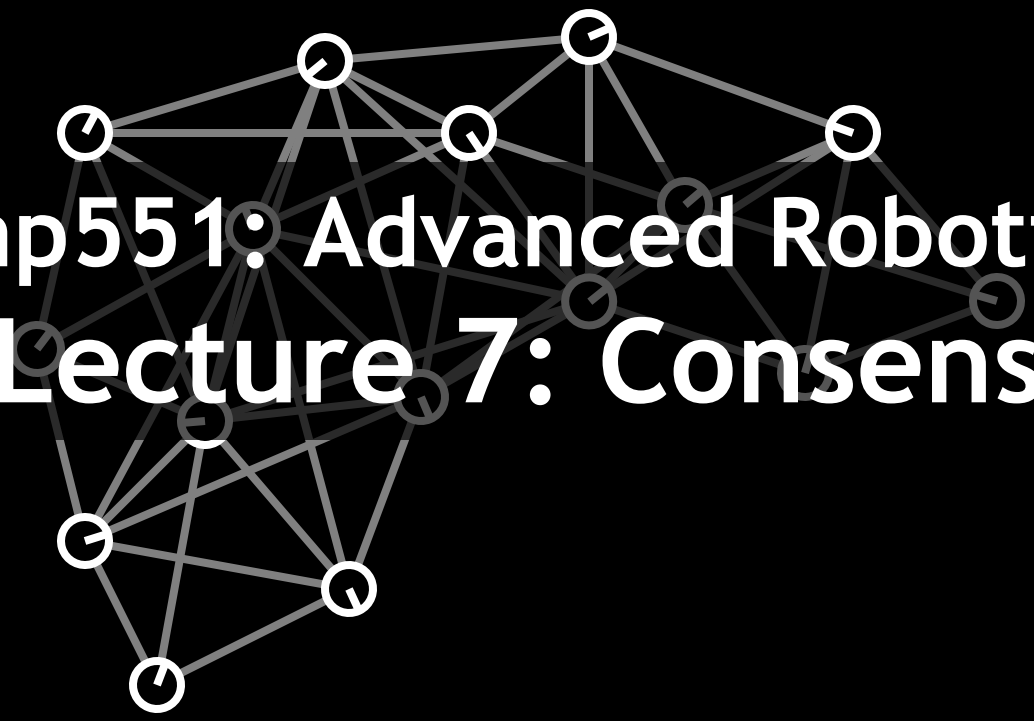


Comp551: Advanced Robotics Lab

Lecture 7: Consensus

intro

multi-robot computation model

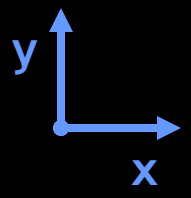
Model: Robot State

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:

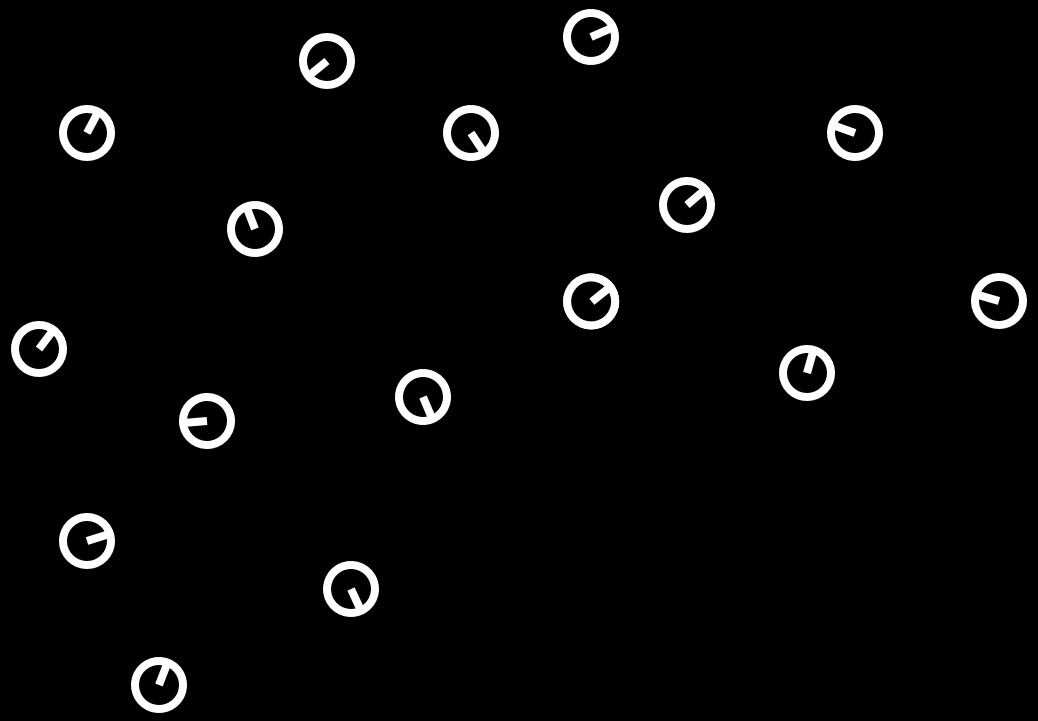
$$s = \langle \text{ID, pose, private vars, public vars} \rangle$$



