COMP2310 Assignment 1 GLIDING IN SPACE
Wenjun Yang u6251843, Friday September 28

Introduction

The assignment is a resource distribution and system communication problem in a distributed system. Every drone in the system needs to have a chance to access the energy globe in a period of time. If there is no coordination between drones, some may never be able to obtain the energy. Thus, the goal in this assignment is to find a strategy to make sure every drone can obtain the energy without starvation. In other words, we have to guarantee that no drones die due to starvation in a long period of time.
The report will demonstrate
1. Problem statement, which analyses the problems in stage b, c, and d and divides the main task into two parts, namely communication problem and motion problem.
2. (1) The method for establishing an efficient communication system for message passing.
   (2) The method for building up a motion system and algorithms for coordinating drones when they prepare to get access to energy globe.
   (3) The way to measure the performance of system and how to strengthen the performance of the distributed system.
3. The strategy to improve the system when there are multiple energy globes in the system.
4. The idea of shrinking the number of drones to a specific size.

Problem Statement

In stage b, the overall design goal is to keep as many drones alive as possible for as long as possible. It is obvious that if all the drones have the chance to charge themselves when they don’t have enough energy, drones may live longer. Hence, the problem becomes how to make drones themselves fly to energy globe for charging when they are lacking in energy.

Since not every drone in the system can know the location of energy globe, the system needs a complete communication system to transfer the position of energy globe between drones so that all of the drones inside the system can know where energy globe is. Here, a new problem that the latest message of a drone may come earlier than the outdated one sent by another drone could arise. If a drone receives an outdated message like a past position of energy globe, it will head to a totally wrong area, which results in a failing charging. Thus, the messages received by drones should be selective.

Once a specific drone receives a correct energy globe location, it will head to the energy globe to obtain energy when its energy is not enough. However, if all the drones fly to the energy globe at the same time, no one can actually ‘touch’ the globe. The phenomenon here is the globe is surrounded by all the drones which try to charge but none of drones succeeds in charging. An efficient coordination algorithm between drones to avoid congestion should be presented.

In stage c, we need further coordination as more energy globes are added. From the perspective of motion system, we could optimise the condition when the drone detects two energy globes. Nothing needs change in terms of communication system.

In stage d, there are three core issues which have to be considered. The first one is how to let the corresponding drone know it’s time to die, the second one is how to let other drones know it’s not their turn to die. The last one is how to make the corresponding drone die. The previous two questions are related to message passing, so the communication system needs to be rebuilt, while the last one is about motion system, we should find a way to make the drone die.

Communication System

Sending and receiving messages are two different parts in the communication system. In my design, each drone will send its vehicle number, current charge and globe position (if the drone finds the energy globe) to the other drones. However, the information of energy globe could be old if the latest message arrives earlier. Hence, a new entity which represents the time when the sending-message drone finds the energy globe is added. So, when the new message arrives, the system will compare it with the time of last received message. If the information is outdated, the system will not update the position of energy globe. At the same time, the system will record the vehicle number and its current charge in a HashMap (Here, HashMap is introduced since the vehicle id is assigned randomly, which means we do not know the range of vehicle id. Array is excluded because the length of array cannot be defined in this condition). The data collected by the system will be used by motion system. The motion system will decide whether the drone should go or not based on the state of drone itself and its neighbour. There are some additional mechanisms which are presented in the flow chart below.

