Distributed Decision Making in A Crisis – AI help the emergency Service

By
Chen Yankun (A1763007)

Supervisor: Aufeef Chauhan, Sheik Mohammad Mostakim Fattah

Clients: Defence Science and Technology Group

Abstract

Information sharing in the distributed network helps agents make decisions. In this paper, it takes Surf Life Saving South Australia (SLSSA) as example to build a graph model. Dijkstra's algorithm is used for finding the shortest path if the connection between agent and headquarter is broken. Further, agents make decisions with average/weighted weight algorithms. Some factors, such as message's delay, message's priority and agents' waiting time, unstable network, which affect agents' decision-making, are discussed. And the agents' waiting time the most important factor which affects agents' decision-making. In future work, several directions, such as information consensus protocols, complex information analysis, applications for test, etc., are introduced.

Background

Nowadays, information superiority is a key factor for mission success. Information collection and sharing help soldiers'/commanders' decision-making, but it consumes significant resources, including expenditures and time, to finish the work from generating hypotheses, to modelling and simulation, laboratory tests and operationally relevant experimentation with Defence systems. A potential surrogate domain that operates in a similar information collection and sharing environment is the emergency services domain (ESD). In Australia, potentially relevant ESD forces include Surf Life Saving Australia (SLSA), the Police Force, the Metropolitan Fire Service (MFS), the Country Fire Service (CFS), the State Emergency Service (SES) and Ambulance services. In this paper, it takes the surf life saving in South Australia as an example. We build a graph network model to simulate relationships among on-site agents and headquarters.

The problems for an agent in a distributed network to make decisions include communications connectivity, limited bandwidth, information quality issues, difficulty in the interpretation of information and the requirement to make quick decisions in an ever-changing environment. Current solution is that information is recorded and shared by smart phone, video camera, cellular network, GPS trackers. The agents send the information to the headquarters (HQ). Then the HQ centers make decisions and send it to the related agents directly. The challenge in the problems: coordination among all of the event members and how an agent makes a decision within a limited time in a distributed network once he loses the direct connection with the HQ (server). The solution of this paper is to transfer the information through the dynamic network based on the shortest path and use the voting algorithm in a distributed network to help an agent to make a decision. We also do some experiments to check the factors which affect agent decision-making in the distributed network.

Surf Life Saving South Australia

Surf Life Saving South Australia has similar aspects of the ADF but in smaller scale. Operations typically involve: water assets (e.g. jet boats and/or jet skis); land assets (e.g. patrol vehicles); land forces (e.g. surf life savers); and a centralised coordinator (e.g. SURFCOM), with incidents coordinated over a range of information pipelines (from voice on digital radios and smart phones through to IP chat). On-site agents face similar decision-making challenges to those in the battlefield in Defence. SLSSA assets regularly have problems in communications connectivity, limited bandwidth, information quality issues, difficulty in the interpretation of information and the requirement to make quick decisions in an ever-changing environment. [1]

In a typical training exercise, SLSSA provided the following assets (with required on-site agent):

- 1 x jet boat (LIFESAVER 3)
- 2xjetskis
- 2 x inflatable rescue boats (IRBs)
- 1 x inflatable rescue boat (Boat in Distress)
- Beach Patrol at Brighton Surf Life Saving Club
- 1 x all-terrain vehicle (ATV)
- Coordination through SURFCOM
- Local incident commander, designated ‘DUTY 10’.

Figure 1 shows the map of asset locations in a rescue emergency event. On-site agents with assets communicate with headquarters (SURFCOM/DUTY10) and follow the commands to rescue the people in the incident.

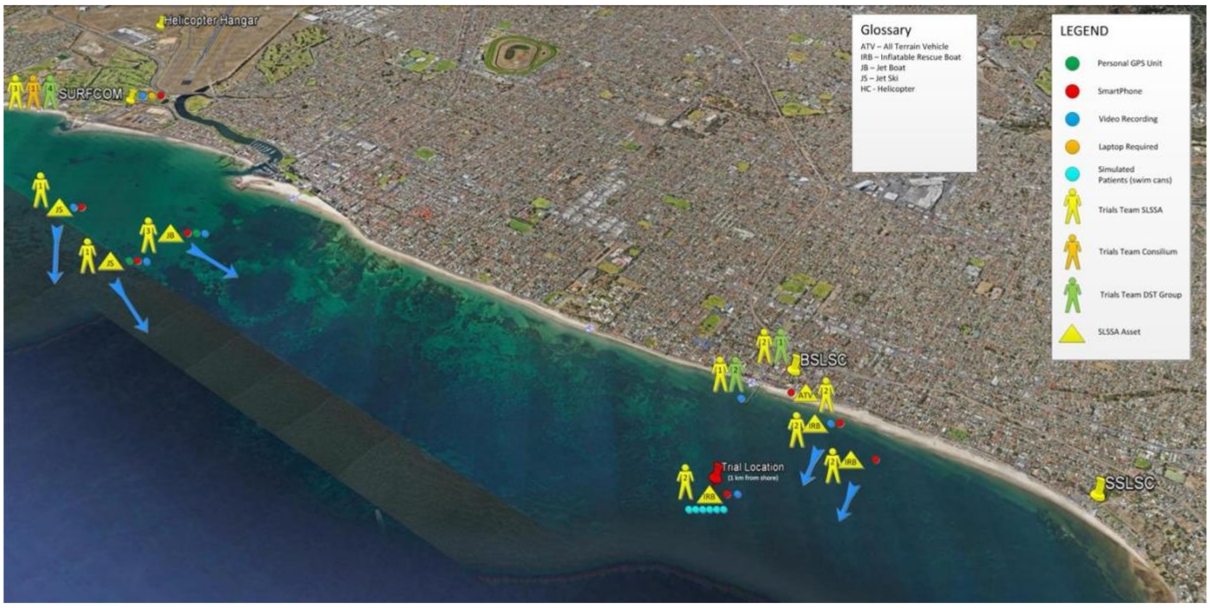


Figure 1. Map of the experiment environment

In order to analyze the problems, we convert the relationships among agents and headquarters to a graph network. In this case, on-site agents and headquarters are considered as vertices in the graph, and the connections among agents and/or headquarters are treated as edges. Consider the delay time of the connections as the weight of edges. Here, the hierarchy of the organization of SLSSA is not considered, as a distributed network is decentralized. Thus, the graph is an undirected, connected, weighted graph. The abstract graph is shown in figure 2 as follows:

