COMP2310

Systems Networks and Concurrency

Assignment 1 Report

Australian National University

1 Porblem Statement

As we know, coordinating behaviors and resources in a concurrent and distributed way is the main area in this course. In this assignment, our goal is utilizing resources that are energy globes in a concurrent and distributed approach to maintain a maximum number of vehicles alive as long as possible. To achieve this goal, we need to take some more specific aspects into consideration, which will be illustrated in the next two subsections.

1.1 Communication

Communication is an abstract concept, but in this concurrent environment, the communicating method we use to exchange information among vehicles determines the efficiency and security of our entire program. An efficient communicating method means all vehicles can get the urgent message as soon as possible so that they will have enough time to decide what to do before they unfortunately out of energy. A safe communicating method means invalid messages like those from a long time ago or those from an invalid source can be easily filtered out. In a word, a good communicating method can guarantee microscopic consistency and macroscopic efficiency and safety. To be more specific, we can summarize a feasible communicating method as the following features:

1. Message structure: messages should include all information that may be useful, which include some vital information like the location of the globe(s).
2. Sending mechanism: vehicles should efficiently send messages, which means the newest message should always be sent first than any other one.
3. Receiving mechanism: vehicles should receive messages from the entire detectable range, which means they will not miss any message that might be helpful.
4. Message storage: vehicles should store all the messages they have received and able to use the storage to select the most helpful one.

1.2 Coordination

How to coordinate all vehicles is also a significant part of this concurrent program. The main goal is to make the entire group of vehicles coordinate every single vehicle to solve time and space problem within a maximum level of concurrency and efficiency. As we know, each unit of the group is racing to get the necessary energy to keep alive, it must obey the overall policy that can keep all vehicles in order. We can also summarize some core features:

1. Overall movement trajectory: vehicles should have reasonable movement trajectory, but not need to be the same one.
2. Strategy in different energy stage: in each energy stage, vehicles should have different instructions to execute, which can ensure the maximum level of time and space utilization.

3. Intelligence(to some extent): vehicles should have the ability to adjust themselves at a high level after analyzing the messages from the concurrent world. The strategy can be changed with the detected information, like more than one globe has been found.

Our specific analysis and detailed statement of this assignment as mentioned above, which should also be the foundation to the next part. We should always consider those crucial features to design and build the most feasible model.

2 Model design

First of all, according to the analysis as mentioned above, Some requirements should be implemented by our design:

1. It can support a small number of vehicles to ensure the lower bound safety of our program.
2. It can support as many vehicles as possible to increase the upper bound of our program.
3. Vehicles can communicate the total number of globes.
4. Vehicles can change their strategy according to the number of globes, which means the single globe mode and multiple globes mode can coexist perfectly.
5. Vehicles can get their 'consensus' by communicating in the concurrent environment.

A feasible and robust model can handle both a large number of vehicles and a rare number of vehicles, which means when the number of vehicles in a range they should be extremely safe that there is almost no death case. After that, the model system should perfectly balance itself between single globe condition and multiple globes conditions. Moreover, the model should also have the ability to shrink the number of vehicles to a target number that is required by the system within a consensus function.

2.1 Pattern

To avoid most of the collision among vehicles, we need to use orbit system to coordinate vehicles around globes. Other than crowded around a globe and racing for energy recharging, all vehicles keep in a moving state within a necessary radius orbit trajectory that is determined by the amount of vehicle. Considering the number of vehicles could be large that one orbit is not enough, a set of parallel orbits can elegantly and evenly distribute all vehicles.

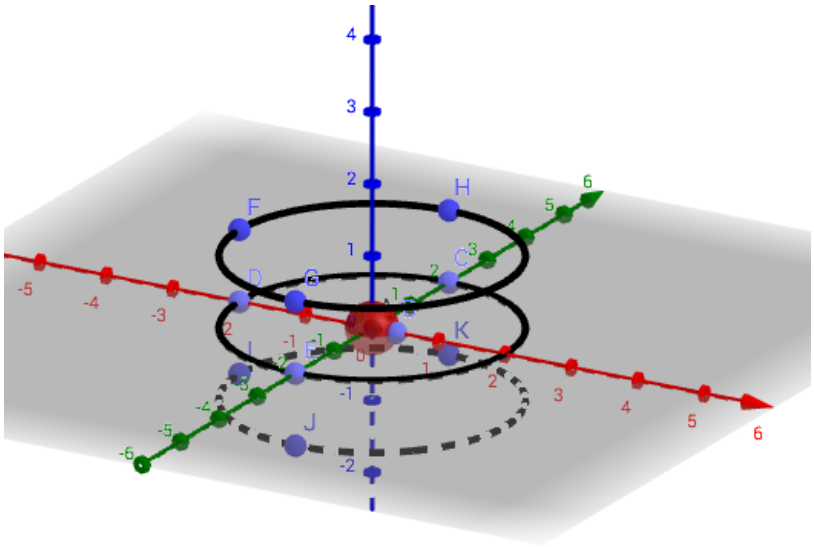


Figure 1: Orbit Schematic Diagram

Besides, if more than one globe had been detected, vehicles should divide into small groups after efficient communication among each unit. In this way, some smaller orbit systems can operate dependently after their consensus to margin into the multiple globes mode.

