A theory of mobile tiny devices

Rachid Guerraoui
EPFL

A world of tiny mobile devices

The march of the Penguins

The march of the Penguins

- The group is threatened if more than a threshold dies on the way back to the sea
- If a penguin starts its trip with a very low temperature, the probability that it reaches the sea is very low

The Escort

- Provide each penguin with a computing device to:
  - measure its temperature;
  - trigger an alert if a threshold (say 5) has a very low temperature

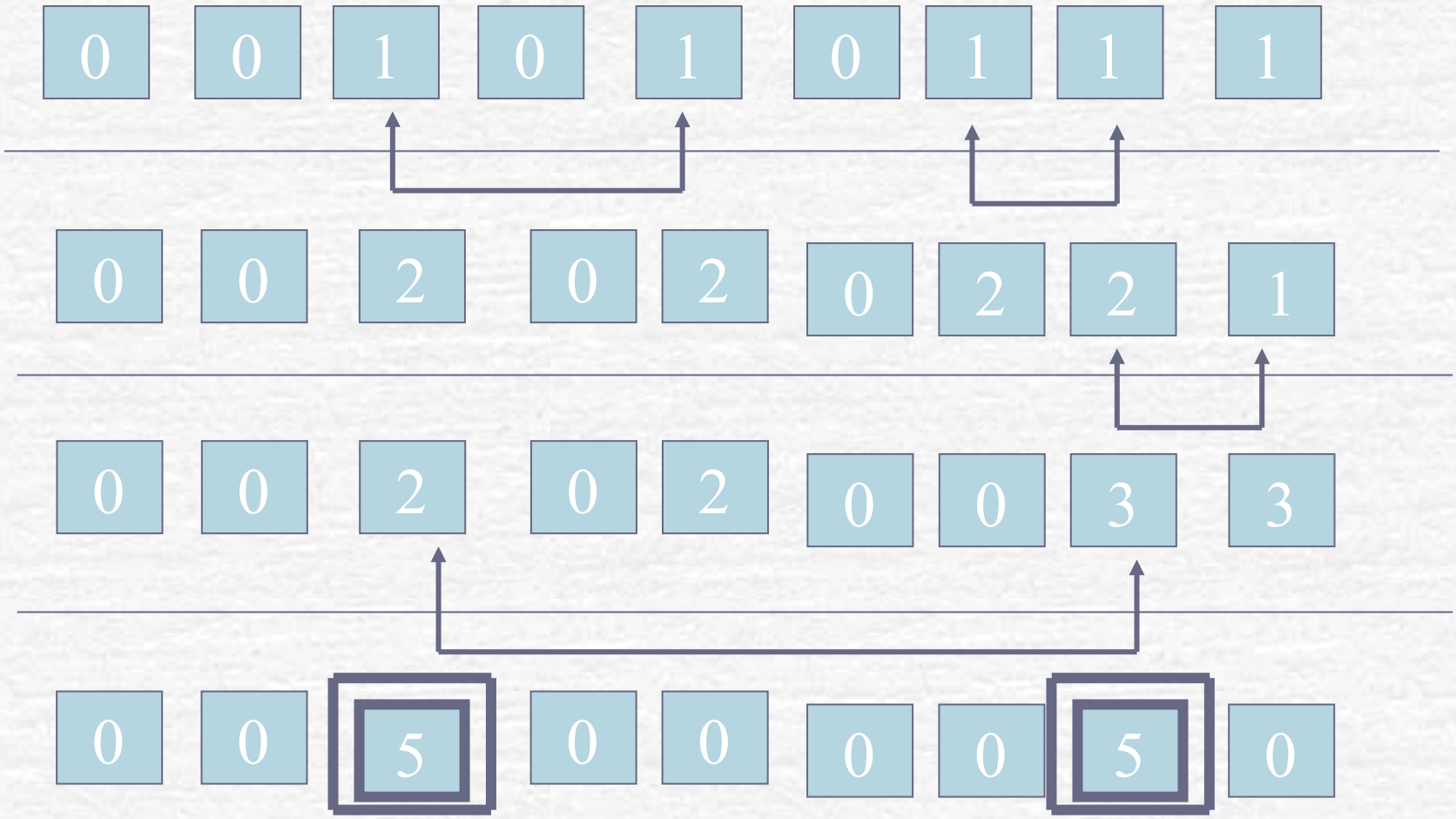
Assumptions

- Every device holds a finite counter (<6)
  - Its initial value is 1 if the penguin has a low temperature and 0 otherwise
- A pair of devices communicate if they get close enough to each other
  - Every pair of devices eventually meet

