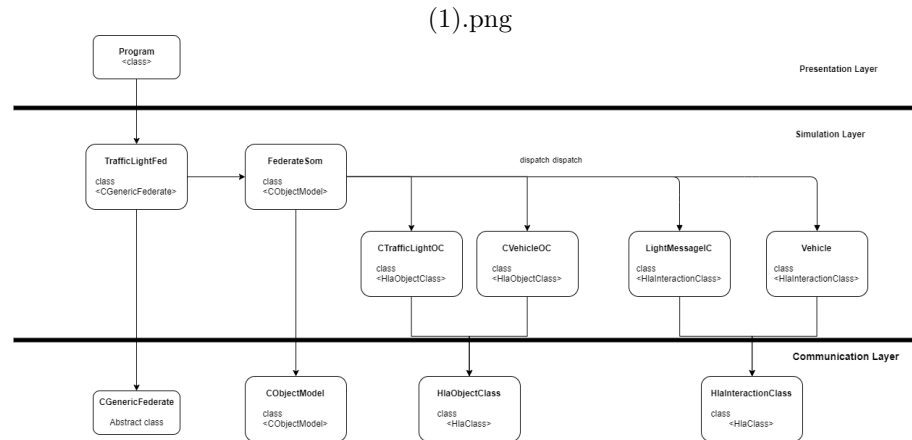


Figure 12: TrafficLight Federate presentation, simulation and communication layers

As it can be seen in the diagram that our federate is consisted of 3 layers. Our simulation is console application, therefore, the presentation layers is consisted of the main Program Class, responsible for displaying possible messages on the screen. Every interaction with the RTI and other object will be displayed as a console message.

The user interaction is also done using console screen. user can enter different types of vehicles and see the resulting behaviour of each Light on each intersection. The messages will be displayed of each action by the federate on the console screen.

Our simulation layer is consisted of TrafficLigthFed Class, which is responsible for communicating with RTI. all callbacks and response functions are implement in that object. FederateSOM contains all the HLA object and interaction classes. so Every class and it's related attribute mentioned in the FDD files are added to HLA Object Model in this class. It also act as an mediator to refer to the classes in the federate. We update the TrafficLight Object and its values are reflected in other federate applications. We also use Interaction class LightMessage, which sends interactions to the coming vehicles about the current sign of the light.

4.1.2 Behavioral Specification

Firstly, our Traffic Lights starts with either creating a new federation execution or joining a federate execution. The Federate is created using CreateFederationExecution method. If the federation execution is already created, then it will use the JoinFederationExecution to join the already created execution. The Starting Federate is responsible for counting the number of federates in the federation execution. After the certain number of federates are reached, 4 in our case, It will interact with the RTI and using RegisterFederationSynchronizationPoint() method to synchronize all the federate to same point. that is all the TrafficLight federates are of Red color at the start. After that our starting point of the federation is reached, that is, East federate will start its process by initializing the flow of turning its light sign to Green, This will updates the Object by AddAttributeValue() method. Other federates will reflect these changes and keep their lights as Red. After certain time (30 Seconds), The East Federate will send message, that it is about to change its sign to Yellow, Other federates will reflect the changes and changes the next TrafficLight, that is the West one, Will changes its sign to Yellow first, than it will send the updates to East Federates and East federate will change its sign to Red. East federate will update send a new message to West. This Interaction will change the West federate sign to Green. and this will continue.

4.1.3 Federate Internal Behavior

At the federate application level, internally the Traffic light has a very basic functionality of sending and receiving object updates. At the initial point, the all the federates are down and then at their start point they are all initially

Red. As the Synchronization completes, which the East Traffic Light initiates, it will turn Yellow and other will turn Red, when it receives other federates callbacks that they are turn Red they It will turn Green. At this point, the vehicle in this region will receive the Green sign as well and they will Go. but other will received Red sign and they will wait. After some time, in our case 30 sec, the East federate will send the updates of changing sign again, the other will received updates and change will occur clock wise other federate. The process will continue in circle.

There is a special case, where an ambulance arrive at the intersection, than it's responsible Federate will received the Interaction, send the updates to other federates and change the corresponding light to Green.

