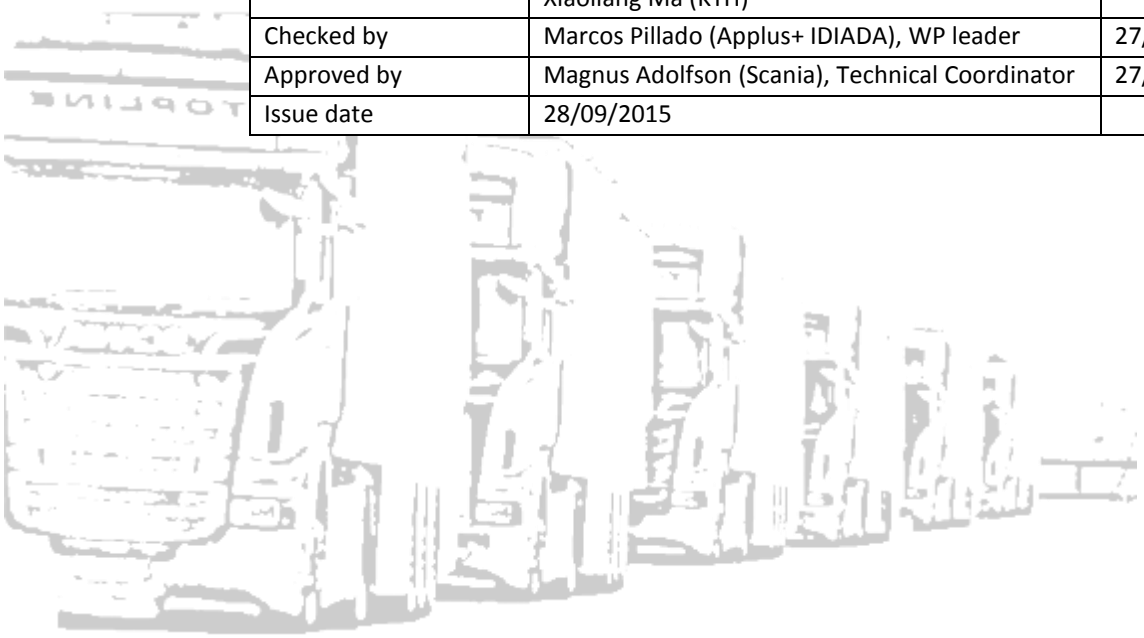
EUROPEAN COMMISSION
SEVENTH FRAMEWORK PROGRAMME
GA No. 610990

Cooperative dynamic formation of platoons for safe and energy-optimized goods transportation

COMPANION

D7.2 Limited results of the off-board platoon coordination system via simulation

Deliverable No.	COMPANION D7.2	
Deliverable Title	D7.2 Limited results of the off-board platoon coordination system via simulation	
Dissemination level	Public	
Written By	Marcos Pillado (Applus+ IDIADA) Jonas Mårtensson (KTH) Xiaoliang Ma (KTH)	
Checked by	Marcos Pillado (Applus+ IDIADA), WP leader	27/09/2015
Approved by	Magnus Adolfson (Scania), Technical Coordinator	27/09/2015
Issue date	28/09/2015	



History log

Name	*Status*	*Version*	*Date*	*Summary of actions made*
Applus+ IDIADA *Marcos Pillado*	Editor	0.1	17/03/2015	First draft. Scheme and descriptions
Applus+ IDIADA *Jonathan Batlle*	Contributor	0.2	18/03/2015	Extended information on Chapter 4
KTH *Jonas Mårtensson (KTH)* *Xiaoliang Ma (KTH)*	Contributor	0.3	04/09/2015	Intro, overview of off-board system, simulator architecture, simulation of transport plan optimization
Applus+ IDIADA *Marcos Pillado*	Reviewer	0.4	10/09/2015	Unified format, merged contributions and corrected syntax
KTH *Jonas Mårtensson (KTH)*	Contributor	1.0	25/09/2015	Rearranged material previously provided
KTH *Xiaoliang Ma (KTH)*	Contributor	1.1	26/09/2015	Rewritten KPIs section and updated images
Applus+ IDIADA *Marcos Pillado*	Reviewer	1.2	27/09/2015	Proofreading and minor changes
Applus+ IDIADA *Marcos Pillado*	Reviewer	1.3	28/09/2015	Unified the keywords used in the document

Executive summary

This document reports the work done inside the task 7.2 Off-board platoon coordination via simulation belonging to work package WP7 – System Integration, Validation, Deployment and Demonstration in the COMPANION project.

The objective of WP7 is the integration, validation and assessment of the full COMPANION system through the deployment and demonstration of platoons in real field trials and simulation of the on-board and off-board systems. The objective of Task 7.2 is to build a simulator to evaluate the capabilities of the off-board system in a larger scale than what is possible with field tests. The simulator will be connected to the off-board platform that is being developed in the COMPANION project. The simulator replaces the real vehicles in the sense that it receives vehicle Assignment Plans, simulates the execution of those plans, and reports back the position and status of the vehicles to the off-board system. The simulator will be used to evaluate how well the off-board system adapts to traffic disturbances and the overall benefit from platoon coordination on larger fleets of vehicles will also be evaluated.

Contents

List of Figures

List of Tables

1. Introduction

This document reports the work done inside the task 7.2 Off-board platoon coordination via simulation, belonging to work package WP7 – System Integration, Validation, Deployment and Demonstration in the COMPANION project.

The objective of WP7 is the integration, validation and assessment of the full COMPANION system through the deployment and demonstration of platoons in real field trials and simulation of the on-board and off-board systems. The objective of Task 7.2 is to build a simulator to evaluate the capabilities of the off-board system in a larger scale than what is possible with field tests. The simulator will be connected to the off-board platform that is being developed in the COMPANION project. The simulator replaces the real vehicles in the sense that it receives vehicle Assignment Plans, simulates the execution of those plans, and reports back the position and status of the vehicles to the off-board system. The simulator will be used to evaluate how well the off-board system adapts to traffic disturbances and the overall benefit from platoon coordination on larger fleets of vehicles will also be evaluated.

2. Overview of the off-board system

In this section we will briefly describe the off-board system. First the system architecture is described. We will then have a more detailed look into the coordinated platoon coordination and to the output that the system generates. The output is a Assignment Plan that specifies, for each truck, the route to travel, the required velocity profile along the route, and information about which trucks to platoon with and where on the route the platooning should take place.

2.1 Off-board system architecture

The figure below depicts the overall component architecture of the COMPANION system platform. The platform can be used either by connecting real vehicles to the Fleet Manager, or by connecting the simulator. More details about the architecture and system components please see [2].

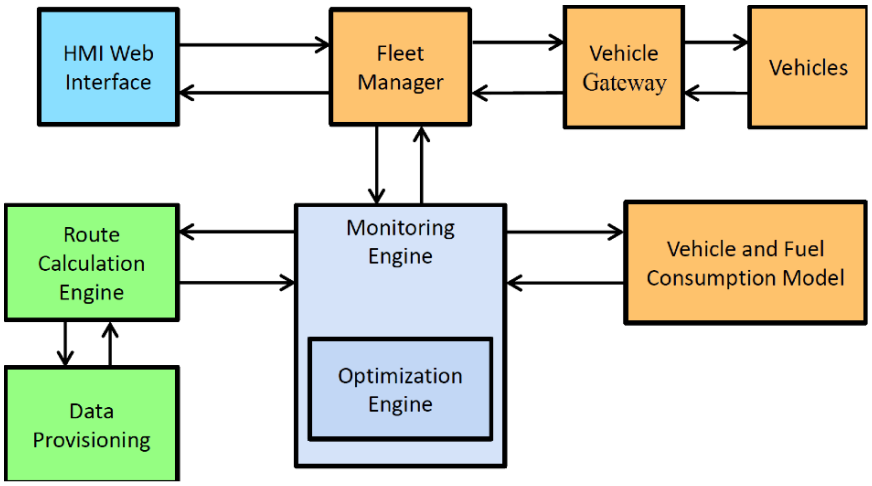


Figure 1. Component architecture of the off-board system

2.2 Scheduling of platoons

This section gives a short explanation of the coordination of platoons algorithm works. The output from this function is the instructions to the vehicles how to drive in order to form platoons. The same information is used as input to the simulator. We also show an example of a scenario and the type of Assignment Plan it will generate.