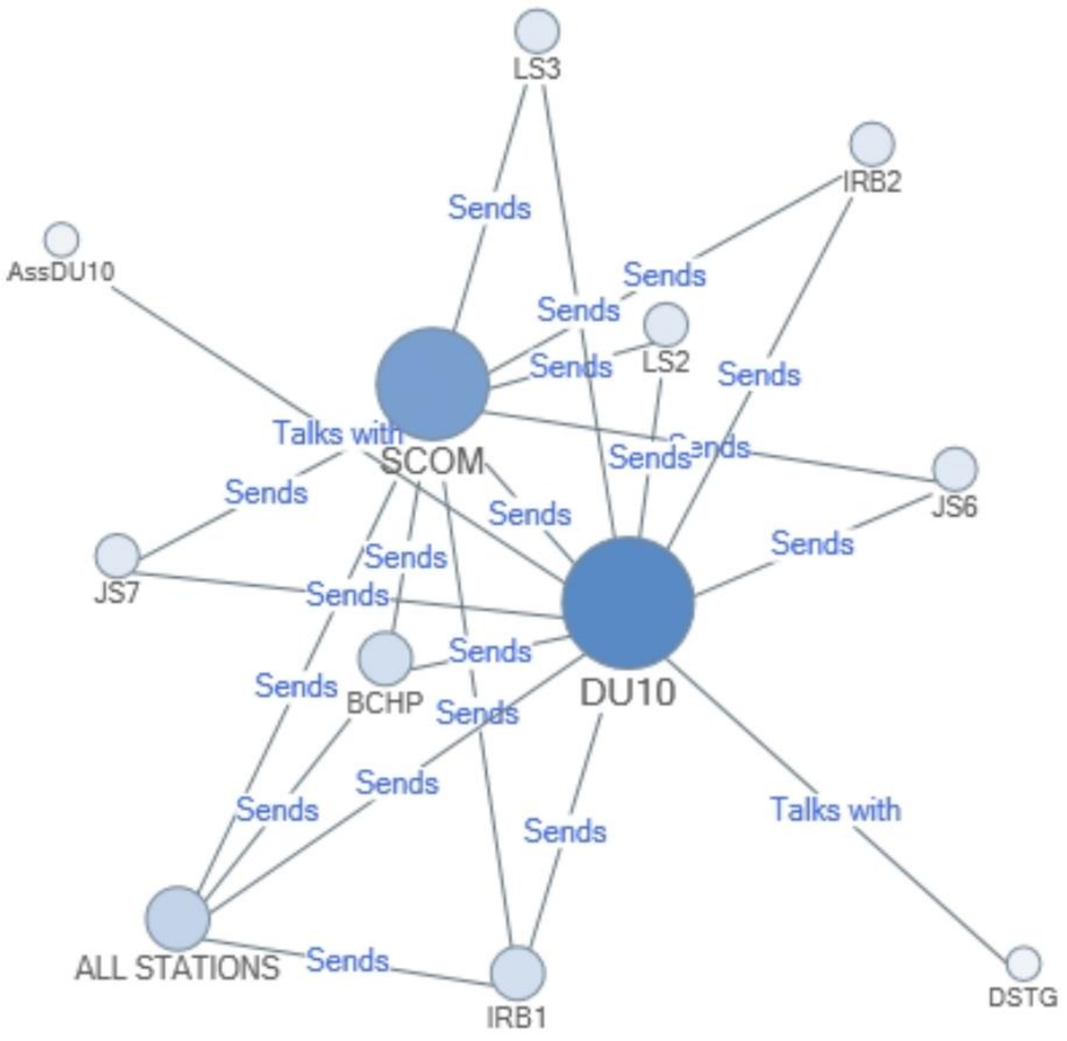


Figure 2. The Graph of Sea Life Saving in S.A.

Find the Shortest Path When the Link Lost

The first problem we want to solve is how to deliver a message if an agent loses connection with the server. There are several factors to cause the network unstable, resulting in the agent losing connections with the server or the whole network. The factors are physical obstructions, electronic gadgets and other devices, distance, technical settings, etc. When the connection is lost, the agent is able to send out messages via other paths. To find an alternative path is one of the solutions for this problem. Obviously, Dijstra's shortest path algorithm in graph theory can be used in this case. Here, we assume the weight of the edge is 'infinity' if the related connection is lost.

Dijkstra algorithm

Dijkstra's algorithm is designed to find the shortest paths between nodes in a graph. And it can be applied to the graph. In the beginning, we'll want to create a set of visited vertices, to keep track of all the vertices that have been assigned their correct shortest path. We will also need to set "costs" of all vertices in the graph (lengths of the current shortest path that leads to it). All the costs will be set to 'infinity' at the beginning, to make sure that every other cost we may compare it to would be smaller than the starting one. The only exception is the cost of the first, starting vertex - this vertex will have a cost of 0, because it has no path to itself - marked as node s.