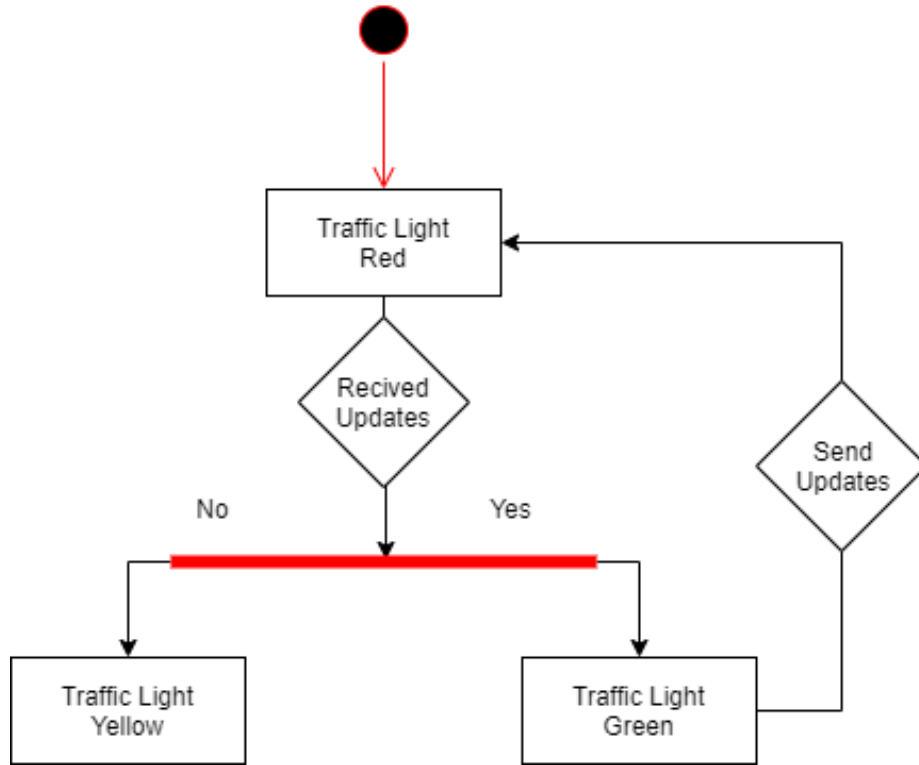


Figure 13: TrafficLight internal behaviour

4.2 Vehicle Federate

4.2.1 Federate Architecture

We don't update the the object of vehicle federate nor we send or receive object to that federate. But we send interaction of VehicleMessages to the TrafficLight

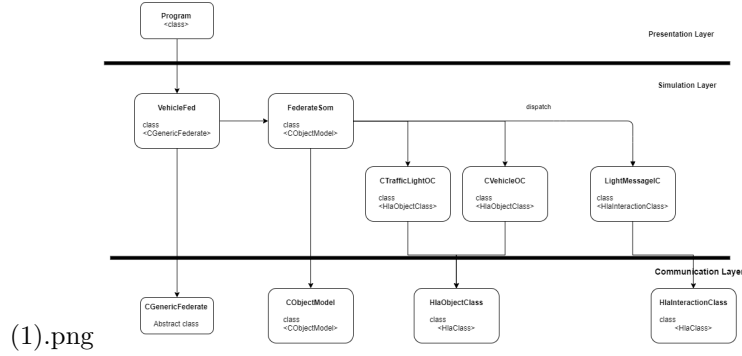


Figure 14: Vehicle Federate presentation, simulation and communication layers

Federate. So we use VehicleMessage for that. it will send the type of vehicle (say car), at its arrival intersection of the area.

4.2.2 Behavioral Specification

There is not much in the Vehicle federate as it only arrive at the intersection and move or wait according to the sign it reads. When this federate joins the execution, the first thing it does is that it send its position to the concerned federate. say East, than that federate's sign will decide if the Vehicle is good to go, or wait for the Green light. Its send Vehicle Message Interaction to the Lightfederates and Recieves Sign and duration Attribute value changes from the federates.

4.2.3 Federate Internal Behavior

Vehicle has its own internal architecture very simple one. It arrives at an intersection and inform it's corresponding TrafficLight via an Interaction. In return, it receives the current sign of its corresponding federate, and either Go or Wait according to the sign.

A vehicle sends its type, (Car, Ambulance, etc.) to its light, If there is a special case, in our case it's ambulance, it will be given special status in the trafficLight federates and all the other federates will change their signs to red. so that it can move as quickly as possible.

5 Federate Implementation

In this section we discuss how we implement our federation and federate applications. We used c for our project implementation. For the project implementation we used .Net Visual studio with a OpenRTI and RaCon library. As we discuss earlier, we have two federate applications namely VehicleFed, TrafficLigthFed.

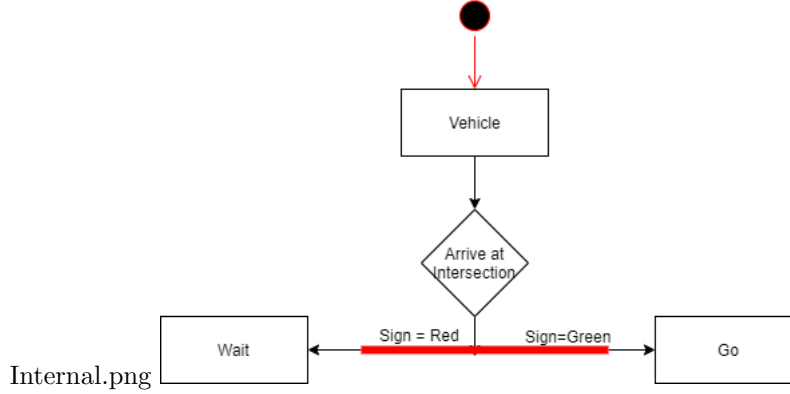


Figure 15: Vehicle Internal Model

5.1 Traffic Light Federate

The directory structure of the trafficLightFed can be seen in the figure 16. We have different that are related to our project. Main files that we will discuss are Program.cs, TrafficLightFed.cs and TrafficLight.cs and others. We will discuss a few of them. As other files are generated by SimGe tool.

