Coordinating robots in disaster areas

CS4501
Final Year Project
BSc in Computer Science
2023-2024

Ivan O Herlihy

Supervised by
Dr. Dan Grigoras

Abstract

Mobile autonomous robots can execute specific tasks in different environments, including disaster areas. Robots coordinate their actions, for example for the completion of a task, or share information about the environment and their status. Communication and coordination can be supported by a cloud.

Declaration of Originality

In signing this declaration, you are conforming, in writing, that the submitted work is entirely your own original work, except where clearly attributed otherwise, and that it has not been submitted partly or wholly for any other educational award. I hereby declare that:

- this is all my own work, unless clearly indicated otherwise, with full and proper accreditation;
- with respect to my own work: none of it has been submitted at any educational institution contributing in any way to an educational award;

- with respect to another's work: all text, diagrams, code, or ideas, whether verbatim, paraphrased or otherwise modified or adapted, have been duly attributed to the source in a scholarly manner, whether from books, papers, lecture notes or any other student's work, whether published or unpublished, electronically or in print.

Signed: 

Date: 4/24/2024

Table of Contents

Introduction

The main goal of the project is to create a messaging protocol for server to robot communication. These robots will be independent entities. They should be able to work together on tasks to complete them but have limited to no communication with each other.

There should be a user interface for an administrator to observe the system and make sure everything is functioning correctly and no errors have taken place. Administrators should be able to add to task while the system is running and enable robots throughout at any point

Robots should be able to perceive their environment and react accordingly. This will help prevent robots getting stuck or causing damage. Robots may have specialised features that are required to complete different tasks and tasks should only be assigned to robots that have these abilities.

The server should be portable. It should be able to be deployed on most if not all cloud service providers and work with robots across the internet.

Analysis

The server should support robots by giving them tasks to do and by helping them communicate with each other. The server will tell robots where and who to communicate with when a task requires multiple robots. Robots will then communicate with each other without server supervision until the task is completed.

Robots will have multiple states that the server and administrator will be able to see. These will include an idle state when the robot has no task usually enacted just after a task is completed. There will be a waiting state whe the robot is waiting for a response from the server or other robots. There is a working state when a robot is actively doing a job. There is a travelling state when a robot is moving from between different locations and finally a breakdown state when a robot is not functional. The server and administrator may not be aware of a breakdown dependent on what has broken on the robot i.e. radio nonfunctional. In this case if a robot has not reached out to the server after a certain amount of time dependent on travel distance and task length the robot is assumed to have broken down and the administrator is notified

The server should store the last known state of all the robots and their current tasks in a database. This will allow an administrator to see how robots are performing and see if there are any errors or breakdowns. This database will be used to create a user interface for the administrator to interact with the server and see what individual robots are doing.

The server should also allow for scheduling in multiple ways. Priority scheduling should be allowed where tasks have a priority and high priority tasks get completed first. Time based scheduling should also be implemented. This scheduling should finish the tasks in the shortest amount of time taking in relative distances between where an idle robot completes a task and where the next task is located.

Design

System Architecture Overview

A mobile cloud service should be developed which can be deployed to control multiple independent robots. These robots should be largely independent and oblivious to each other. The server will have the ability to organise robots to work together on tasks while not directly together. The server should be able to react to robots failing or breakdown..

System Components

Sever

Figure one shows the State machine for the server. It can be seen that there is a general loop from “Waiting for Robots” to “Communicating with a Robot” to “Tasks Available”. This is the main loop the server will run through. The server starts by listening for a robot to inform it that it is offline. Then the server retrieves the next task to be given and tells the robot of the task. There is then a check to see if another robot has come to idle or if there are tasks left. If both are false the server goes back to its default listening state. This ensures that the server will give out tasks to robots as soon as the robots become available.

If a task requires multiple robots then the server will make sure that enough robots are assigned to a task and are ready to work before beginning the task. This makes sure that robots don’t begin group tasks themselves, possibly

breaking or damaging something or itself.

