Table of Contents

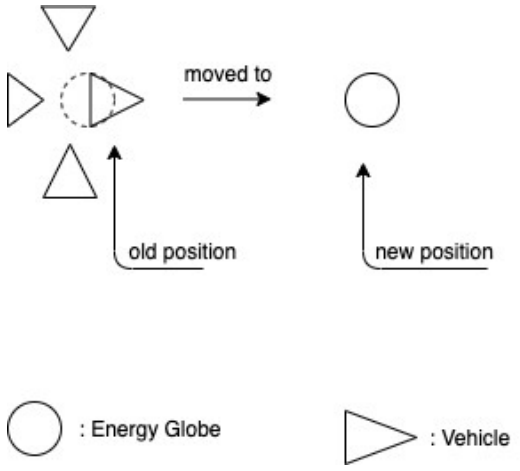
1.Brief Introduction

A swarm of vehicles need to be charged to keep them alive by passing “energy globes”. But coordinating their behaviours and resources in a concurrent way are required to avoid collision. This report consists of four stages to solve this problem. Firstly, a problem set that need to be considered. Secondly, we need to design stage to deal with this problem. Thirdly, do the stage implementation. Finally, testing stage for supporting the implementation.

2. Problem Set

2.1 Exchange Globe Location between vehicles

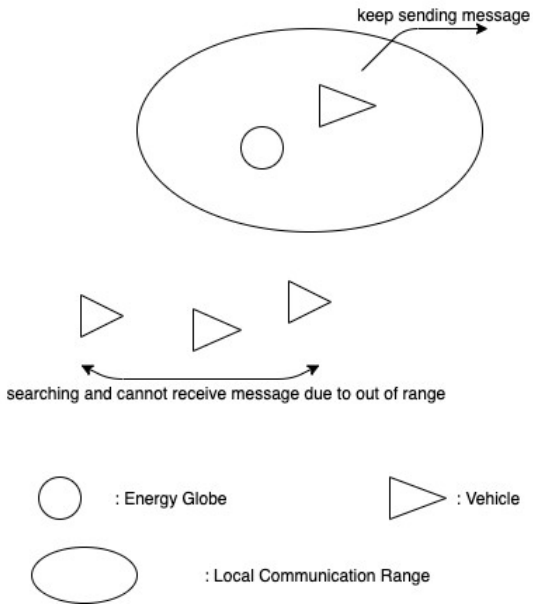
Vehicles need to be recharged by passing an energy globe. If every vehicle finds energy globes by their own, then the way is quite inefficient compared to mutual cooperated way. By cooperation, vehicles can share energy globe information to others once they find an energy globe. Due to every vehicle is able to send and receive message in a small range, message passing is a good tool to share information. In term of message, the contend of message should be consider carefully. For example, the location of energy globe is important that should be sent. What's more, the validity of message is essential. If a message is outdated, the vehicles who receive those messages may fail to get recharged as the corresponding energy globe may move away and this vehicle will run out of energy eventually (Figure 2.1.1).



(Figure 2.1.1)

2.2 Running Track

Running track is an important aspect of vehicle's live. The behaviours of vehicles after recharging would affect efficiency of vehicle's charging because message can only be exchanged in a small range. If a vehicle stays closed to energy globe after recharging instead of travelling around, message may fail to be successfully received by other vehicles (Figure 2.2.1). Moreover, due to collision-free system, vehicles may get stuck if many vehicles rush to same destinations. Therefore, design a model that controls the track of vehicles is really necessary.



(Figure 2.2.1)

2.3 Multiple Energy Globes

If the number of energy globes are more than one, the behaviours of vehicles should be changed. For example, when multiple energy globes are found, vehicles should pick one globe's position to go and recharge. Moreover, if an energy globe disappeared, some decisions should be made by vehicles.

2.4 Control Size of Vehicles

Due to the total number of vehicles are not given, it is hard to decide which vehicle should run out of energy and vanish. So, the total number of vehicles becomes a problem for vehicles to find.