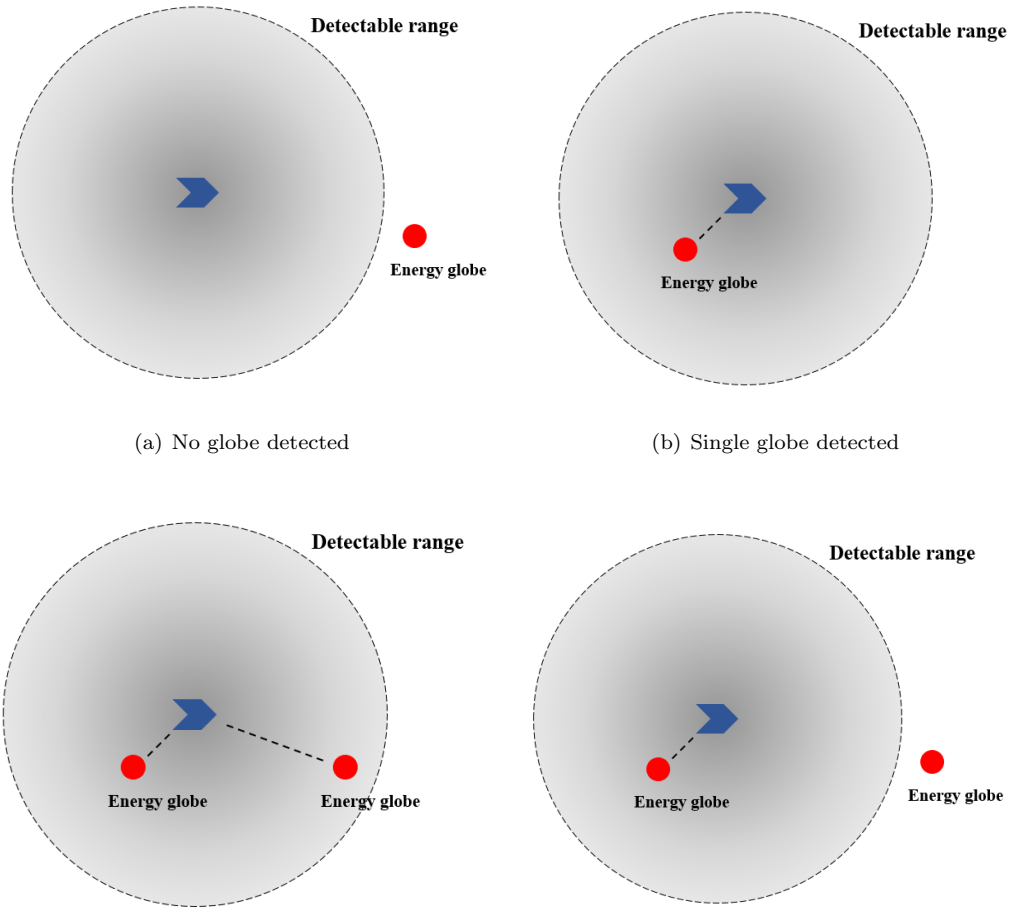
2.2 Communication

Communication is an essential part of model designing. We can actually divide this part into several subsections, which means there are smaller modules other than an entire complex communication system we need to implement in the future.

2.2.1 Globe Detection and Message Passing

First of all, detecting globe or globes is the basic functionality for all vehicles. Then, message passing should be executed right after the detection stage. Considering the number of globes is not fixed, there can be some different situation:

1. If no globe had been detected, vehicles should stay in receiving stage.
2. If one globe had been detected, vehicles should update their information record and send messages to other vehicles in the detectable range.
3. If more than one globe had been detected, vehicles should update their information record and send messages to other vehicles in the detectable range to divide into more than one group.



(a) No globe detected

(b) Single globe detected

(c) Dual globe detected

(d) Single globe detected but exist undetected globe

Figure 2: Globe Detection Schematic Diagram

2.2.2 Message Receiving

Like the diagram shown in figure 3, a vehicle in the concurrent environment is always sending messages while it is also dealing with a lot of received messages from other ones. A vehicle is requested to analyze all the messages it has been received because in the real-time concurrent environment everything can change in a transient time. Vehicles can abandon all old messages when they receive a new one, or they can store all messages in a queue or stack to analyze each message one by one. Ultimately, in the implementing stage, the message receiving part should be carefully built and consider much more details.

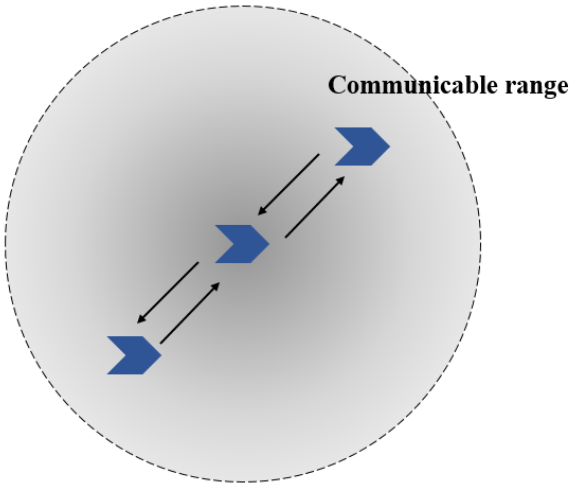


Figure 3: Message Receiving Schematic Diagram

2.3 Vehicle Energy Stage

In short, three main actions for each vehicle matching three energy stages:

1. Normal operation stage: with a high level of energy, vehicles can running in the orbit trajectory at a relatively low speed.
2. Energy supplementary stage: with a low level of energy, vehicles need to approach to a globe for energy supplementary at a relatively high speed.
3. Return stage: after energy recharging, vehicles need to go back to their orbit to make room for other ones who need to get recharged.