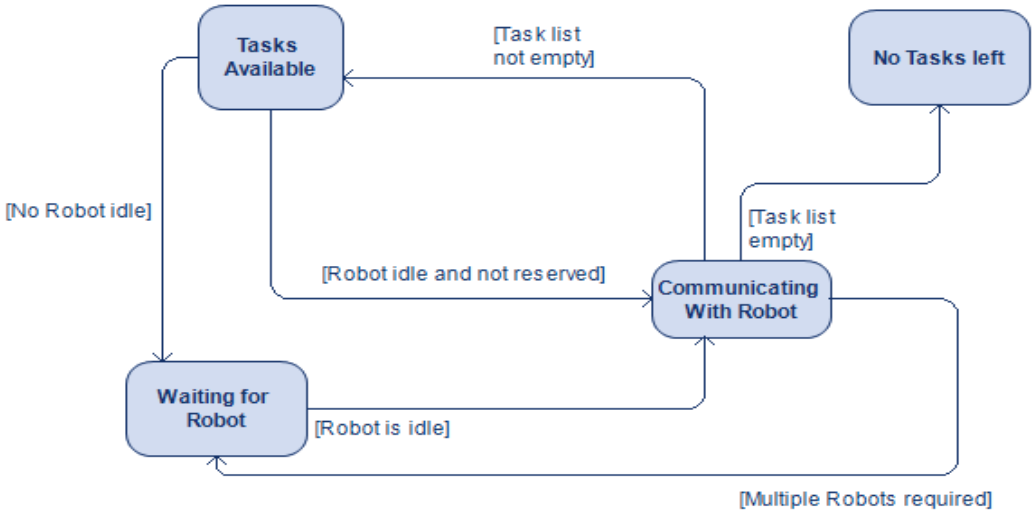

Fig. 1 Sever State machine

Robots

Figure two shows the robot state machine. There are also three different states that the robot loops through. They are “Idle” to “Traversing” to “Working” then repeats. Robots start at idle. They then immediately request a task from the server. If the robot isn’t at the location of the task it will begin travelling to the task. Once at the location of the task it will begin working. Upon finishing a task the robot will become idle and once again request a task from the server.

At any point during this process a robot may have an error. This may include radio malfunction or a physical breakdown. If this is to occur it will stop till the issue is resolved. After the issue is resolved it will go straight to the idle state as the task may have been assigned to a new robot.

