IDS Projects

Vania Marangozova-Martin

2020-2021

Schedule

S12	22/3	7	Discuss LABS, Choice of project, P2P	1.5	////	
S13	29/03	8	P2P II, Complex dist systems I	3	////	
S14	5/4	9			Project - design 1st version	1.5
S15	12/4	10	Complex dist systems II	1.5	Project	3
S16	19/4	Spring VACATION	** TURN IN PROJECT **		** TURN IN PROJECT **	
S17	26/4	11	SOUTENANCES	1.5	SOUTENANCES	6
S18	3/5	EXAMENS		15.		18.

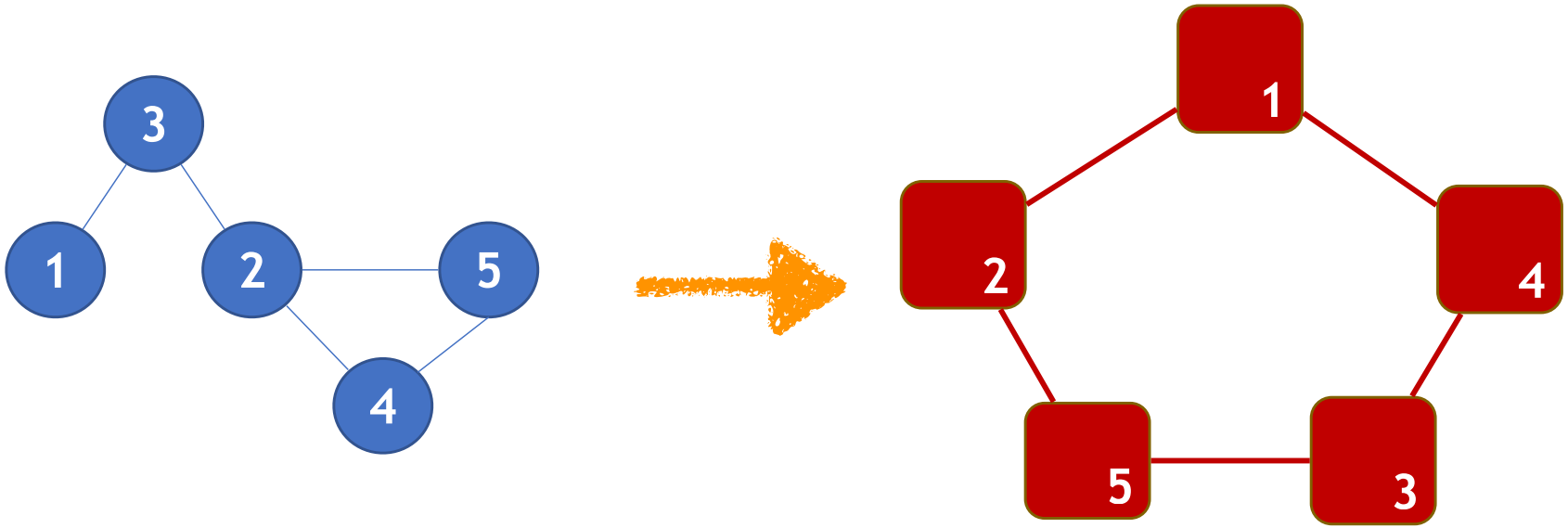
Administrativia

- Organisation
  - 2 members per group
- Project advancement:
  - We will organize discussions per group
- Turn in your project
  - turn in code and a report
  - prepare live remote demo session (15 minutes)
  - report = description of architecture, design, choice of techno
- There are three "algo-oriented projects"
  1. you need to work on the algorithmic logic
  2. implement it with the techno (RMI or RabbitMQ) you want
  3. be capable of explaining the choices (algo/techno) you made

Project 1 [ALGO]: Network Overlay

1. The goal is to construct a virtual network over a physical one

2. We start with a graph that gives the physical topology and build a virtual topology

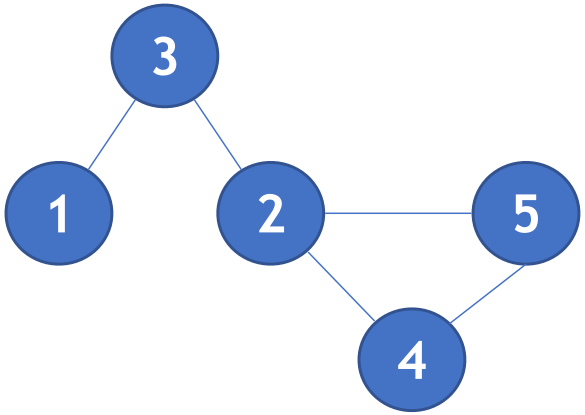


Overlay : the input graph

1. You need an input to describe a graph for example a matrix
2. As the graph gives the physical topology of your system, you need to instantiate it with RabbitMQ or Java RMI to have a running distributed system

	1	2	3	4	5
1			1		
2			1	1	1
3	1	1			
4		1			
5		1			

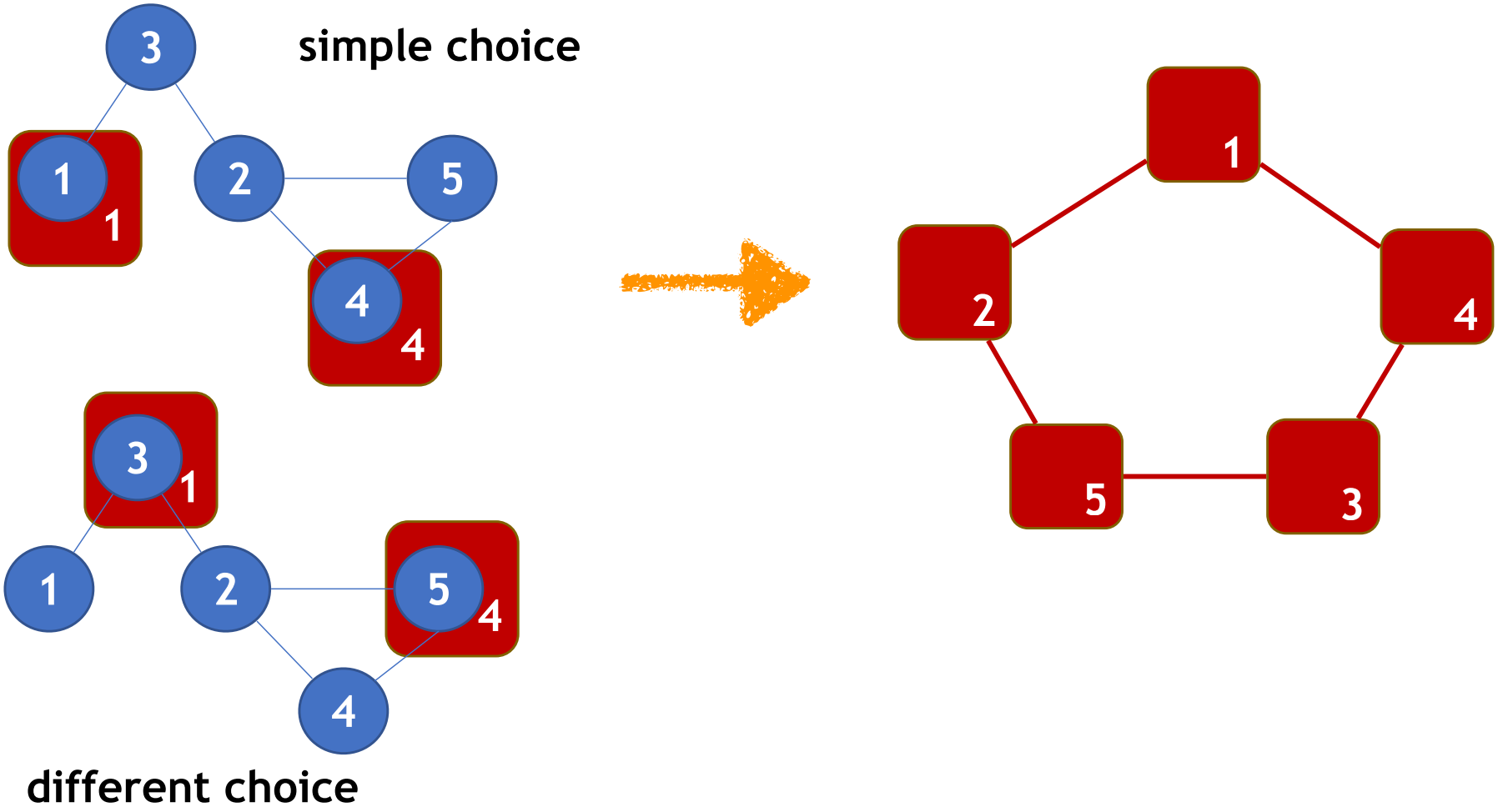
input matrix



running system, each node is a separate running entity (e.g. a remote object in RMI)