The SOM FOM for our project are generated using SimGe tool the report of which can be seen in the last section. We will not discuss that part as it is pretty straight forward and due the helpful tool it is just filling the tables.

We start by creating a new federation execution. following code shows that we are either creating new federation execution or joining new execution. the code is shown below;

We either create a new federation or join an already existing one. line 50 and 56 is the call back functions to RTI using RaCon Library. We will have 4 federate execution from this federate application. each one will have its own name and process to follow. The functionality is very similar in different prospects that is why we kept them using 1 federates.

Next we run the application and using different code we start 4 federates. We use synchronization point at the start to make sure every federate is running and ready to start when required. our synchronization can be seen in the next figure 18

In the main simulation loop we call he synchronization point. line 105 in the figure is the calling function, with parameter of EveryLightUp. This method will be reflected as call backs in every federate and than using those call back we will inform RTI from other federates. When all the federates send the call back to the RTI and RTI in turns inform all the federates of each others status we will make every federate's Sign as Red and Duration as 0. 19

After synchronization we use Data distribution management to update different attributes and send Interactions between federates to share the light Sign and duration. East federates will create the federation execution and than

other will join. As soon as 4 federates are up the synchronization point will be achieved at that point East Federate will turn yellow. Than using Command Line interface we will use "Send" To turn its sign to Green and Yellow. After it turns "Yellow" The West federate will turn Yellow and East will turn Red. After that West will turn Green and so on.

5.2 Vehicle Federate

Vehicle federate has a very simple task in our federation execution. the directory sturcture of this federate can be seen in Figure 20. Similar to the previous federate application, we have some important files that we changes and use for our implemenation. Program.cs is used to join the federate execution. 20

After joining the vehicleFed will have its type of vehicle (Car, Bus, Ambulance...) and the direction it is coming from. If it is coming form East and it's type is Car the East TrafficLightFed will receives its update and then it will send its Interaction to the VehicleFed. The Interaction will contain Sign and Duration of TrafficLightFed. If the Vehicle receives Red sign it will stop the vehicle and if the Sign is Green it will go and leave the federation execution. The implementation can be seen in Figure 21

6 References

- [1] O. Topçu and H. Oğuztüzün, *Guide to Distributed Simulation with HLA*. Springer International Publishing, 2017.


Appendix

OMT REPORT



Generated by SimGe at 19/12/2019 07:58:25

Object Model Identification Table

Object Model Name	TMSFOM	Application Domain	HLA General
Object Model Type	FOM	Use Limitation	NA
Version	1.0	Use History	
Modification Date	11/5/2019 12:00:00 AM		
Security Classification	Unclassified		
Release Restriction			
Purpose	The purpose of this simulation is to monitor traffic	Description	The purpose of this simulation is to monitor traffic in a 4-way road. The traffic lights will generate the stoping and starting signs and vehicles will follow the lights to either stop or continue...
		Other	Created by SimGe at 11/5/2019 7:26:55 PM

Keywords

Taxonomy	Keyword Value
----------	---------------

References

Type	Identification
------	----------------

Generated by SimGe at 19/12/2019 07:58:25

1/17

Appendix

OMT REPORT



Generated by SimGe at 19/12/2019 07:58:25

POCs

Name SimGe
Organization oToT
Type Standards Sponsor

Phones

Type	Number	Email
Business	+1 (111) 111-1111	otot.support@outlook.com

Emails

Email
otot.support@outlook.com

Object Class Structure Table

Parent	Name	PS
	HLAobjectRoot	Neither
HLAobjectRoot	TrafficLight	PublishSubscribe
	Vehicle	PublishSubscribe

Interaction Class Structure Table

Generated by SimGe at 19/12/2019 07:58:25

2/17

Appendix

OMT REPORT



Generated by SimGe at 19/12/2019 07:58:25

Parent	Name	PS
	HLAinteractionRoot	Neither
HLAinteractionRoot	LightMessage	PublishSubscribe
	VehicleMessage	PublishSubscribe

