Lab 3: Leader Election and Agreement

Due: February 3, 2009

The goal of this lab is to design three different types of agreement algorithms: leader election, agreement by sharing data, and agreement by sharing physical quantities..

1 Setup

Before you start, update your copy of the SwarmBotAPI subversion repository, unzip the lab 3 source, and download the new SwarmOS to the robot. Review the documentation's main file, index.html, located in the lab3\doc folder.

This document will be a work in progress. If something seems broken or impossible, it probably is. Please let me know about potential mistakes as soon as possible.

1.1 Downloading and Remote Commands

There are a lot of robots in the lab these days. We need to be careful about sharing radio bandwidth. Use the command `dwir` to download software using the IR transmitters. This command uses the highest power setting on the IR system, which has a range of around 3 meters. Use the foam barriers to block your robots from other groups. Currently, all `s` commands are transmitted via RF. I'll work on a fix for this, but in the meantime, refrain from using the `s` command while other groups are working in the lab.

2 Leader Election

Design a leader election algorithm. Your algorithm should elect a single robot as leader, and all other robots should know which one is the leader. You can use your lab 2 code for broadcast flood tree construction, but don't necessarily need to. This algorithm needs to be self-stabilizing. If the leader is moved or removed, a new leader should be selected. If the original leader is replaced, or any number of additional robots are added or removed, the algorithm should maintain a single leader in the group.

Use the lights to indicate which robot is the leader. Test your algorithm on a network with at least 12 robots and a diameter of at least 4. Print the current leader ID to the console so you can see if it is working.

3 Agreement with Messages

Design a distributed averaging algorithm similar to the calculator agreement demonstration we did in lecture. Use each robot's ID as the initial value. Each robot computes a pairwise average with a random neighbor. Have the process continue indefinitely. Decide how your algorithm will handle missed messages - it can either make an error that will increase the global average, or one that will decrease the global average. Select one approach for the lab, but coordinate with the other teams to be sure that each approach is tested. Test your algorithm under two conditions:

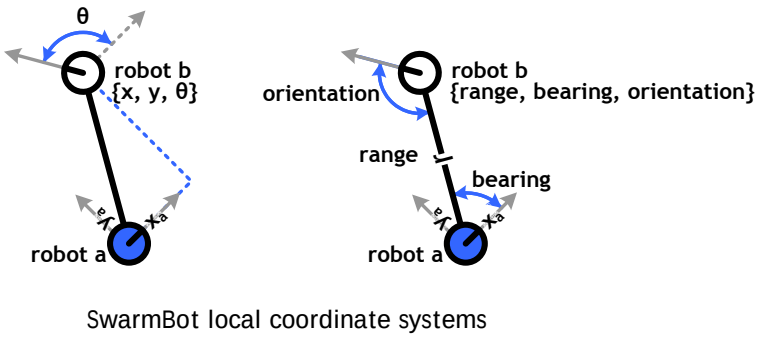

Figure 1: SwarmBot local coordinate systems

1. A static configuration.
2. A dynamic configuration using the `randomReflectMotion` behavior.

Run your algorithm long enough to measure the convergence rate and the divergence rate due to communications errors. This rate might not be linear. Test your algorithm on a network with at least 12 robots and a diameter of at least 4. Design a test for local convergence, and use the lights to indicate when the average has converged. Use the datalogger to record the average from all the robots simultaneously. Sample datalogger code and a data logger instruction handout will be available next week.

4 Agreement with Physical Quantities

4.1 Orienting

This is a warm-up. Using two robots, use the `rotateAngle()` behavior and the `nopClosest()` neighbor operation to write a behavior that makes one robot face the same direction as another robot. If you want, you can use your leader election algorithm from section 2 to select the leader. The local coordinate system for the swarmBots is shown in Figure 1.

4.2 Agreement

Write a behavior that has each robot face the average heading of all of its neighbors, but does not have the same type of errors: portions of the global quantity lost (or gained) because of missed messages. Heading is the direction a robot is facing in a shared global coordinate system. Be careful how you compute the average of angles, there are many clever-sounding ways, but only one correct way to do this. Use the math functions provided in the API, they are optimized for integer computation.

5 Check-off and Write-up

5.1 Check-off (Complete as a group)

1. Demonstrate leader election. Use the lights to indicate which robot is the leader.

2. Demonstrate leader election. Use the datalogger to show that all robots know which one is the leader.
3. Demonstrate agreement using message passing. Show convergence with the lights on the robots.
4. Demonstrate agreement using message passing. Use the datalogger to show that all robots know the global average.
5. Demonstrate agreement using motion. Show the robots agreeing on a heading.

5.2 Write-up (Complete individually)

Make a network with a diameter of at least four using 12 robots. Collect enough data from section 3 to demonstrate the convergence time to the global average, and then the divergence from the global average due to communications errors. Make two plots. One plot should show the 12 global average estimates, one from each robot, on the same plot. Make a second plot with all the data combined that shows the global mean and variance over time.

Write a **1.5 page** report. Explicate the sources of error and potential utility of these algorithms in real applications. Explain the differences in the errors from agreement with messages and agreement with physical quantities. Conjecture if any message-passing algorithm can compute a perfect global average on these robots. Provide a strong counter argument if it is not possible, or an algorithm sketch if it is. The sketch can be short, around 2-3 lines. Note the advantages and disadvantages of this perfect algorithm, if it exists.