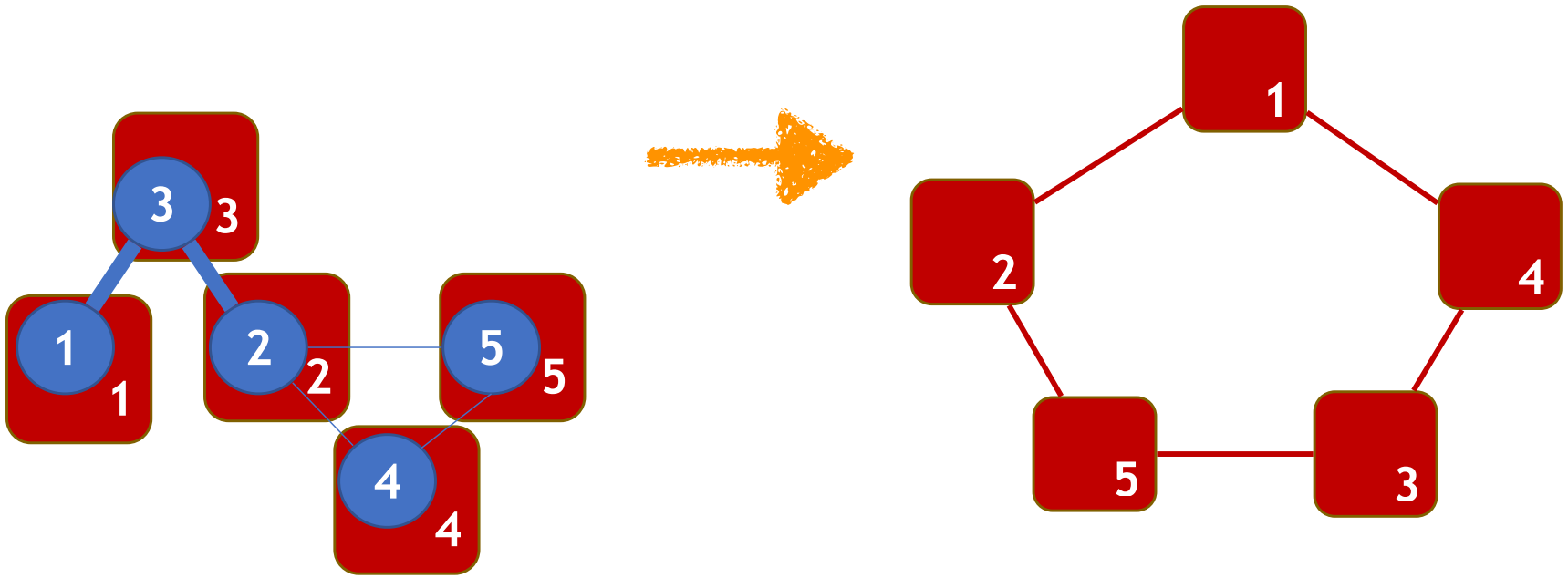
Overlay : virtual ring

- Decide of the nodes identifiers



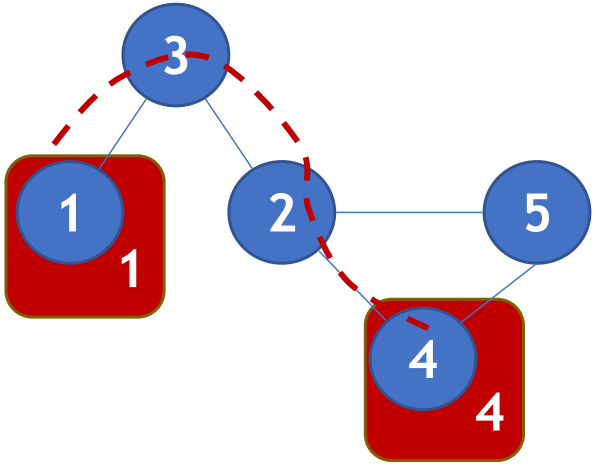
Overlay : virtual ring

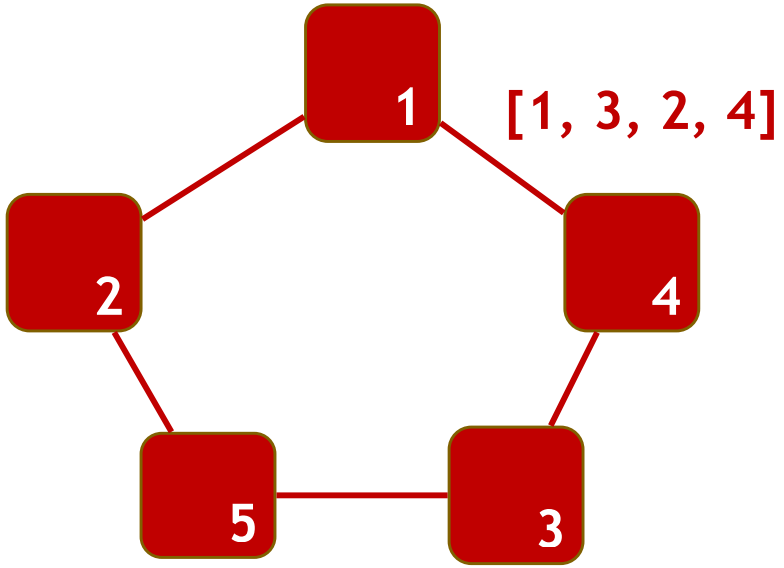
- Decide who is neighbor of who in the virtual ring
  - May be 1-2-3-4-5-1 or, as given in the picture, 1-4-3-5-2-1



Overlay : virtual ring

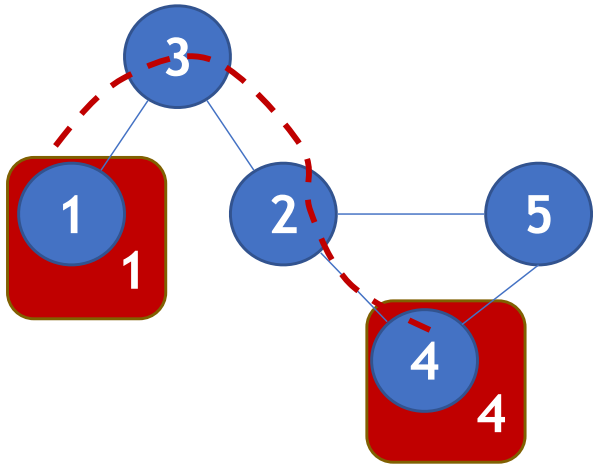
- Compute the routing between a ring node and its two neighbors

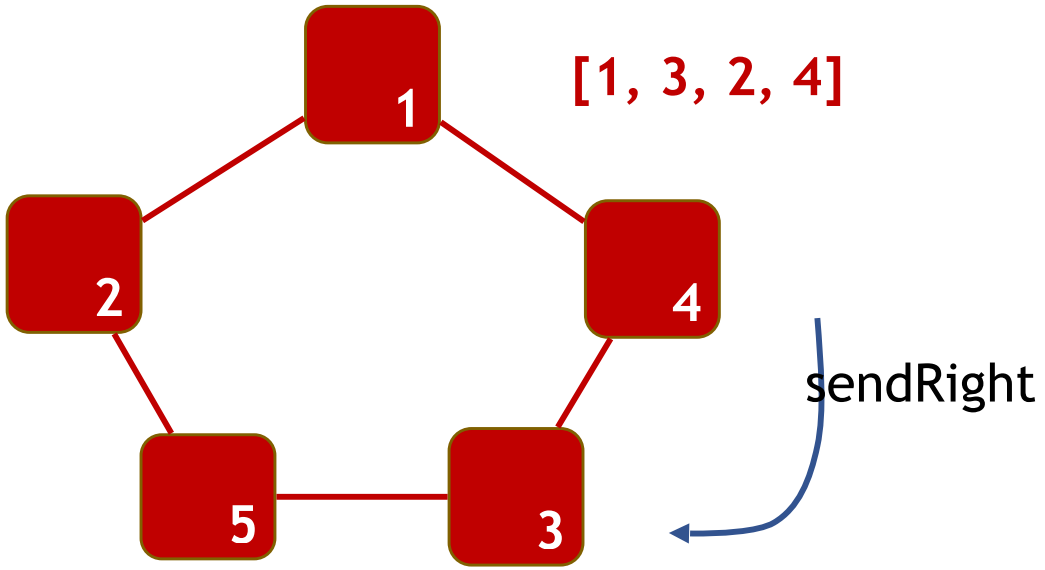




Overlay : virtual ring

- Implement the two basic primitives
  sendLeft(Message m) and sendRight(Message m)