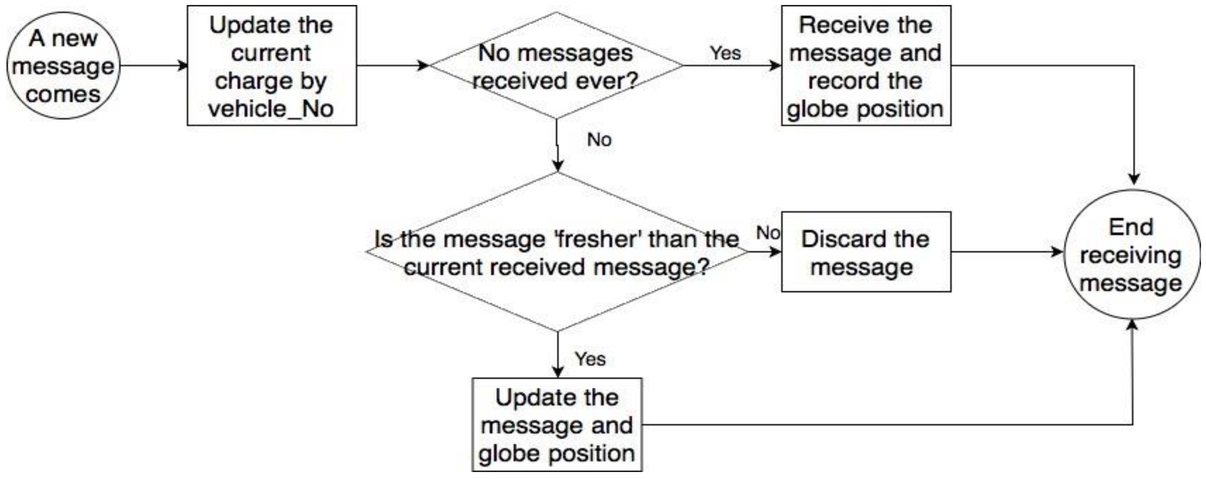

A new message comes
Update the current charge by vehicle_No
No messages received ever?
Yes
Receive the message and record the globe position
No
Is the message 'fresher' than the current received message?
No
Discard the message
End receiving message
Yes
Update the message and globe position

Figure 1: Basic working flow of communication system

Motion System

The motion system is responsible for controlling the behaviour of drones. According to the analysis in the problem statement, drones' action without coordination may lead to failing charging. My algorithm used here is really simple, that is every drone will check whether its energy is the lowest one with the help of HashMap which stores the charge of all the other drones when the charge of that drone is less than a value. If that drone finds its energy is the lowest one, it will head to energy globe for charging. Otherwise, it has to wait until the next round and then decide if it is its turn for charging. For instance, there are d0 – d9, ten drones in all. After the first comparison, if d1 has the lowest energy among d0 – d9, it will head to the energy globe and the remaining drones have to wait for the next round comparison. The algorithm is efficient because in every round, only the drone with the lowest energy has the chance to charge. This not only solves the congestion problem since only one drone can head to the globe each round, but also can keep the system stable for a very long time. However, the comparison could cause some of the drones to lose an amount of energy and then die before touching the energy globe. Hence, I add an emergency mode to the system, that is if a drone's energy is less than a specific value, it will head to the globe with the highest speed directly without comparing energy with other drones.

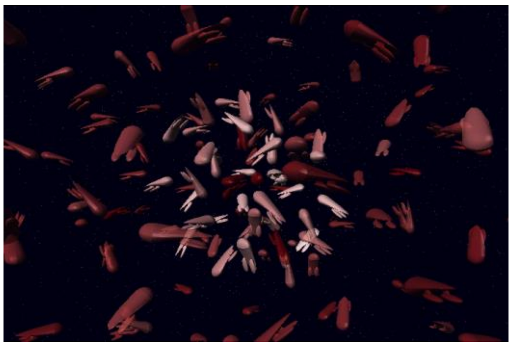


Figure 2: Drones head to globe without coordination

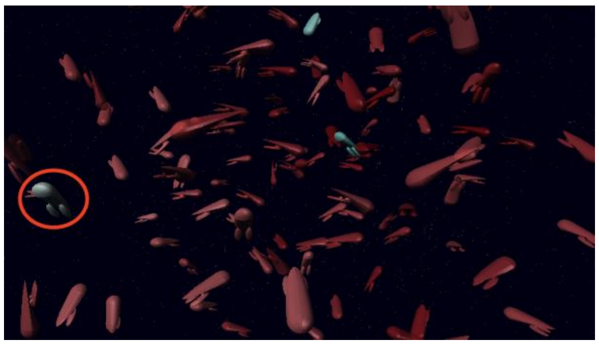


Figure 3: Drones head to globe without emergency mode

(Figure 3 shows that when all the drones try to head to the globe, they are stuck outside of the globe, Figure 4 shows that when system only has the comparison mechanism, the drone in red circle gradually die without reaching the globe)

Performance measure & Results

Measure 1 -> Set the initial number of drones to a fixed value and check how many drones stay alive after a certain time.