external global coordinate system

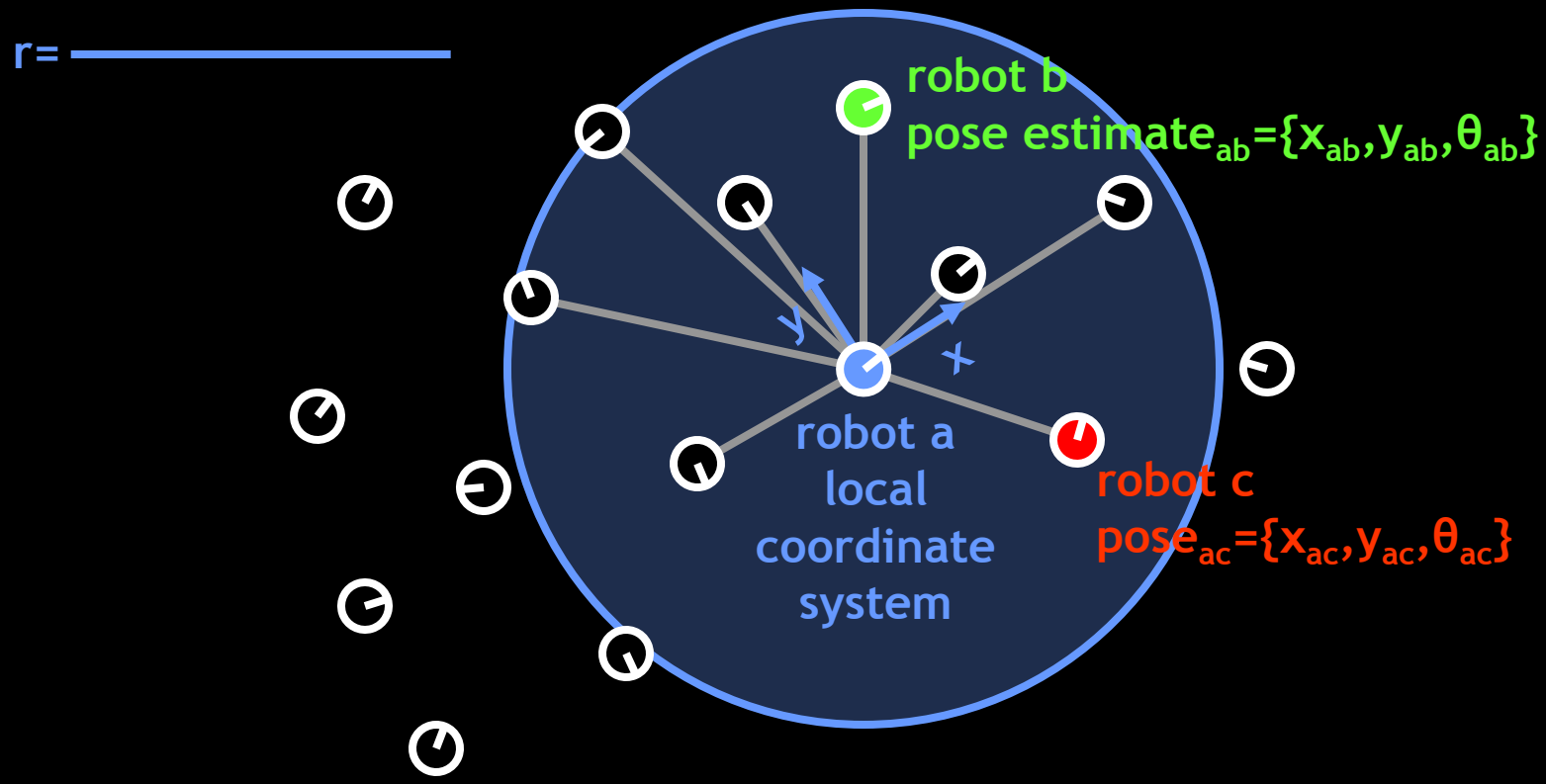
Model: Configuration

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



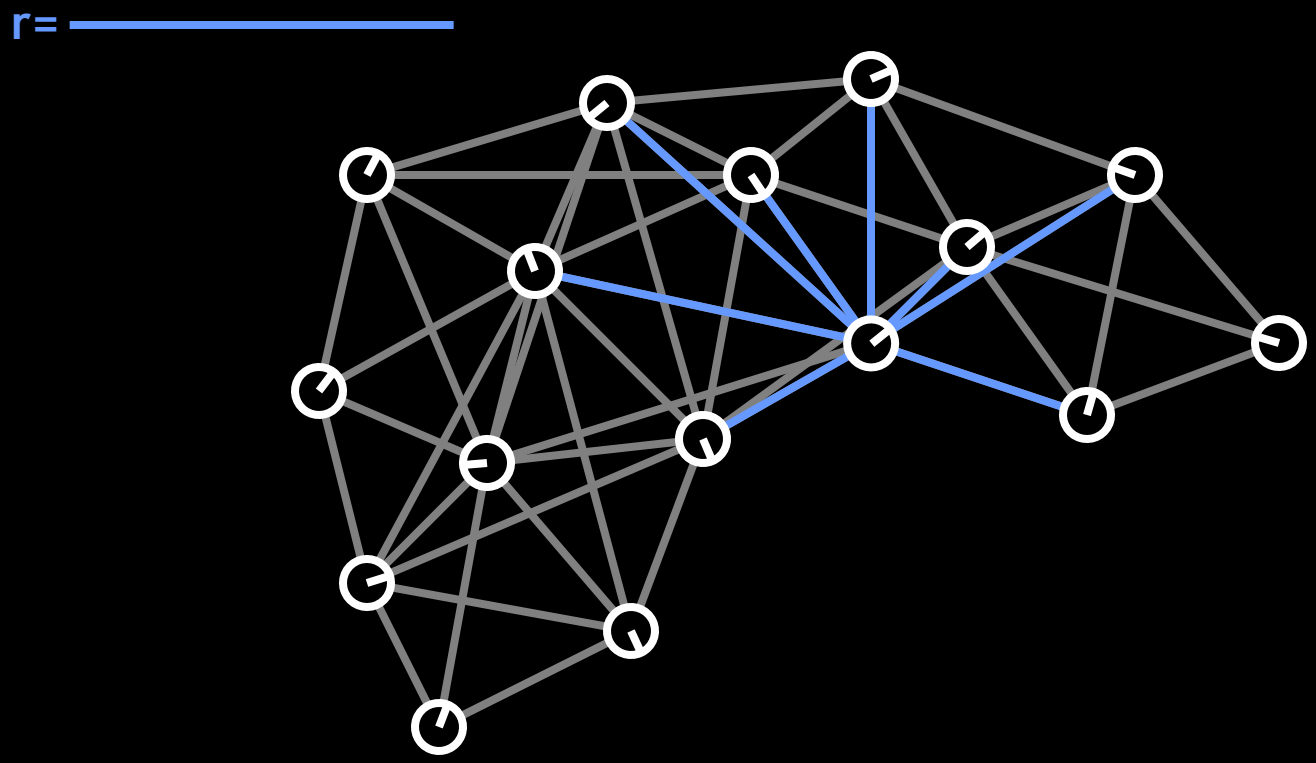
Model: Local Network Geometry

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

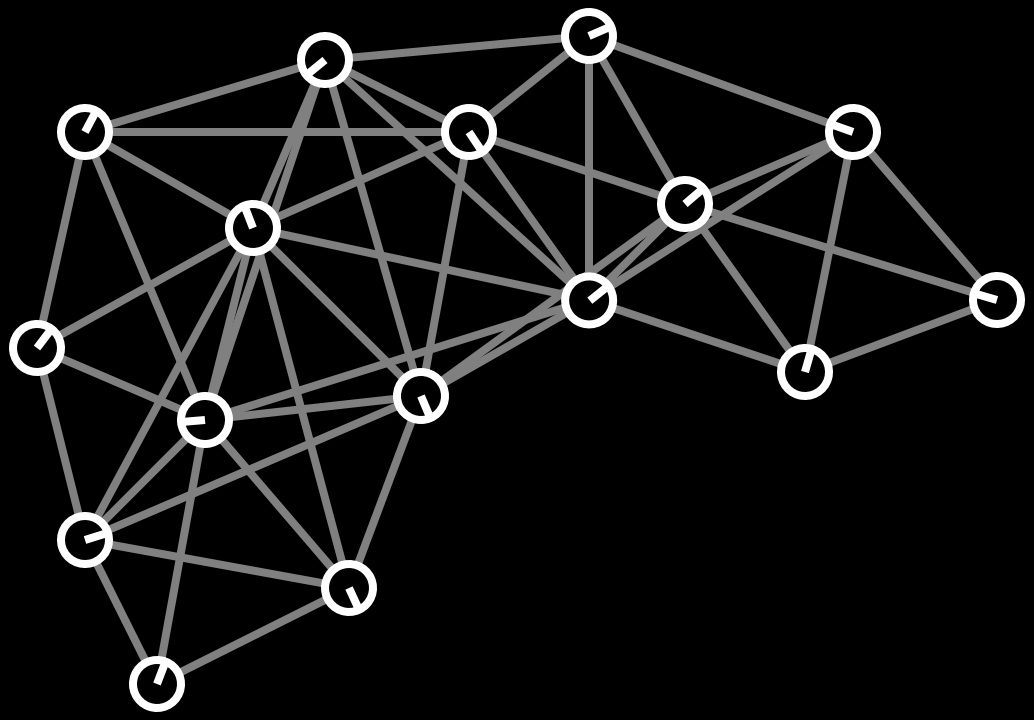


Model: Periodic Communications

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

Three Types:

- process (robot) failures
- communications failures
- network changes

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
- all processes know that the algorithm is done

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?

How to deal with communications loss?