Then, we repeat two main steps until the graph is traversed (as long as there are vertices without the shortest path assigned to them):

1. We pick a vertex with the shortest current cost, visit it, and add it to the visited vertices set.
2. We update the costs of all its adjacent vertices that are not visited yet. For every edge between n and m where m is unvisited - if the cheapestPath(s, n) + cheapestPath(n,m) < cheapestPath(s,m), update the cheapest path between s and m to equal cheapestPath(s,n) + cheapestPath(n,m).

In this algorithm, we pass each edge once, which results in time complexity of O(|E|), where |E| represents the number of edges. We also visit each node once, which results in time complexity of O(|V|), where |V| represents the number of vertices. Each vertex will be put in a priority queue, where finding the next closest vertex is going to be done in constant O(1) time. However, we use O(Vlog|V|) time to sort the vertices in this priority queue. This results in *time complexity* of this algorithm being O(|E|+|V|log|V|). [2]

Data structure

There are three main data structures used to store graphs in memory. The first data structure is called the adjacency matrix. As the name suggests, adjacency matrices are helpful when we need to quickly find whether two nodes are adjacent (connected) or not. The adjacency matrix is a 2D Boolean array of a size V^2. The second data structure is the adjacency list. In this data structure, we don’t aim to store the value of all different pairs u and v. Instead, for each node, we store the neighboring nodes only. The last data structure is the edges list. From the name, we can infer that we store all the edges from our graph inside a linked list.

The table below shows space complexity and time complexity for each structure.

Data structure	Space complexity	Time complexity	
		Connection checking	Neighbours findings
Adjacency Matrix	$O(V^2)$	$O(1)$	$O(V)$
Adjacency List	$O(V+E)$	$O(N_u)$	$O(N_u)$
Edge List	$O(E)$	$O(E)$	$O(E)$

Find the shortest path

As edges have two features: weight and bandwidth, we can use the Dijkstra's algorithm to find the shortest path with edges' weights. Also, we slightly change Dijsktra's algorithm to find the shortest path with maximum bandwidth. Thus, one method is to find shortest path only. It is used for sending text messages or audios which the message bytes are not very long, and the bandwidth doesn't have great impact to deliver the messages. The steps are as follows:

1. Find a shortest path, check the bandwidth b
2. Remove all edges with bandwidth <= b in the graph
3. Re-find shortest path
4. Compare the distance of the new shortest path with the previous one, if they are the same, go to step 5; otherwise, the previous path is the solution
5. Compare the bandwidth with the previous one, if the bandwidth b is greater or equal than the previous one, remember this new path.
6. Repeat 2 to 5 step to find the shortest path with maximum bandwidth

Find the shortest path with maximum bandwidth

The other option is to find the shortest path with maximum bandwidth by using modified dijkstra's algorithm. We define a ratio = maximum bandwidth / distance of the path. It is used for sending images or videos, which require broad bandwidth. We need to find the highest ratio among all the possible shortest path choices. Here are the steps for this method:

1. The steps are similar to option 1. Find the minimum bandwidth b in the shortest path.
2. Calculate the ratio, and find the maximum one.
3. If there is no path from start vertex to end vertex, stop finding.

Both methods can be used to find an alternative path to send messages when the connection is lost. We can decide which method to be used to improve the message transformation efficiency.

Decision Making in a Distributed Network

The second problem is how an agent makes a decision in a distributed network. In a scenario, when an agent wants to make a decision in a crisis event, he/she needs information about the situation as much as possible. How can the agent obtain the information? Here is one method to collect the information. The agent sends information requirements (such as how the situation is, who is injured, etc, or a yes/no question) to other distributed agents. Other distributed agents will send back the related information required. However, during a crisis event, there are many uncertainties. For example, the network may not work properly, and become unstable. The direct links between agents may be disconnected. In this case, to ensure that the agent receives as much information as possible on time, an algorithm is designed to solve this problem: the Dijkstra's algorithm to find the shortest path. Due to the unstable network, the agent may not receive all the information. The information includes text files, voices, images, or videos. With information, the agent needs to analyze it and then make a decision. However, to analyze text files, voices automatically, may involve natural language processing, and to process images or videos are more difficult as they require image processing and involve lots of computing. Due to the project time constraint, in this thesis, these analyses are simplified. Suppose the agent receives a Boolean value (true or false) from other agents. In this case, the agent can make a decision through a voting algorithm. Figure 3 shows an agent broadcasts requests/gets feedback to/from other agents or headquarters through a dynamic network.

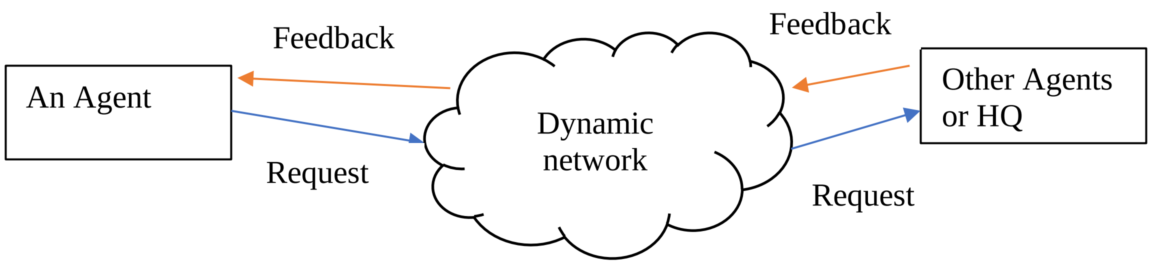


Figure 3. An Agent Broadcasting Request and Getting Feedback

The process of an agent in a distributed network broadcasting request, getting feedback is shown in figure 4.