3. Design Stage

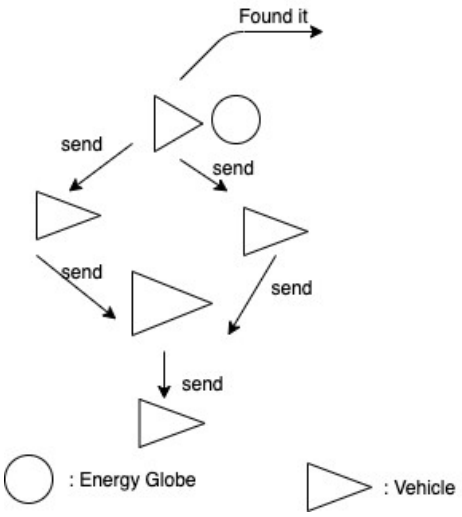
3.1 Message Passing

I decided to use message passing method to handle information sharing problem because message passing is friendly to distributed physical system for implementation.

Firstly, the content of message should be decided. I designed a message structure containing the following parts:

1. An array of positions of Energy Globes to help other vehicles to recharge (used for 3.3: multiple Energy Globes).
2. Time of creating message. In order to check the validity of the message, the time of initialising message is a way to deal with this problem. If the initialising time is too old, this message definitely outdated, and vehicles should wait for new message to avoid situation like in Figure 2.1.1.
3. Number of Energy globes found. This is for calculating closed Energy Globe for multiple Energy Globes (used for 3.3: multiple Energy Globes).
4. A conditional flag that points out whether it finds an Energy Globe or not.
5. An array of vehicles. This is for reducing the size of vehicles (used for 3.4: Reduce size of vehicles).
6. An array that indicates which vehicles should run out of energy (used for 3.4: Reduce size of vehicles).
7. An emergency array that indicates which vehicles energy are in danger level.
8. Emergency time.

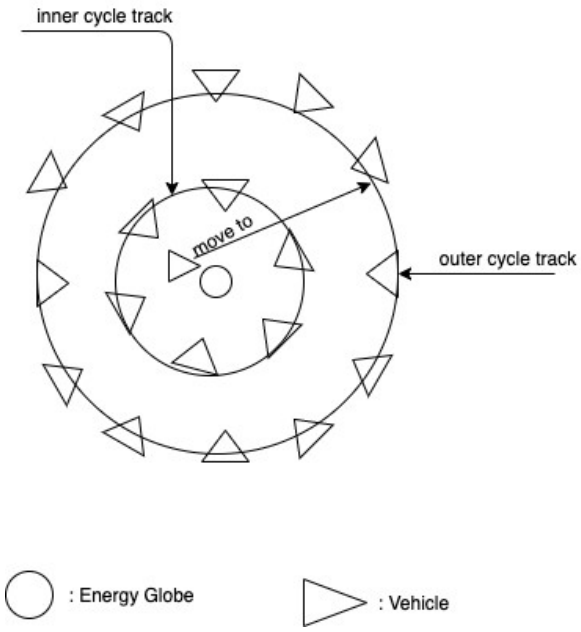
Secondly, once the structure of message is confirmed, the issue of time of sending message and receiving message should be solved. To spread the message efficiently, a vehicle who find an energy globes should repeatedly send message to other vehicles who are in the same range. And once other vehicles receive this message, they also need to send this message continuously to vehicles nearby. In this way, message can be transferred far and wide (Figure 3.1.1).



(Figure 3.1.1)

3.2 Circle Model for Controlling Running Track.

To improve performance of message passing and to avoid collision, I designed a circle model. Here is a figure shows this circle model.



(Figure 3.2.1)

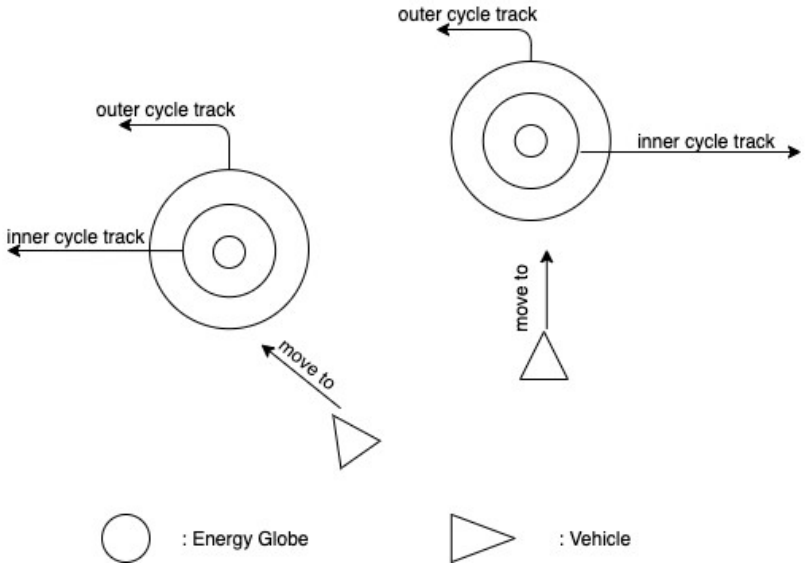
Firstly, a vehicle should go to outer circle track after find an Energy Globe in order to spread message far away. That means the message would be updated to the whole vehicles once a vehicle just gets recharged.