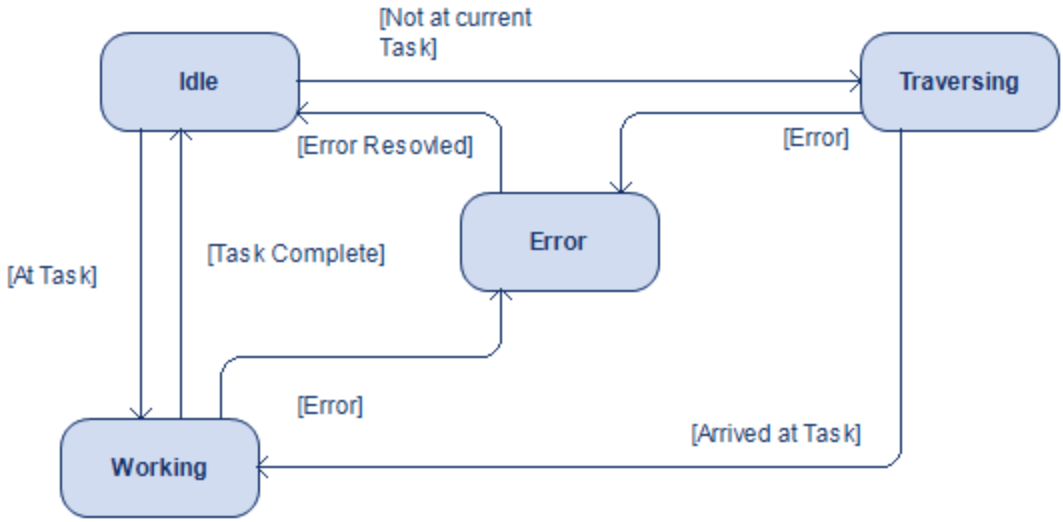

Fig. 2 Robot State Machine

Messaging

There are two main messages the robot will send to the server. Requesting a new task and Informing the server they have arrived at a task. If a robot informs the server that they are idle and the robot happens to already be at the task the server is about to assign to them, the server will instead send the required info to begin the task and the robot will begin the task immediately.

In figure three we see how task starting is performed when multiple robots are required. One robot will get their message to the server first as it arrives at the destination. It might get to the destination first because it was given the task first or it was closer to the destination than the second robot. Once the last robot required to complete the task has arrived at the location the server will send a message to each robot individual to begin the task.