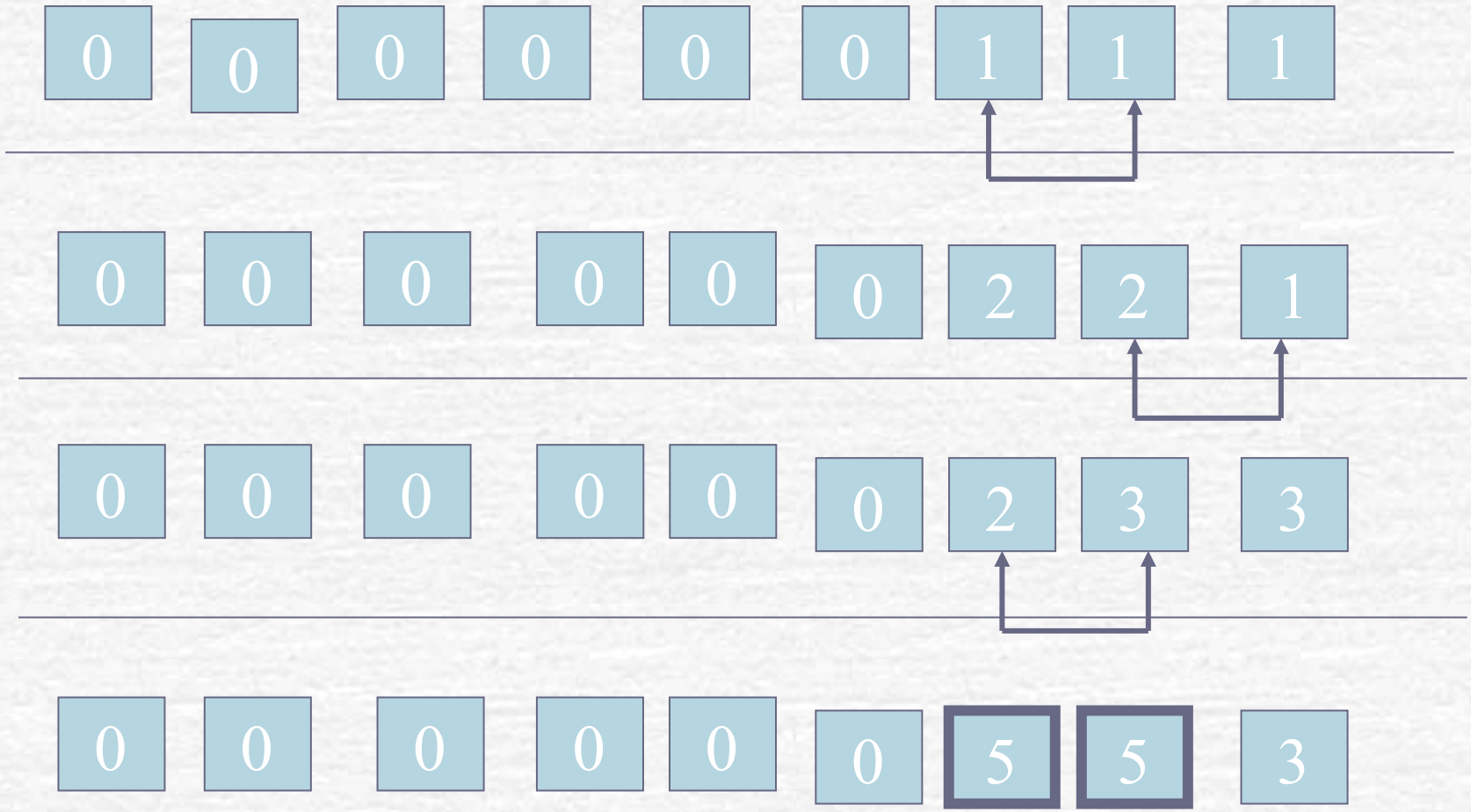
The problem

- All devices eventually output “alert” iff at least 5 initial values are 1

Algorithm ?