Secondly, in order to update message consistently, vehicles on inner cycle track should always go to energy globe. In this way, message validity can be guaranteed and reduces the chance of situation like figure 2.1.1 happened. Thirdly, conditions of determining which vehicles are on the inner cycle track should be considered. I have decided to make vehicles that have high level energy on the outer cycle track and make vehicles with low level energy on the inner track. In this strategy, low level energy vehicles can be recharged timely because they are on inner cycle track. And once vehicles on outer cycle track have low level energy, they would come to inner cycle track and wait for recharging. Finally, in order to avoid collision, vehicles after recharged should go fast to outer cycle track to give enough space to other vehicles to recharge. And if

some vehicles are in emergency of charging, the vehicles on inner cycle track should stop going to Energy Globe to make these emergent vehicles go to Energy Globe and recharge faster. An array of vehicles in emergency would be helpful (Covered in 4.1).

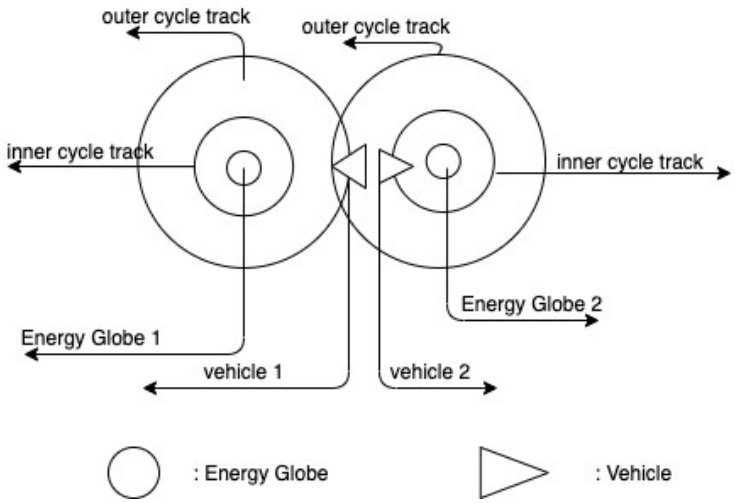
3.3 Solving Multiple Energy Globes Issue.

Firstly, for multiple Energy Globes, vehicles should find the closest Energy Globe to go for saving energy and time (Figure 3.3.1). Therefore, each vehicle should calculate distance between their current location and indicated positions in the messages of founded Energy Globes in order to find the closest Energy Globe before going anywhere.



(Figure 3.3.1)

Secondly, if radius of outer cycle track is large, the newest message would mislead other vehicles to find closest Energy Globe. For example (Figure 3.3.2), in a situation, vehicle2's energy is under emergency level and is going to closest Energy Globe. However, if vehicle1 just got charged on the outer cycle track and sent message to vehicle2 (both in a small range), then vehicle2 will go to Energy Globe 1 because vehicle1's message is the newest one. In this situation, vehicle2 would run out of energy because Energy Globe 1 is not the closest one.



(Figure 3.3.2)

To prevent this situation happening, I decided to make the radius of outer cycler track as smaller as possible to avoid overlapping.

3.4 Method for Reducing Size of Vehicles

The key point of this problem is to find some ways to get the total number of vehicles. Once we get the total number of vehicles, then we can know how many vehicles should be reduced. So, I planned to choose a vehicle X for counting the number of vehicles: Firstly, every vehicle should modify message to record themselves before sending. Then, vehicle X would know the recorded number of vehicles when receive a message, however, this number may not be the total of number of vehicles because of message delay. Finally, Vehicle X could know the total number of vehicles after a period time because message would transfer to every vehicle and back to vehicle X eventually. After counting the total number of vehicles, vehicle X could decide which vehicle should vanish.