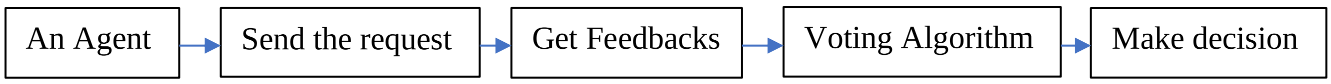


Figure 4. The Process of an Agent Making Decision in Distributed Network

Voting Algorithms and Decision Making

Two voting algorithms are implemented to help on-site agent to make decision. One is the average weight voting algorithms. In this algorithm, the messages have the same weight no matter which nodes send back to the requesting agent. The other is the weighted weight voting algorithm, in which the weight of the message depends on the number of the edges in the sending node. The more edges a node has, the more weights the message is. The reason for this algorithm is that, in our case, the headquarters have many connections with other on-site agents and there is more information and more resources to analyze the current situation. Thus, their information gets more weights.

On-site agents make decisions based on the messages they got. For example, in the average voting algorithm, if the number of messages voting for true is more than 50%, then the requesting agent makes a true decision, otherwise a false decision. For the true decision, the agent may take further action according to situations. For the false decision, the agent does not take any action. Similarly, agents make decisions based on the weighted weight voting algorithm. If the ratio of weights voting for true to voting for false is greater than 50%, agents make a true decision, otherwise, false decision.

Results Analysis

When an on-site agent sends a request to other agents/headquarters for information before making decisions or taking actions, other agents/headquarters should reply to the message as soon as possible in an emergent service event. However, the message may be delayed for many factors, such as message's delay, message's priority, waiting time, and unstable network, etc. We have designed some experiments to check how these factors affect agents making decisions.

i. **Decision Making Affected by Message Delay**

   Assume that each connection between two agents in the unstable dynamic network has the probability of 2% for disconnection from each other.

Given a limited time t, an on-site agent will make decisions based on the information he gets.

In the experiment, we suppose that the on-site agent waiting time t is 5 cycles, and the chance for delay on each agent is 50%. the on-site agent may miss the replies from other agents in the network.

The result is shown in the table 1:

Table 1 Missing relies under different probability of delay within t = 11

Prob. delay	Missing replies (total 11 replies)									
0%	1	0	0	1	0	0	0	0	1	2
50%	5	7	6	7	3	7	3	6	5	1
90%	9	11	11	11	10	11	11	11	10	9

The data shows when the waiting time is 11 cycles, the agent can receive all the messages from the network if no delay, but can receive 1 to 7 messages from the network with the delay probability of 50%, and 0-2 messages with the delay probability of 90%.

This indicates the delay time is key factors for the agent to receive the information. The more information the agent gets, the better decision he makes. It is suggested that in the real world, we should reduce the delay time of responding or speed up to process the event information time to achieve better distributed decision making in the network.

ii. Decision Making Affected by Message Priority

The priority of the message affects its sending order through the intermediate agents on paths.

Assume that each connection between two agents in the unstable dynamic network has the probability of 2% for disconnection from each other. Assume that an agent with weight less than 4 returns false for reply, greater than 4 return true. (The number of edges equals the weights of an agent). Under the assumptions above, the ratio, the weights of nodes voting for true to the weights of nodes voting for false, is 20/23. Since the majority weights of nodes voting for false, the agent does nothing. Otherwise, the agent takes action.

In the experiment, we add an extra agent A to the LS3 agent (the agent A is only connected with LS3), and suppose that the on-site agent waiting time t is 8 cycles, and there is no delay for the messages reply.

1. Message’s priority follows a Uniform distribution

   Table 2 the Agent A Make decision within t = 8 cycles

times	1	2	3	4	5	6	7	8	9	10
decision	T	T	F	F	T	T	F	F	F	F

   T = True, F = False

   The result shows the message priority affects the agent A decision making when the message priority follows a uniform distribution.

2. No priority for messages:

   Table 3 the Agent A Make Decision within T = 8 cycles

times	1	2	3	4	5	6	7	8	9	10
decision	T	T	T	T	T	T	T	T	T	T

   T = True, F = False

   Without message priority, the agent A always takes an action because the agent A receives messages from DU10 and SCOM (the most weighted agent in the network).

iii. Decision Making Affected by Limited Waiting Time

In the experiment, suppose that the on-site agent waiting time t is different, the network is stable and there is no delay for the message's reply. If the waiting time is long enough, the agent will receive all replies (total 11 replies) from other agents.

Table 4 the Agent Receive Relies under Different Waiting Time

Waiting Time	1	2	3	4	5	6	7	8	9	10	11	12
Average Received Replies (times)	0	1	2	3	5	7	7	8	9	10	11	11

The result shows that the longer the waiting time, the more messages the agent receives. It suggests that the agent will get more information from other agents if waiting long enough. However, when the waiting time is greater than 11 cycles, there are no more messages to receive.

iv. Decision Making Affected by Unstable Network

In the experiment, suppose that the on-site agent waiting time t = 11 cycles, and there is no delay for the message's reply. The network is unstable, that is, each link between two agents will be disconnected with different probabilities. (If the network is stable, the agent will receive 11 replies in total.)

Table 5 the Agent Receive Relies under Different Links Dropping Rate

Link Dropping rate	0%	1%	2%	3%	4%	5%	6%	7%	8%	9%	10%
Average Received Replies(times)	11	10.5	10.3	10.0	9.8	9.4	8.9	8.5	8.1	7.5	6.4

The result shows that the agent receives the messages decreasing when the links of dropping rate increases. In the worst case, the agent drops off the whole network and can't receive any messages. Once the agent is connected to the network, he/she can receive the messages within the waiting time.