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

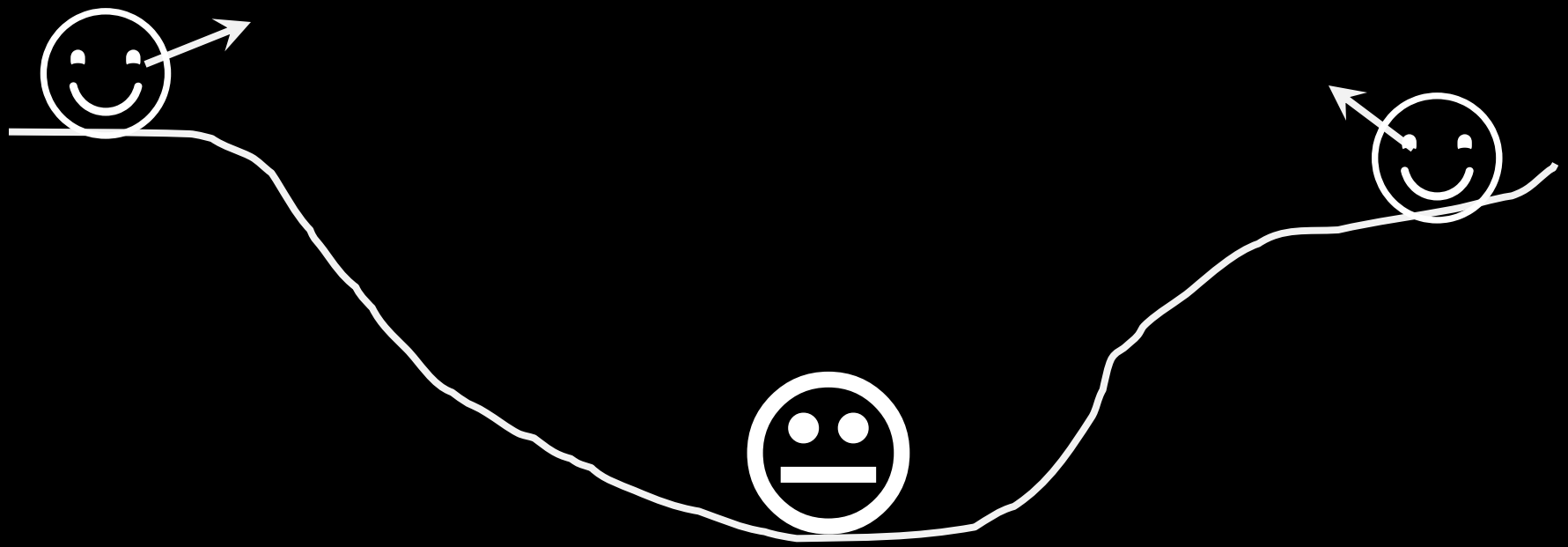
  All processes eventually decide, in bounded time

The Byzantine Generals Problem

Two Generals need to attack at the same time, or be defeated

They can send messengers back and forth, but the messengers might not make it.

Can they agree on a time to attack?



Consensus #2: The Byzantine Generals Problem

Two Generals need to attack at the same time, or be defeated

They can send messengers back and forth, but the messengers might not make it.

Can they agree on a time to attack?

Consensus: The Skit

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”

Are you serious, it really doesn't work?

Nope. Consensus is not possible with faulty communications

- One of the most famous results in distributed algorithms
- This is called an "Impossibility Proof"

But what about my bank records?

- Databases deal with this by using "transactions" and the concept of "rollback"

Ok, Consensus stinks. Agreement is better, right?

agreement algorithms

Reference

Consensus and Cooperation in Networked Multi-Agent Systems

Algorithms that provide rapid agreement and teamwork between all participants allow effective task performance by self-organizing networked systems.

By Reza Olfati-Saber, *Member IEEE*, J. Alex Fax, and Richard M. Murray, *Fellow IEEE*

ABSTRACT | This paper provides a theoretical framework for analysis of consensus algorithms for multi-agent networked systems with an emphasis on the role of directed information flow, robustness to changes in network topology due to link/node failures, time-delays, and performance guarantees. An overview of basic concepts of information consensus in networks and methods of convergence and performance analysis for the algorithms are provided. Our analysis framework is based on tools from matrix theory, algebraic graph theory, and control theory. We discuss the connections between consensus problems in networked dynamic systems and diverse applications including synchronization of coupled oscillators, flocking, formation control, fast consensus in small-world networks, Markov processes and gossip-based algorithms, load balancing in networks, rendezvous in space, distributed sensor fusion in sensor networks, and belief propagation. We establish direct connections between spectral and structural properties of complex networks and the speed of information diffusion of consensus algorithms. A brief introduction is provided on networked systems with nonlocal information flow that are considerably faster than distributed systems with lattice-type nearest neighbor interactions. Simulation results are presented that demonstrate the role of small-world effects on the speed of consensus algorithms and cooperative control of multivehicle formations.

KEYWORDS | Consensus algorithms; cooperative control; flocking; graph Laplacians; information fusion; multi-agent systems; networked control systems; synchronization of coupled oscillators

Manuscript received August 8, 2005; revised September 7, 2006. This work was supported in part by the Army Research Office (ARO) under Grant W911NF-04-1-0316.
R. Olfati-Saber is with Dartmouth College, Thayer School of Engineering, Hanover, NH 03755 USA (e-mail: olfati@dartmouth.edu).
J. A. Fax is with Northrop Grumman Corp., Woodland Hills, CA 91367 USA (e-mail: alex.fax@ngc.com).
R. M. Murray is with the California Institute of Technology, Control and Dynamical Systems, Pasadena, CA 91125 USA (e-mail: murray@caltech.edu).
Digital Object Identifier: 10.1109/JPROC.2006.887293

I. INTRODUCTION

Consensus problems have a long history in computer science and form the foundation of the field of *distributed computing* [1]. Formal study of consensus problems in groups of experts originated in *management science and statistics* in 1960s (see DeGroot [2] and references therein). The ideas of *statistical consensus theory* by DeGroot reappeared two decades later in aggregation of information with uncertainty obtained from multiple sensors[1] [3] and medical experts [4].