4. Implementation Stage

The implementation stage only contains implementation for Multiple Energy Globes Mode because Single Energy Globe Mode is the foundation of Multiple Energy Globes Mode and the implementation of Multiple Energy Globes Mode would support Single Energy Globe Mode.

4.1 Message Passing

To deal with multiple Energy Globes, I added seven variables as I mentioned above (3.1) to “vehicle_message_type”. And I designed a procedure “vehicle_search_globe” for writing message and sending message. In this procedure, vehicles could save information of globes

once they found them by “Energy_Globes_Around” function, write this information to a message based on the structure above, and then send this message to around vehicles. To avoid message outdated, I set a constant value of duration. If the difference between the time of received message and the time of message created is larger than this duration, then the received message is considered as outdated and vehicles need to search new message with search speed. And the reason of message outdated is because there is no vehicle recharging for a long time (message would be updated to all vehicles immediately once vehicle after recharged due to recharged vehicles go to outer cycle track through inner cycle track).

In terms of searching new message, I implemented a search strategy. Once the message received is outdated, vehicles would go to the position of closest Energy Globe based on a function that indicated the closest globe by using this outdated message and keep listening message because Energy Globe cannot escape to the circle tracks.

In order to avoid collision, emergency array in message is quite significant. I defined a function called “Find_Emergency” to count the number of vehicles in emergency based on emergency array. If the number of vehicles in emergency is larger than the number as defined, the rest of vehicles on inner cycle track would stop rushing to Energy Globe in order to give more spaces to these vehicles in emergency. And these emergency vehicles would keep message updating after recharged. And by using emergency time, vehicles can update emergency array to new message if emergency time larger than message time.

4.2 Circle Model

Firstly, I defined a few radiuses based on the number of vehicles. The greater number of vehicles and the larger radius of cycles because many vehicles running on a small cycle would cause collision. Secondly, I defined two functions: “Find_Position” and “Find_Angle”. Find_Angle function is for calculation angle that used for calculation next position a vehicle should go. Find_Position function calculates position based on sin function, cos function and an angle because vehicles running on cycle tracks. Finally, I used “Set_Destination” function to set vehicles’ position based on “Find_position” function and “Set_Throttle” function to adjust running speed.

4.3 Reducing Size of Vehicles

Firstly, due to message contains an array of vehicles, I defined a function called “Find_Vehicle” to calculate the number of vehicles based on this array. As I mentioned in 3.4, the total number of vehicles could be calculated eventually. Secondly, I picked one vehicle X to decided which vehicle need to vanish based on Vehicle_No, the number of vehicles should be vanished can be calculated based on total number of vehicles. So, if Vehicle_No is less than or equal to reduced

number, vehicle X should set the location of run_out_of_energy array to true based on Vehicle_No and send modified message to nearby vehicles (if Vehicle_No of vehicle X is less than or equal to reduced_number, ignore setting and reduced_number pluses 1). Finally, once a vehicle finds itself should be vanished, it would never go to an Energy Globe and run out of energy.

5 Testing Stage

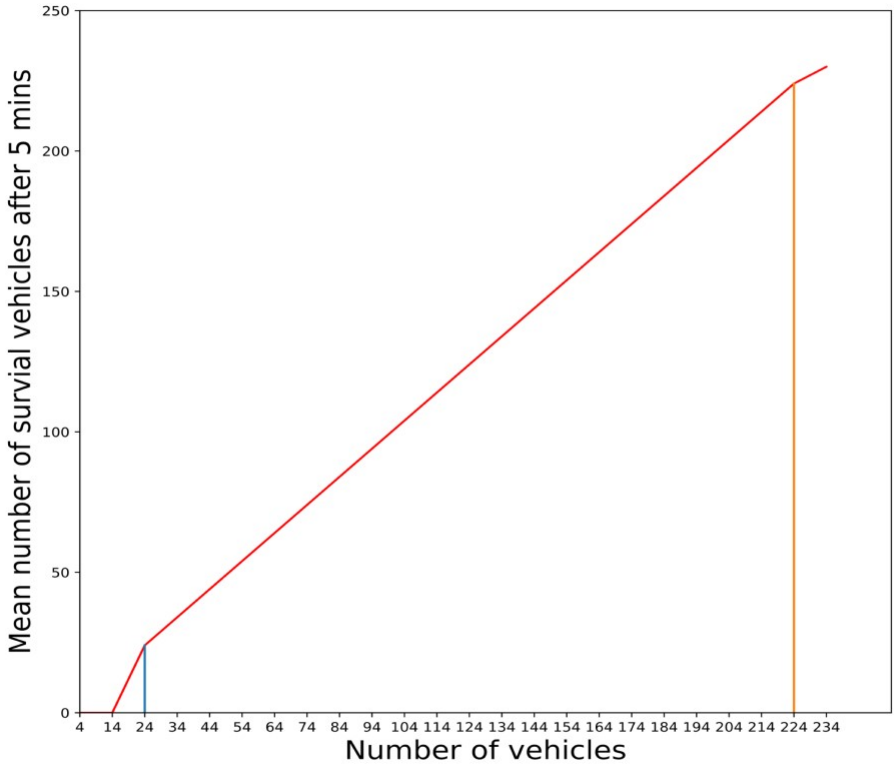
Testing stage is an important part to validate design and implementation of this system. To make this testing persuasive, we should make the testing times enough long because all vehicles could survival in a short time even though in a poor system. And to test this system's scalability, we should do enough testing cases with different number of vehicles.