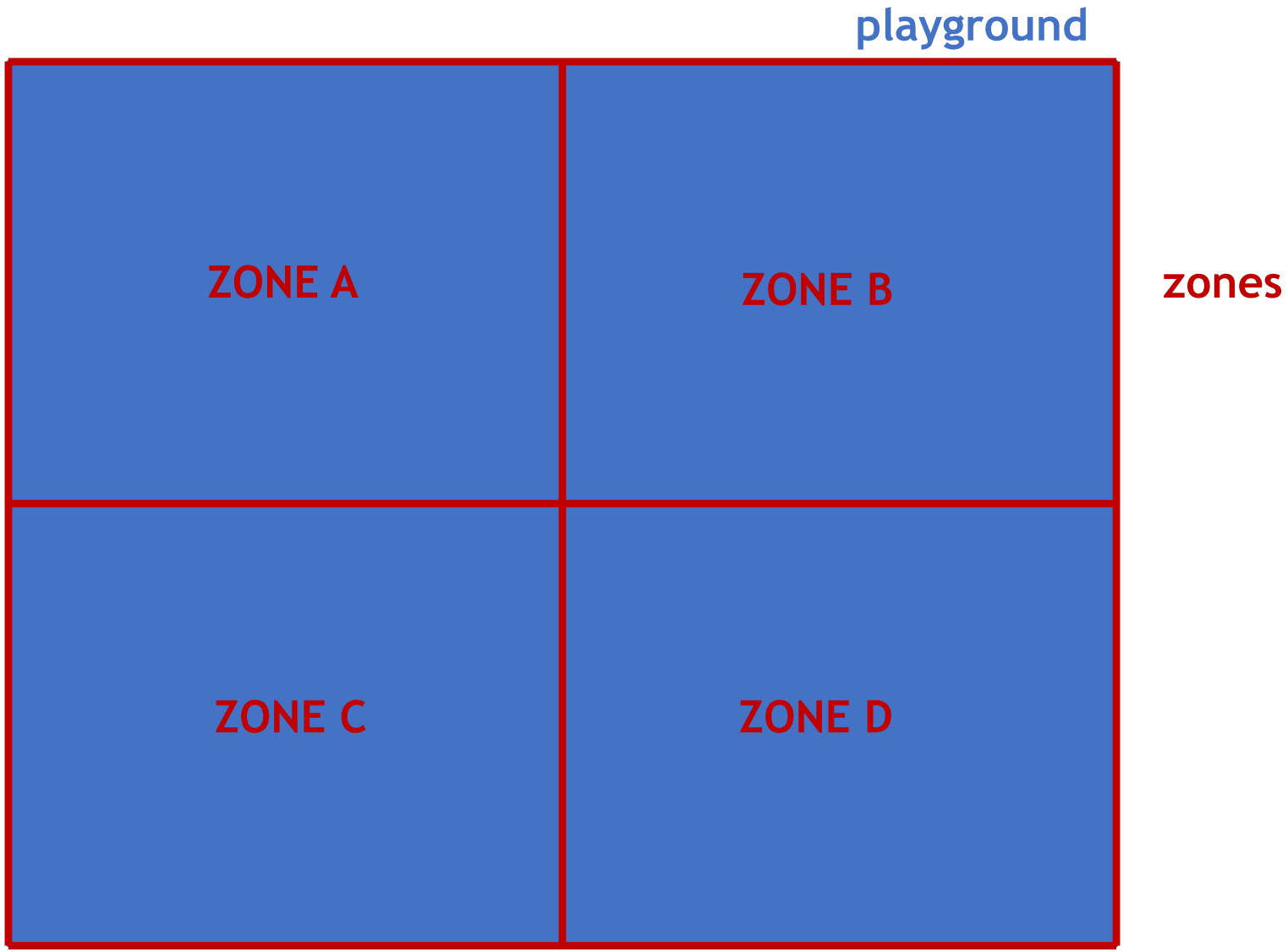


Project 2 [ALGO]: Multi-player Game

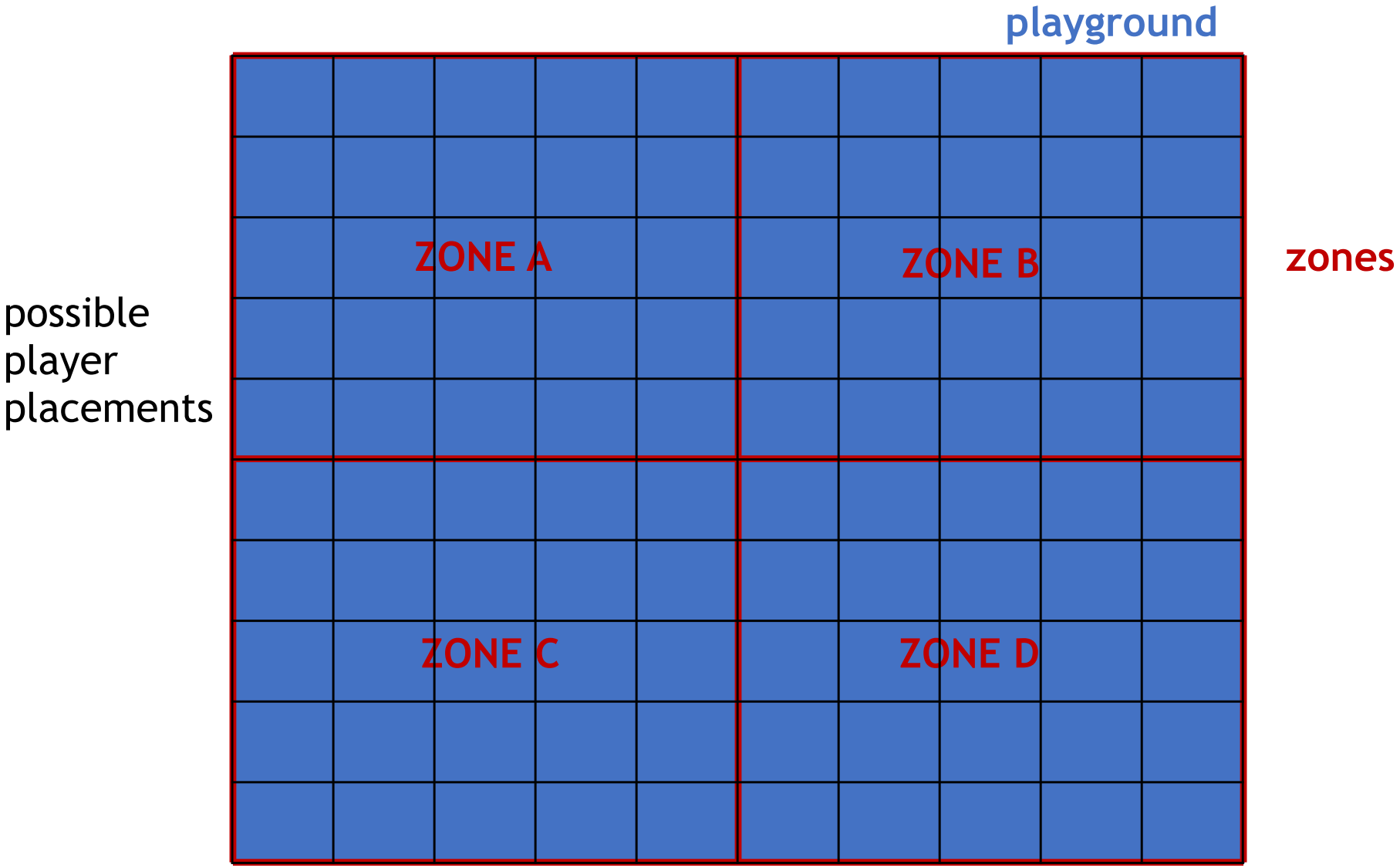
- Goal : Follow the movements of players
- Setting
  - The playground is separated in zones
    - you need to decide how
  - Players move around and pass from one zone to another
    - how do they move?
    - should avoid collisions
    - should say "Hello" to each other
  - Each zone is managed by a node
  - The nodes that manage the set of zones are connected in a distributed topology
    - You need to choose what topology

Project 2 [ALGO]: Multi-player Game (2)
EXAMPLE of a setting

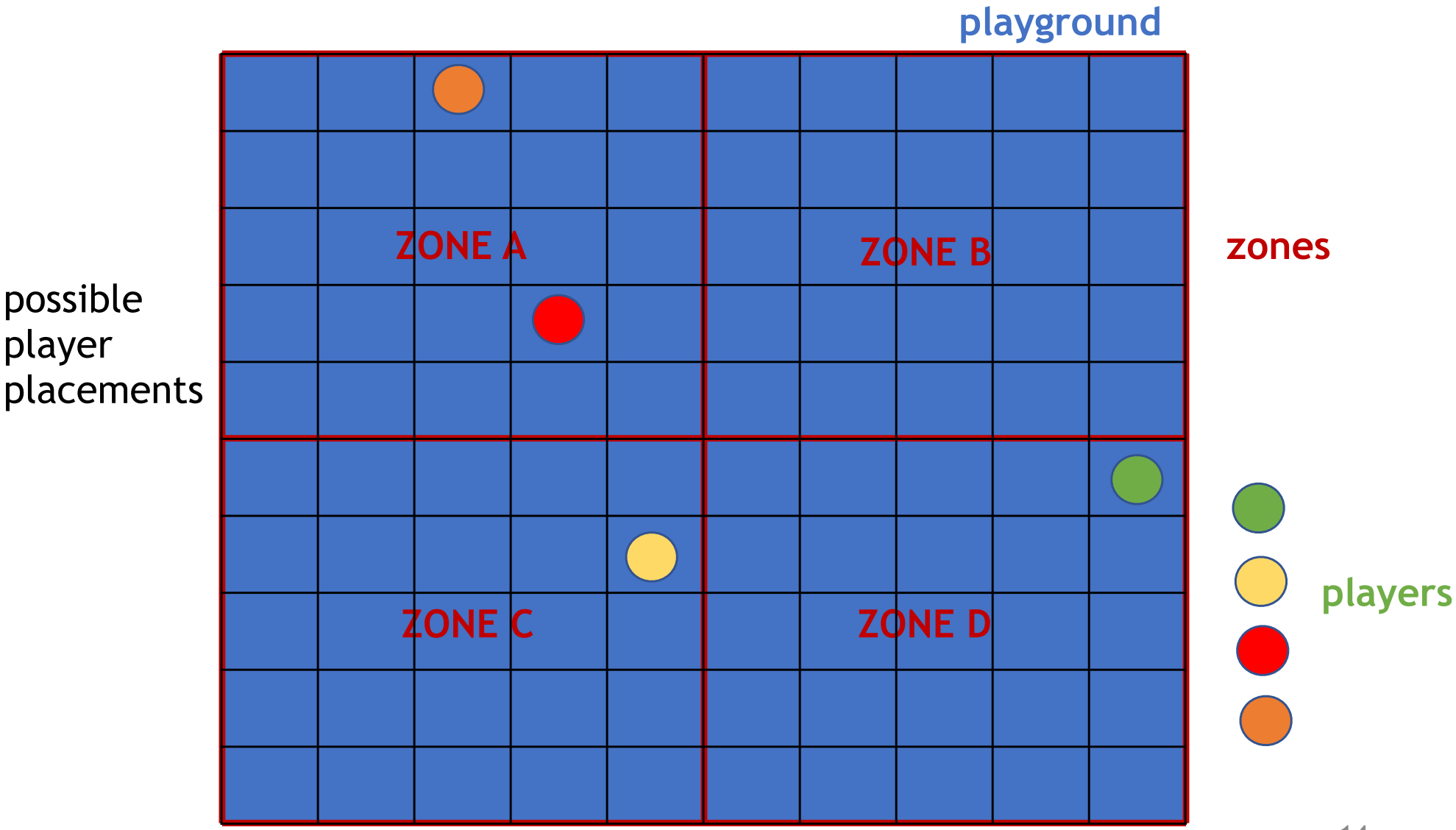
Project 2 [ALGO]: Multi-player Game (2)
EXAMPLE of a setting