2.2.1 Nomenclature

A transport assignment defines the origin and destination of a transport, and it also gives timing constraints in terms of destination deadline and time for earliest departure. A route describes the path from origin to destination. The *routes* from the Route Calculation Engine (RCE) are given as sequences of *links*. The Monitoring and Optimization Engine (MOE) will not work on link level, but on larger *segments*, consisting of several links. The connection nodes between segments are called *waypoints*. This nomenclature is depicted in Figure 2. The output from the MOE is a Platoon Plan, consisting of, for each vehicle, a list of segments (which in turn are defined as lists of links). For each segment the requested average velocity. Merge and split only occurs at the waypoints, which means that each segment has a fixed platoon configuration. This configuration is given as a list of vehicle IDs of the other vehicles in your platoon. These are called the *platoon companions* of a vehicle.

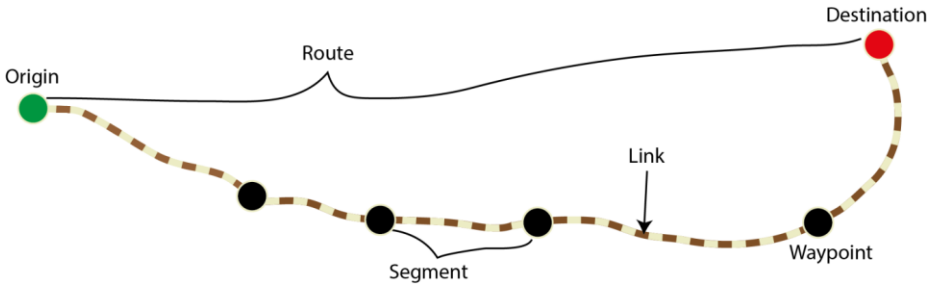


Figure 2. Definitions of link, segment and route

2.2.2 Coordinated platoon planning

In order to efficiently obtain platoon configurations and the corresponding average velocities for each vehicle, a three-step approach is taken.

Route planning

The first step comprises finding the most suitable route for each vehicle, taking factors such as road topography and traffic information into account. This is done by the Route Calculation Engine in the off-board platform

Pairwise speed planning

Second, for a given vehicle, which will be referred to as a coordination leader, its fuel-optimal velocity profile is computed. The starting time and arrival deadline are taken into account together with constraints such as driver resting times. The profile specifies the desired average velocity on larger segments of the route. Then, for each vehicle with a partially overlapping route with the coordination leader, a pairwise analysis is used to determine whether it is beneficial to adapt its velocity profile to form or join a platoon with the coordination leader. If so, this vehicle is referred to as a coordination follower. In this pairwise analysis, the coordination leader does not adapt its

velocity profile, such that several coordination followers can be assigned to a single coordination leader. Arrival deadlines are taken into account when adapting the velocity profiles of the coordination followers. A conceptual sketch of the pairwise speed adaptation is shown in Figure 3 where the vertical axis represents the position of a truck along its own path. The grey area shows the overlapping parts of the two vehicles' routes, which means that platooning only can take place in that region. The blue line corresponds to the fixed speed profile of the coordination leader. The coordination follower has some freedom to adapt its speed, indicated by the red shaded area. The speed profile is optimized so that platooning with the coordination leader is possible. The optimized profile of the coordination follower is shown as the dashed red line. When the blue and red lines coincide the vehicles are platooning.

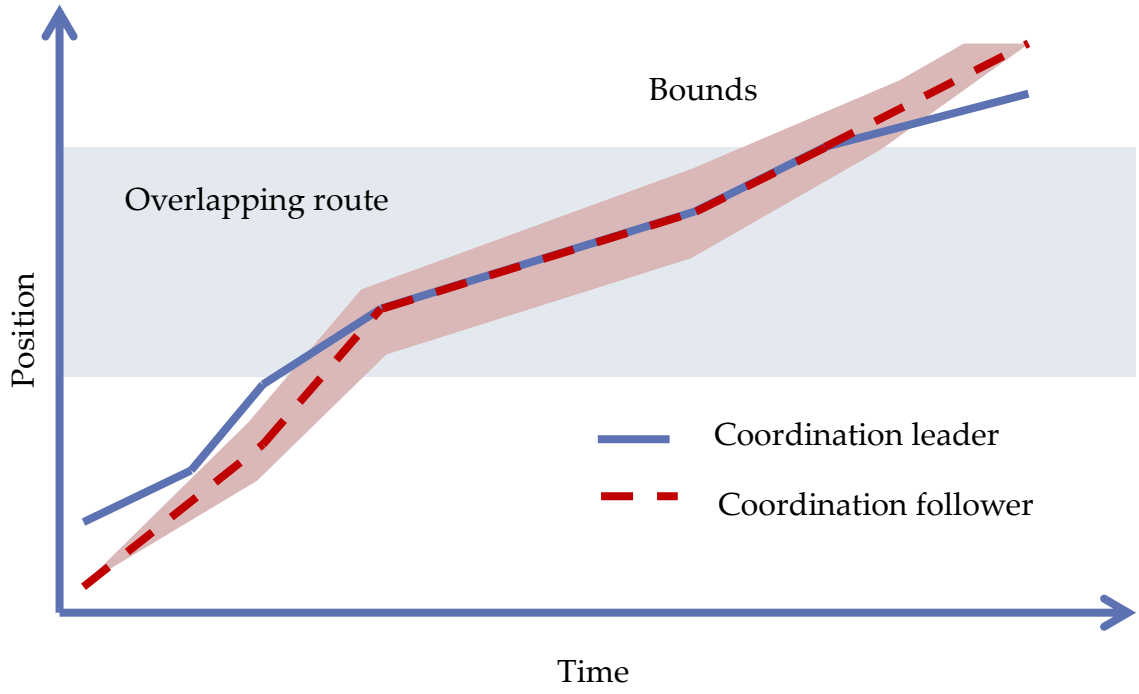


Figure 3. Pairwise speed adaption

Coordination leader selection

The selection of the most suitable coordination leaders is crucial in obtaining significant fuel savings. This selection forms the third step. Repeating the pairwise analysis for every potential coordination leader leads to a data set that can be conveniently represented as a graph. In this graph, the nodes represent the vehicles and their incoming edges denote the fuel savings obtained when this vehicle is selected as a coordination leader. The left plot of Figure 2.4 shows such a coordination graph.

From graph clustering algorithms, an algorithm can be derived to compute a suitable set of coordination leaders. Specifically, a greedy algorithm that incrementally adds or removes individual vehicles from the set of coordination leaders provides a computationally efficient and scalable approach. The right plot of Figure 4 shows the result of that algorithm, where there are three coordination leaders selected (green nodes), and each of those has a number of coordination followers associated to it (red nodes). **On the left:** Each node represents a vehicle. The directed edges represent the obtained fuel savings if the start node were selected as a coordination follower and the end node as a coordination leader. The edge weight corresponds to the fuel saving obtained

from that pairwise fuel-optimized plan. The algorithm works iteratively by adding or subtracting coordination leaders. The values in green at each node represent the incremental benefit of choosing that vehicle as a coordination leader. In this case the green node achieves the best benefit and that is selected as a coordination leader. **On the right:** The algorithm stops when no more fuel savings can be obtained by adding or subtracting a coordination leader. In this example there are three coordination leaders, one with three followers and two with one follower each. One vehicle is not coordinated at all and will drive according to its own optimal speed profile.

