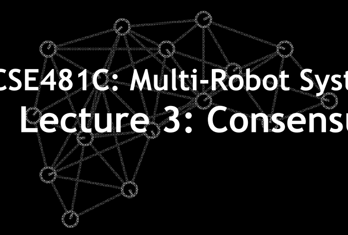


CSE481C: Multi-Robot Systems
Lecture 3: Consensus

intro

Misc

Announcements
- on reading papers
- on packet communications

Paper review
- convergecast aggregation
- database queries

Finish Lec2 Content

multi-robot computation model

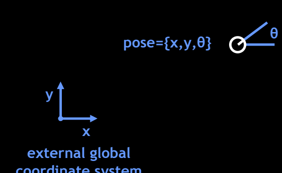
Model: Robot State

We can describe the **state**, *s*, of a single robot as a tuple of its ID, pose, and private and public variables:

Model: Robot State

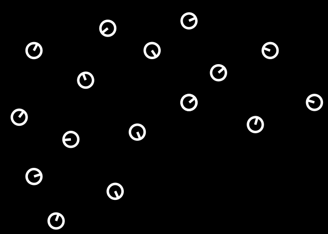
We can describe the **state**, *s*, of a single robot as a tuple of its ID, pose, and private and public variables:

$s = \langle \text{ID, pose, private vars, public vars} \rangle$



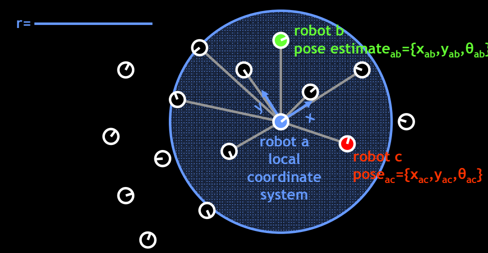
Model: Configuration

We define a **configuration**, *C*, as the states of *n* robots
All the robots use the same software and hardware



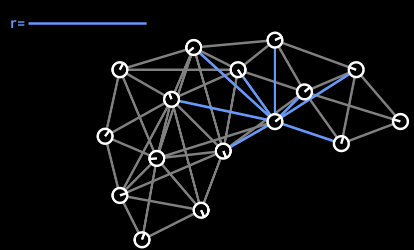
Model: Local Network Geometry

Each robot can communicate with and localize neighboring robots within radius r



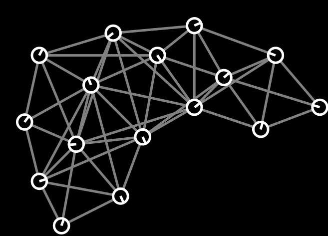
Model: Configuration Graph

A configuration *C* and communication radius *r* produces a **configuration graph** *G*
C is **valid** iff *G* is connected



Model: Periodic Communications

Each robot broadcasts its **public vars** every τ seconds
We assume local communications are reliable
This creates a synchronizer, giving us global **rounds**



definition of terms

Self-Stabilizing Algorithm

Assume:
- Any initial configuration (state, position)
- That robots operate properly
- communications are reliable (perfect)

Provide:
- Proof that the system will stabilize to a desired configuration
- Show time and communications complexity

Complexity Measures

Computation:
- computation per round
- number of rounds
- time for robots to achieve final configuration

Communication:
- total number of messages
- messages per robot per round (bandwidth)

Errors

Two Types:
- process (robot) failures
- communications failures

Two Flavors:
- bounded quantity:
  "At most one robot will fail"
- probabilistic
  "Messages arrive with probability p"

leader election

Leader Election

Requirements:
- one process becomes leader
- other processes become not-leader

Bonus Requirement:
- all processes know which one is the leader

approaches

1. All processes start with same initial state
   - If you have two identical processes, design an algorithm to elect one of them a leader.
   - But only one execution possible on both processes
   - Can't break symmetry → Impossibility proof – not possible to elect leader
2. Randomized Algorithm
   - 1 random bit
   - 50/50 change of electing leader on each flip
   - How long will it take if graph is fully connected?
   - How long will it take if graph is not fully connected?
3. Unique IDs
   - break symmetry with deterministic algorithm
   - can elect leader in bounded time
   - how long will it take?