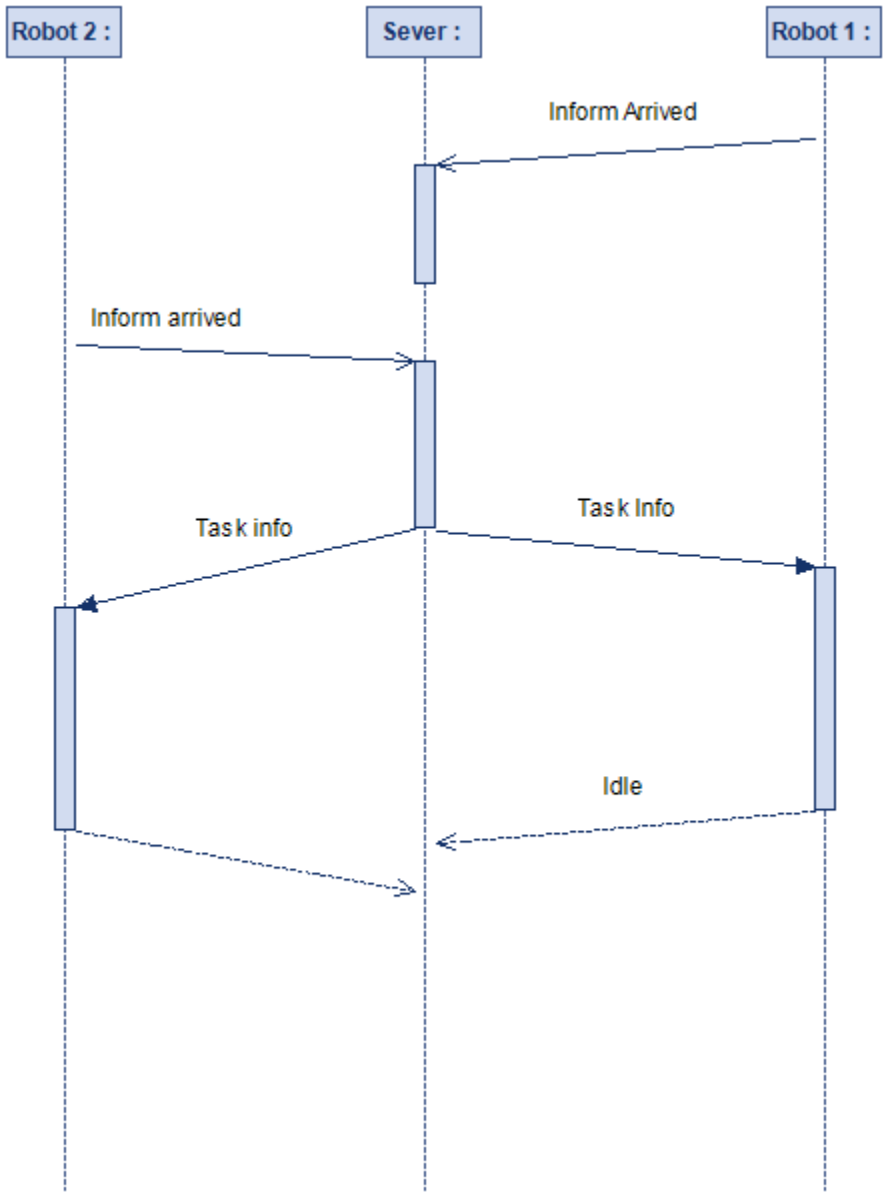

Fig 3. Inform Arrived Message two Robots

Administration

To allow for an administrator to see what is happening between the robots and the server an administration interface was created. The interface will only display what is known by the server. This includes what tasks there are and how they are progressing as well as the last known state of each robot. As robots only update the server on state change the server will always know what a robot is doing when it is in a functional state. However if a robot is to break down and enter the server state it may take a period of time before the server is aware of this. There will also be log files of all messages received and sent from the server.

Implementation

Server Startup

To start the server, it only requires a small amount of information. There must be a starting list of tasks. This can be passed to the server from another script as a list of task objects or they can be placed into a Json file with the correct formatting. The server must also be informed of its default IP and Port number so it knows where to begin to listen from.

Robot Startup

Robot startup is slightly more complicated. The robot will require more information as it is on the robot to begin the connection to the server. It requires its own IP and Port, the servers IP, Port and its starting location and a unique identification number. Robots may also have extra functionality where they can do special tasks i.e. have saws to remove trees installed.

When a new robot connects to the server it immediately enters the idle state. The server stores a new entry in its robots table about the robot and immediately gives it a task.

Messaging

Both the robots and the server use the same messaging functions to communicate. There is a simple send message function and a receive message function. This uses Python Sockets to set up and tear down TCP connections.

Over the sockets a Json object is sent allowing for highly human readable messages. As long as the messages from the robots are correctly formatted they can communicate with any type of robot.
An example of message from the server might look like this:

```
{
    "location":[25,56],
    "task":"Clear Tree",
    "robots Required": 1,
    "Info": null
}
```

This is the minimum required information a robot needs to complete a job. If extra information is required for a job it can be added to the info section of the Json. This can be used to create new robots that still function with the same server.

Information Storage

All information about the robots and tasks are stored in Json files. This is updated every time there is a change in the robots or information. This allows the administration user interface to see what each robot is doing and how the tasks are progressing. The interface is created using a Flask web application.

This method of storing information allows for the server to recover in the event of a crash. There is enough information in the files to restore the state of the server if this is to occur. However it is best practice to not restore the robots and only the tasks as there may have been messages sent to the server that were never received and will have to be reset.

Robot Crash

The server stores a time to live for every robot in the system. This is estimated based on the distance to the next job and the estimated time required to complete this job. If the robot were to take longer than the time to live the robot is assumed to have crashed. When a robot crashes if possible the task will be reassigned to a new robot. For some tasks it may not be possible as the state of the task is no longer known i.e. delivering a unique object to an area. In this case the task is marked failed and the administrator is notified. The administrator is also notified about the robot crash though its information is retained

If a robot comes back online after a perceived crash the server will recognize it by its identification number and re-assign new tasks as if it had never gone offline

Evaluation

System Testing

To test the project a simulated environment was created. A random set of tasks were generated and the server was started. Multiple robots were then launched to test if it could complete every task. Once all the tasks were completed constantly other features were tested. This included manual deactivating robots mid task and then reactivating them, launching new robots mid simulation and more.

Some limitations were found. In the case that a large proportion of the robots fail there is a chance that the tasks can become uncompletable. This is due to robots not listening when they are busy. All robots may be waiting for more robots to come to complete a task but if there is no robot able to complete a task after multiple robots have gone offline they will wait indefinitely.

Scheduling of tasks was never implemented due to time constraints. This means tasks are assigned based on their position in the list of tasks not off location relative to robots. It is possible to have a simple scheduling system where tasks are ordered before the server is able to see them as this order is preserved
All other features that were designed have been implemented and are working to their full extent. There does not seem to be any other bugs that affect the system.