5.1 Single Energy Globe

This part will test performance of this system corresponds to single Energy Globe. There are two aspects that need to be tested: all vehicles survival without any vanish (stage b) and specific number of vehicles survival (stage d).

5.1.1 All Vehicles Survival without Any Vanish

To test scalability, I started testing from 4 vehicles and did each iteration by plus 10 vehicles to 154. And to make the tests persuasive, each test would last for 5 mins and repeat for 5 times, and then recorded the average result. Here is a figure shows the result of each testing



(Figure 5.1.1.1)

From this figure, when the number of vehicles under 24, this system not works well. And there are several reasons as follow: Firstly, vehicles cannot find Energy Globe. In this case, I cannot do any improvement; Secondly, small population of vehicles just spread message in a small range and not very efficiently and some vehicles cannot receive any message all the time. To solve this issue, radius of outer circle track should be increased for spreading message further; Thirdly, small population of vehicles cannot trap Energy Globe in a circle well. For example, if message is outdated, vehicles cannot search and find Energy Globe because Energy Globe may already escape out of these cycle tracks. If I decrease radius of cycle tracks, message cannot spread further; if I increase radius of cycle tracks, this issue become more serious. So, the key for solving these problems is to find a proper radius of cycle tracks although this is quite hard. Unfortunately, I still cannot figure out a perfect radius to solve these problems for small population of vehicles.

The approximate upper bound of this system is 224 vehicles. There are some reasons cause this problem. Firstly, due to the large population of vehicles, the probability of collision would be increased. During test, I found that many vehicles with low-level energy rush to Energy globe at

the same time. This caused some vehicles stuck and run out of energy. So, I increased the radius of these circles, but it doesn't work well and even worse (Figure 5.1.1.2).

Size	outer radius	inner radius	survial		outer radius	inner radius	survial
234	0.35	0.1	230		0.45	0.1	233
234	0.4	0.1	225		0.45	0.15	210
234	0.45	0.1	233		0.45	0.2	210
234	0.5	0.1	230		0.45	0.25	205
234	0.55	0.1	231		0.45	0.3	198

(Figure 5.1.1.2)

5.1.2 Reduce Size of Vehicles.

To test this part, I set a specific size 30 and started testing from 35 to 135 because the bound of vehicles for this system is approximately from 24 to 144 (Each test runs for 5 mins). The following figure shows the result of this part test.

Size	Result	Cost time for reducing
35	30	57
45	30	59
55	30	66
65	30	75
75	30	80
85	30	90
95	30	105
105	30	110
115	30	115
125	30	116
135	30	120
145	30	128
155	30	126
165	30	128
175	30	136

185	30	136
195	30	139
205	30	146
215	30	148
225	30	160

(Figure 5.1.2.1)