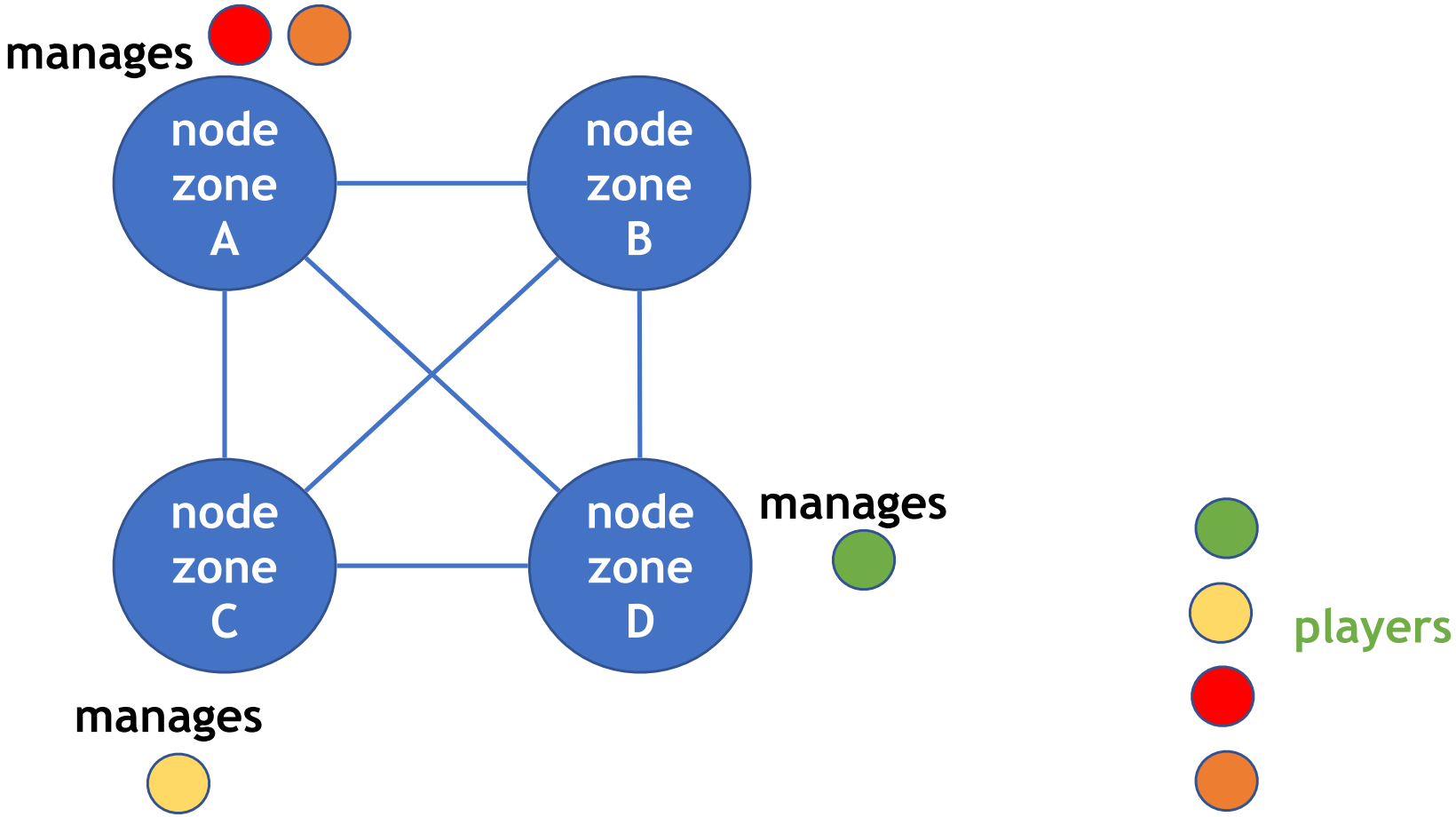
Project 2 [ALGO]: Multi-player Game (2) EXAMPLE of a setting



Project 2 [ALGO]: Multi-player Game (2) EXAMPLE of a setting

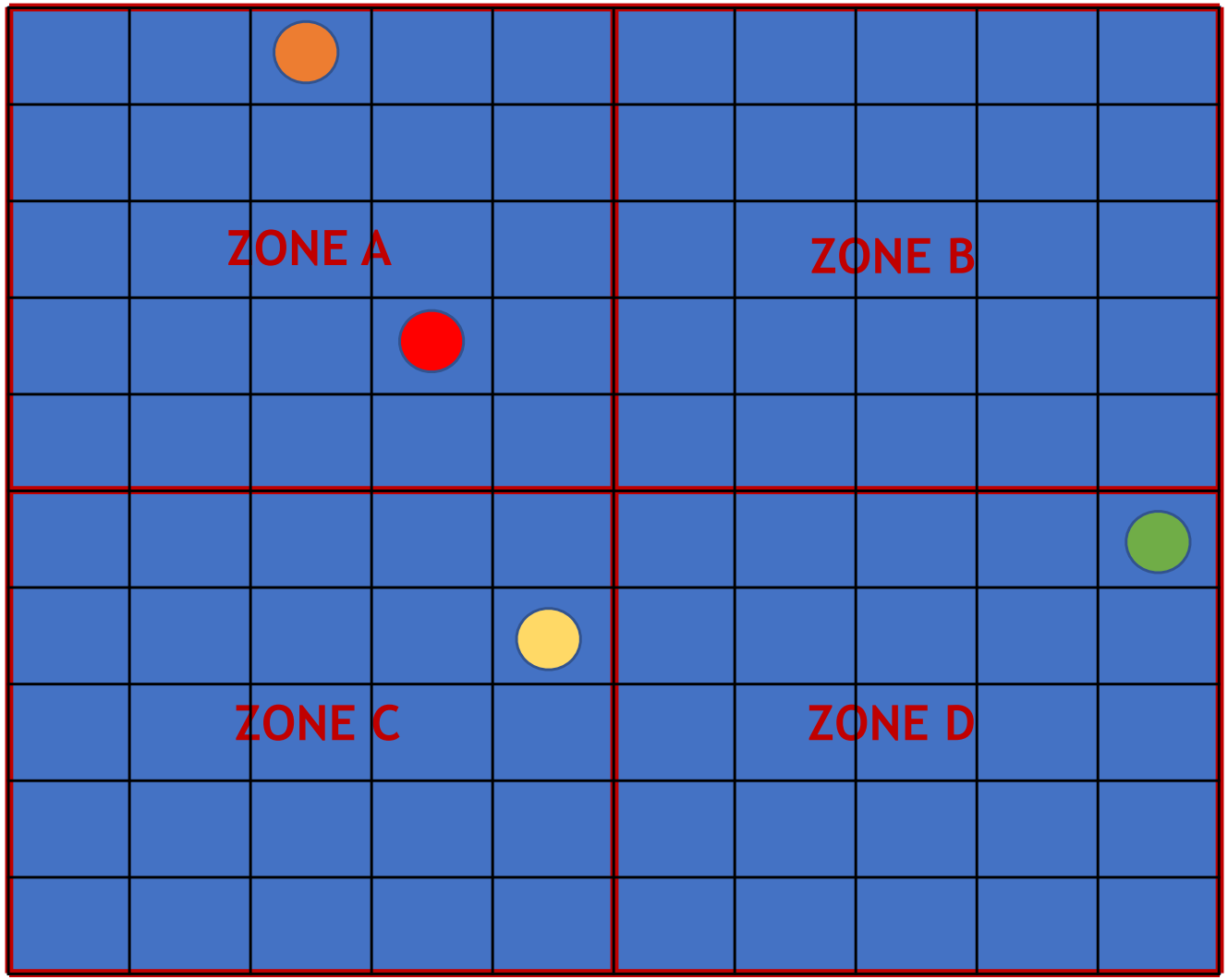


Project 2 [ALGO]: Multi-player Game (2)
EXAMPLE: The corresponding dist. architecture



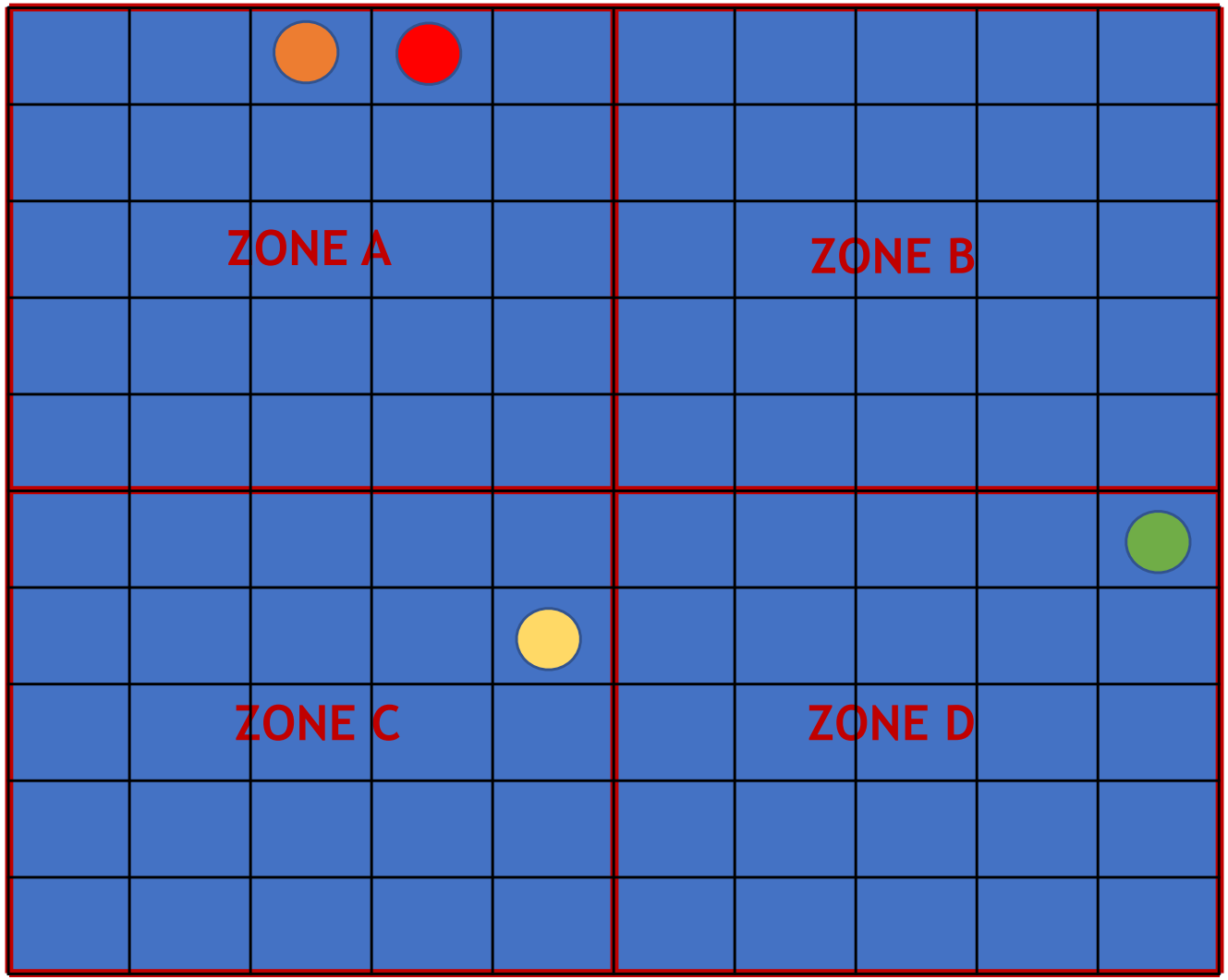
Project 2 : Multi-player Game (2)