Distributed computation over networks has a tradition in systems and control theory starting with the pioneering work of Borkar and Varaiya [5] and Tsitsiklis [6] and Tsitsiklis, Bertsekas, and Athans [7] on *asynchronous asymptotic agreement problem* for distributed decision-making systems and parallel computing [8].

In networks of *agents* (or dynamic systems), "consensus" means to reach an agreement regarding a certain quantity of interest that depends on the state of all agents. A "consensus algorithm" (or protocol) is an interaction rule that specifies the information exchange between an agent and all of its neighbors on the network.[2]

The theoretical framework for posing and solving *consensus problems* for networked dynamic systems was introduced by Olfati-Saber and Murray in [9] and [10] building on the earlier work of Fax and Murray [11], [12]. The study of the *alignment problem* involving reaching an agreement—without computing any objective functions—appeared in the work of Jadbabaie *et al.* [13]. Further theoretical extensions of this work were presented in [14] and [15] with a look toward treatment of directed information flow in networks as shown in Fig. 1(a).

[1]This is known as *sensor fusion* and is an important application of modern consensus algorithms that will be discussed later.
[2]The term "nearest neighbors" is more commonly used in physics than "neighbors" when applied to particle/spin interactions over a lattice (e.g., Ising model).

Agreement Algorithms

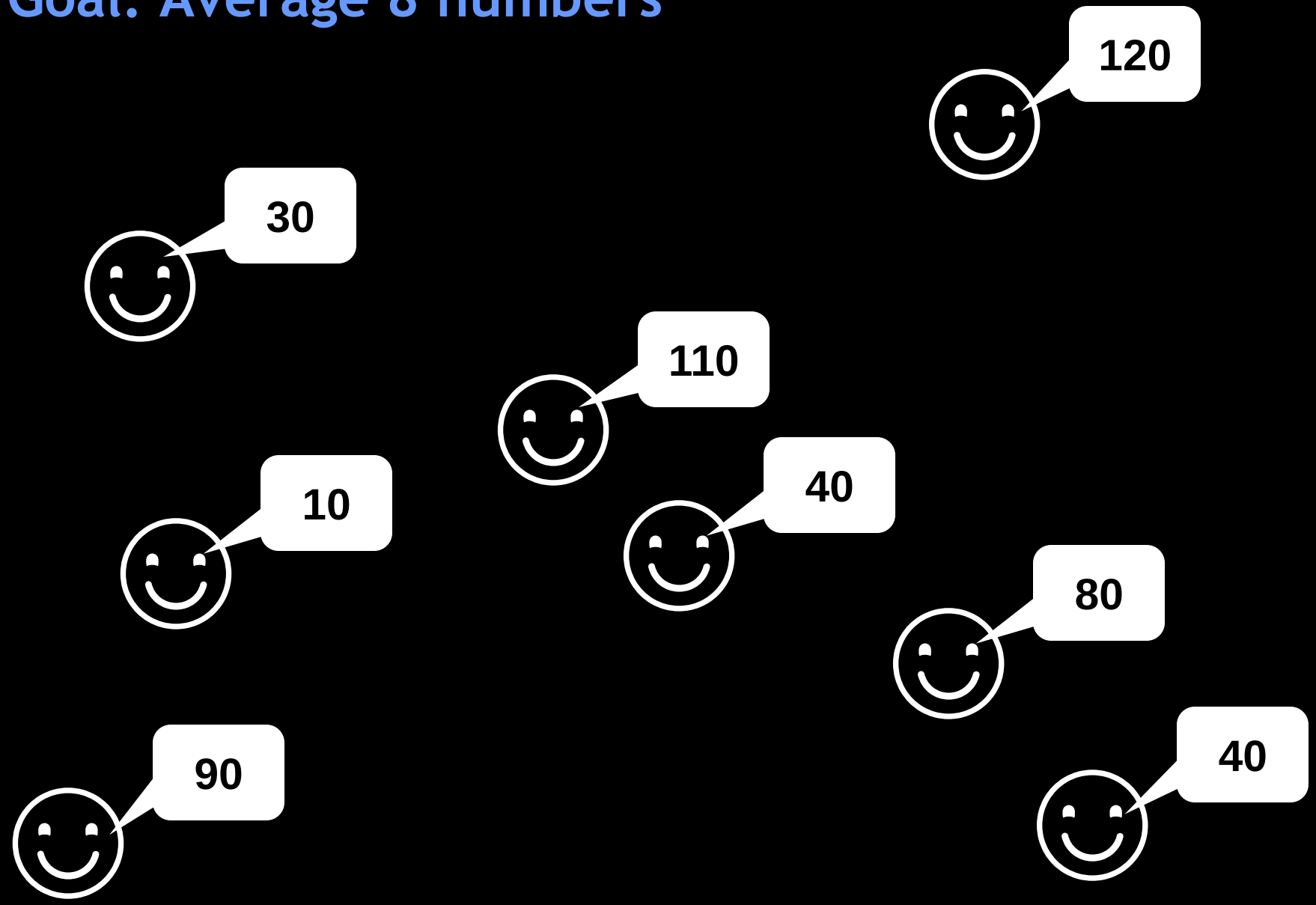
It’s like consensus for real-valued quantities

- Processors share real-valued quantities
- All processors converge to the same quantity.
- The final quantity might not be one of the initial values