Figure 4 presents a brief schematic diagram.

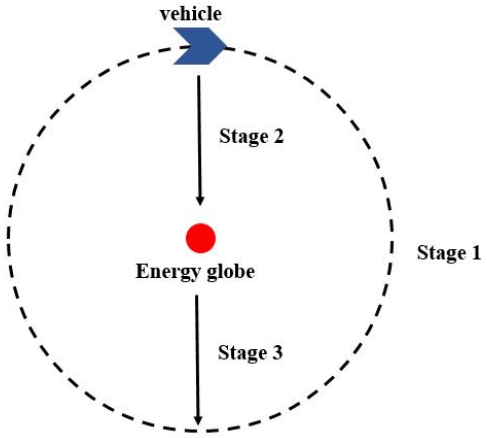


Figure 4: Energy Stage Schematic Diagram

2.4 Possible Design Flaws

After illustration as mentioned previously, we should analyze possible design flaws. The main purpose of this part is that in the implementation stage we need to face some difficult challenges what we have not taken into our consideration. So, some possible solutions countering those design flaws.

Firstly, we can enumerate all possible design flaws blow:

1. If the radius of orbit was larger than a vehicle's detectable range, the location of energy globes will not be updated in time.
2. Vehicles are running in their orbit trajectory while globes are also moving, so vehicles might lose the location of globes which has the possibility of causing a disaster.

Then, our countering solutions:

1. The maximum radius of orbit must not be larger than a vehicle's detectable range, although the upper bound of the number of vehicles that our system can support will also be limited.
2. Introducing the concept of 'Flagship', which means there are some vehicles always chasing globes. In this way, vehicles will never lose their target.

Within all these design work done, we can attempt to build a suitable model in the next part.

3 Implementation

In the implementation part, three main sections are single globe mode, dual globes mode, and consensus problem. We still choose the regular programming way to attempt to finish the implementation work of these three stages.

3.1 Single Globe Mode

Single globe mode is the fundamental part of this entire system including all essential functions and features, and dual globes mode can be regarded as several dependently operating single globe mode. In the next parts, we will build some necessary functions.

3.1.1 Orbit Trajectory

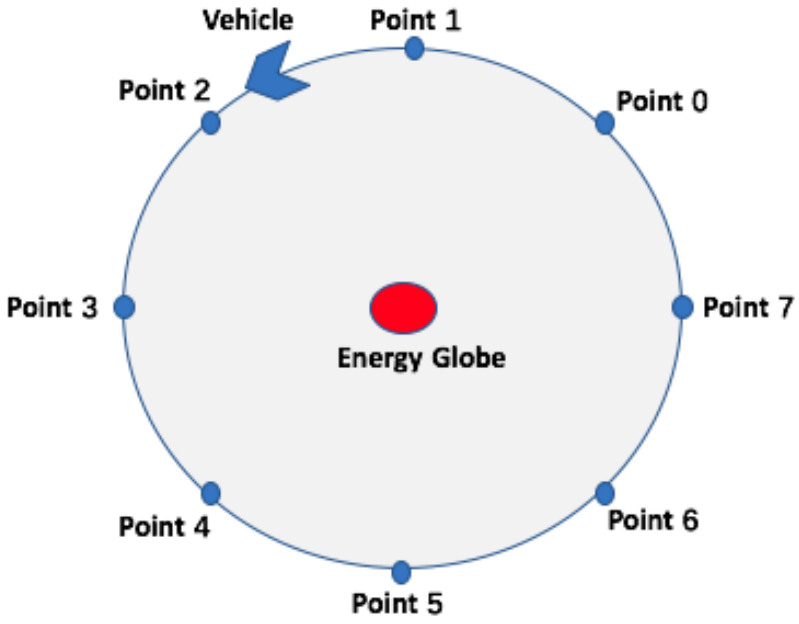
A crucial variable need to be created, which is the radius of the orbit Trajectory. This radius is automatically calculated by the total number of the vehicles. However, the orbit trajectory is not a real circle, it is an **internal octagon**, which could make the orbit seems like a circle. As we could not let the vehicles surround the energy globe in a strict circle, we need to simplify the geometric problem. An intuitionistic comparison could be found in Figure 5 on the next page.

There should be three orbit trajectories, so we need to define three arrays and arrange three groups of vehicles, which could be easily solved by mod three calculation. Each array should include eight points on that orbit like trajectory. We also use a mod eight operation to calculate which point vehicles should go, and an abs operation to calculate a value to show whether it can go to the next point of the orbit. This orbit function is the most critical implementation in this step because it makes all the movement regular and recurrent.