Red says Hello to Orange



Project 2 : Multi-player Game (2)

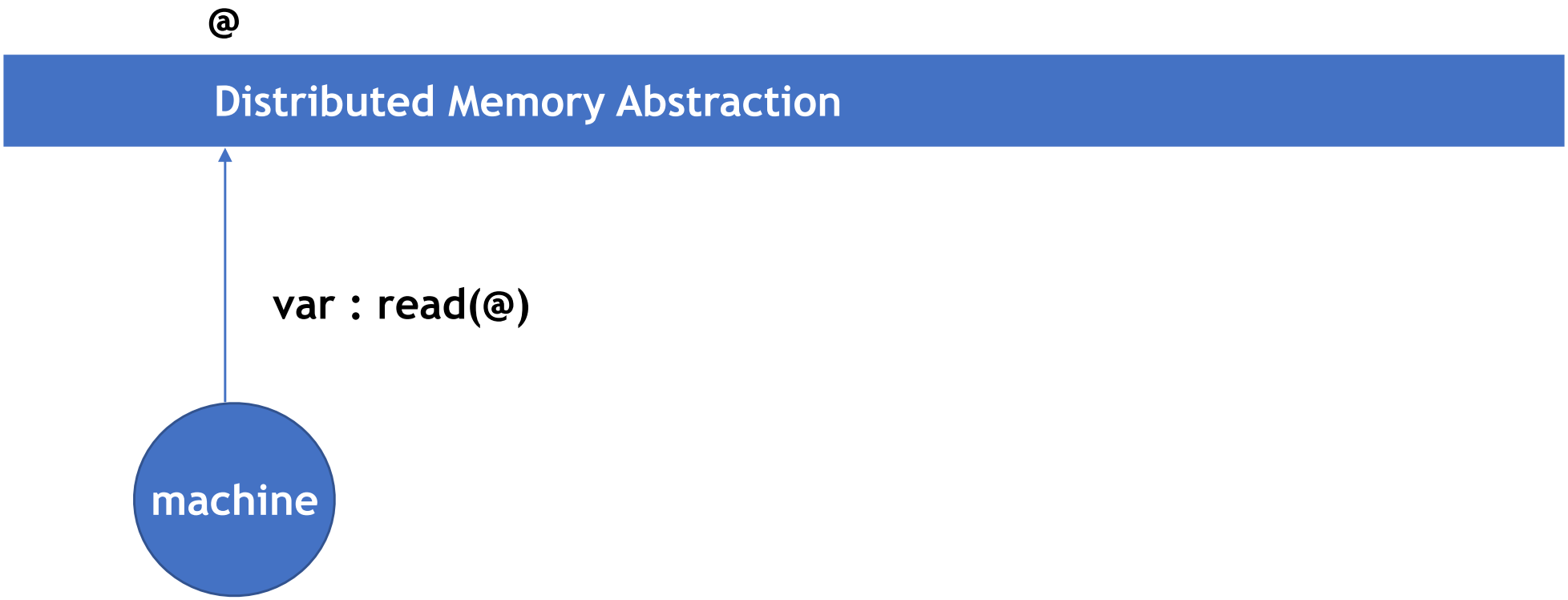
Red says Hello to Orange



Green enters zone C

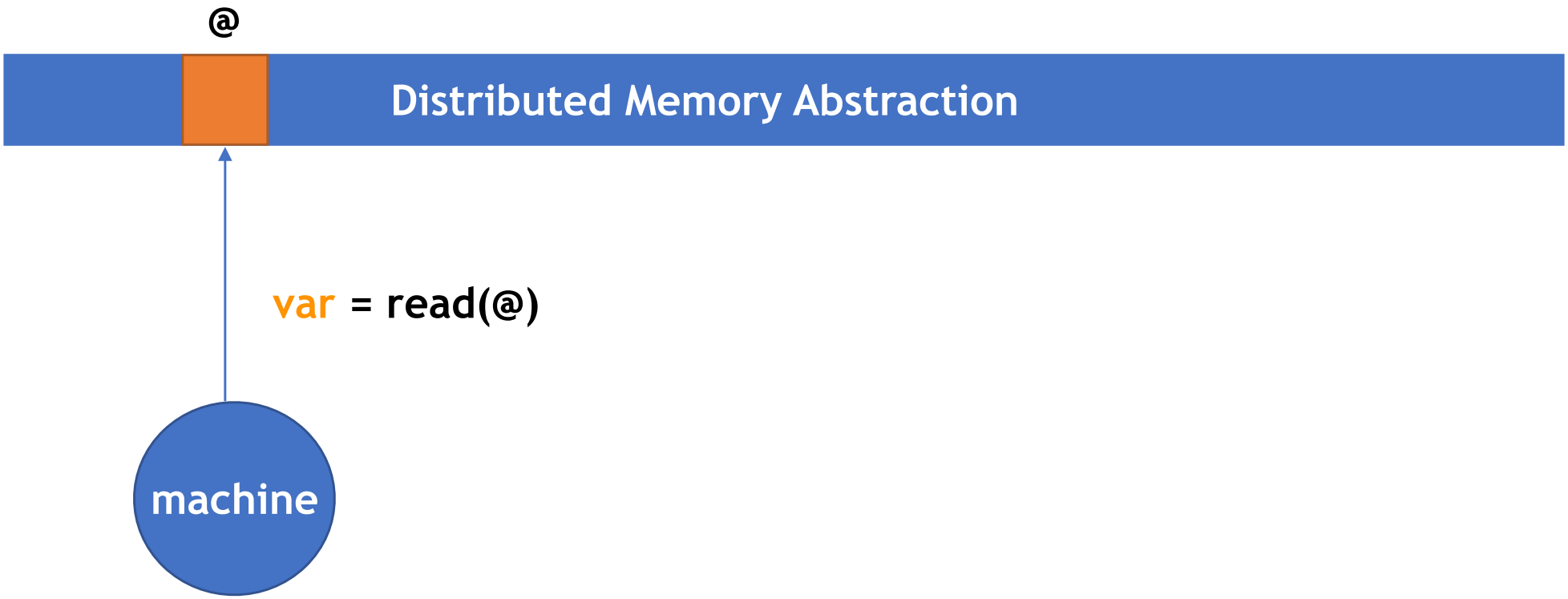
Project 3 [ALGO]: DSM (Distributed Shared Memory)

- Distributed Shared Memory Abstraction



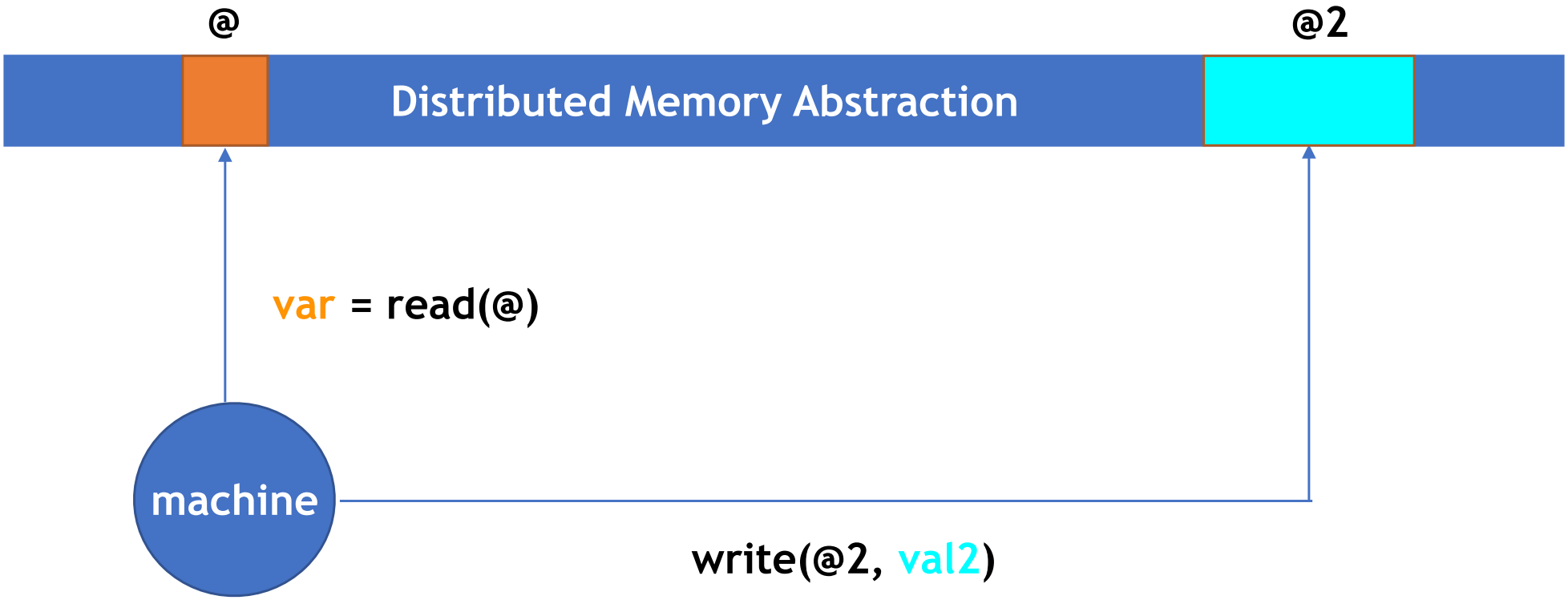
Project 3 [ALGO]: DSM (Distributed Shared Memory)