The papers this week are *hard*

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

Goal: Average 8 numbers



Goal: Average 8 numbers

120

30

110

Agreement: The Skit

80

90

40

Average Agreement

The simplest agreement algorithm

Start with *n* robots, whose state is stored in *n* variables:

$$x_1; x_2; \ldots; x_n$$

Find a partner. Run the update rule:

$$x_1^0 = \frac{x_1 + x_2}{2}$$

$$x_2^0 = \frac{x_2 + x_1}{2}$$

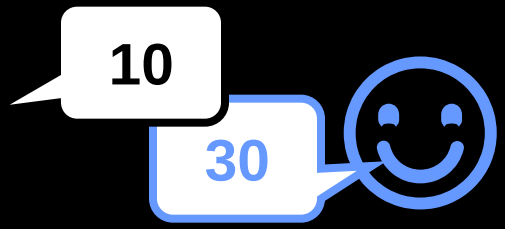
Then switch partners.

Repeat.

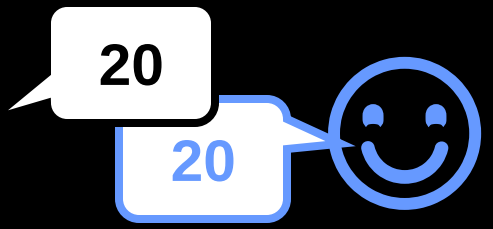
calculator agreement

Instructions:

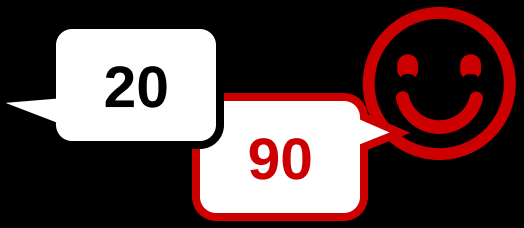
1. Enter your starting number into your calculator.



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

Save

$$\sigma^2 = \frac{1}{2}\left[(x_i - \bar{x})^2 + (x_j - \bar{x})^2 \right]$$

$$\sigma^2 = \frac{1}{2}\left[x_i^2 - 2x_i\bar{x} + \bar{x}^2 + x_j^2 - 2x_j\bar{x} + \bar{x}^2 \right]$$

$$\sigma^2 = \frac{1}{2}x_i^2 - x_i\bar{x} + \bar{x}^2 + \frac{1}{2}x_j^2 - x_j\bar{x}$$

$$\sigma^{2\prime} = \frac{1}{2}\left[\left(\frac{x_i + x_j}{2} - \bar{x} \right)^2 + \left(\frac{x_i + x_j}{2} - \bar{x} \right)^2 \right]$$

$$= \left(\frac{x_j + x_j}{2} - \bar{x} \right)^2 = \frac{x_i^2 + 2x_ix_j + x_j^2}{4} - \bar{x}(x_i + x_j) + \bar{x}^2$$

$$\sigma^{2\prime} < \sigma^2$$

$$\frac{x_i^2 + 2x_ix_j + x_j^2}{4} - \bar{x}(x_i + x_j) + \bar{x}^2 < \frac{1}{2}x_i^2 - x_i\bar{x} + \bar{x}^2 + \frac{1}{2}x_j^2 - x_j\bar{x}$$

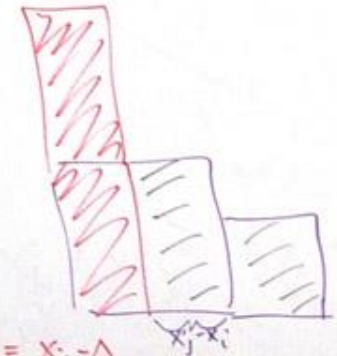
$$\frac{1}{4}(x_i^2 + 2x_ix_j + x_j^2) - \bar{x}x_i + \bar{x}x_j + \bar{x}^2 < \frac{1}{2}x_i^2 - \bar{x}x_i + \bar{x}^2 + \frac{1}{2}x_j^2 - \bar{x}x_j$$

$$x_i^2 + 2x_ix_j + x_j^2 < 2x_i^2 + 2x_j^2$$

$$2x_ix_j < x_i^2 + x_j^2$$

$$x_ix_j < x_i^2 + (x_j - x_i)x_j$$

$$x_ix_j < x_i^2 + x_j^2 - x_ix_j$$



$x_i = x_j - \Delta$

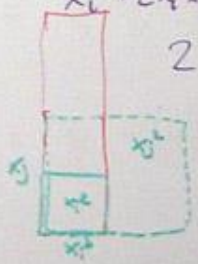
$x_j - x_i = \Delta$

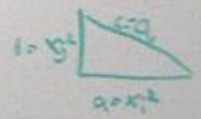
$$2(x_j - \Delta)x_j < (x_j - \Delta)^2 + x_j^2$$