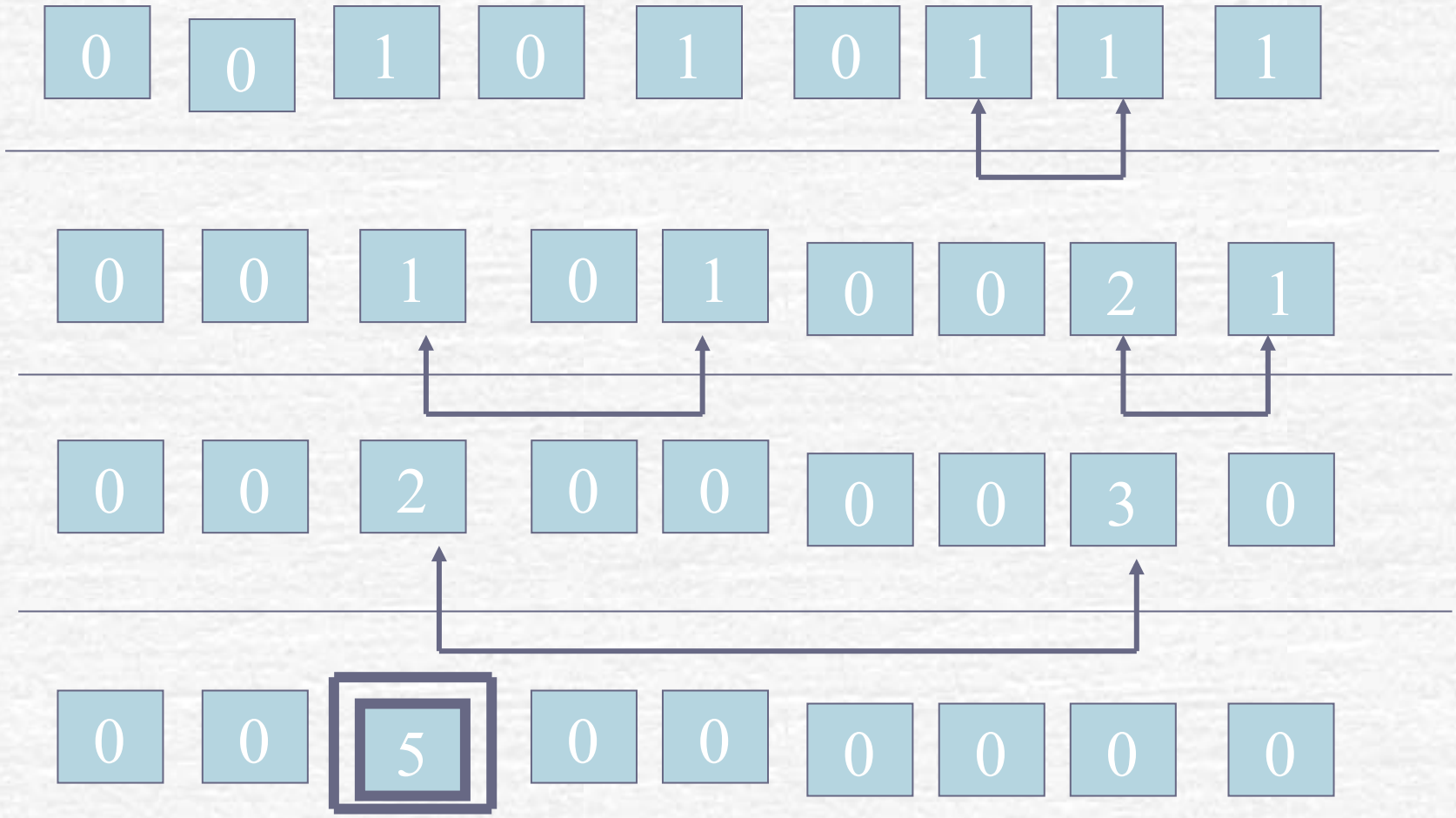
Algorithm?