- Distributed Memory Abstraction



Project 3 [ALGO]: DSM (Distributed Shared Memory)

- Distributed Memory Abstraction



Project 3 [ALGO]: DSM (Distributed Shared Memory)

To Implement the primitives

- `value read(address)`
- `write(address, value)`

1. You need to decide what is an address and how to manage addresses
2. You need to decide what kind of values you will manipulate

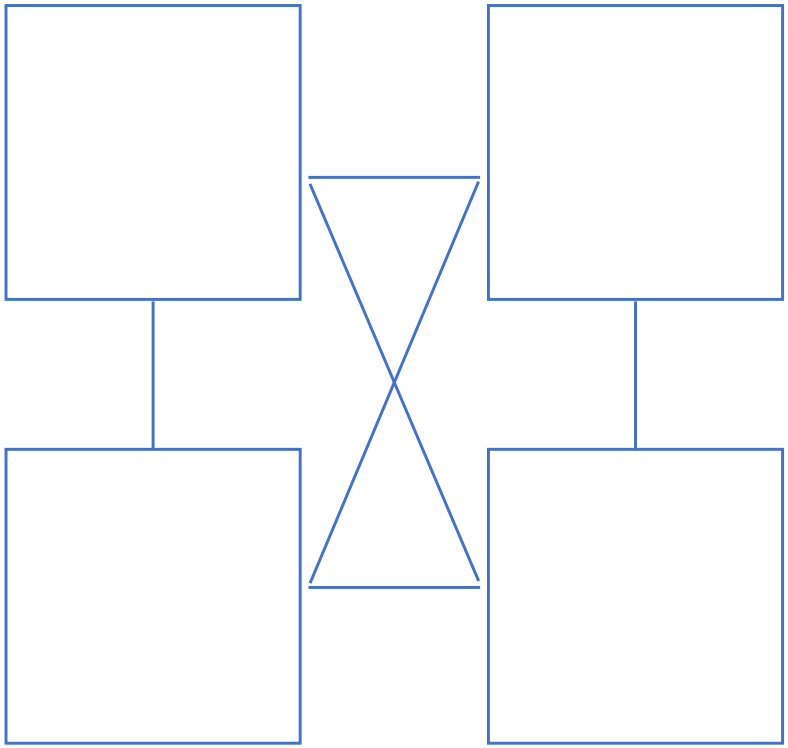
Project 3 [ALGO]: DSM (Distributed Shared Memory)

- The memory is actually distributed over several physical machines
  - You need to decide how to distribute the memory
- The machines are interconnected into a distributed topology
  - You need to choose the topology

- You need to decide what is an address and how to manage addresses
- You need to decide what kind of values you will manipulate

