CSCI 251: Concepts of Parallel and Distributed Systems

Alvin Lin

September 2017 - December 2017

Project 3

Introduction

This project involved a robot simulation in which the objective of the robots was to surround a target at an unknown location. Given that there was no global coordinator and each robot acted independently, this was accomplished by having the robots communicate their search space to each other to efficiently search the grid for the location of the target.

Outline

This simulation runs the robots in two phases, a search phase where they perform a clustered search for the target block, and a convergence phase where they converge onto the block to surround it.

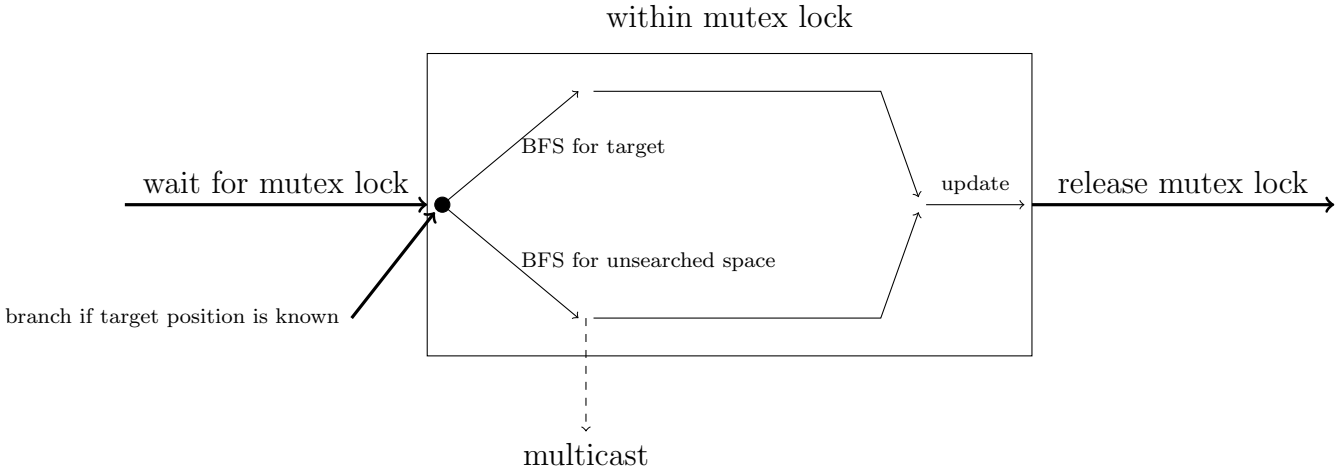
Search Phase

During the search phase, the robots each independently use breadth first search to find the closest unsearched square. After proceeding there, they mark their respective surrounding squares as searched and communicate it via multicast to the other robots. (As an implementation specific detail, this was accomplished with a shared memory space). Each robot acquires a mutex (shared among all of them) to ensure data stability before computing the next location it moves to. The first robot to locate the target block broadcasts it to all the other robots.

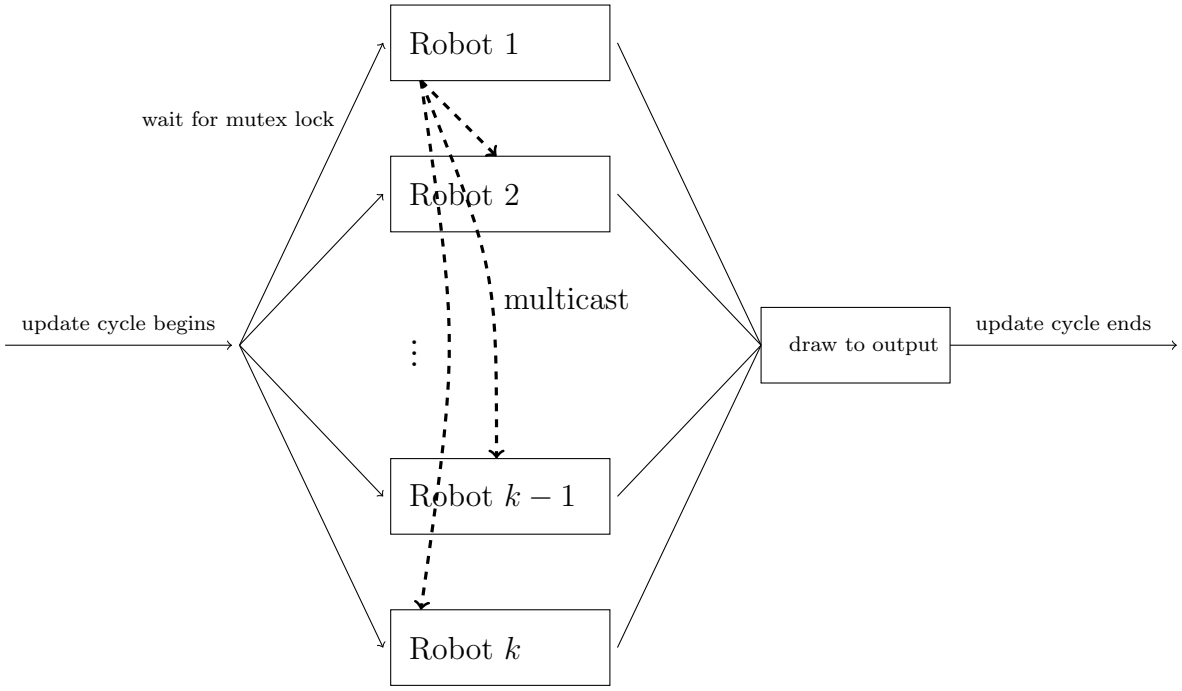
Convergence Phase

During the convergence phase, the target block location has been broadcasted to all the robots. Each robot will independently use breadth first search again to converge towards a location whose Euclidean distance to the block is smallest. At this point, if the robot determines that it is as close as it can possibly be to the block, then it will declare itself as done. When all robots have declared themselves as done, the simulation ends. Because of the nature of the search algorithm, the robots converge onto the target to surround it on a first-come first-serve basis.

Single Robot



A Single Update Cycle



During the update cycle, each robot maintains whether or not it has updated during that cycle. If it has already updated, then it will not update again until the next update cycle begins. This ensures that each robot updates once during each cycle and has a chance to move one square. The only exception to this occurs if the robots are in the convergence phase, since robots that are "done" will be skipped by this update counter.

Since the state of the grid is stable at the end of each update cycle, we draw the state of the grid at the end of each update cycle. The robots use an election process to delegate drawing to the *last* robot that updates during each update cycle.

Results

To run the simulation, compile the program by invoking `make` in the program directory, and then invoke the `simulate` executable. The executable takes 3 parameters and an optional random seed. Note that the delay represents a minimum delay in milliseconds between each update cycle.

```
./simulate <grid size> <robots> <delay> [seed]
```

In the simulation, each robot is denoted by a letter that uniquely identifies it. The grid itself is bounded with the `=` and `|` characters, while the target block is identified with that `#` character. Squares that the robots have searched are marked with the `-` character.

Given a grid size $l \times b$ with k robots, the search phase's running time is upper bounded by $\frac{lb}{k}$ steps. Once a robot finds the target block, it communicates the position via multicast and all robots begin the convergence phase. The convergence phase's running time is bounded by $l + b$ since that is the maximum possible distance a robot may need to travel to reach the target block. In terms of steps taken, the simulation runs in $O(\frac{lb}{k} + l + b)$ time.