Conclusion

Agents make decisions based on the information they have. All these factors affect the information that an agent can receive. We can improve agents decision-making by reducing or eliminating these factors. For example, if necessary, the agent could wait long enough to get all the information before making a decision. Or reduction of message's delay time can be done by improving processing of the information with AI technology. In our case, through the shortest path algorithm, agents can receive messages through unstable networks. Even if the agent receives messages sent by part of agents, he/she can make a decision if the total weight of received messages is greater than 50%. When considering all the factors, the most important factor is waiting time as the agents connect with the headquarters in our case, message priority have less impact on agents' decision-making. Also, if the waiting time is long enough, the agent can receive the message . However, the limitation is the agent may not get enough information if he/she loses all connections from the network, and can't make a proper decision.

GUI Design

There are many libraries, such as Tkinter, in python to develop GUI. Tkinter is a combination of the Tcl and Python standard GUI frameworks. It is suitable for developing desktop apps. [4]
The Advantages of Tkinter are:

1) Supporting the recent version of Python, it is most likely already installed.
2) It offers a vast collection of well-known widgets, including all the most common ones, like buttons, labels, checkboxes, etc
3) Adding code to each widget is straightforward.

But it also has limitations, that is, the version of Tkinter should be ensured for supporting the extended set of Ttk widgets.

Introduction of Program

To simulate the functions and solve the problem, the program contains four classes: Graph, Vertex, Edges, Message.

The Graph class includes the other three classes: Vertex, Edges, Message. The Graph object stores all of vertices and edges objects and has functions to manipulate them, such as add, delete, modify functions. It also includes the network status updating function. A vertex object includes vertex name, address, connecting edges, send/receive buffers, etc. It can generate a broadcasting request or reply message object, and handles received messages. An Edge object contains a start vertex object and a send vertex object, weight, bandwidth, etc. A Message object has message id, source, destination, priority, vote, delay, and reply, etc., parameters.

The main function includes initializations of graph and GUI. The procedure is shown in figure 5.

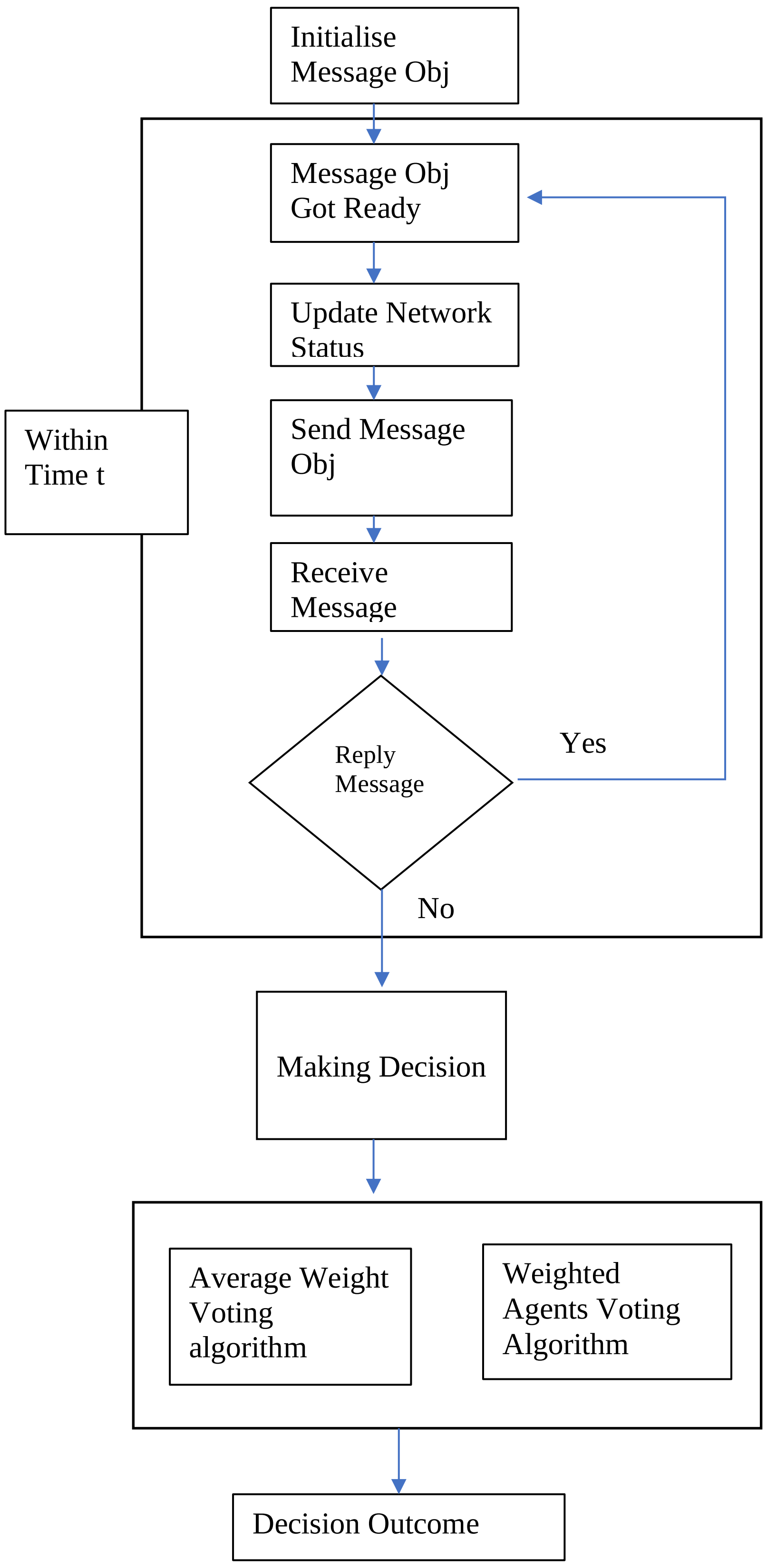


Figure 5. The Procedure of the Program

Explanation of GUI

When we simulate the dynamic network, and want to modify the vertices or edges or show the shortest path from one vertex to the other. We could use the console terminal to check the result. However, it is more convenient to use a self-defined GUI to test all functions we need, showing as figure 6.