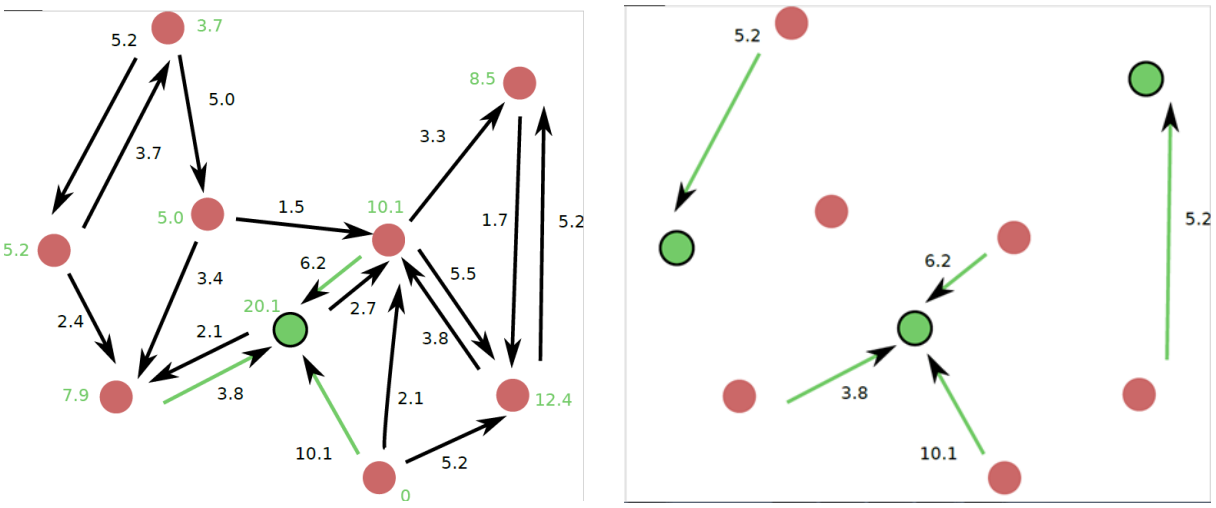


Figure 4. Algorithm for coordination leader selection

Platoon Plan creation

The Platoon Plans are created for each of the clusters in the coordination graph. The pairwise speed optimization has already defined segments for which average speed references are given. If two or more coordination followers are signed to the same coordination leader then these must be combined and some segments may be divided in smaller segments.

An example is shown in the upper plot of Figure 5 where the routes of three assignments overlap. In this case the green route R2 corresponds to a coordination leader and the two other routes correspond to two coordination followers. When these are combined it could look as in the lower plot. There are 11 segments all together. A Platoon Plan example for the three vehicles is described in Table 1. Note for example the shaded row for segment 6, where all vehicles have the same velocity and they are platoon companions of each other.

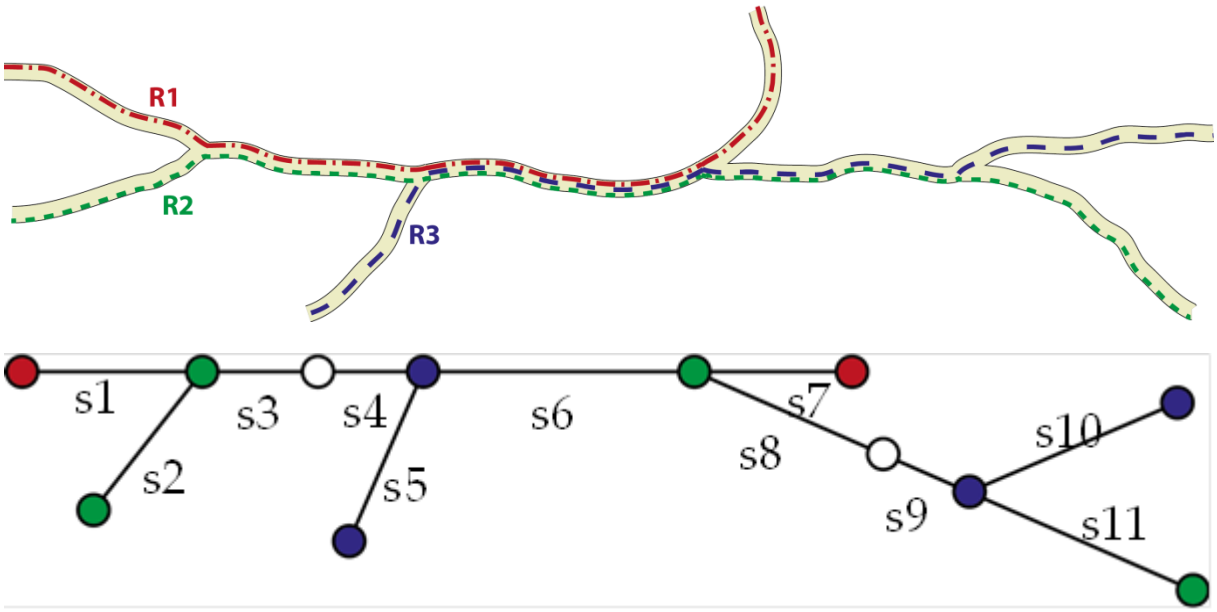


Figure 5. Example of three assignments

Vehicle 1 (red)			Vehicle 2 (green)			Vehicle 3 (blue)		
Segment	Velocity	Companions	Segment	Velocity	Companions	Segment	Velocity	Companions
S1	75							
			S2	60				
S3	75		S3	70				
S4	70	Vehicle 2	S4	70	Vehicle 1			
						S5	85	
S6	80	Vehicle 2+3	S6	80	Vehicle 1+3	S6	80	Vehicle 1+2
S7	75							
			S8	85	Vehicle 3	S8	85	Vehicle 2
			S9	80		S9	80	
						S10	80	
			S11	75				

Table 1. Platoon Plan example

2.2.3 Simulation example of platoon coordination

The platoon control and coordination algorithms presented in this paper are demonstrated by means of a simulation scenario representing a part of the highway network of Sweden, see Figure 2.6. On this network, 200 heavy-duty vehicles originating from six locations in the Stockholm area in the east travel to five destinations in the west. The starting times for these vehicles are taken from a two-hour interval.

The methodology for coordinated platoon planning, described above, is used to select suitable coordination leaders and their respective followers. Herein, pairwise plans are considered in which the coordination followers catch up with their leaders and platoon until either of their routes end or their routes split up.

For this scenario, the coordination algorithm selects 54 coordination leaders and 139 coordination followers, which adjust their velocity profiles to catch up with the coordination leaders in order to form platoons. The maximum number of coordination followers per coordination leader is 8, whereas the median is 2. The remaining 7 vehicles do not platoon but traverse their routes individually.