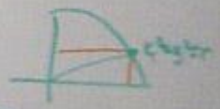
$$2x_j^2 - 2\Delta x_j < x_j^2 - 2\Delta x_j + \Delta^2 + x_j^2$$

$$x_j^2 < \Delta^2 + x_j^2$$

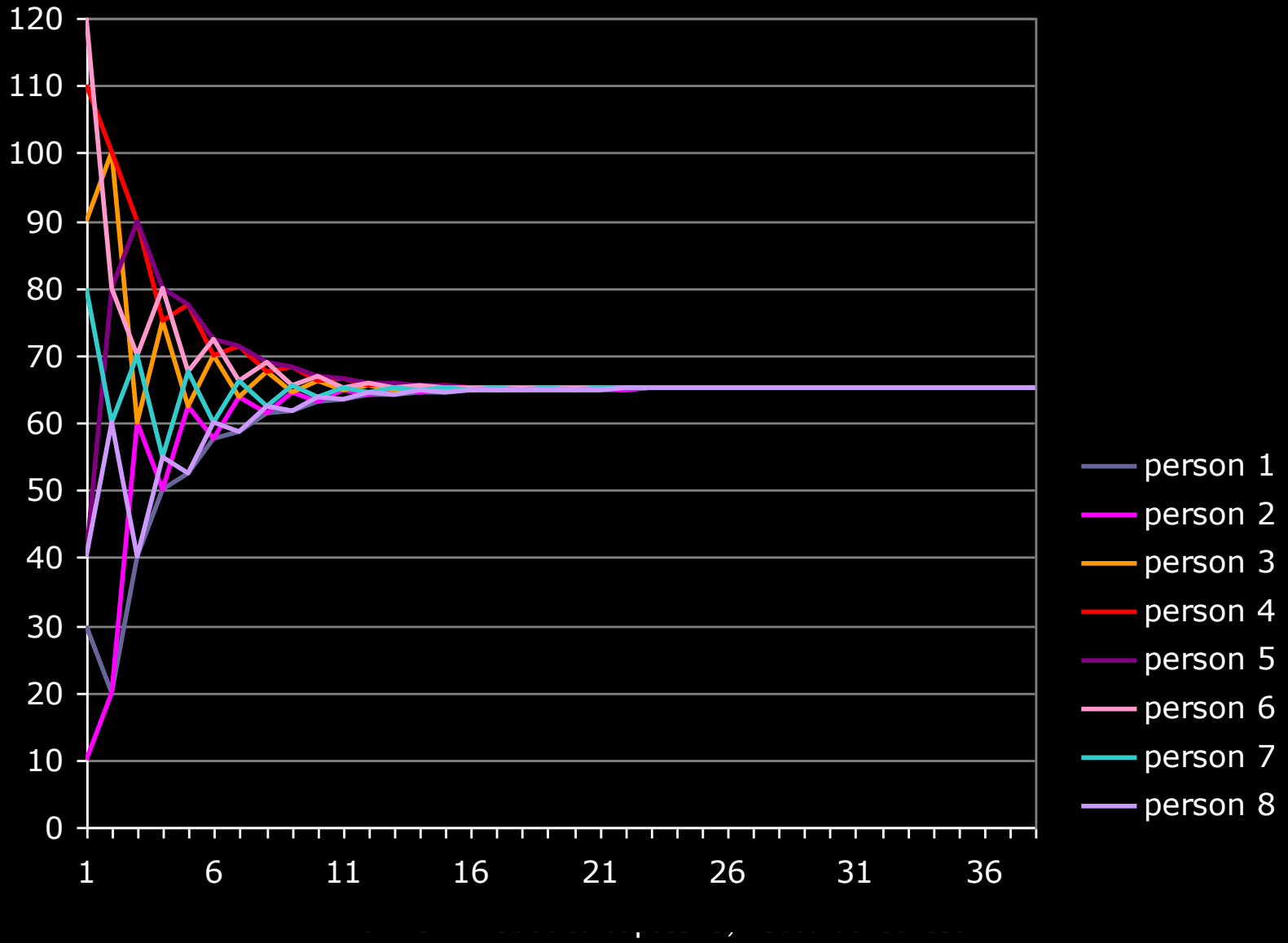
$$0 < \Delta^2$$







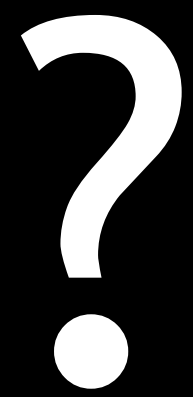
Simulation


120
110
100
90
80
70
60
50
40
30
20
10
0
1
6
11
16
21
26
31
36
person 1
person 2
person 3
person 4
person 5
person 6
person 7
person 8

Who Would Compute an Average Using this Crazy Technique?