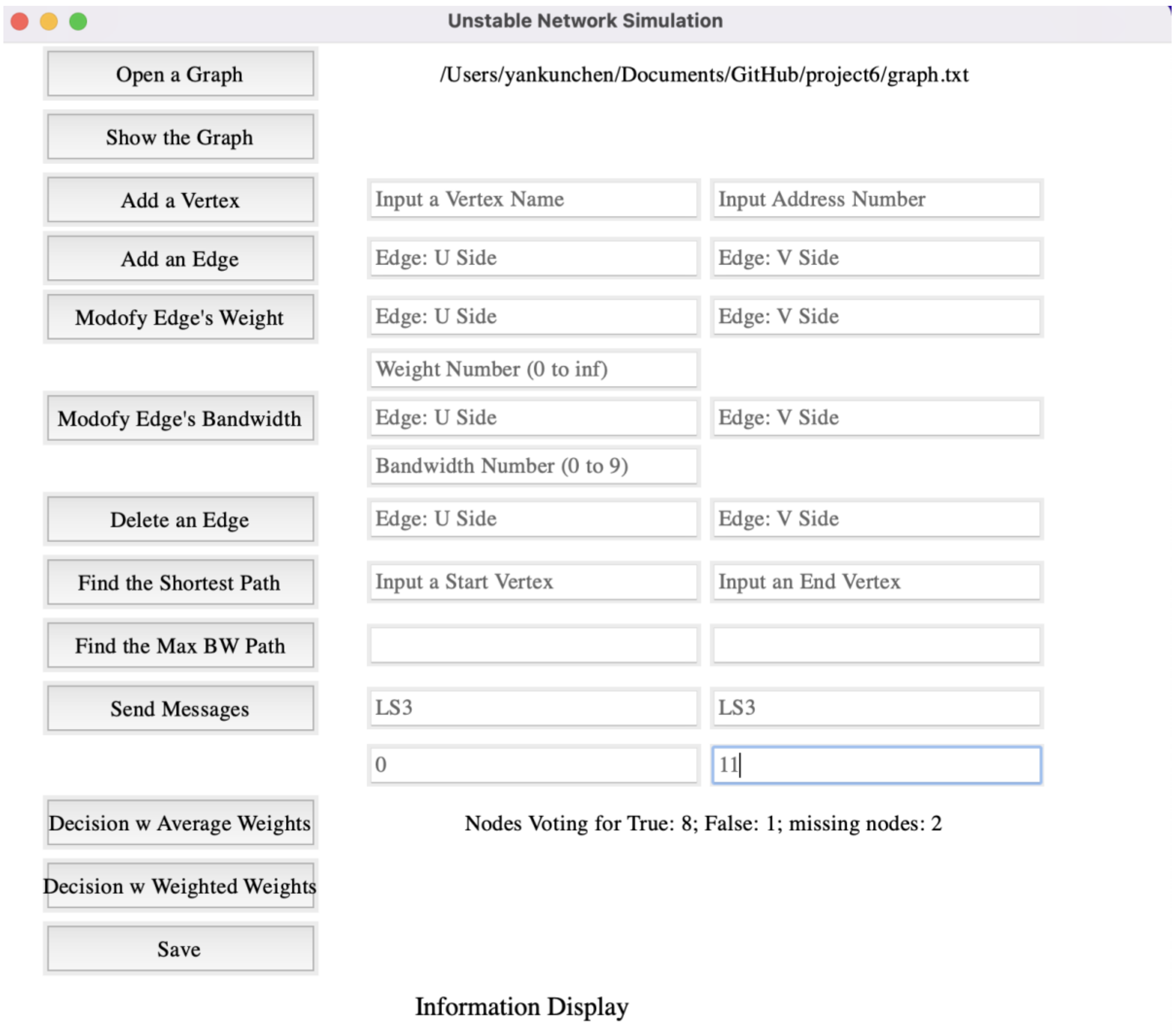
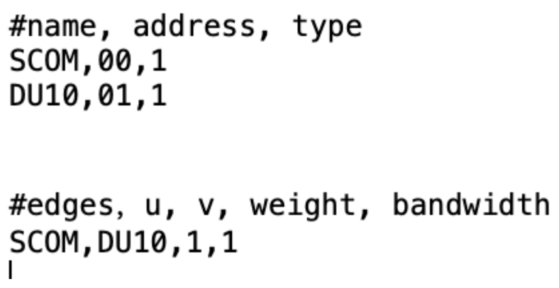


Figure 6. GUI for Testing Functions of Dynamic Network

Here is an explanation of the GUI. Firstly, the GUI can open a graph in a txt file, shown in figure 7. The .txt file format is as follows:

There are two parts in the txt file. The first part includes nodes (name, address, type), the second part includes edges (u, v, weight, bandwidth)..



```
#name, address, type
SCOM,00,1
DU10,01,1


#edges, u, v, weight, bandwidth
SCOM,DU10,1,1
```

Figure 7. Graph in .txt File Format

After we load a proper graph txt file, click the ‘Show the Graph’ button. It shows the visualized graph of the txt file. One of the examples is shown as Figure 8.