Attribute Table

Object	Attribute	Data Type	Update Type	Update Condition	D/A	P/S	Available Dimensions	Transportation	Order
HLAobjectRoot	HLAprivilegeToDeleteObject	HLAToken	Static	NA	DivestAcquire	PublishSubscribe		HLAreliable	TimeStamp
TrafficLight	Duration	HLAinteger64Time	Static	NA	NoTransfer	PublishSubscribe		HLAreliable	Receive
	Sign	HLAASCIIstring	Static	NA	NoTransfer	PublishSubscribe		HLAreliable	Receive
	Position	HLAASCIIstring	Static	NA	NoTransfer	PublishSubscribe		HLAreliable	Receive
Vehicle	Direction	HLAASCIIstring	Static	NA	NoTransfer	PublishSubscribe		HLAreliable	Receive
	Status	HLAASCIIstring	Static	NA	NoTransfer	PublishSubscribe		HLAreliable	Receive
	VehicleName	HLAASCIIstring	Static	NA	NoTransfer	PublishSubscribe		HLAreliable	Receive

Parameter Table

Generated by SimGe at 19/12/2019 07:58:25

3/17

Appendix

OMT REPORT



Generated by SimGe at 19/12/2019 07:58:25

Interaction	Parameter	Data Type	Available Dimensions	Transportation	Ordering
LightMessage	DateTime	HLAInteger64Time		HLAreliable	Receive
	Duration	HLAInteger64Time		HLAreliable	Receive
	Sign	HLAASCIIString		HLAreliable	Receive
	Type	HLAASCIIString		HLAreliable	Receive

Dimension Table

Name	Data Type	Dimension Upper Bound	Normalization Function	Value When Unspecified
AreaOfResponsibility	AreaEnum	4	LinearEnumerated(Location, [West, East, North, South])	[0..4)

Time Representation Table

Category	Data Type	Semantics
timeStamp	HLAInteger64Time	
lookahead	HLAInteger64Time	

User-Supplied Tag Table

Category	Data Type	Semantics
updateReflectTag		NA
sendReceiveTag		NA
deleteRemoveTag		NA

Generated by SimGe at 19/12/2019 07:58:25

4/17

Appendix

OMT REPORT



Generated by SimGe at 19/12/2019 07:58:25

divestitureRequestTag		NA
divestitureCompletionTag		NA
acquisitionRequestTag		NA
requestUpdateTag		NA

Synchronization Table

Label	Tag Data Type	Capability	Semantics
EveryLightUp	HLAASCIIstring		

Transportation Type Table

Name	Reliability	Semantics
HLAreliable	Reliable	Provide reliable delivery of data in the sense that TCP/IP delivers its data reliably
HLAbestEffort	Best Effort	Make an effort to deliver data in the sense that UDP provides best-effort delivery

Update Rate Table

Name	Maximum Update Rate

Switches Table

Switch	Setting
Auto Provide	False

Generated by SimGe at 19/12/2019 07:58:25

5/17

Appendix

OMT REPORT



Generated by SimGe at 19/12/2019 07:58:25

Convey Region Designator Sets	False
Convey Producing Federate	False
Attribute Scope Advisory	False
Attribute Relevance Advisory	False
Object Class Relevance Advisory	False
Interaction Relevance Advisory	False
Service Reporting	False
Exception Reporting	False
Delay Subscription Evaluation	False
Automatic Resign Action	NoAction

Data Type Tables

Basic Data RepresentationTable

Name	Size	Interpretation	Endian	Encoding
HLAinteger16BE	16	Integer in the range $[-2^{15}, 2^{15} - 1]$	Big	16-bit two's complement signed integer. The most significant bit contains the sign.
HLAinteger32BE	32	Integer in the range $[-2^{31}, 2^{31} - 1]$	Big	32-bit two's complement signed integer. The most significant bit contains the sign.
HLAinteger64BE	64	Integer in the range $[-2^{63}, 2^{63} - 1]$	Big	64-bit two's complement signed integer first. The most significant bit contains the sign.
HLAfloat32BE	32	Single-precision floating point number	Big	32-bit IEEE normalized single-precision format. See IEEE Std 754-1985

Generated by SimGe at 19/12/2019 07:58:25

6/17

Appendix

OMT REPORT



Generated by SimGe at 19/12/2019 07:58:25

HLAfloat64BE	64	Double-precision floating point number	Big	64-bit IEEE normalized double-precision format. See IEEE Std 754-1985
HLAoctetPairBE	16	16-bit value	Big	Assumed to be portable among devices.
HLAinteger16LE	16	Integer in the range $[-2^{15}, 2^{15} - 1]$	Little	16-bit two's complement signed integer. The most significant bit contains the sign.
HLAinteger32LE	32	Integer in the range $[-2^{31}, 2^{31} - 1]$	Little	32-bit two's complement signed integer. The most significant bit contains the sign.
HLAinteger64LE	64	Integer in the range $[-2^{63}, 2^{63} - 1]$	Little	64-bit two's complement signed integer first. The most significant bit contains the sign.
HLAfloat32LE	32	Single-precision floating point number	Little	32-bit IEEE normalized single-precision format. See IEEE Std 754-1985
HLAfloat64LE	64	Double-precision floating point number	Little	64-bit IEEE normalized double-precision format. See IEEE Std 754-1985
HLAoctetPairLE	16	16-bit value	Little	Assumed to be portable among hardware devices.
HLAoctet	8	8-bit value	Big	Assumed to be portable among hardware devices.