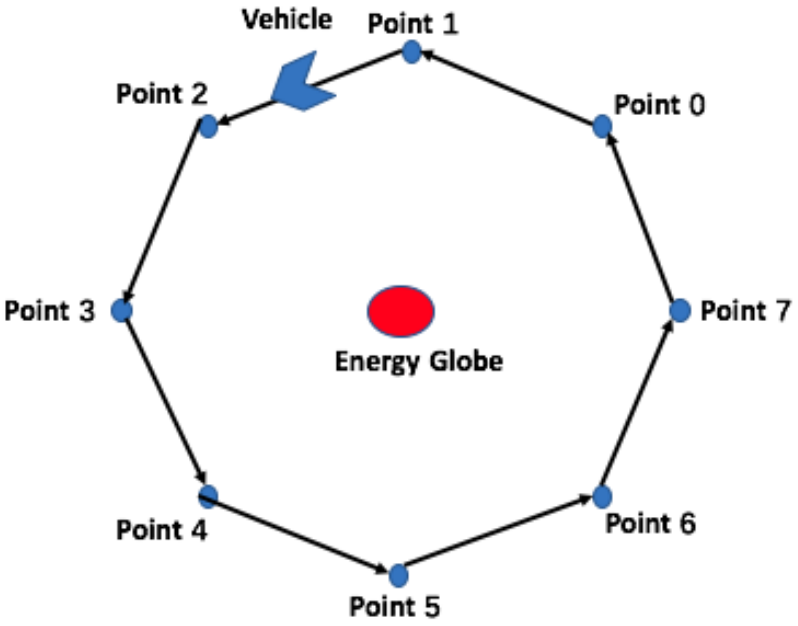
3.1.2 Message Structure

In the single globe mode, vehicles need to record three things:

1. Globe position: globe position is the target coordinates that vehicles can get recharged.
2. Message update time: message update time is aiming to filter the latest message.
3. point of the orbit: point of the orbit is used to show vehicle which point of the orbit it should go.



(a) Ideal Orbit Trajectory



(b) Real Orbit Like Trajectory

Figure 5: Orbit Trajectory and Internal Octagon Schematic Diagram

3.1.3 Globe Detection and Message Sending

In the initial stage, all vehicles should scan their detectable range to find whether there is a globe. Vehicles need to send a message right after detecting a globe, which means each globe be discovered and each message be sent. If vehicles cannot recognize any globe, they will stay in Waiting state.

There is a while loop for receiving messages, vehicles jump out the loop when then receiving messages from other ones. As many messages a vehicle can receive at the same time, it should be able to find the latest useful message. Then, after finishing the receiving loop, Vehicles need to send their local messages after an update, which means this message can be sent to a further place.

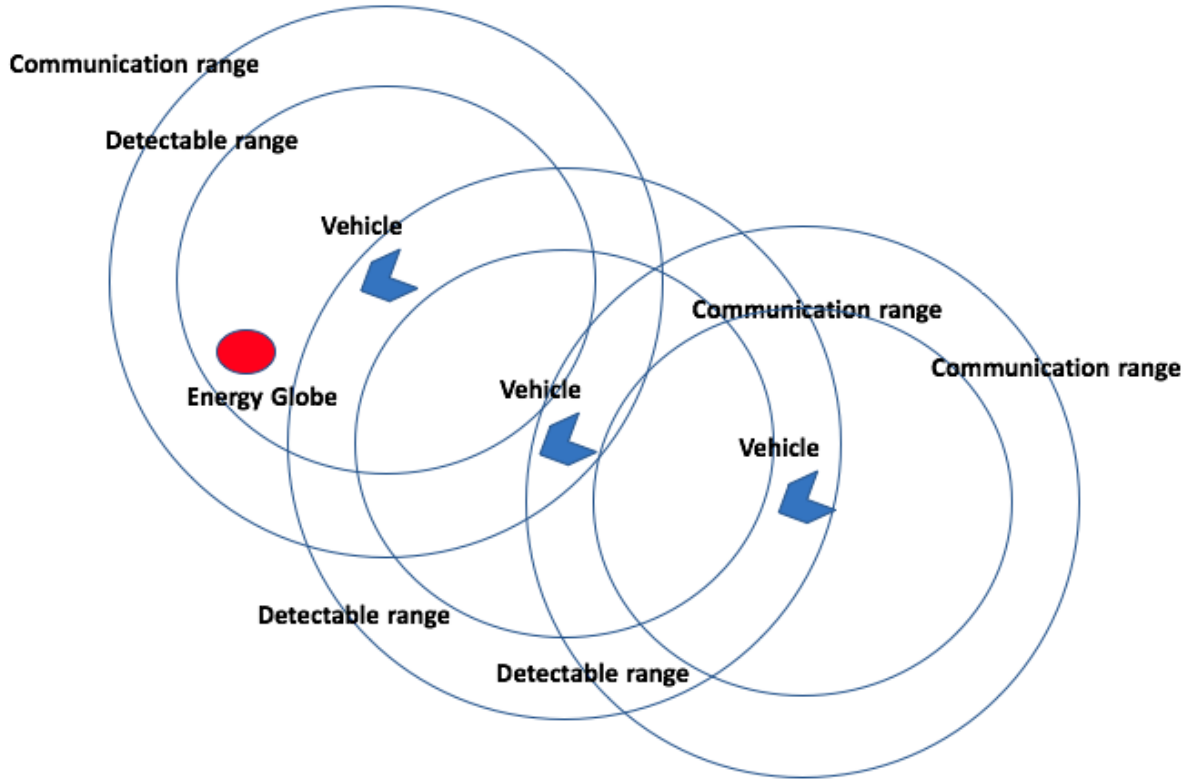


Figure 6: Message Passing Schematic Diagram

Besides, there must be someone always chasing the globe so that it can sending globe position all the time., which is known as 'Flagship'. We choose vehicle 1 to be the 'Flagship', because when the number of vehicles is one the system will still be able to support the only one lonely vehicle in the concurrent world.