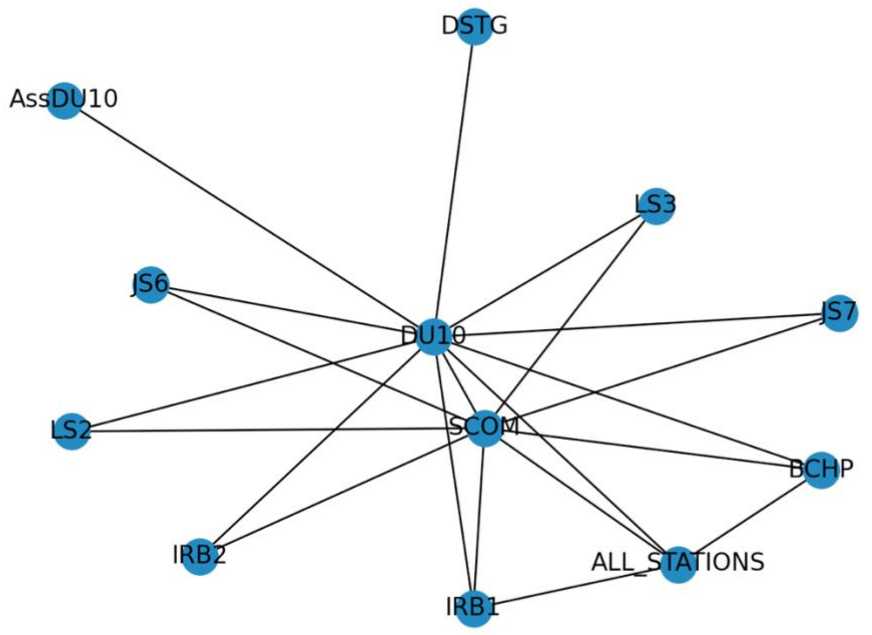
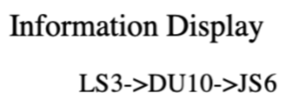


Figure 8. A Graph from the Txt File Input

We can add a vertex, edges to the graph. The weight and bandwidth of an edge can be modified as well. If modification is done, the message about modifying succeeded or failed will be displayed in the ‘Information Display’ box.

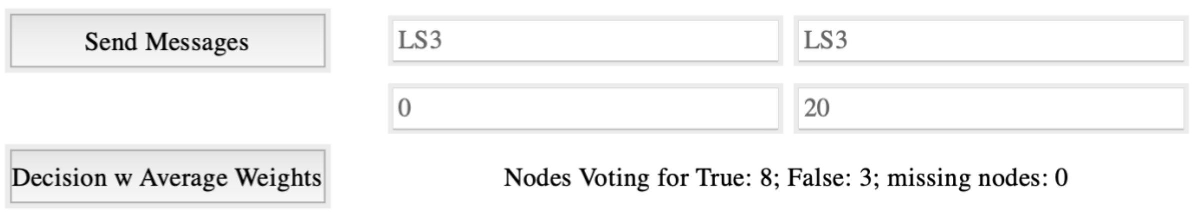
An edge in the network also can be deleted to simulate the lost connection between two nodes. We can find the shortest path with/without maximum bandwidth from a vertex to any other vertex in the network. The result is shown in the ‘Information Display’ box. For example, we can find the shortest path between the LS3 node and the JS6 node, and the path is shown as figure 9.


Information Display

LS3->DU10->JS6

Figure 9. The Shortest Path Between LS3 and JS6

The most important function is the voting algorithm with average weights. That is, an agent broadcasting his/her request to the whole network, and get feedback from other agents. For example, showing in Figure 7, LS3 broadcast his/her request to the whole network(LS3 is start node and also the destination node to receive the feedback), put 0 (1 for sending message from start vertex to end vertex) parameter in the left-bottom box for broadcasting type, and waiting time is set to 20 cycles. LS3 can make a decision based on the received messages. Click the 'Send Message' button, then click the 'Decision w Average Weights' button, LS3 receives 8 nodes voting for true, 3 nodes voting false. The result is shown as figure 10. For example, LS3 can take a pre-set action based on the result. If it receives less than 50% True, the LS3 doesn't take any action.



LS3 LS3

0 20

Nodes Voting for True: 8; False: 3; missing nodes: 0

Figure 10. LS3 Broadcasts and Gets Feedbacks

The function of the voting algorithm with weighted nodes is the same as the one with average weights. Click the 'Send Message' button, then click the 'Decision w Weighted Weights' button. The result will be shown at the right of the 'Decision w Weighted Weights' button.

Future Work

In the future, there are several directions to focus on.