The routes of one particular coordination leader and its two coordination followers are highlighted in Figure 6. Starting locations and destinations are indicated by red circles and blue squares, respectively, and the boldface numbers represent intersections. The two numbers next to the road segments indicate the number of vehicles that traverse this segment and the average platoon size on this segment, respectively, as a result of the applied coordination algorithm. The road segment between nodes 7 and 8 is traversed in both directions and the statistics for vehicles travelling in either direction are indicated separately. Three routes, indicated by dashed lines, are highlighted as an example for a group comprising a coordination leader (black) and two coordination followers (blue and red). The corresponding trajectories are presented in Figure 2.7, where the time gaps with respect to the coordination leader as a function of the position on the road are shown. The graph shows the time gap to the platoon leader as a function of the position on the road, where this position is taken along the routes of the individual vehicles. The dashed lines denote the position of the nodes representing road intersections in Figure 2.6, with the top labels denoting the node number. As an example, note that the coordination leader starts from Norrtälje (node 2) and drives to Trollhättan (node 15). When the time gap is zero and the routes of the vehicles overlap (between nodes 5 and 10), the vehicles operate in a platoon.

Note that the first coordination follower (blue) shares the first part of its route with the coordination leader (black), but as it starts 1.25 hours later it catches up at maximum speed, indicated by a decreasing gap to the leader in Figure 7It then meets the platoon consisting of the coordination leader and the other coordination follower (red) between nodes 5 and 7, in which it stays until its destination at node 10 is reached.

The route of the second coordination follower intersects with the route of the coordination leader at node 5 and the coordination follower's start time is such that it catches up to the coordination leader at a velocity that is lower than the maximum speed. The coordination follower and coordination leader form a platoon at node 5 and platoon until node 10 where their routes split up.

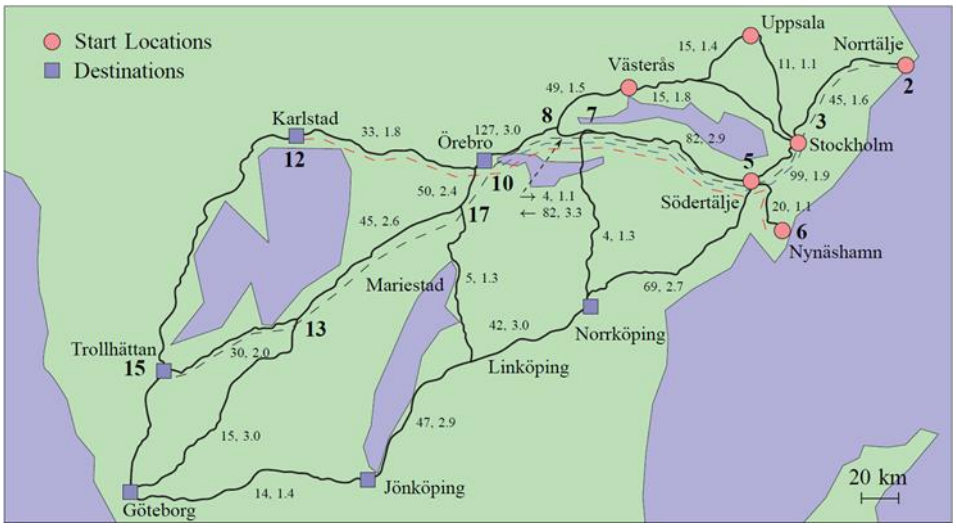


Figure 6. The Swedish road network used in the case study

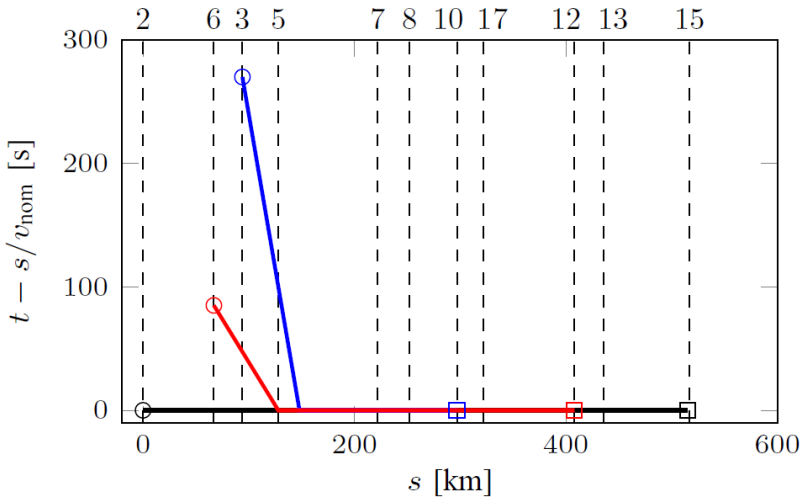


Figure 7. Platoon plans for a coordination leader (black) and two coordination followers (blue and red)

3. Overview of COMPANION Simulator

The main purpose of the COMPANION Simulator is to evaluate the Monitoring and Optimization Engine (MOE) and its application in platooning at large network level. In this section, the simulator design and implementation will be described with limited results in testing theAssignment Plans. The objective is to be able to finally simulate many COMPANION trucks and platoons coordinated in a large network and evaluate system performance. While the current document focuses on simulation system design and software simulation implementation, a comprehensive description and the final evaluation will be reported in WP5 Off-Board System for Platoon Coordination within task T5.2 "*Coordinated fault-tolerated real-time scheduling of platoon*".

3.1. Capabilities

In general, the COMPANION simulator is designed as a discrete event simulation tool with two main capabilities:

- Take Assignment Plans as inputs and predict the online states of COMPANION HDV trucks and platoons under dynamic traffic and weather conditions;
- Demonstrate the off-board decision-making (route) and information guidance (speed) to individual HDVs while online Platoon Plans are updated according to scheduling and platooning requirements.

In order to demonstrate the COMPANION platooning system, especially the off-board system, and visualize the real-time states of HDVs, a discrete-time computer simulation tool has been developed:

- To mimic the assignment of individual COMPANION truck with OD and initial itinerary (input from COMPANION system);
- To simulate real-time traffic congestion and its impacts on COMPANION trucks;
- To reflect and predict the movement of COMPANION trucks and platoons on the network;
- To predict the real-time travel delay status and fuel consumption.

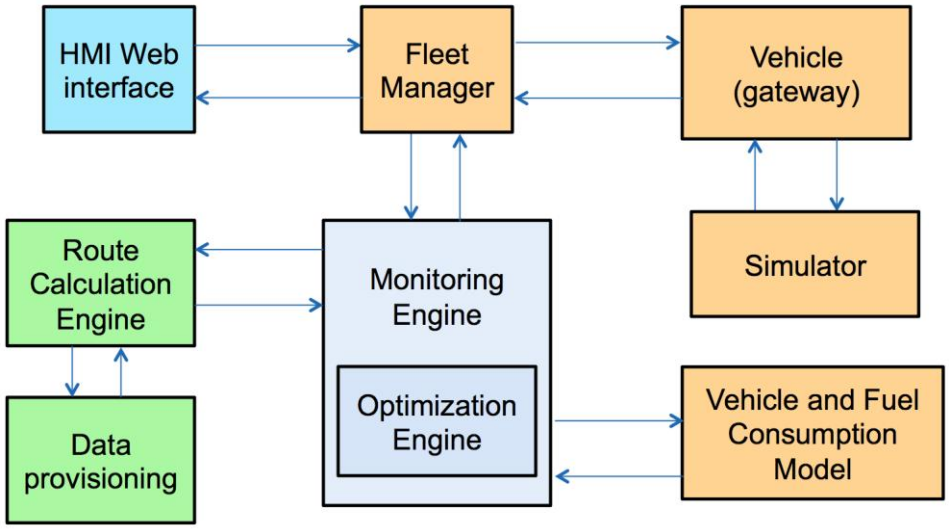


Figure 8. The simulator as an evaluation component to communicate with the COMPANION platooning system