3.1.4 Coordination

Our coordination strategy has been mentioned above, which can be quickly reviewed part 2.3. Vehicles check their current energy to know which energy state they are in and decide what to do. Actually, there are different throttle value in different energy state, which will make vehicles operating neatly and orderly.

Within all these functions done, single globe mode is finished. The dual globes mode will be implemented in the next part, which should be improved from the single globe mode.

3.2 Dual Globes Mode

In this part, there is not much change in message sending function and message receiving function, but we still need to improve many things. We still need more information from the concurrent world to teach vehicles how to survive when there are more than one globe exist.

3.2.1 Message Structure

In the dual globes mode, vehicles need to record three more things:

1. Source vehicle number: source vehicle number records which vehicle is the source of this message.
2. Sending vehicle number: sending vehicle number records which vehicle sent this message to the current vehicle.
3. Vehicle1 position: flagship position records the position of the 'Flagship' vehicle 1.

3.2.2 Mode Recognition

To survive in dual globes mode, a new feature that can distinguish the difference between single globe mode and dual globes mode need to be implemented. So, a boolean variable called *dualglobedetected* was created to show whether we are in single globe mode or dual globe mode, which uses a straightforward Physical theory.

If the globe was detected in different positions in a short time interval, there must exist more than one globe.

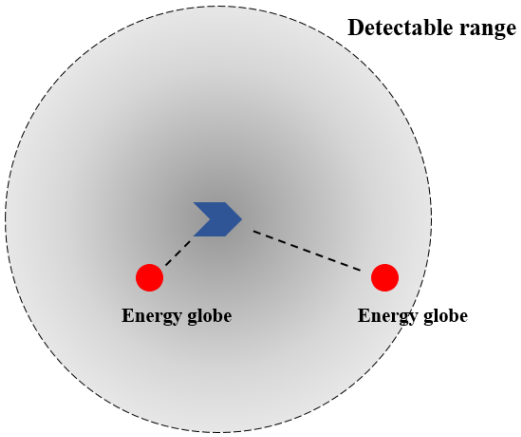


Figure 7: Dual Globes Detected Schematic Diagram

As more than one globe has been found, another vehicle should take the responsibility to be a 'Flagship'. We determine vehicle 2 to be a 'Flagship' vehicle with the same reason.

3.2.3 Grouping Method

Aiming to perfectly utilize the globes' resources, dividing all vehicles into two groups is required. After using a mod two operation, and all vehicles tend to use the message from vehicles in the same group so that they can naturally go to a different globe and obit system guided by different Flagship.

However, if they cannot receive any message from the same group, they are still able to use the messages from vehicles in the other group, which ensure they can still survive if they miss their direction.

3.2.4 Flagship Mechanism

To avoid an embarrassing situation that two flagships are chasing the same globe and ignoring the other one, we still need to add a new mechanism. There are two special vehicles in this mode, which are vehicle 1 and vehicle 2.

Vehicle 1 has the highest priority, and once it is close to an energy globe, it will filter out all the messages including a globe position which is far away from it. And vehicle 2 has the second highest priority. It will do the same thing as vehicle 1, but it will never chase the same globe with vehicle 1, which is judged by math calculation between the position of vehicle 1 and the position of energy globe(Presented in Figure 8). This function will need to use the **Vehicle 1 position** in the messages.

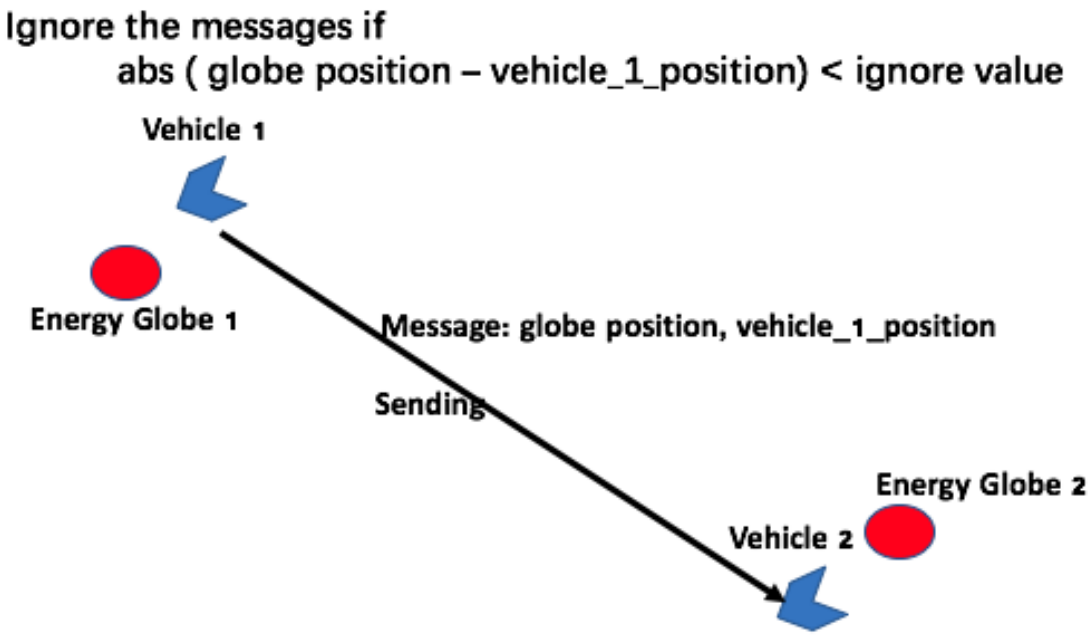


Figure 8: Flagship Mechanism Schematic Diagram

Within all these functions done, dual globes mode is finished.