Test results:

Single Globe & 5 mins			
Test_No \ Initial Drone Number	50	100	150
1	50	100	143
2	50	100	147
3	50	100	145
4	50	100	143
5	50	100	143
Average	50	100	144.2

Table 4: Three different initial drone numbers tests, each one has five times tests

Test analysis:
From the table, we find the algorithm works well when the initial drone number is 50 or 100. None of the drones dies in the 5 minutes. However, when the initial number is 150, the system cannot stay stable. I notice that 3-7 drones die in the first two minutes and after that, the system stays stable. After analysing, I find the phenomenon occurs because my algorithm only allows one drone touch the globe each round, but initially, a lot of drone use up their energy at the same time. They all set themselves to emergency state and head to the globe at the same time, so congestion is generated. Hence, how to improve the usage of globe is a way to improve the system, which will be covered in the improvement part afterwards.

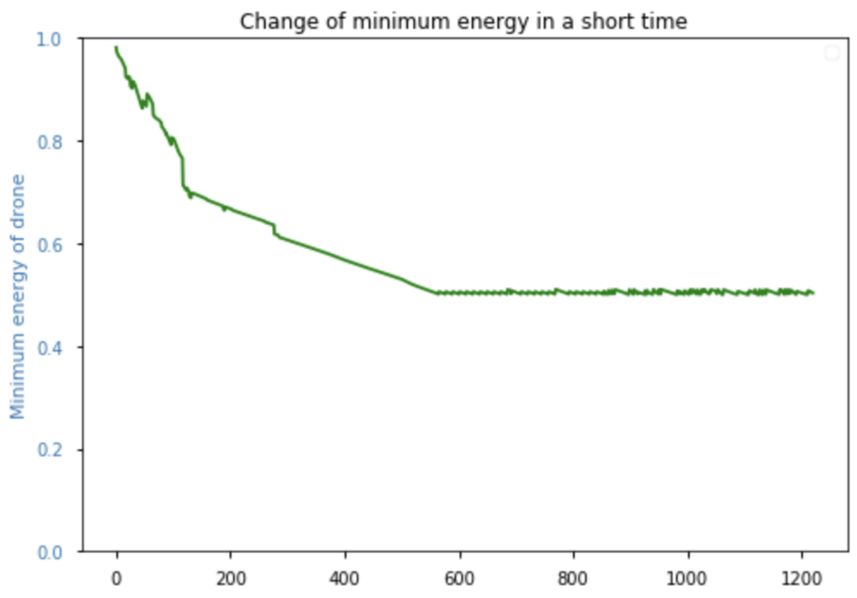
Measure 2 -> Start with a low number of drones and then slowly increase them until one of drones die.

Test results:
We start with 100 drones and increase 5 drones after the system stays stable each time. The last outcome is when the number of drones reaches 210, some of drones die quickly and the number of drones drops to 200 in the end and system keeps stable, which means the maximum weight the distributed system can bear is around 200.

Test analysis:
The result of measure 2 is not surprising when it is compared to measure 1. The system I design is hard to deal with the condition when there are lots of drones at the beginning. However, if I increase the number of drones when the system is stable, the system works better because the probability of which all the drones run out of their energy at the same time is low when the system has been stable.

Measure 3 -> Record the minimum charge among the drones and test the stability of the value (64 drones).

Test results:



Graph 5: Change of minimum energy of drones in a short time

Test analysis:
The line chart shows that the minimum energy of drones drops very quickly at first, after half of the time, the value remains stable at around 0.5 and there is no significant change since then. The result indicates that the system is stable and no drones in the system die.

Improvements

As mentioned before, the use ratio of energy globe is low because every time only the drone with the lowest energy can have chance to touch the globe. And this will cause the majority of drone head to the globe by emergency mode and congestion happens. One of possible solutions is each time we allow five drones with the lowest energy get the energy rather than one drone. The solution will improve the usage of globe in the usual time and reduce the possibility of emergency.

There are several arguments which might affect the performance, that is when the drone should begin to charge and when the drone is in the emergency state. Suppose when the energy of drone is less than L, it begins to try charging and when energy is less than E, it is in emergency state. I prepare four groups (L, E) pairs, (0.85, 0.5), (0.8, 0.4), (0.85, 0.4) and (0.8, 0.5), then test each one by drawing the diagram of minimum energy change. The result shows that when L is equal to 0.8, there is a significant fluctuation at the beginning of line and the line converges to 0.4 in the end while no obvious fluctuation happens, and the line converges to 0.5 when L is equal to 0.85, E is equal to 0.5. The (L, E) pair (0.85, 0.5) is the best one since the energy of most drones is above 0.5 when the drones need charging and most drones obtain the energy without triggering the emergency (E = 0.5), which means the use ratio of globe is high.