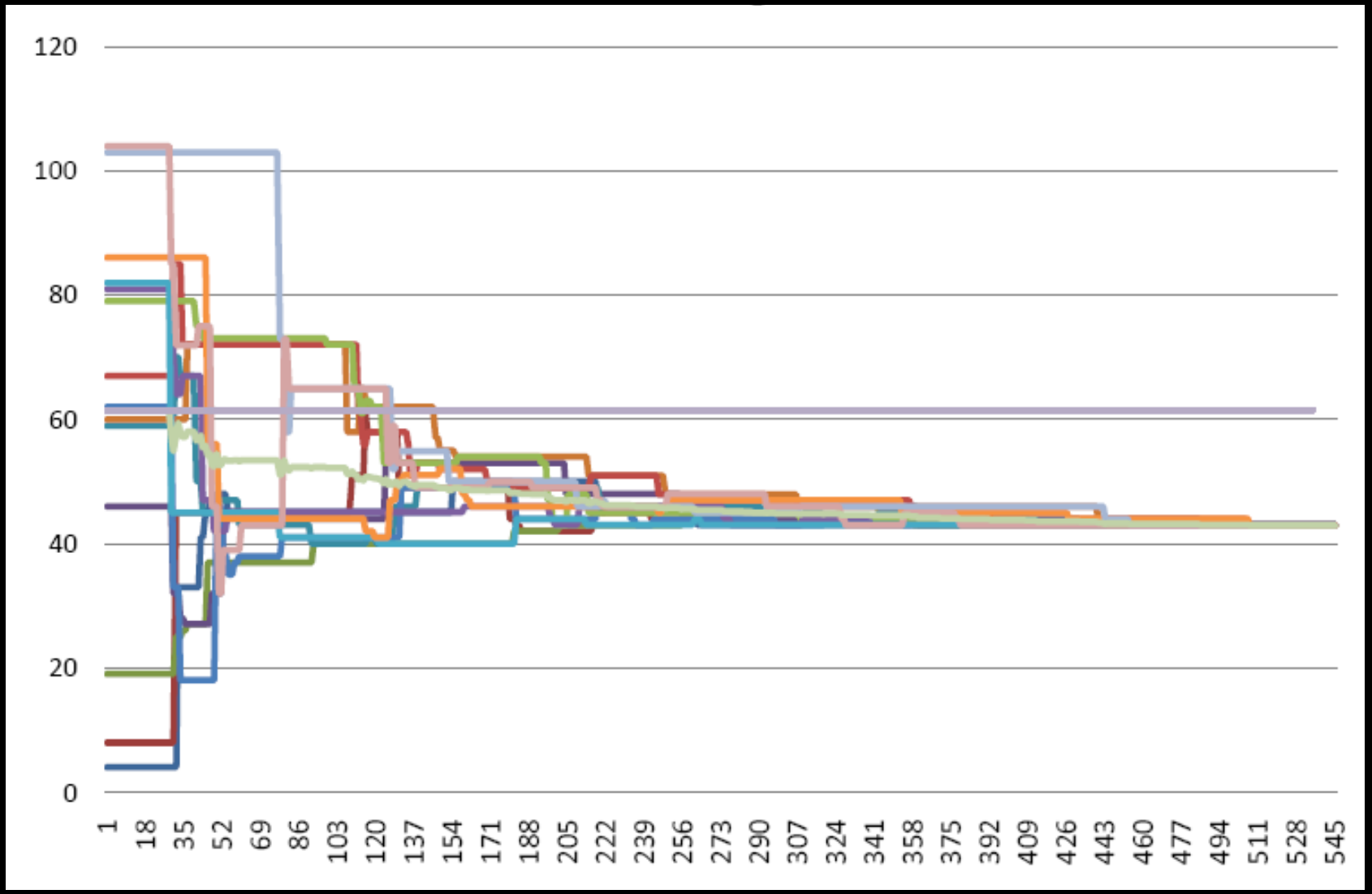
Honeybees! Workers share food all the time, computing a global average. This lets an individual worker know when the ***hive*** is hungry by measuring when ***she*** is hungry.

Will This Work on the Robots?



Agreement Experiment

Perfect agreement on systems that lose messages is impossible



PS02: leader election and agreement

Flocking and Consensus

Can we use consensus for more physical algorithms?

Like flocking

[white board]

Google “boids”…

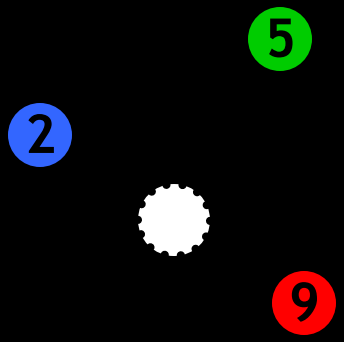
Agreement is everywhere

Can we use consensus for even more physical algorithms?

Like sorting?

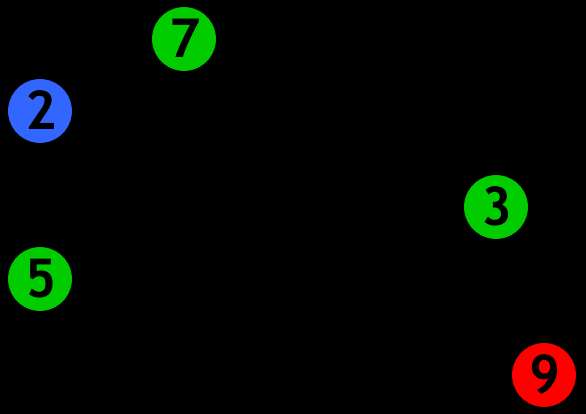
Physical Bubble Sort

Goal: Sort the robots by their robot ID



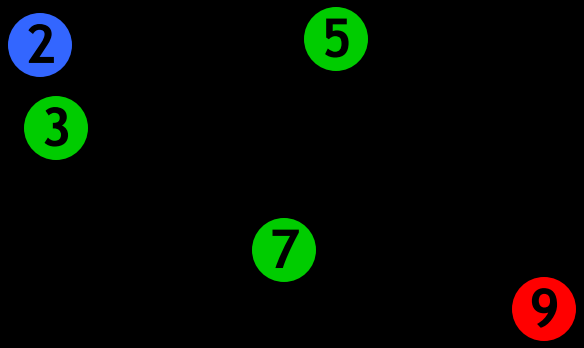
Physical Bubble Sort

Goal: Sort the robots by their robot ID



Physical Bubble Sort

Goal: Sort the robots by their robot ID



Physical Bubble Sort

Goal: Sort the robots by their robot ID