3.3 Consensus Problem

After countless attempts, we finally fail to build a good function to solve the consensus problem, which can make our system automatically shrink its size. However, it is still beneficial to discuss the efforts we did.

We will discuss more details in the last part of our report.

4 Experiment

Experiment part is the most critical section of this entire project. All the designs and implementation need to be evaluated by a large number of tests. We need to make sure our system can support not only normal cases but also some bonding cases. Besides, the test unit should last for enough time, because any bad work can luckily maintain a short time. We still evaluate our model in two main parts.

Each single test unit last at least 20 minutes.

4.1 Single Globe Mode Experiment

For the fundamental single globe mode, we start with the default number 42, and try to upscale and downscale the number of vehicles to find the upper bound and lower bound of our model. Each test unit we only add or subtract only one.

The downscale process is quite good, and our system supports the bounding case, which is just one vehicle in the space. And for the upper bound, we finally found that someone, unfortunately, disappeared when the number of vehicles came into 175. Here is a graph for the test result of the single globe mode.

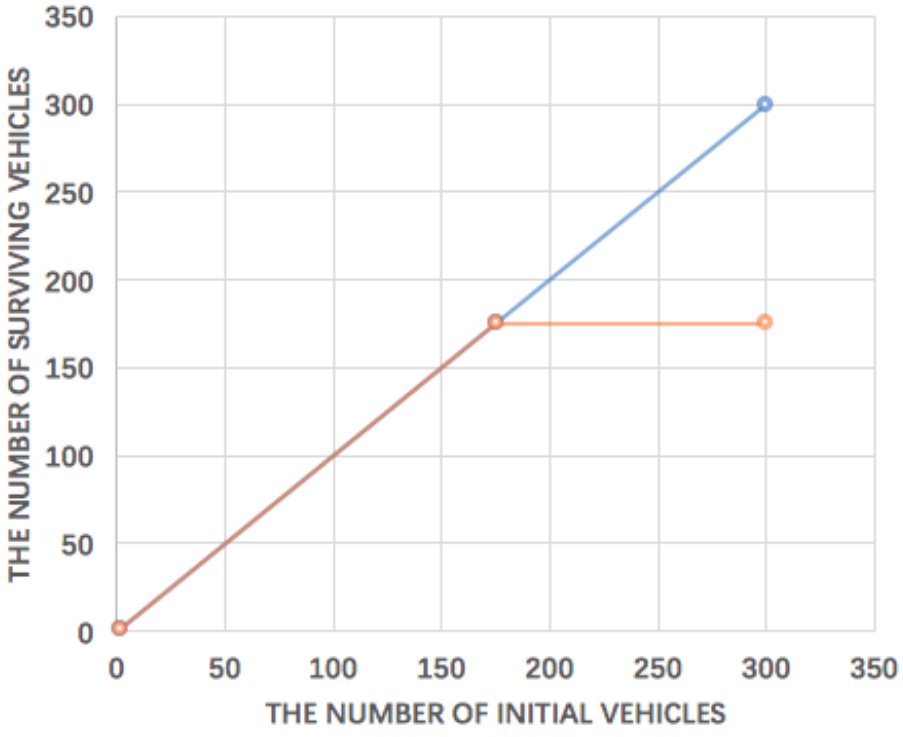


Figure 9: Test result of single globe mode

4.2 Dual Globes Mode Experiment

Dual globes mode is much more difficult than the single globe mode. After some similar testing processes. We finally found that someone, unfortunately, disappeared when the number of vehicles came into 267. We get the test result of the dual globes mode, which is presented in Figure 10.