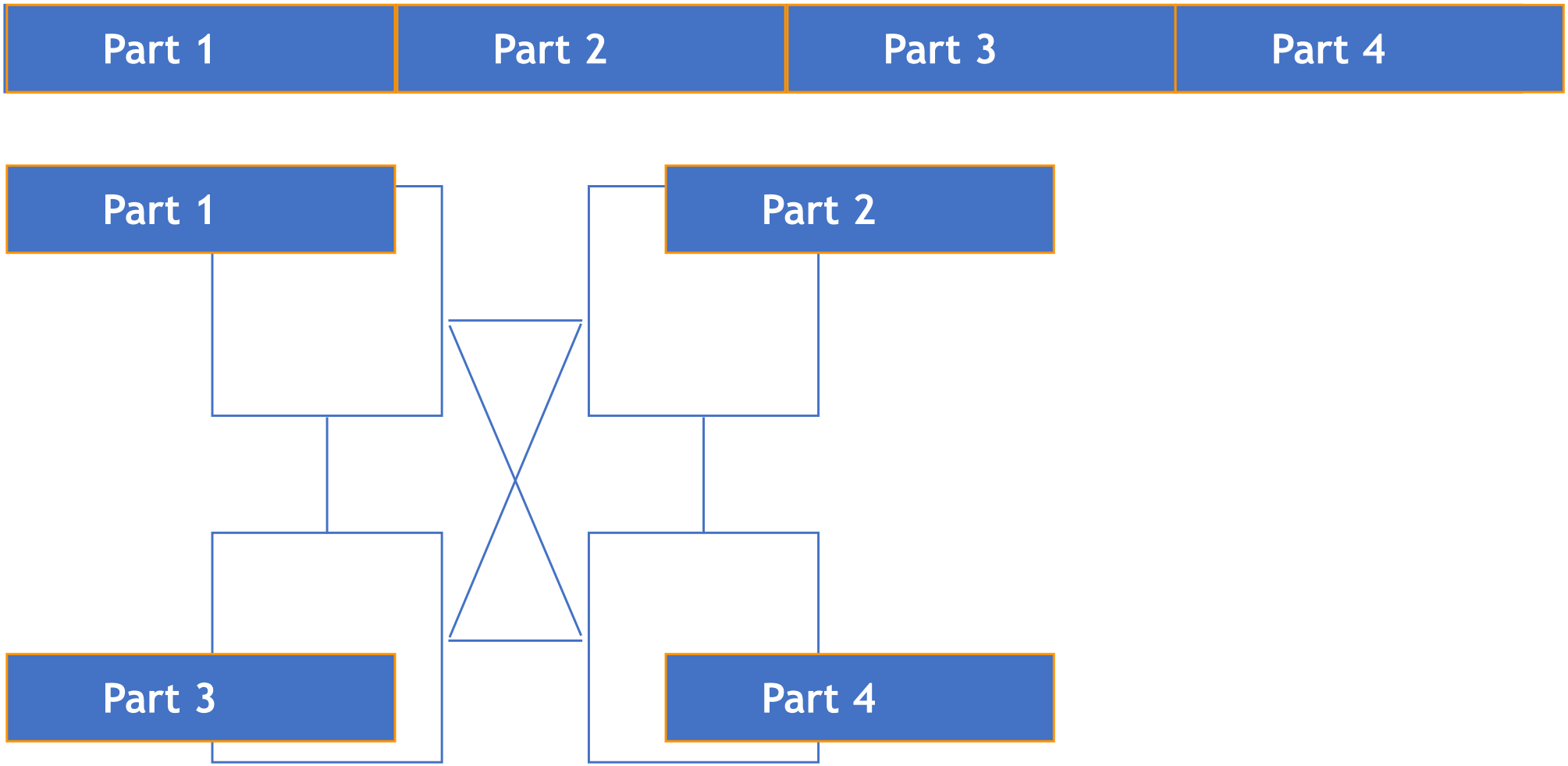
Project 3 [ALGO]: DSM EXAMPE Setting

Distributed Shared Memory Abstraction

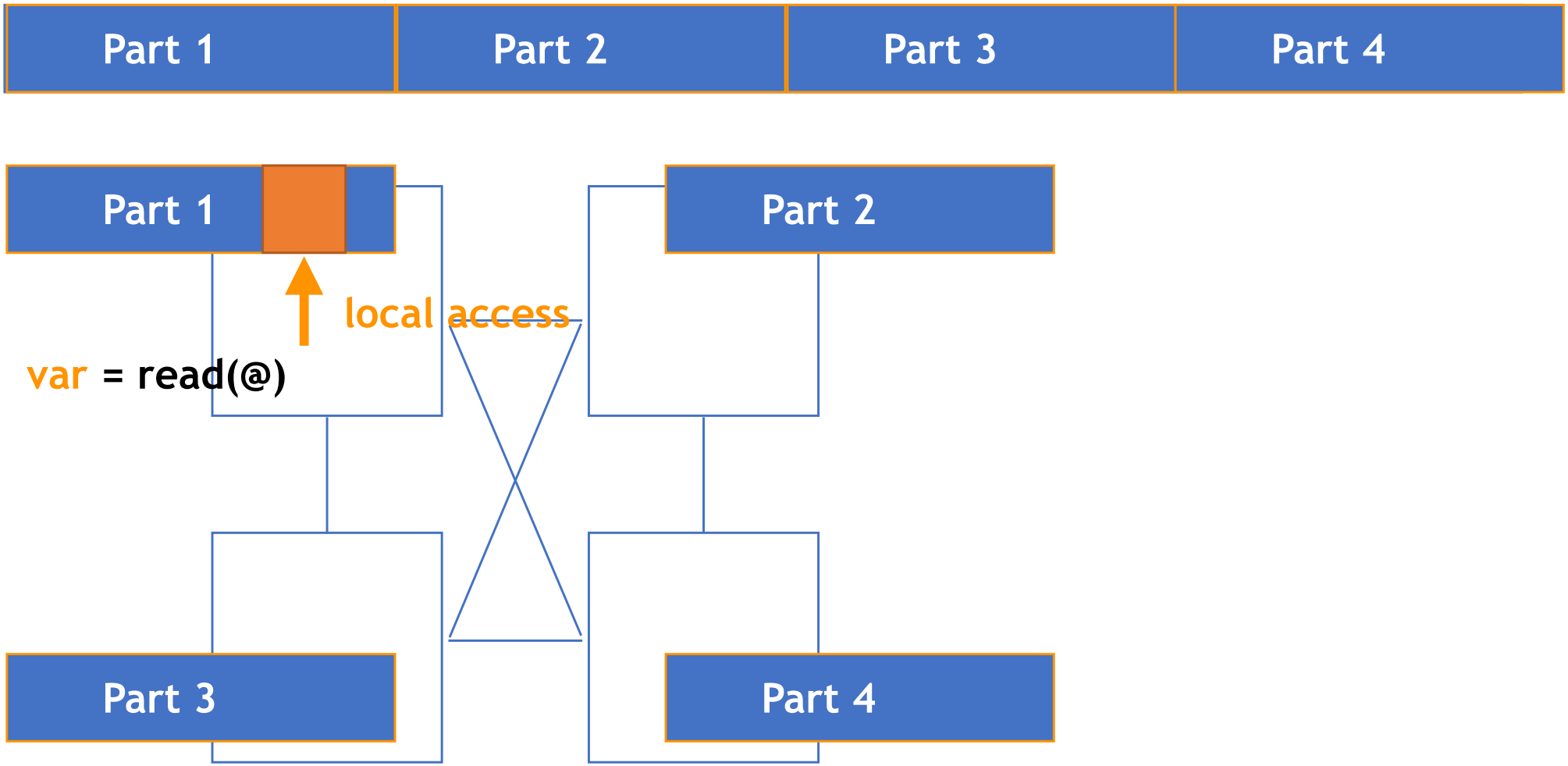


Physical machines in a grid

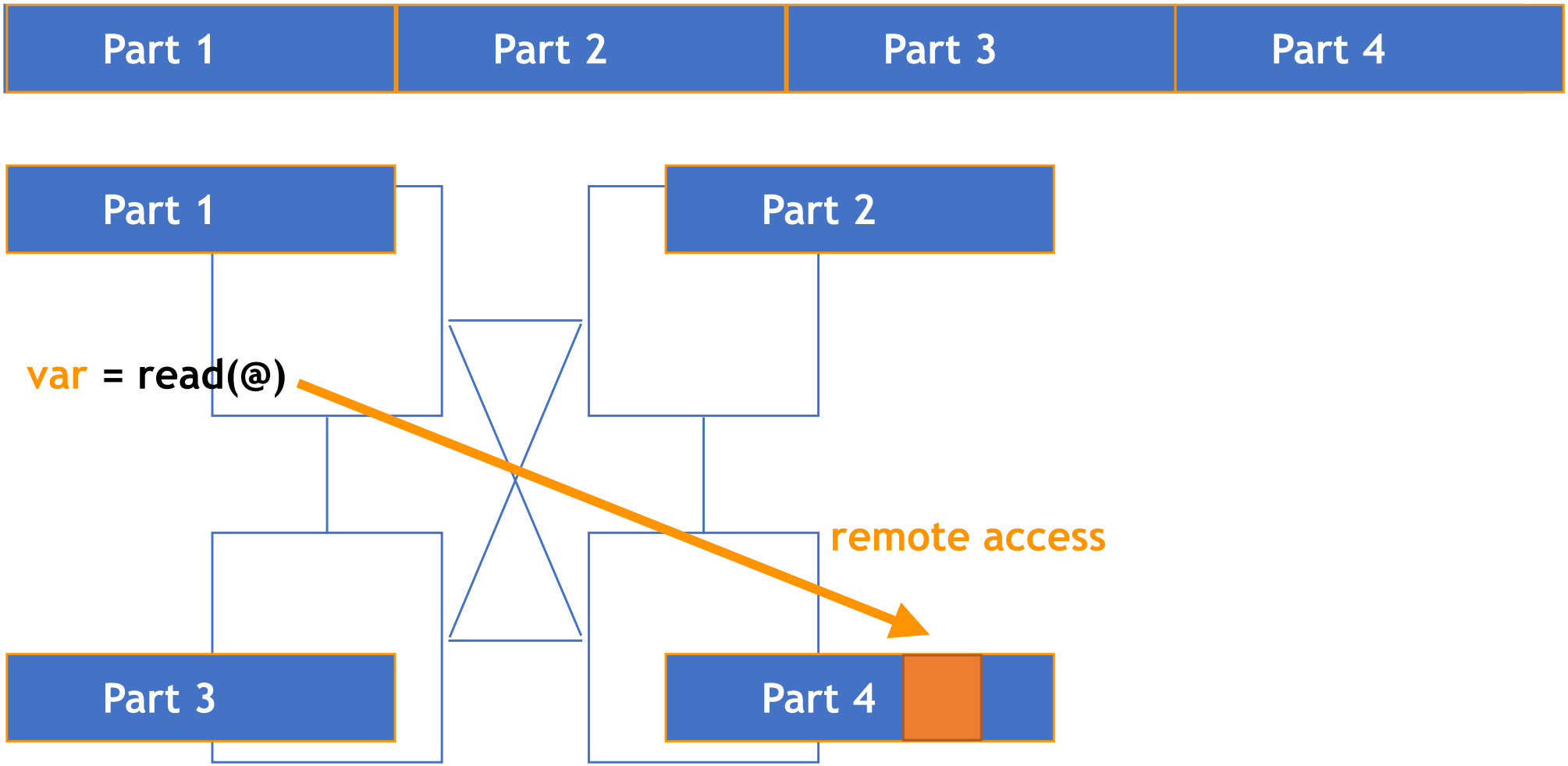
Project 3 [ALGO]: DSM EXAMPE with Partitioning



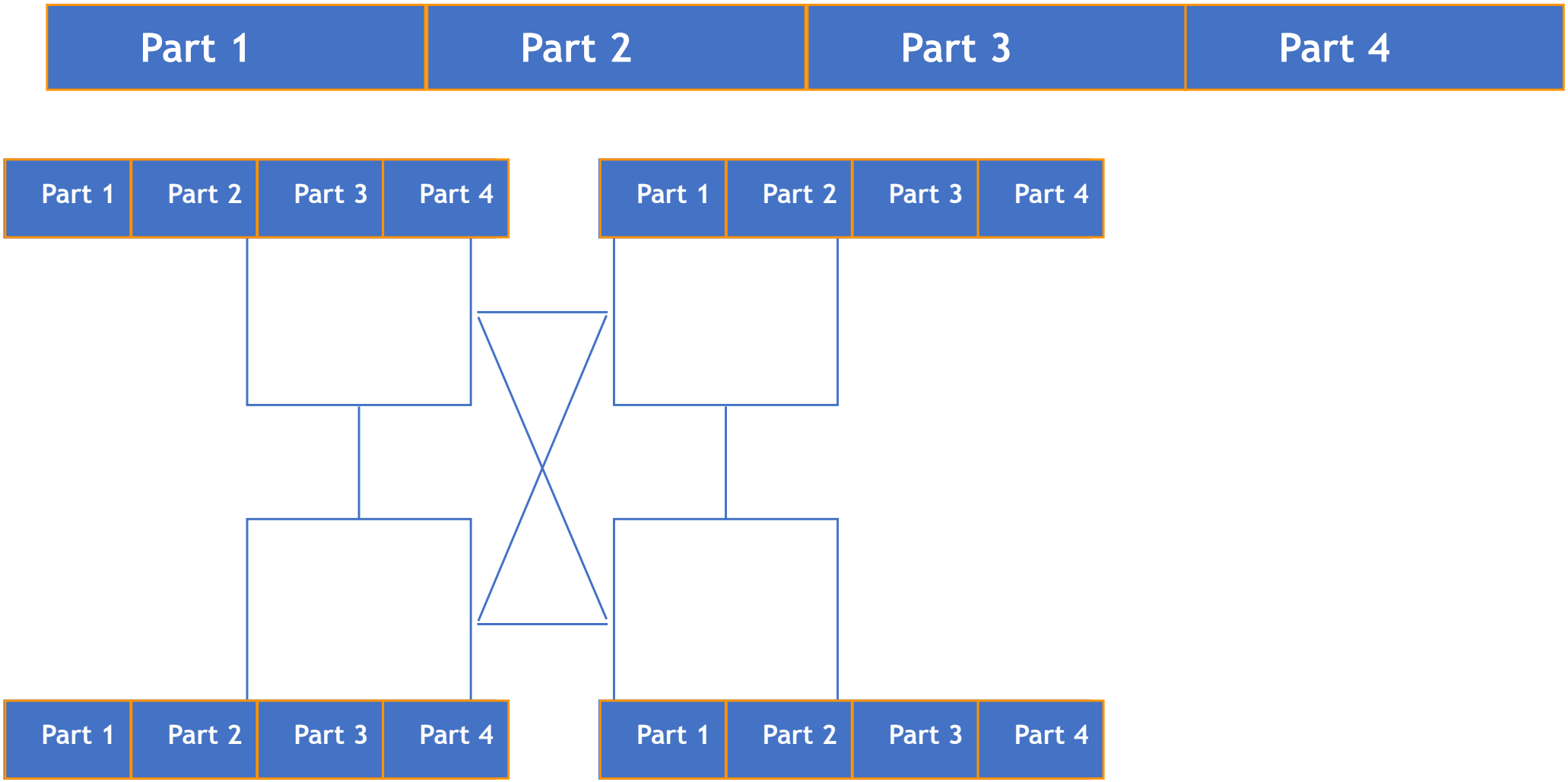
Project 3 [ALGO]: DSM EXAMPE with Partitioning