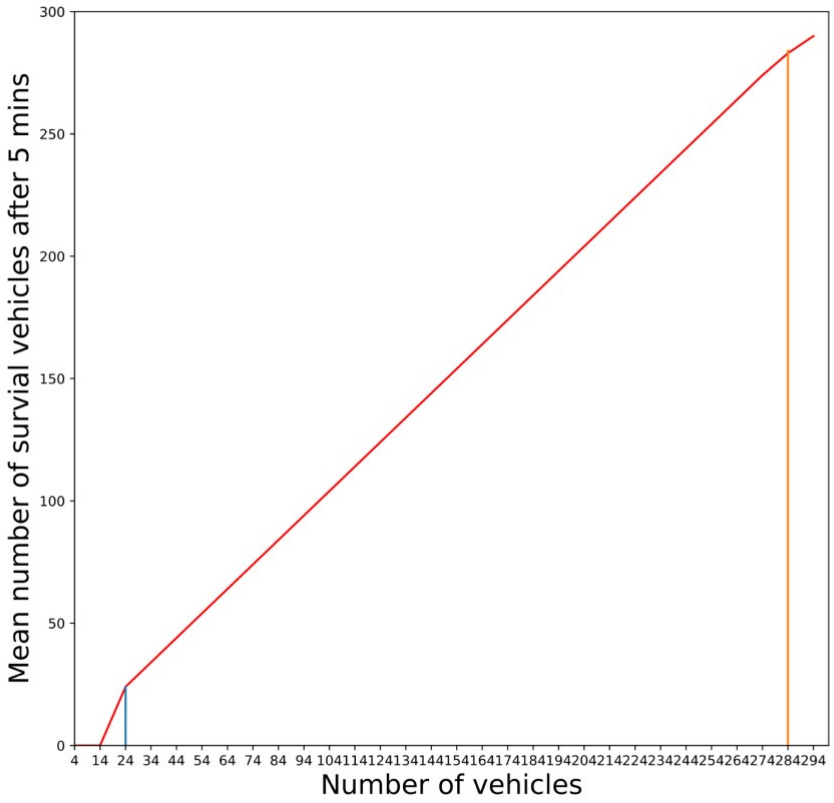
From this figure above, this system support reducing size very well. But duration of reducing is not good because of message delay. To solve this issue, we can set more vehicles to count the total number of vehicles for reducing time.

5.2 Multiple Energy Globes

Similarly, there are also two parts need to test. Firstly, test for survivability. Secondly, test for reducing size.

5.2.1 All Vehicles Survival without Any Vanish

I used the same setting as 5.1.1. And following figure shows the result of this testing.



(Figure 5.2.1.1)

From this figure, this system not supporting quite well. The bound for this functionality is approximately from 24 to 284. The reason for upper bound is that decreasing radius for large population of vehicles not works. Due to larger population of vehicles and collision-free system, the radius of cycles I set is not real radius when this system running. So, decreasing radius method cannot avoid overlapping for larger population of vehicles. What's more, the reason for lower bound is same as Single Energy Globe mode.

5.2.2 Reduce Size of Vehicles.

Similarly, this part is same as Single Energy Globe mode for testing ability of accurately reducing size. I set a specific size 30 and started testing from 35 to 275 because of the bound of this system. The following figure shows the result of this testing (figure 5.2.2.1).

Size	Result	Cost time for reducing
35	30	86

45	30	90
55	30	130
65	30	157
75	30	150
85	30	160
95	30	155
105	30	167
115	30	154
125	30	150
135	30	146
145	30	161
155	30	175
165	30	163
175	30	175
185	30	156
195	30	178
205	30	165
215	30	169
225	30	183
235	30	173
245	30	198
255	30	230
265	30	238
275	30	268

(Figure 5.2.2.1)

This figure proves that reducing size functionality work well. Similarly, duration of reducing can be decreased by picking more vehicles counting the total number of vehicles.

6 Conclusion

Based on performance that proved in test stage, this system has following functionality:

1. Fully Distributed: By using message passing, scheduling vehicles to recharge.
2. Supporting multiple Energy Globes.
3. Shrinking size of vehicles to a specific size.

However, as I mentioned in test stage, there are also some drawbacks:

1. Bad Scalability: this system only can support a small range of vehicles. But this performance can be improved by finding suitable radiuses for Single Energy Globe mode. For Multiple Energy Globe mode, vehicles may not update their message even though this message is not newest until this message is too outdated.
2. Long duration of reducing size: The cost time of shrinking vehicles to a specific size is too long. To improve this, adding more vehicles to count the total number of vehicles may be a good solution.

Due to time constrains, this system still can be improved by these ideas I talked above. Overall, this system works well and have good expansibility.