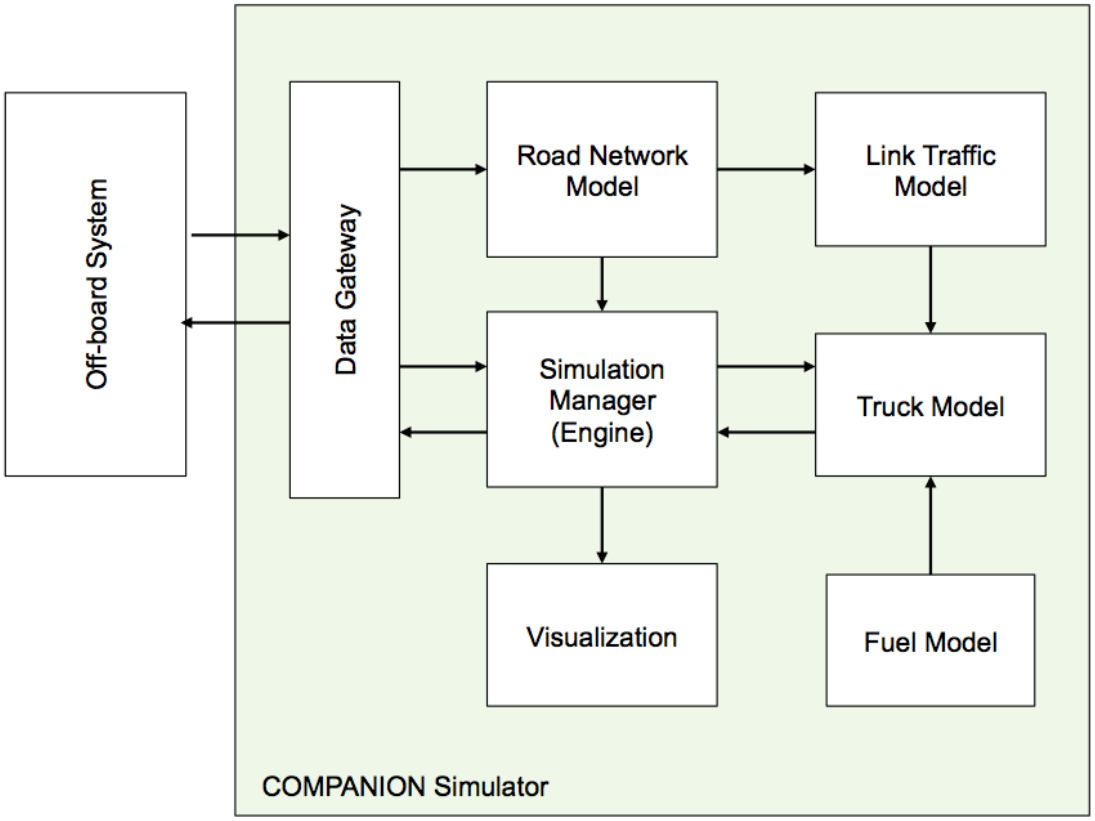


Figure 9. Architecture of the COMPANION simulator

3.2. Architecture

While not part of the real COMPANION system, the simulator plays essential roles to demonstrate the off-board decision system, especially the monitor and optimization engine. But according to the general system development, the off-board system requires direct communication with vehicle through its gateway, where information from real or virtual vehicles can be retrieved safely. This is shown in Figure 8.

Figure 9 illustrates the major components of the simulator. At the concept level, the simulator indeed retrieves from and provides to off-board decision system necessary information through its own data gateway. The communication is achieved through Microsoft Azure Cloud. The initial data received for Assignment Plan will be used to construct a road network graph, composed of links and nodes. Segment, a set of links, is also introduced. While the off-board optimization decision is based on road segments, traffic is modelled at the link level. A heavy-duty vehicle (also called truck) is modelled as a unit that encapsulates its own information obtained from real or virtual trucks. The Fuel Model component is an interface where a fuel model can be implemented or fuel calculation software can be embedded. The running of the simulator is managed and coordinated by the simulation engine. Visualization is an independent, web-based tool interacting with others through the simulation engine. The communication between the visualization tool and simulation engine is implemented using shared memory where simulated truck data is reported to the visualization unit every simulation step.

3.3.1. Road network model

The road network being simulated is represented by a graph of nodes and directed links. A node includes the coordinate of the central point while a link connects nodes for a certain direction. Traffic information is represented at the link level. Each link has its own properties like road grade, speed limit etc. Segment is introduced as an aggregate set of links in order to support representation of route choice and platooning decision. When an Assignment Plan arrives before a simulation starts, it will be interpreted by going through the route of each vehicle. Road segments and links will be abstracted and used to generate the road network. When getting new Assignment Plans during simulation, the route of each vehicle will be traversed and new road segments with the composed links will be added to the existing road network model.

3.3.2. Link traffic model

As mentioned, traffic flow is described at the link level. Currently, the traffic condition is simply represented by the speed (V_f) of general traffic flow according to traffic information from the route calculation engine (RCE). In fact, this is incorporated in the Assignment Plan generated using real-time traffic information, whereas the simulator doesn't directly receive real-time traffic information from RCE. However, speed limits are coded for each road segment in the network as V_L. During simulation, the speed of a truck is updated according to the minimum of speed limit on links and the running speed planned at the level of road segment.

The user can also introduce incident at a link of the selected road segment, which affects the speed level of the road link during certain period. This function mainly supports the demonstration of route optimization engine, which will generate new plans for each truck.

3.3.3. Truck and platoon models

Only COMPANION trucks are represented in the simulator while other vehicles on road, though not simulated, have direct impacts on the general traffic condition. A truck unit is represented by an object model, which encapsulates basic properties of the truck. The basic properties include truck parameters such as length, weight, engine power, freight parameters etc. as well as its route and trajectory (position and speed).

The truck object will also have a property to reflect whether it is in platooning mode. If it is in a platoon, the reference to its preceding truck and/or following truck will be added. Each truck also has a property of its delay value on the road segment compared to the planned estimated time of arrival (ETA). This helps the visualization tool to monitor the delay of all running trucks. In particular, significant delay may trigger the route optimization engine to generate a new plan.

In addition, a platoon object model is also introduced and encapsulates a sequence of trucks in order. This is mainly to support visualization so that no sorting or searching is needed for seeking the state of platoons. The platooning operations such as merging and splitting are simply represented. For example, when two trucks at a certain simulation step run close enough to each other (within a threshold of certain distance), a platoon will be formulated. The state variables of each truck including platooning mode, preceding or following truck will be updated. A new platoon object will be created with those two trucks as members.

3.3.4. Fuel model

The fuel model is only designed as an interface. Currently, a fuel model based on vehicle dynamics is implemented, which calculate the fuel being used during each simulation time step mainly according to its speed and acceleration. It is a thermal engine based fuel consumption model that integrates with vehicle mechanical properties. The model computes fuel consumption based on thermal efficiency, mechanical efficiency on friction resistance as well as the motion of vehicle kinetics. The air drag resistance therefore has a direct impact on fuel usage. When a truck runs in the platooning mode, the fuel saving due to platooning will be estimated. The interface allows further extension to apply the fuel model developed by Scania.

3.3.5. Simulator engine

The simulator engine (manager) plays a central role of controlling the discrete-time simulation, threading and simulation clock; in addition, it hosts the road network being simulated and a data structure for a sequence of truck objects being assigned and update truck states every simulation time step according to dynamic traffic information encapsulated in the Assignment Plan. It also includes a sequence of platoon objects.

As mentioned, delay of truck compared to the plan is estimated and updated every simulation step. This information will be feeded back, which can be used by route optimization engine to decide whether route optimization is required.