Reflections

There are two basic tasks for drones to meet the requirement, finding the globe and heading to globe in order.

1. Why the design will solve the problem described in the assignment sheet?

The communication system in my design is capable of finding the correct position of globe and the motion system in my design can coordinate all the drones, deciding which drone touch the globe first and in which order drones touch the globe. My design solves two basic tasks of the problem, so the design solves the problem.

2. Why I pick the design I picked?

The communication system in my design, using the real-time clock to exclude the outdating messages can quickly help drones find the position of globe. The drone will not find or receive a wrong position and waste energy. The motion system will find the drone which is most in need of energy rather than find a random one. The design is more efficient and targeted in terms of motion system than those models which pick a fixed number of drones with random vehicle_No.

Improvement when there are multiple globes (Stage C)

From the perspective of communication system, nothing has to be changed. An additional feature is added to the motion system. Originally, when a drone detects multiple energy globes around, it will only choose the first one in the globe array (Globe (1)). Now, the drone will calculate the distances between the globes and itself, choose the closest globe and head to it.

Performance measure & Results when there are multiple globes

Test result of measure 1:

Dual Globes & 5 mins			
Initial Drone Number / Test_No	50	100	150

1	50	100	143
2	50	100	147
3	50	100	144
4	50	100	144
5	50	100	149
Average	50	100	145.6

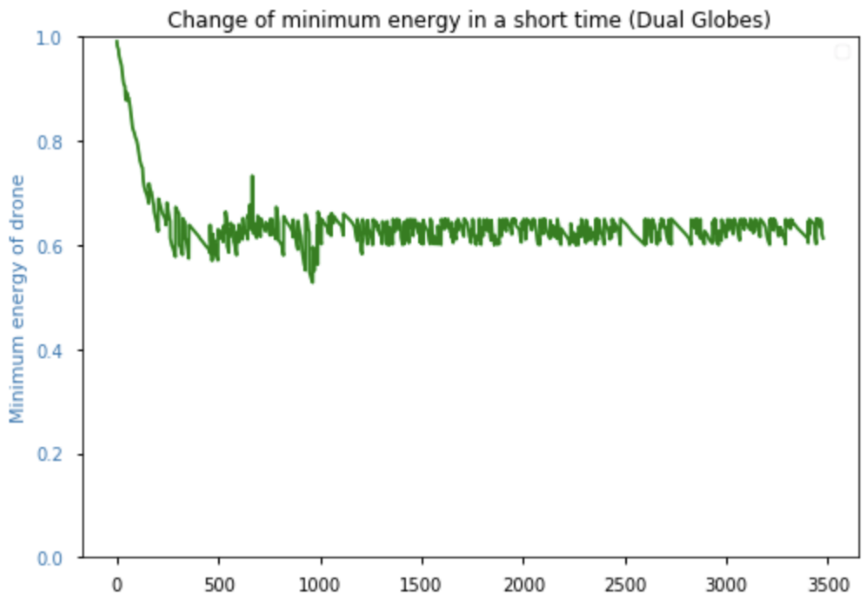
Table 6: Three different initial drone numbers tests for dual globes, each one has five times tests

Test analysis:
The same problem in single globe happens here as well. The usage of two globes is low and a lot of drones head to globes in emergency state, the congestion occurs.
However, there are two globes in the system, so the number of drones remained is greater than that of single globe (145.6 vs 144.2).

Test result of measure 2:
We start with 120 drones and increase 10 drones each time. The result is when the number of drones reaches 300, all the drones still survive. I didn't increase the number of drones further.

Test analysis:
The dual globes are the same as two 'single' globes if we add new drones after the system becomes stable. Theoretically, dual globes can support more than 400 drones if a single globe supports 200 drones.