Information Consensus in Distributed Network Protocols

In distributed computing or multi-agent systems, consensus makes sure coordinating processes agree on some data value during computation. Real-world applications requiring consensus include cloud computing, clock synchronization, PageRank, opinion formation, smart power grids, state estimation, control of UAVs (and multiple robots/agents in general), load balancing, blockchain, and others. One method to generate consensus is for all processes(agents) to agree on a majority value. In this case, it requires more than 50% votes (where each process is given a vote). However, if one or more processes fail, it may lead to the uncertain consensus, and cause the incorrect outcome. Protocols are designed to solve such consensus problems under limited numbers of faulty processes. These protocols need to meet a number of requirements to be useful. E.g. A trivial protocol could have all processes output binary value 1. Another requirement is that a process may decide on an output value only once and this decision is irrevocable. [3]

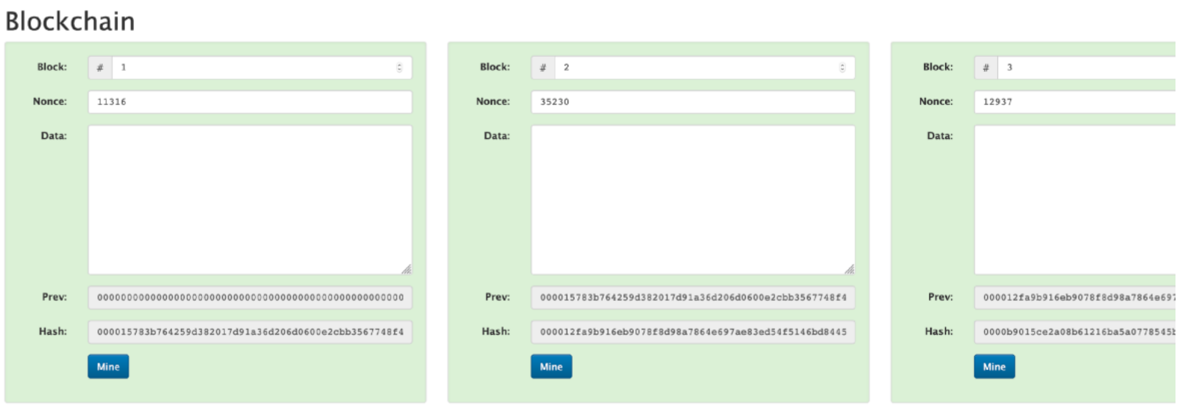
There are some properties for a consensus protocol:

1. Termination: eventually, every correct process decides some value
2. Integrity: If all the correct processes proposed the same value v, then any correct process must decide v.
3. Agreement: every correct process must agree on the same value

Blockchain is one of the most famous information consensus protocols.

Blockchain Technology

The blockchain is a database, a distributed ledger indicating a network agreement of data (e.g., transaction) that has ever happened. To be more specific, blockchain combines many blocks, each block includes a block number, a nonce number, a cryptographic hash of the previous block, a timestamp, and cryptographic hash of data, shown in figure 11.The cryptographic hash function is SHA-256, which is computed with eight 32-bit words.


Blockchain

Block:	# 1	Block:	# 2	Block:	# 3
Nonce:	11316	Nonce:	35230	Nonce:	12937
Data:		Data:		Data:	
Prev:	0000000000000000000000000000000000000000000000000000	Prev:	000015783b764259d382017d91a36d206d0600e2cbb3567748f4	Prev:	000012fa9b916eb9078f8d98a7864e697
Hash:	000015783b764259d382017d91a36d206d0600e2cbb3567748f4	Hash:	000012fa9b916eb9078f8d98a7864e697ae83ed54f5146bd8445	Hash:	0000b9015ce2a08b61216ba5a0778545

Figure 11, A Simple Blockchain Visual Structure [5]

The blockchain data is stored across its peer-to-peer network, and it eliminates many risks that come with data being held centrally. The decentralized blockchain may use ad hoc message passing and distributed networking. Each node in a decentralized network has a copy of the blockchain. No centralized "official" copy exists, and no user is "trusted" more than any other. Blockchains use consensus methods to serialize changes, such as proof-of-work, proof-of-stake, etc.

Blockchain is decentralized, transparent, secure technology. The blockchain data is trusted and very difficult to modify, agents in the network are confident to trust each other. The data on blockchain distributed networks is transparent, and each agent in the network can access it and make their decisions.

It needs lots of effort to learn about the blockchain and apply it in decision-making the distributed network. This is one of the directions for future research.

More Complex Messages Analysis

In this paper, one of the assumptions is that the agents reply to messages with Boolean values. However, in the real world, the replies could be texts, voices, photos, or videos.
It is important to analyze the message to get useful information and to help the agent to make a decision. For the text and voices (can be converted to text), the natural language process (NPL) technology is applied to get useful information. This is a big area that many scientists focus on. One of possible methods is collecting as much as possible texts/words used in the emergency events and apply machine learning technology, such as decision trees, to abstract useful information to help agents make decisions.
For photos and videos analysis, it is more complex and needs mass of computation. Machine learning in image processing is a popular field. The challenge here is how to get key information from photos or videos without humans involved.

Development of Applications for Devices and Test
One of possible future work for this project is to build a real application that can run on Android or IOS devices in a real network. Validate the application and test the application work properly and help to solve the coordination problems among different teams.

Others

Except directions mentioned above, other directions include, what is a good distributed decision, how agents' decisions/ actions coordinate in the emergency event, how an agent makes a coordinating decision if all of his/ her connections are lost. Etc.

Reference:

1. 2018. Conducting Information Superiority Research in an Unclassified Surrogate Defence Environment. DST Group.

2. GeeksforGeeks. 2022. *Dijsktra's algorithm*. [online] Available at: <https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/> [Accessed 17 February 2022].

3. En.wikipedia.org. 2022. *Wikipedia:Consensus - Wikipedia*. [online] Available at: <https://en.wikipedia.org/wiki/Wikipedia:Consensus> [Accessed 17 February 2022].

4. Docs.python.org. 2022. *tkinter — Python interface to Tcl/Tk — Python 3.10.2 documentation*. [online] Available at: <https://docs.python.org/3/library/tkinter.html> [Accessed 17 February 2022].

5. Brownworth, A., 2022. *Blockchain Demo*. [online] Andersbrownworth.com. Available at: <https://andersbrownworth.com/blockchain/blockchain> [Accessed 17 February 2022].