Simple Data Type Table

Name	Representation	Units	Resolution	Accuracy	Semantics
HLAASCIIchar	HLAoctet	NA	NA	NA	Standard ASCII character (see ANSI Std x3.4-1986)
HLAunicodeChar	HLAoctetPairBE	NA	NA	NA	Unicode UTF-16 character (see The Unicode Standard, Version 3.0)
HLAbyte	HLAoctet	NA	NA	NA	Uninterpreted 8-bit byte
HLAinteger64Time	HLAinteger64BE	NA	1	NA	Standardized 64 bit integer time

Generated by SimGe at 19/12/2019 07:58:25

7/17

Appendix

OMT REPORT



Generated by SimGe at 19/12/2019 07:58:25

HLAfloat64Time	HLAfloat64BE	NA	4.9E-308	NA	Standardized 64 bit float time
----------------	--------------	----	----------	----	--------------------------------

Enumerated Data Type Table

Name	Representation	Semantics	Enumerator	Value
AreaEnum	HLAinteger32BE	It enumerates areas of responsibility	Aor1	0
			Aor2	1
			Aor3	2
			Aor4	3
HLAboolean	HLAinteger32BE	Standard boolean type	HLAfalse	0
			HLAtrue	1

Array Data Type Table

Name	Element Type	Cardinality	Encoding	Semantics
HLAASCIIstring	HLAASCIIchar	Dynamic	HLAvariableArray	ASCII string representation
HLAunicodeString	HLAunicodeChar	Dynamic	HLAvariableArray	Unicode string representation
HLAopaqueData	HLAbyte	Dynamic	HLAvariableArray	Uninterpreted sequence of bytes
HLAtoken	HLAbyte	0	HLAfixedArray	

Fixed Record Data Type Table

Record Name	Encoding	Semantics	Field Name	Field Type	Field Semantics
-------------	----------	-----------	------------	------------	-----------------

Generated by SimGe at 19/12/2019 07:58:25

8/17

Appendix

OMT REPORT



Generated by SimGe at 19/12/2019 07:58:25

Variant Record Data Type Table

Record Name	Encoding	Semantics	Discriminant Name	Discriminant Type	Alternative Enumeration	Alternative Name	Alternative Type	Alternative Semantics
-------------	----------	-----------	-------------------	-------------------	-------------------------	------------------	------------------	-----------------------

Interface Specification Services Usage Table

IEEE Std 1516.1-2010 clause	Service	Usage	Is Callback?
4.2	connect	False	False
4.3	disconnect	False	False
4.4	connectionLost	False	True
4.5	createFederationExecution	True	False
4.6	destroyFederationExecution	True	False
4.7	listFederationExecutions	False	False
4.8	reportFederationExecutions	False	True
4.9	joinFederationExecution	True	False
4.10	resignFederationExecution	True	False
4.11	registerFederationSynchronizationPoint	False	False
4.12	confirmSynchronizationPointRegistration	False	True
4.13	announceSynchronizationPoint	False	True
4.14	synchronizationPointAchieved	False	False
4.15	federationSynchronized	False	True
4.16	requestFederationSave	False	False

Generated by SimGe at 19/12/2019 07:58:25

9/17

Appendix

OMT REPORT



Generated by SimGe at 19/12/2019 07:58:25

4.17	initiateFederateSave	False	True
4.18	federateSaveBegun	False	False
4.19	federateSaveComplete	False	False
4.20	federationSaved	False	True
4.21	abortFederationSave	False	False
4.22	queryFederationSaveStatus	False	False
4.23	federationSaveStatusResponse	False	True
4.24	requestFederationRestore	False	False
4.25	confirmFederationRestorationRequest	False	True
4.26	federationRestoreBegun	False	True
4.27	initiateFederateRestore	False	True
4.28	federateRestoreComplete	False	False
4.29	federationRestored	False	True
4.30	abortFederationRestore	False	False
4.31	queryFederationRestoreStatus	False	False
4.32	federationRestoreStatusResponse	False	True
5.2	publishObjectClassAttributes	False	False
5.3	unpublishObjectClassAttributes	False	False
5.4	publishInteractionClass	False	False
5.5	unpublishInteractionClass	False	False
5.6	subscribeObjectClassAttributes	False	False