Problems

How to deal with removal of leader?

How to deal with multiple leaders?

How to elect two leaders?
- Running time and communications complexity?

k leaders?
- Running time and communications complexity?

consensus

What is consensus?

Simple:

- All processes agree on a quantity
- All processes know that they agree

Formal:

- Agreement:
  no two processes decide on different values
- Validity
  1. If all processes start with 0, then 0 is the only possible decision value
  2. If all processes start with 1 and all messages are delivered, then 1 is the only possible decision value
- Termination
  All processes eventually decide

the consensus card game

Whoa… Another Impossibility Result!

Consensus is not possible with faulty communications

- One of the most famous results in distributed algorithms
- (How do you get anything done with these systems, anyway?)

The proof uses the concept of “indistinguishable executions”

agreement algorithms

Agreement Algorithms

It’s like consensus for real-valued quantities

- Processors share real-valued quantities
- All processors converge to the same quantity.

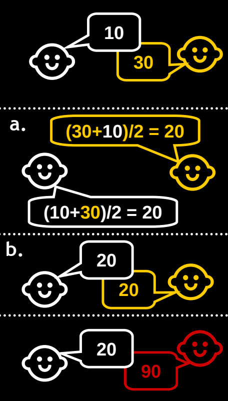
The papers this week are *hard*

- So I will introduce this content with two fun activities…

calculator agreement

Instructions:

1. Enter your starting number into your calculator.



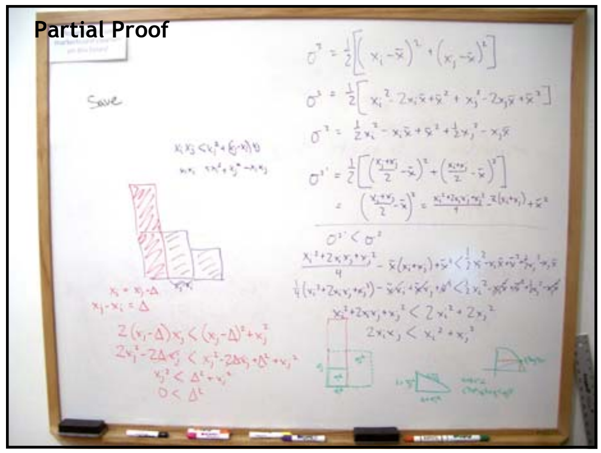
2. Pick another person and average your two numbers. (Add theirs to yours and divide by two) Don't round off, keep all the digits. Both people should end up with the same number.

3. Repeat 12 times. Try to visit different people.

The answer is

65

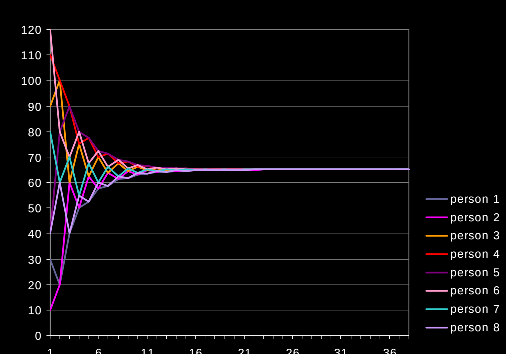
Partial Proof



Reference


Distributed Asynchronous Deterministic and Stochastic Gradient Optimization Algorithms

Simulation



Who Would Compute an Average Using this Crazy Technique?

Honeybees! Workers share food all the time, computing a global average. This lets an individual worker know when the *hive* is hungry by measuring when *she* is hungry.

Will This Work on the Robots?

?

poker chip agreement

lab 3: leader election and agreement