3.3.6. Visualization

An independent visualization tool is developed, which draws the road network on a map and represents live trucks and their state information during simulation. When a simulation runs, the simulation engine will share the live information of truck states including position, speed, delay, platooning or not, and fuel consumption for demonstration on web-based visualization.

3.3.7. Data gateway

This module represents an internal function unit to obtain information of Assignment Plans and provide real-time vehicle states via Azure. The unit includes a message receiver and a message sender, both of which run on a new thread. The receiver listens and reacts to new Assignment Plans whereas the sender sends truck states for each simulation step.

3.3. Simulation procedures

Figure 10 summarizes the detailed procedures when the simulator starts. First, once the simulator receives the first Assignment Plan, the simulation engine starts to initialize timing and interpret the “vehicle-to-infrastructure” information posted through Azure, and moreover construct a sequence of vehicles by calling the truck model. In the meantime, a road network graph is also generated using detailed route information for each truck. The graph model can be extended if there is new link and road segment in the updated Assignment Plan. After road network and truck models are initialized, the simulation process starts and vehicles run on the network and update their state each simulation time step.

The visualization unit will get updated information about vehicle states and represent them in the map-based visualization graph. In each time step, the simulated vehicle trajectory information will be sent back to Azure to be used by off-board system for decision-makings. When certain event such as

traffic incident, delay of a truck etc. causes an update of Assignment Plan, the simulation engine will update the network graph and vehicle properties (routes, speed plan etc.) accordingly. Vehicles run until they reach the destinations. All results per simulation step will also be stored for post-process analysis.