Generated by SimGe at 19/12/2019 07:58:25

10/17

Appendix

OMT REPORT



Generated by SimGe at 19/12/2019 07:58:25

5.7	unsubscribeObjectClassAttributes	False	False
5.8	subscribeInteractionClass	False	False
5.9	unsubscribeInteractionClass	False	False
5.10	startRegistrationForObjectClass	False	True
5.11	stopRegistrationForObjectClass	False	True
5.12	turnInteractionsOn	False	True
5.13	turnInteractionsOff	False	True
6.2	reserveObjectInstanceName	False	False
6.3	objectInstanceNameReserved	False	True
6.4	releaseObjectInstanceName	False	False
6.5	reserveMultipleObjectInstanceNames	False	False
6.6	multipleObjectInstanceNamesReserved	False	True
6.7	releaseMultipleObjectInstanceNames	False	False
6.8	registerObjectInstance	False	False
6.9	discoverObjectInstance	False	True
6.10	updateAttributeValues	False	False
6.11	reflectAttributeValues	False	True
6.12	sendInteraction	False	False
6.13	receiveInteraction	False	True
6.14	deleteObjectInstance	False	False
6.15	removeObjectInstance	False	True

Generated by SimGe at 19/12/2019 07:58:25

11/17

Appendix

OMT REPORT



Generated by SimGe at 19/12/2019 07:58:25

6.16	localDeleteObjectInstance	False	False
6.17	attributesInScope	False	True
6.18	attributesOutOfScope	False	True
6.19	requestAttributeValueUpdate	False	False
6.20	provideAttributeValueUpdate	False	True
6.21	turnUpdatesOnForObjectInstance	False	True
6.22	turnUpdatesOffForObjectInstance	False	True
6.23	requestAttributeTransportationTypeChange	False	False
6.24	confirmAttributeTransportationTypeChange	False	True
6.25	queryAttributeTransportationType	False	False
6.26	reportAttributeTransportationType	False	True
6.27	requestInteractionTransportationTypeChange	False	False
6.28	confirmInteractionTransportationTypeChange	False	True
6.29	queryInteractionTransportationType	False	False
6.30	reportInteractionTransportationType	False	True
7.2	unconditionalAttributeOwnershipDivestiture	False	False
7.3	negotiatedAttributeOwnershipDivestiture	False	False
7.4	requestAttributeOwnershipAssumption	False	True
7.5	requestDivestitureConfirmation	False	True
7.6	confirmDivestiture	False	False
7.7	attributeOwnershipAcquisitionNotification	False	True

Generated by SimGe at 19/12/2019 07:58:25

12/17

Appendix

OMT REPORT



Generated by SimGe at 19/12/2019 07:58:25

7.8	attributeOwnershipAcquisition	False	False
7.9	attributeOwnershipAcquisitionIfAvailable	False	False
7.10	attributeOwnershipUnavailable	False	True
7.11	requestAttributeOwnershipRelease	False	True
7.12	attributeOwnershipReleaseDenied	False	False
7.13	attributeOwnershipDivestitureIfWanted	False	False
7.14	cancelNegotiatedAttributeOwnershipDivestiture	False	False
7.15	cancelAttributeOwnershipAcquisition	False	False
7.16	confirmAttributeOwnershipAcquisitionCancellation	False	True
7.17	queryAttributeOwnership	False	False
7.18	informAttributeOwnership	False	True
7.19	isAttributeOwnedByFederate	False	False
8.2	enableTimeRegulation	False	False
8.3	timeRegulationEnabled	False	True
8.4	disableTimeRegulation	False	False
8.5	enableTimeConstrained	False	False
8.6	timeConstrainedEnabled	False	True
8.7	disableTimeConstrained	False	False
8.8	timeAdvanceRequest	False	False
8.9	timeAdvanceRequestAvailable	False	False
8.10	nextMessageRequest	False	False

Generated by SimGe at 19/12/2019 07:58:25

13/17

Appendix

OMT REPORT



Generated by SimGe at 19/12/2019 07:58:25

8.11	nextMessageRequestAvailable	False	False
8.12	flushQueueRequest	False	False
8.13	timeAdvanceGrant	False	True
8.14	enableAsynchronousDelivery	False	False
8.15	disableAsynchronousDelivery	False	False
8.16	queryGALT	False	False
8.17	queryLogicalTime	False	False
8.18	queryLITS	False	False
8.19	modifyLookahead	False	False
8.20	queryLookahead	False	False
8.21	retract	False	False
8.22	requestRetraction	False	True
8.23	changeAttributeOrderType	False	False
8.24	changeInteractionOrderType	False	False
9.2	createRegion	False	False
9.3	commitRegionModifications	False	False
9.4	deleteRegion	False	False
9.5	registerObjectInstanceWithRegions	False	False
9.6	associateRegionsForUpdates	False	False
9.7	unassociateRegionsForUpdates	False	False
9.8	subscribeObjectClassAttributesWithRegions	False	False