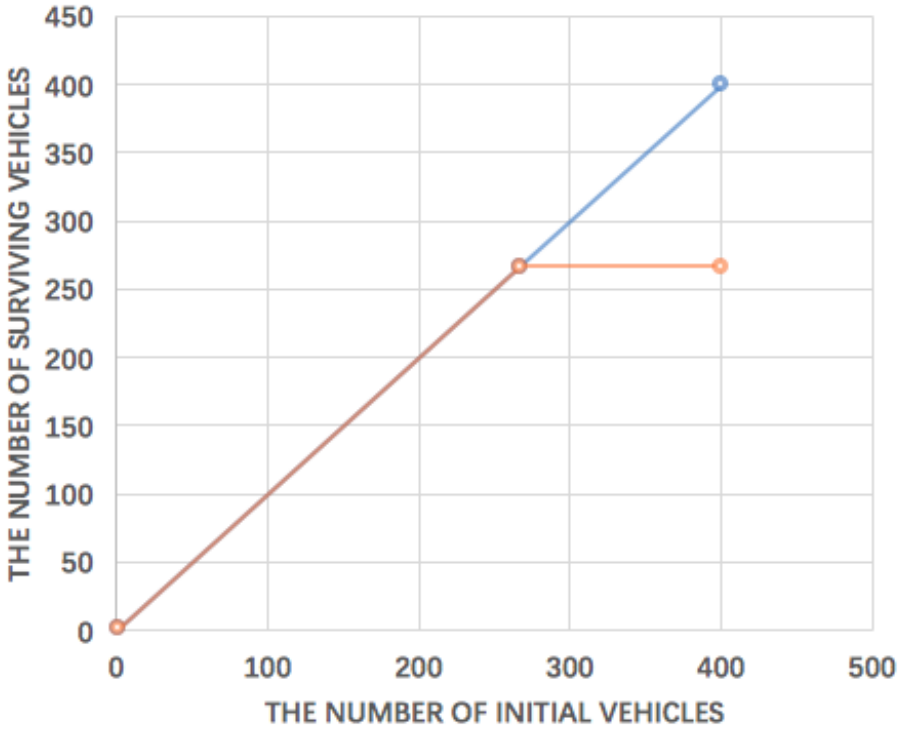
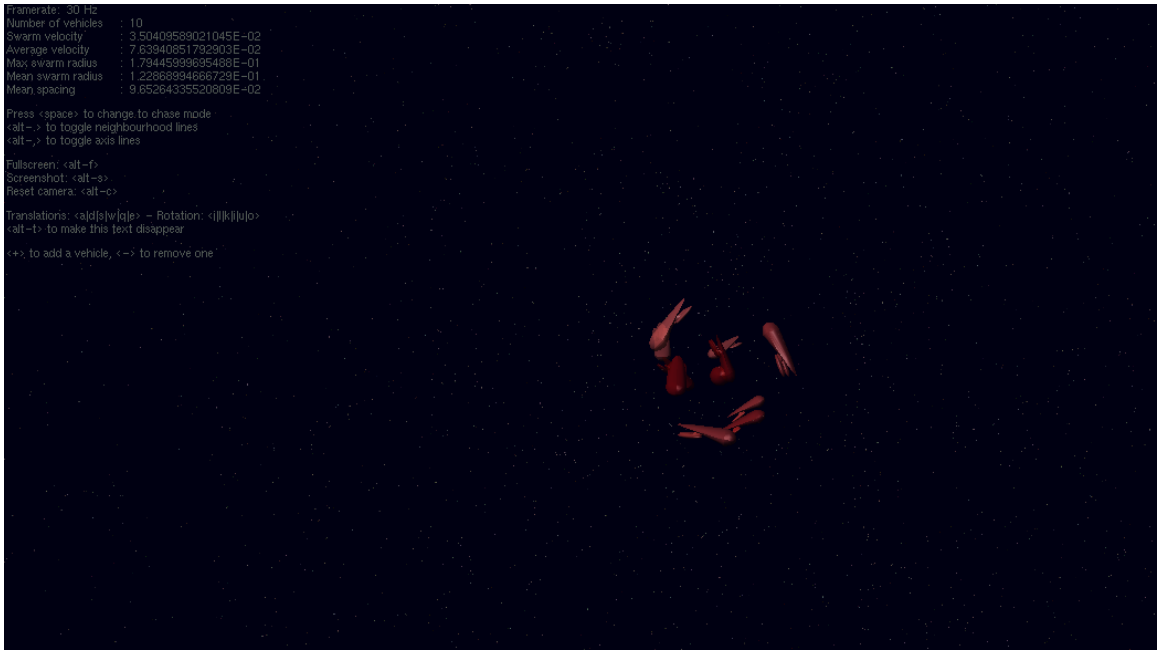


Figure 10: Test result of dual globes mode

4.3 Test Screenshots

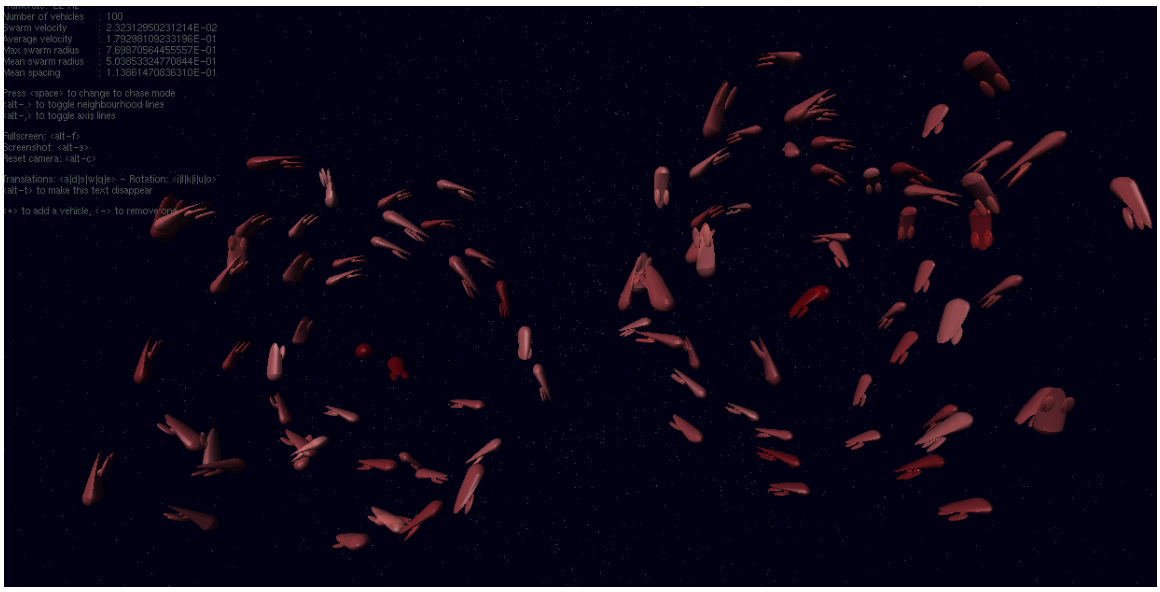
There are some screenshots to present our result intuitionistically. This model not only solves most of the problems that we have met during the implementation stage but also meets nearly all my requirements that I listed initially.