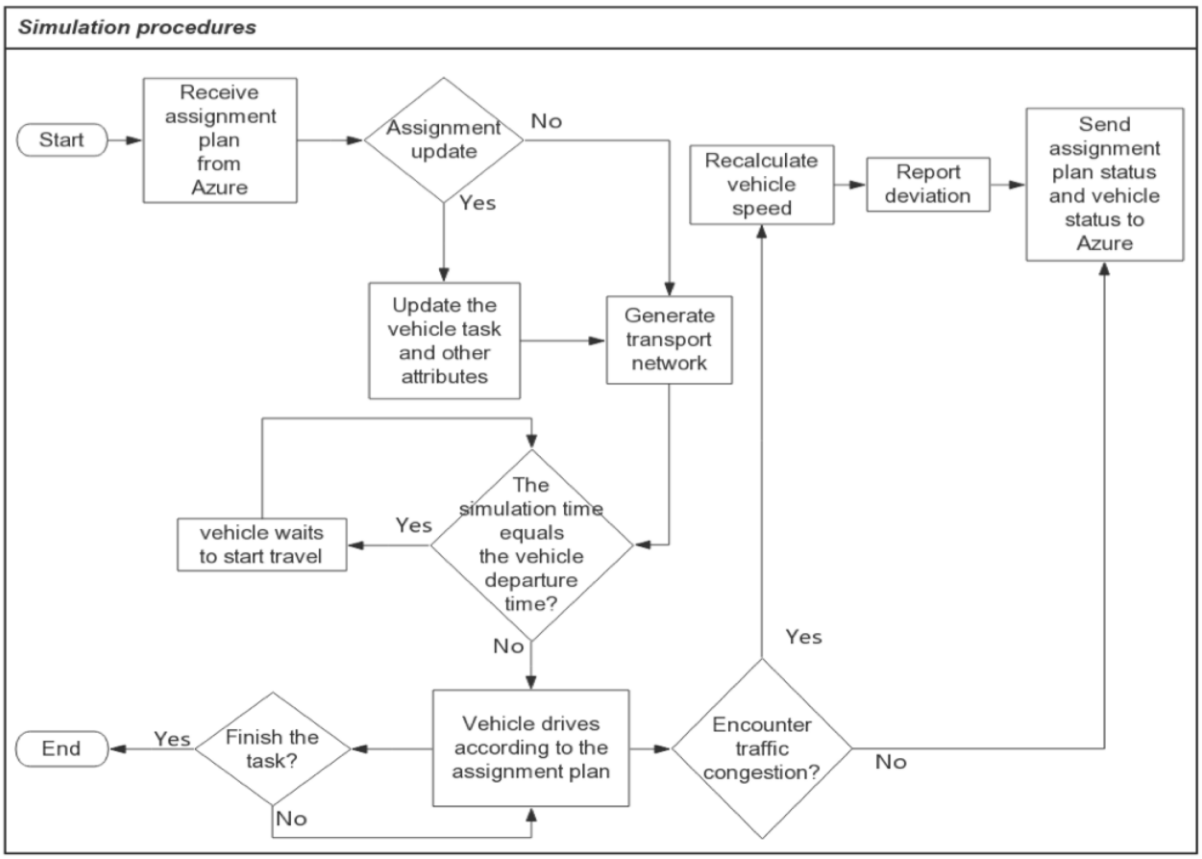


Figure 10. Diagram demonstration of the simulation procedures

3.4. Implementation

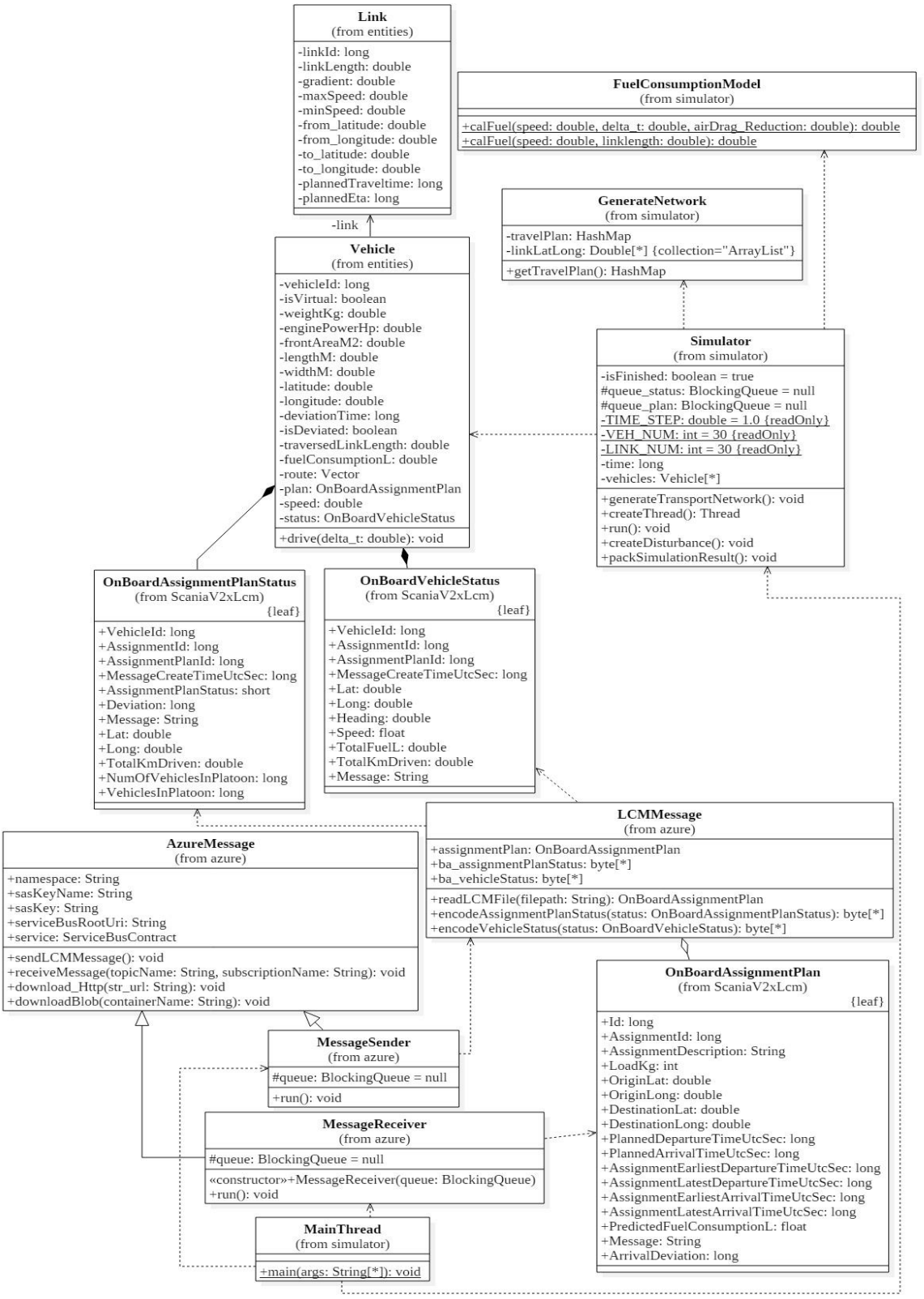


Figure 11. UML diagram for the design of COMPANION simulator

The implementation of the COMPANION simulator is done in Java. Microsoft Azure is used for communication, receiving from vehicle gateway and sending information to the Monitoring engine.

Web-based visualization is developed as an independent program using the Django framework and Python Script.

Figure 11 shows the UML design diagram of the COMPANION simulator. The simulator uses a multithreading framework to increase the computational efficiency. While the main thread controls the whole program, there are three additional threads that support the program, namely the Simulator, the MessageReceiver and the MessageSender. The Simulator thread is responsible for the execution of the simulation engine, the MessageReceiver is designed to receive message from Azure and the MessageSender will handle the message sending work to Azure. The working flow can be presented as the follow steps:

1) The main thread starts the MessageReceiver, the Simulator and the MessageSender threads. Then the MessageReceiver thread starts to listen to certain subscriptions under the topic "infrastructure_to_vehicle_topic" in Azure. (The word "certain" means a convention reached by KTH and SCANIA, who creates the "infrastructure_to_vehicle_topic".) Each thread will use the BlockingQueue to put messages and (or) take messages, as a typical producer-consumer design pattern. At the very beginning, since there are no messages passed to the Simulator and the MessageSender, both of the threads will be blocked until the MessageReceiver receives OnBoardAssignmentPlans for a set of vehicles.

2) When the MessageReceiver receives messages from Azure, it will pre-process the messages and put them into the BlockingQueue. The pre-processing goes as follows: firstly, the MessageReceiver gets one message from each subscription (the name of the subscription will be a string containing the vehicle id.) and the message is the link to Azure storage. Then the MessageReceiver will download the Assignment Plan file using the URL. By calling LCMMessage.readLCMFile(filepath) function, an OnBoardAssignmentPlan object will be returned. All the OnBoardAssignmentPlan objects will be stored in a HashMap, where the key is the vehicle id and the value is the OnBoardAssignmentPlan object. Finally the HashMap is put into the BlockingQueue.

3) When the BlockingQueue is not empty, the Simulator starts to take the HashMap from the BlockingQueue. The HashMap contains everything necessary for the simulation engine. According to the HashMap, the engine creates a set of vehicles with vehicle id and other attributes like OnBoardAssignmentPlan, OnBoardAssignmentPlanStatus and OnBoardVehicleStatus. At the same time, the engine will find the earliest departure time of all these vehicles and the value will be passed to the member variable time as the start time of the simulation.

4) When a simulation begins, the engine builds the graph model using link-level information from OnBoardAssignmentPlan objects. As a discrete event-based simulator, the simulation engine could create random disturbances so that some links encounter traffic congestions during a predefined period. For each simulation run, the simulation time increases step by step. When the simulation time equals the departure time of the vehicle(s), the vehicle(s) starts to move on the network graph model. If the vehicle comes across the disturbances, the speed of the vehicle will be reset according to the traffic flow model. Under such circumstances, the vehicle will encounter deviation. Otherwise the vehicle drives as planned. Every 10 seconds, the Simulator thread puts the OnBoardAssignmentPlanStatus and OnBoardVehicleStatus (contains fuel consumption) information into the BlockingQueue while the MessageSender thread then takes the information from the

BlockingQueue. By calling functions like LCMMessage.encodeAssignmentPlanStatus(parameter) and LCMMessage.encodeVehicleStatus(parameter), the OnBoardAssignmentPlanStatus object and the OnBoardVehicleStatus object will be encoded into byte arrays. Those byte arrays will become the body of the BrokeredMessage and sent to the Azure cloud.

5) In terms of the transport plan update, the simulator has functionalities to handle it in the following way. In the real world, the plan update refers to the route adaption, speed and platoon plan changes. Since the route adaption is decided by the Monitoring engine, the adaption of speed and platoon plans is the main result being demonstrated by simulation. When the MessageReceiver thread is listening to the subscriptions and detects a new message, it will send the message to the Simulator thread as designed before. The Simulator will find the vehicle with the same ID and replace the OnBoardAssignmentPlan attribute with the new one. If the new plan changes speed profiles, the simulation engine will carry out the new plan step by step. If the new plan changes route, then the situation becomes more complicated. First, the simulation engine needs to check whether the graph model contains the new links. If so, new links will be added to the graph model. Then the vehicle will carry out the new Assignment Plan. The program will not delete the old links. The new plan could not change the event that has happened in the simulation. If the simulation is used for testing other transport plans, the simulator has to be restarted to reconstruct the simulated time and space. During evaluation, the simulation can run much faster than real-time but the optimization engine is designed to handle real-time case. So, when the new plan gives order to change speed, which is supposed to happen right after an unacceptable deviation reported to the monitor engine, the clock in the simulation engine may run far ahead the time when the new plan should be carried out due to optimization computation and communication delay. Therefore, in our implementation, when the deviation is detected, the simulation runs with a slow-down speed close to real-time while continues recording the simulation results at the very moment for the possible rollback. When a new plan arrives, the simulation may return its speed and starts from the exact simulation time that the plan arrives.

4. Scenario evaluation and initial test

4.1 Test cases

The tests using the COMPANION simulator will be designed based on test cases that take place in Europe. Scania has provided departure and arrival places in Sweden to create the test cases. Furthermore Transportes Cerezuela has provided real transport routes in Europe with departure and arrival places in Spain, UK, France and Germany. So Transport Assignments are created using provided data. Each assignment has ID, origin, destination, departure and arrival times with boundaries, load and route deviation option. Grouping these assignments has been created the test cases splitting the ones from Sweden from the ones for the rest of Europe. The test cases do not consider the route calculation so the off-board system takes care of defining the most optimum path. The final evaluation of the test cases will be carried out with the task in WP5.

4.2 Simple test using the COMPANION simulator

Since the integration of the COMPANION off-board system and simulator is still an ongoing task, simple Assignment Plans are therefore formulated to demonstrate the development of COMPANION simulator as independent program. In Figure 12, the web-based visualization shows a simple case that two trucks are running on E4 and E20 and plan to meet at Södertalje and form a platoon. The simulator demonstrates the successful platooning when there is no delay at the links. When a delay is introduced on a link by setting lower speed limit due to traffic incident, one truck will be affected with a significant delay and therefore miss the merging point to form a platoon. Such a simple test has proved the functionality of the COMPANION simulator. The continuous work will make it more robust and integrated with real platooning system for final demonstration.