Generated by SimGe at 19/12/2019 07:58:25

14/17

Appendix

OMT REPORT



Generated by SimGe at 19/12/2019 07:58:25

9.9	unsubscribeObjectClassAttributesWithRegions	False	False
9.10	subscribeInteractionClassWithRegions	False	False
9.11	unsubscribeInteractionClassWithRegions	False	False
9.12	sendInteractionWithRegions	False	False
9.13	requestAttributeValueUpdateWithRegions	False	False
10.2	getAutomaticResignDirective	False	False
10.3	setAutomaticResignDirective	False	False
10.4	getFederateHandle	False	False
10.5	getFederateName	False	False
10.6	getObjectClassHandle	False	False
10.7	getObjectClassName	False	False
10.8	getKnownObjectClassHandle	False	False
10.9	getObjectInstanceHandle	False	False
10.10	getObjectInstanceName	False	False
10.11	getAttributeHandle	False	False
10.12	getAttributeName	False	False
10.13	getUpdateRateValue	False	False
10.14	getUpdateRateValueForAttribute	False	False
10.15	getInteractionClassHandle	False	False
10.16	getInteractionClassName	False	False
10.17	getParameterHandle	False	False

Generated by SimGe at 19/12/2019 07:58:25

15/17

Appendix

OMT REPORT



Generated by SimGe at 19/12/2019 07:58:25

10.18	getParameterName	False	False
10.19	getOrderType	False	False
10.20	getOrderName	False	False
10.21	getTransportationTypeHandle	False	False
10.22	getTransportationTypeName	False	False
10.23	getAvailableDimensionsForClassAttribute	False	False
10.24	getAvailableDimensionsForInteractionClass	False	False
10.25	getDimensionHandle	False	False
10.26	getDimensionName	False	False
10.27	getDimensionUpperBound	False	False
10.28	getDimensionHandleSet	False	False
10.29	getRangeBounds	False	False
10.30	setRangeBounds	False	False
10.31	normalizeFederateHandle	False	False
10.32	normalizeServiceGroup	False	False
10.33	enableObjectClassRelevanceAdvisorySwitch	False	False
10.34	disableObjectClassRelevanceAdvisorySwitch	False	False
10.35	enableAttributeRelevanceAdvisorySwitch	False	False
10.36	disableAttributeRelevanceAdvisorySwitch	False	False
10.37	enableAttributeScopeAdvisorySwitch	False	False
10.38	disableAttributeScopeAdvisorySwitch	False	False

Generated by SimGe at 19/12/2019 07:58:25

16/17

Appendix

OMT REPORT			
Generated by SimGe at 19/12/2019 07:58:25			
10.39	enableInteractionRelevanceAdvisorySwitch	False	False
10.40	disableInteractionRelevanceAdvisorySwitch	False	False
10.41	evokeCallback	False	False
10.42	evokeMultipleCallbacks	False	False
10.43	enableCallbacks	False	False
10.44	disableCallbacks	False	False
Notes Table			
Label	Semantics		