(a) Lower bound of single globe mode

(b) Upper bound of single globe mode

(c) Lower bound of dual globes mode



(d) Upper bound of dual globes mode

Figure 11: Screenshots and Achievement Exhibition

5 Discussion

Firstly, we need to check all the requirements quickly. The quality of our model should be related to these requirements:

1. It can support a small number of vehicles to ensure the lower bound safety of our program. **Done**
2. It can support as many vehicles as possible to increase the upper bound of our program. **Done**
3. Vehicles can communicate the total number of globes. **Done**
4. Vehicles can change their strategy according to the number of globes, which means the single globe mode and multiple globes mode can coexist perfectly. **Done**
5. Vehicles can get their 'consensus' by communicating in the concurrent environment. **Failed**

The conclusion is our model works. The system can always support all vehicles between an upper bound and a lower bound. Also, it can switches smoothly from single globe mode and to dual/multiple globes mode.

Next, we will discuss what we have accomplished and what we can improve in the future.

5.1 Accomplished Work

First of all, the most important reason for this success model is the excellent design for message structure. All the information included by one piece of message helps vehicles survive in each stage. For example, **Source vehicle number** and **Sending vehicle number** help two groups of vehicles efficiently follow their guider and group. **V1 position** helps the Flagship that is vehicle two easily avoid the embarrassing situation that all of the 2 Flagships are chasing the same energy globe and wasting the other one.

Secondly, the dual detected mechanism is highly efficient. Under this mechanism, only a short time need to be spent before more than one globe founded. It is an excellent concurrent application in real life because we don't know how many globes there are and it should be detected by vehicles themselves. Also, this mechanism can easily make two different policy for the two different situations so that we can do some specific things in different part.

Last but not the least, Flagships can always send the newest messages for all vehicles and ensure all vehicles in the right track. Also, Flagship mechanism is helpful to the grouping process. The orbit design is pretty beneficial too because vehicles can always utilize the space and time resource in a high level of efficiency without useless collisions.

5.2 Remaining problem

From my point of view, the Remaining problem is more valuable than out accomplished Work. Because these problems are always alerting us that there is still something we need to improve.

Firstly, vehicles in this system cannot detect more than two globes, but this problem can be solved elegantly by adding some more modular operations. Also, the dual globes detecting mechanism is not perfect so that sometimes there may be some mistake.

Secondly, Although this model performs extremely well in the small group of vehicles situation, it is not able to support a considerable number of vehicles. Here are some reasons below:

1. Flagships always occupy and waste the best time and space resources, because they are always following closely to energy globes.
2. Filtering useful messages demand time, especially when the total number of receiving messages is quite large.
3. The mathematical relationship between the radius of orbit trajectory and the number of vehicles.
4. The recharging policy is not at the best performance level.
5. The communicate distance is an essential factor, but not get used perfectly.

5.3 Consensus Problem

The consensus is a fundamental problem in distributed and concurrent systems, which is to achieve overall system reliability in the presence of many faulty processes. This often requires processes to agree on some data value, which is the task number who need to vanish during the group shrinking stage. As we did not successfully make this function, in this part, we will discuss what we have learned from the literature review stage and some ideas for solving this problem.

Our main goal is to shrink the size of the vehicle group to a required number. There are some stages during this process:

1. Proposers send a new proposal that which vehicle should vanish.
2. Accepters need to make their consensus that this vehicle should vanish.
3. If a new proposal has been made before a consensus being made, all accepters need to remake their choice.
4. If there is no new proposal before a consensus being made, the vehicle which required to vanish in that proposal should vanish.

Here we come across several problems:

1. This model requires a highly efficient method to calculate the number of remaining vehicles.
2. This model requires at least one proposer to propose periodically.
3. This model requires all other vehicles as acceptors to efficiently respond to each proposal.
4. This model requires a checking mechanism for multiple proposals check.
5. This model requires all accepters having a highly efficient method to respond to their vanishing instruction from a new consensus proposal.
6. This model requires a timer to check the condition of the entire concurrent environment periodically.

Considering all those requirements, the consensus problem is a good practice in the future, which means we need to design a much better model.

6 Conclusion

This assignment is a good practice that we indeed look underneath the hood in the concurrent world. By coordinating behaviors and resources in a concurrent and distributed way, we indeed deeper understanding of this course. We also have a more comprehensive understanding of our deficiencies. In the next few weeks, we need to learn more knowledge to handle more complex problems in the virtual environment and the real world. Indeed, we will acquire more useful expertise for concurrent and distributed systems.