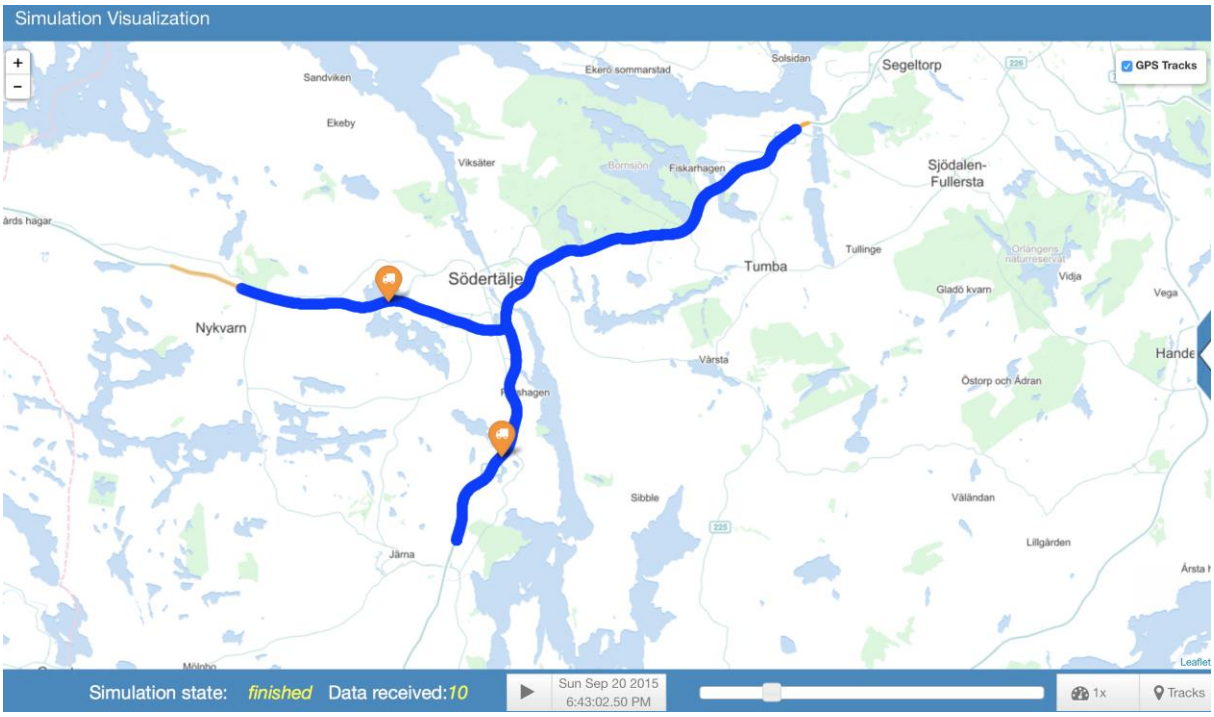


Figure 12. Web-based demonstration of a platoon formulation by 2 trucks

4.3 Key Performance Indicators

This section provides an overview of the Key Performance Indicators (KPIs) identified to assess the performance of the off-board System using COMPANION simulator. Defined from end-user perspective, these KPIs reflect the system performance on platooning, operation costs, fuel consumption and mobility.

4.2.1. Computational time for a single platooning plan

This KPI shows the total time that takes the off-board system for performing the procedures of generating a platoon plan. Simulation makes it possible to estimate average computational time for generating a platooning plan.

$$KPI = Computational\ time\ for\ a\ platooning\ plan$$

4.2.2. Messages processed per minute

One of the problems that should be avoided is the communication traffic congestion at the vehicle gateway, so in order to quantify how faster is the vehicle's gateway when processing messages going to / coming from the off-board system, the following KPI is proposed:

$$KPI = Number\ of\ messages\ the\ vehicle\ gateway\ can\ process\ per\ minute$$

Simulation makes it easy to estimate average number of the messages being processed in simulated traffic network.

4.2.3. Fuel savings platooning vs. coordinated platoon

This indicator compares the performance of fuel saving by spontaneous platooning and coordinated platooning respectively. Simulation evaluation can be carried out under different platooning strategies. Fuel consumption for each vehicle can be estimated using instantaneous vehicle trajectory.

4.2.4. Average time for re-calculation

This indicator refers to how long a recalculation of route may take. The simulation makes iterative route calculation convenient. Average computational time can be obtained.

$$KPI = Average\ time\ for\ route\ recalculation$$

4.2.5. How many recalculations were performed for a vehicle/platoon in average?

The indicator may refer to recalculation of route and transport plans. It can be considered to evaluate the quality of route calculation and optimization of transport plans. It is also an index that reflects the processing ability of the platooning system. Simulation has big advantage in estimating this value due to its capabilities to carry out different computational experiments.

$$KPI = \frac{Recalculations\ per\ vehicle}{Number\ of\ platoons\ in\ average}$$

4.2.6. Predicted fuel consumption by off-board system and simulator

This KPI shows the difference between fuel consumptions predicted by off-board system and simulated result. Thus the indictor evaluates the quality of the optimization of transport plans

$$KPI = \frac{Simulated\ fuel\ consumption}{Offboard\ predicted\ fuel\ consumption} * 100$$

4.2.7. Change in journey time and variability

The first KPI shows the journey time gain/loss when platooning against when non-platooning. It's measured in percentage.

$$KPI = \frac{Journey\ time\ before\ platooning - Journey\ time\ after\ platooning}{Journey\ time\ before\ platooning} \times 100$$

The variability is estimated by the following indicator:

$$Coeff\ of\ variation = \frac{Standard\ deviation\ of\ journey\ time\ observations}{Mean\ journey\ time\ observation}$$

$$KPI = \frac{Coeff\ of\ variation\ before\ platooning - Coeff\ of\ variation\ after\ platooning}{Coeff\ of\ variation\ before\ platooning} \times 100$$

4.2.8. Initial planned/real meeting points

This parameter shows the meeting points that have been achieved for platooning against the ones that has not been achieved due to traffic issues or another problems. It's measured in percentage.

$$KPI = \frac{Real\ meeting\ points}{Planned\ meeting\ points} * 100$$

4.2.9. Successful merging rate

This KPI shows the platoons that were planned to perform versus the ones that have been actually done. The parameter is calculated as a percentage value:

$$KPI = \frac{Number\ of\ platoons\ formed}{Number\ of\ platoons\ planned\ to\ be\ formed} \times 100$$

4.2.10. Quality of ETA

This KPI estimates how much delay the truck may experience when finishing a journey taking into account the total journey time. It's a percentage value. Simulation can be used to estimate statistically average performance:

$$KPI = \left| \frac{Time\ at\ arrival - Predicted\ time}{Journey\ time} \right|$$

4.2.11. Km driven in a platoon

This KPI counts the kilometres driven in platoon mode (as the following truck) versus the kilometres driven alone. Simulation facilitates the estimation on large network. The value is in percentage.

$$KPI = \frac{Km\ platooning}{Length\ of\ the\ route} \times 100$$

5. Conclusions

In fleet management and truck platooning, the off-board system plays an essential role to support decision-makings in updating Assignment Plans and Platoon Plans. Computer simulation has the capability to evaluate such decision-makings considering a large number of trucks being affected by different traffic incidents on the network. In order to evaluate the final performance of the COMPANION platooning system, a simulator is designed and implemented for later large-scale evaluation. This report presents detailed design and software implementation of the COMPANION simulator. The relation of the simulator with other part components is also illustrated. A simple test is performed to prove the functionalities of the simulator. For more comprehensive evaluation and demonstration of the COMPANION system, performance indicators are designed. In the final evaluation of the off-board system, these performance indictors will be evaluated using the COMPANION simulator for more elaborated scenarios in WP5.

5. References

[1] *Description of Work*. COMPANION project.
[2] *D3.1 Component specification for the overall architecture*. COMPANION project