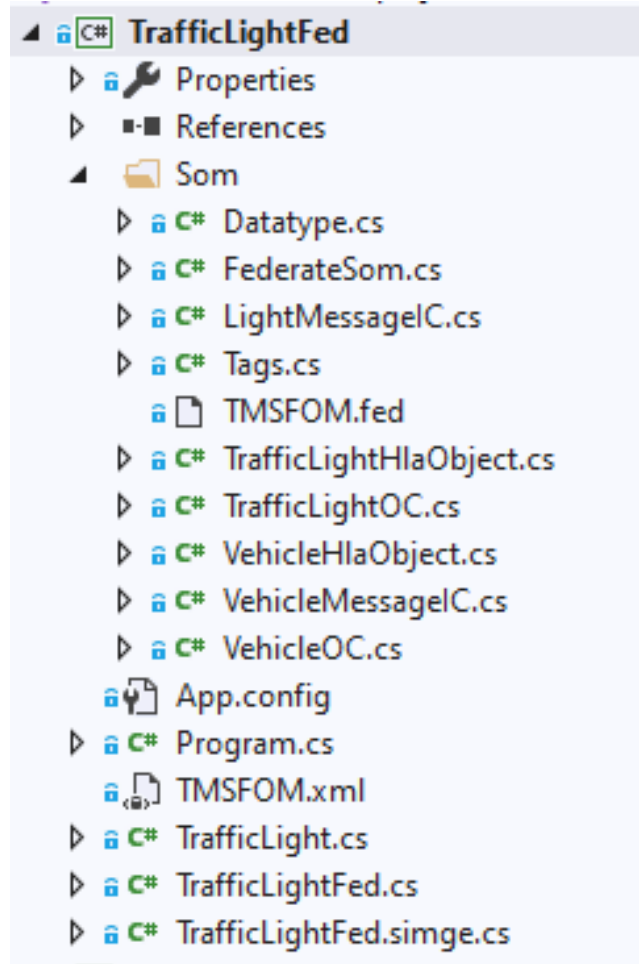


Figure 16: trafficLightFed Directory Structure

```

38 //trafficLightFed = new CTrafficLightHlaObject(trafficLightFed.Som.TrafficLightOC);
39
40 Console.WriteLine("Traffic Light Position [East, West, North, South]?");
41 name = Console.ReadLine();
42 // Initialize the federation execution
43 trafficLightFed.FederationExecution.FederateName = "TrafficLightFed" + name;
44 trafficLightFed.FederationExecution.Name = "tmsFederation";
45 trafficLightFed.FederationExecution.FederateType = "TrafficLight";
46 trafficLightFed.FederationExecution.ConnectionSettings = "rtti:///0.143.6.187";
47 trafficLightFed.FederationExecution.FDD = @"*.TMSFOM.xml";
48
49 // Connect
50 trafficLightFed.Connect(CallbackModel.EVOKED, trafficLightFed.FederationExecution.ConnectionSettings);
51 // Create federation execution
52 bool isCreated = trafficLightFed.CreateFederationExecution(trafficLightFed.FederationExecution.Name, trafficLightFed.FederationExecution.ConnectionSettings);
53 Console.WriteLine(isCreated);
54
55 // Join federation execution
56 trafficLightFed.JoinFederationExecution(trafficLightFed.FederationExecution.FederateName, trafficLightFed.FederationExecution.ConnectionSettings);
57

```

Figure 17: Federation Creation

```

38 //trafficLightFed = new CTrafficLightHlaObject(trafficLightFed.Scm.TrafficLightOC);
39
40 Console.WriteLine("Traffic Light Position [East, West, North, South]?");
41 name = Console.ReadLine();
42 // Initialize the federation execution
43 trafficLightFed.FederationExecution.FederateName = "TrafficLightFed" + name;
44 trafficLightFed.FederationExecution.Name = "tmsFederation";
45 trafficLightFed.FederationExecution.FederateType = "TrafficLight";
46 trafficLightFed.FederationExecution.ConnectionSettings = "rtti://10.143.6.187";
47 trafficLightFed.FederationExecution.FED = @"..\TMSFCM.xml";
48
49 // Connect
50 trafficLightFed.Connect(CallbackModel.EVOKED, trafficLightFed.FederationExecution.ConnectionSettings);
51 // Create Federation execution
52 bool isCreated = trafficLightFed.CreateFederationExecution(trafficLightFed.FederationExecution.Name, trafficLightFed.FederationE
53 Console.WriteLine(isCreated);
54
55 // Join federation execution
56 trafficLightFed.JoinFederationExecution(trafficLightFed.FederationExecution.FederateName, trafficLightFed.FederationExecution.Fe
57

```





































Figure 18: Federation Synchronization

```

306 base.FdAmb_SynchronizationPointAnnounced(sender, data);
307
308 #region User Code
309 //Report.WriteLine($"Ready for zone transfer. Label: {data.Label}");
310 Console.WriteLine("We are changing lights to red all");
311 Program.TrafficLights[0].TrafficLight.Sign = "Red";
312 SynchronizationPointAchieved(data.Label, true);
313 #endregion //User Code

```

Figure 19: Synchronization Point Announced

- ▲   VehicleFed
 - ▶   Properties
 - ▶   References
 - ▲   Som
 - ▶   Datatype.cs
 - ▶   FederateSom.cs
 - ▶   LightMessageIC.cs
 - ▶   Tags.cs
 -   TMSFOM.fed
 - ▶   TrafficLightHlaObject.cs
 - ▶   TrafficLightOC.cs
 - ▶   VehicleHlaObject.cs
 - ▶   VehicleMessageIC.cs
 - ▶   VehicleOC.cs
 -   App.config
 - ▶   Program.cs
 -   TMSFOM.xml
 - ▶   Vehicle.cs

```

42 // Call the base class handler
43 base.FdAmb_InteractionReceivedHandler(sender, data);
44
45 if (data.Interaction.ClassHandle == Som.LightMessageIC.Handle)
46 {
47     Console.WriteLine("Recieved: " + Som.LightMessageIC.Handle);
48
49     if (data.IsValueUpdated(Som.LightMessageIC.Sign))
50     {
51         Sign = data.GetParameterValue<string>(Som.LightMessageIC.S
52         Console.WriteLine("Traffic Light Sign: " + Sign);
53         if(Sign == "Green")
54         {
55             Console.WriteLine("Green ! GO now");
56         }
57         else if(Sign == "Red")
58         {
59             Console.WriteLine("RED! Wait");
60         }
61     }

```

Figure 21: VehicleFed Interaction Received Handler