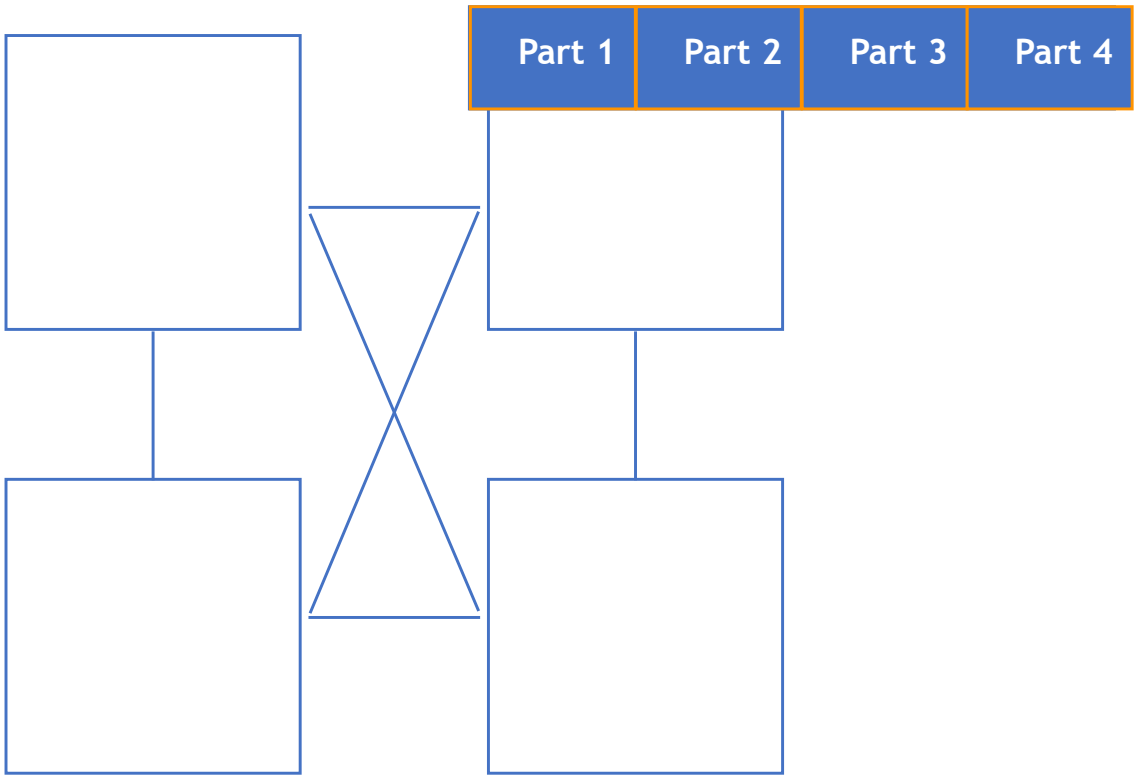
Project 3 [ALGO]: DSM EXAMPE with Partitioning



Project 3 [ALGO]: DSM EXAMPE with Replication



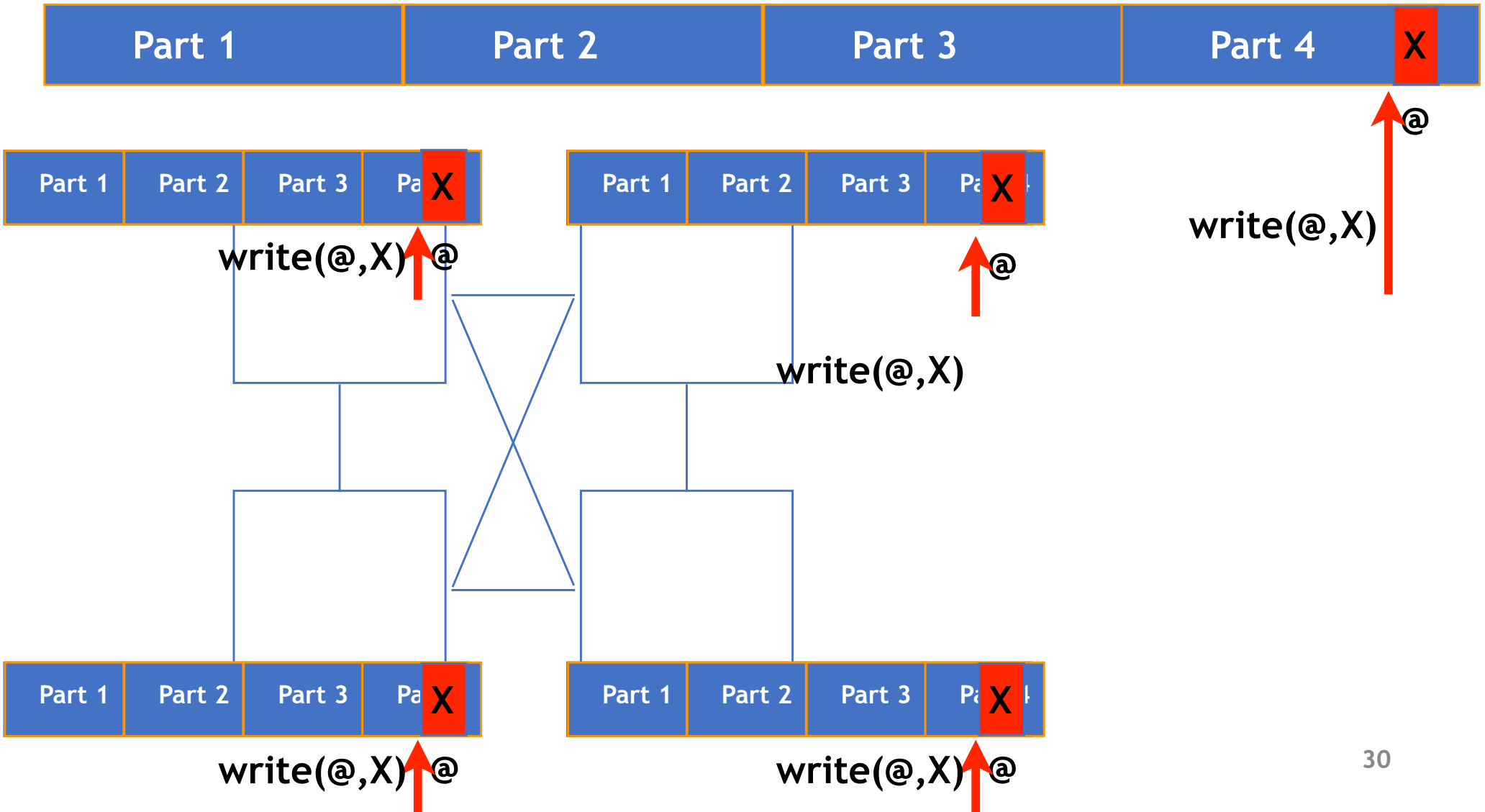
Project 3 [ALGO]: DSM EXAMPE with Non Replication



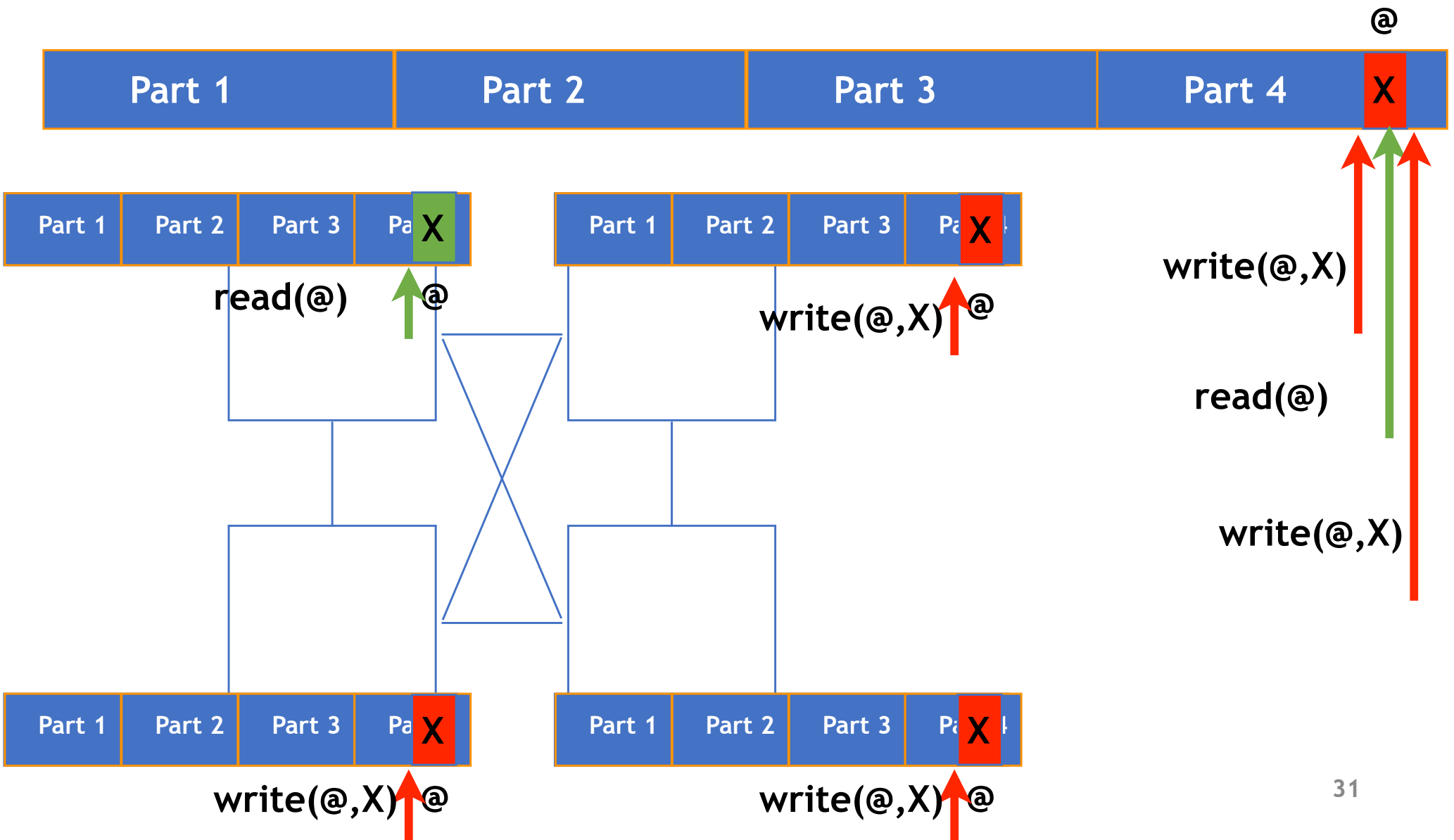
The big problem is consistency
(this is really complicated)

- If there are only read operations, no problem
- The pb is that there are write operations
- What if two write operations concern the same address?
  - Which one will go first?
  - What value will the others observe? (if they read from this address)

Strong consistency



Consistency?



References (on Moodle)

1. Fekete, A. D., Ramamritham, K.: Consistency Models for Replicated Data. In Charron- Bost, B., Pedone, F., Schiper, A., eds. : Replication LNCS 5959. Springer-Verlag, Berlin (2010) 1-17

2. Burckhardt, S.: Principles of Eventual Consistency. Foundations and Trends Programming Languages 1(1-2), 1-150 (2014)