Test result of measure 3: (64 drones)



Graph 7: Change of minimum energy of drones in a short time (Dual Globes)

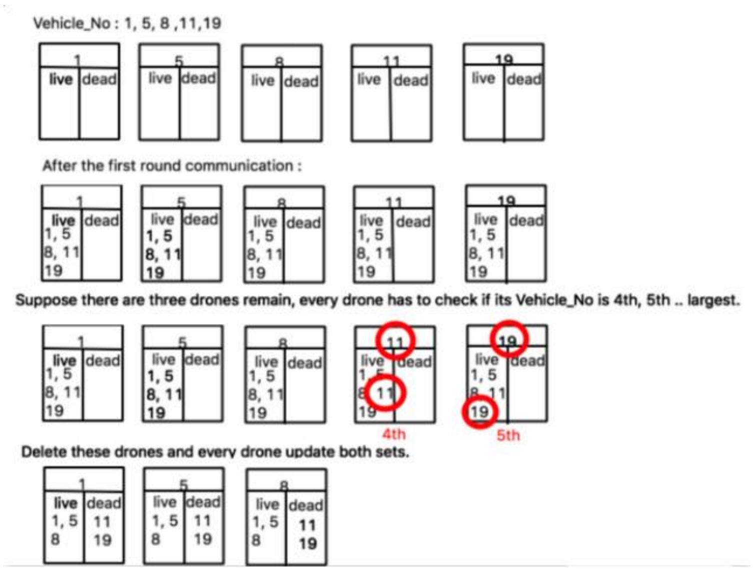
Test analysis:
The line chart shows that the minimum energy is higher than the minimum energy in single globe (around 0.6). The reason is there are two globes, so the drones are divided into two parts on the screen. The same drone spends less time touching the globe when it is compared to the single globe condition, which leads to higher minimum energy.

Shrinking task Implementation (Stage D)

There are two core problems in stage d which should be solved, the first one is how to let the specific drone know it needs to die and the second one is how to make the drone die.

For the first problem, it depends on the communication since without communication, every drone only knows its own Vehicle_No, which is useless to solve the first problem. My algorithm used here is to arrange two sets to store the live vehicles' number and dead vehicles' number. Additionally, every drone will communicate with others, swapping the information of live vehicles and dead vehicles. Suppose there are *m* drones alive now and the requirement states only *n* drones remain in the end. Every drone will look in its own set of live vehicles and check whether its Vehicle_No is the n+1 .. m largest in the set. If so, the drone is the one which needs to die.

For the second problem, it depends on motion system. Every drone is a task in the code, if we would like to make a drone die, we should finish the corresponding task immediately. Thus, my solution is that the drone drops out the loop and finishes the task. Here is an example for 5 drones and 3 drones remain in the end.



Vehicle_No : 1, 5, 8 ,11,19
After the first round communication :
Suppose there are three drones remain, every drone has to check if its Vehicle_No is 4th, 5th .. largest.
4th
5th
Delete these drones and every drone update both sets.
live
dead

Figure 8: An example to show how the algorithm in stage d works

Performance measure & Results

Measure -> Set initial drone number to m, remaining drone number to n (m >= n), testing whether there are n drones remains.

Test result of measure 1:

Drone Number Change / Test_No	64 -> 42	100 -> 50	120 -> 40
1	1	1	1
2	1	1	1
3	1	1	1
4	1	1	1
5	1	1	1

Comment: 1 represents success, 0 represents failure.

Table 9: The test result for drone number change implemented by stage D

Test analysis:
Test 15 times and all pass. The result shows the algorithm works fine.

What I have learnt from this assignment

1. Asynchronized message passing

The assignment helps me review the asynchronized message passing. The communication between the drones is asynchronized, so the outdated information may occur and disturb the operation of system. I learnt how to solve some problems by real-time clock and buffer.

2. Resources coordination in the system

I learnt how to coordinate the resources in the system for all the drones by priority. The drone with the lowest energy is set to the highest priority in the system.

3. Consensus problem solving

The stage d is the consensus problem in a distributed system. I solved the problem by introducing two sets to record the live drones and dead drones. Here, I learnt how to solve a problem with complex data structures.

4. Arguments optimisation

There are some arguments in the system which determines the system performance. I learnt to adjust and optimise the arguments in the system, trying to get the best performance for the system.

5. Ada programming and debugging

I learnt how to define and use some data structures like HashMap and HashSet in the assignment. At the same time, in order to optimise arguments, I learnt the basic Ada Input and Output. Additionally, I met many warnings and errors during the process and debugged in Ada so as to run the program.

Reference

Code Reference:
Discussing with *Chenhao Tong*, a student who is interested in distributed system design, about the overall design of system and basic algorithms used for coordination.
Discussing with *Zhaoyu Feng* (u6392260) and *Yiluo Wei* (u6227375) about the design and implementation of stage d.

Report Reference:
None.