User-Friendliness

In general the system was not designed with user friendliness in mind. The robots are launched with terminal commands that are relatively long and there are large text outputs. However the information server side is relatively readable and can be understood by someone who doesn't have a large technical background.

Conclusions

This section shall be a summary of the project and what has been accomplished. You should also outline what conclusions can be drawn from your work: what does your project contribute to the sum of human knowledge? You should also sketch any future developments or lines of enquiry suggested by your work. In particular, you should include:
The project as a whole was a success. There are limitations in the design that would not make it a viable product to be deployed in a real world scenario. However it is a good proof of concept and jumping off point for further development to create a product that could be useful and help in real problems.

The original idea of disaster zones ended up not being explored to its fullest but a more general approach was taken. The product could be applied to a variety of situations.

In figure four we see the Gantt chart used for this project. It was used as a general guide for the entire project. In retrospect I should have known that the implementation was a bigger task than originally imagined. I ended up writing code much closer to the open day than originally planned. I also failed to write the report as quickly as I had imagined. I should have begun the implementation at least two weeks earlier allowing for a greater time to write the report. However all in all the project was effective in what was set out to be achieved

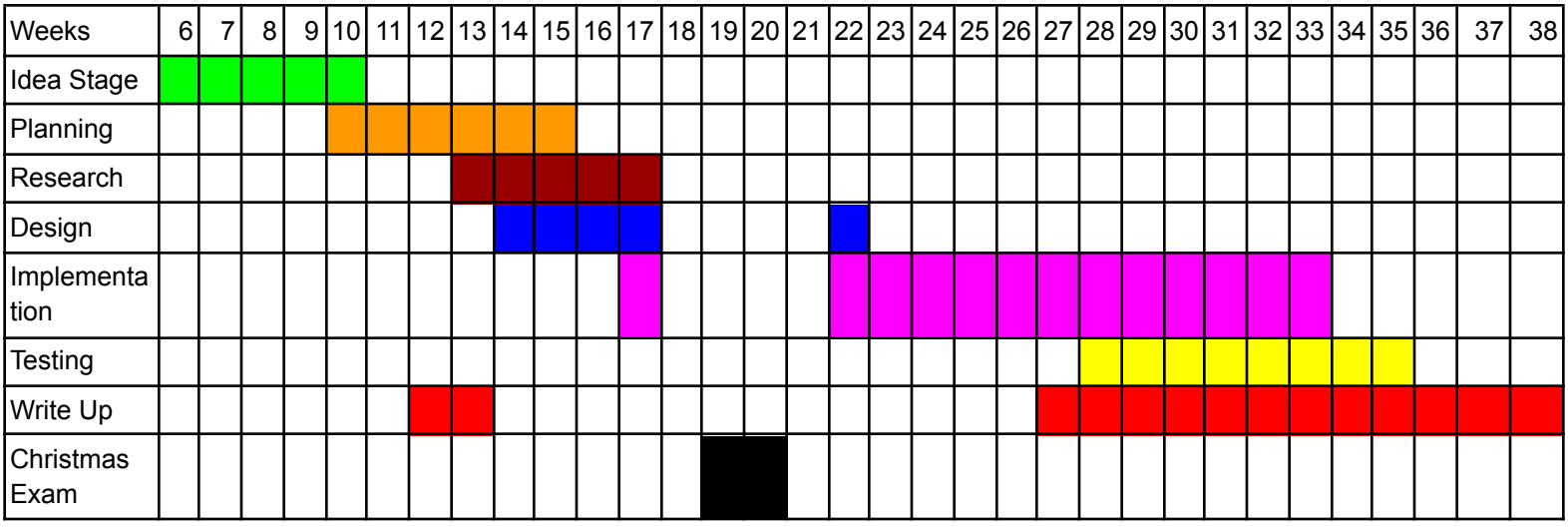

Fig 4 Gantt Chart