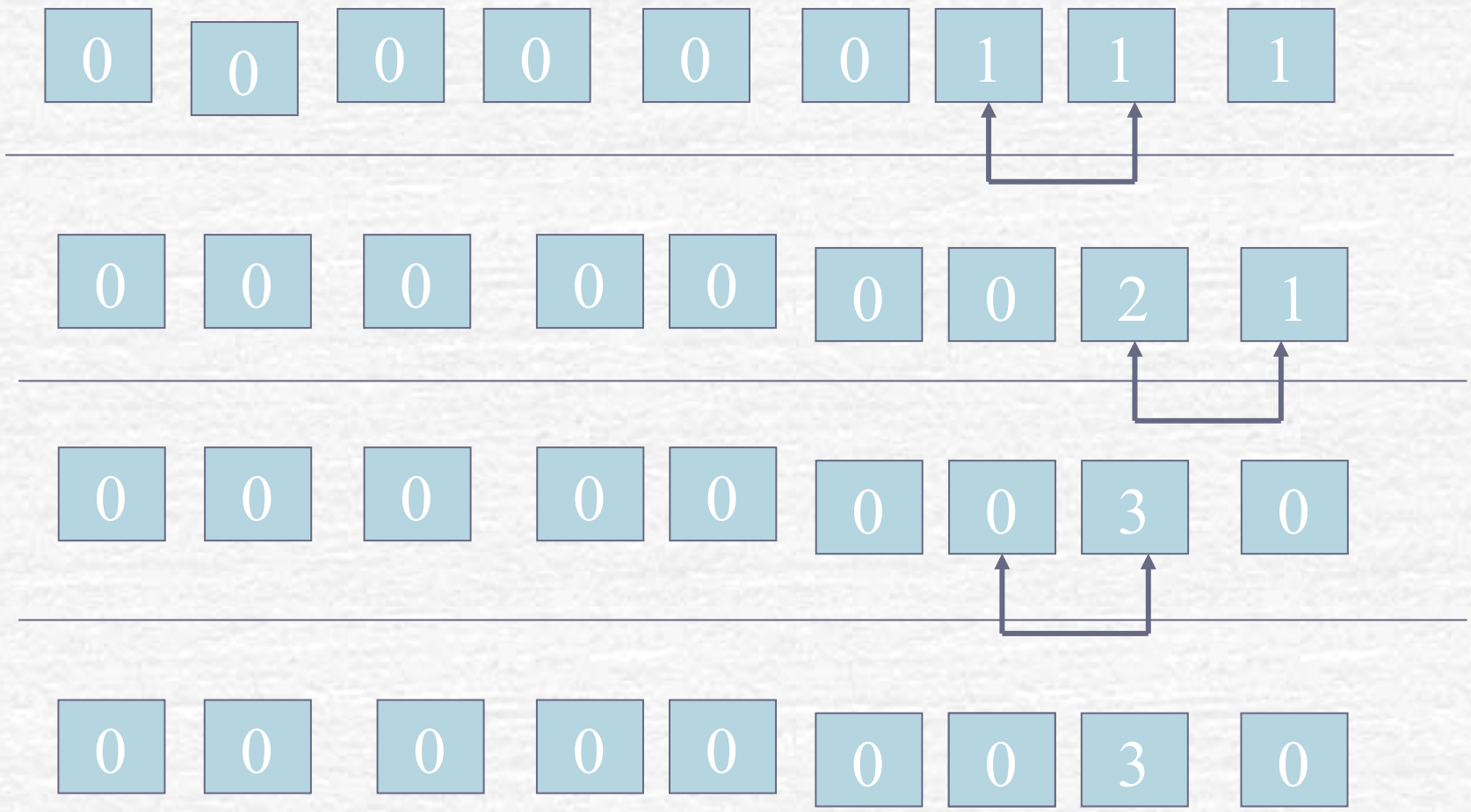
Algorithm

- When two devices meet, one keeps in its counter the sum of the values whereas the other puts it back to 0
- Any device with value 5 triggers the alert

Algorithm



Algorithm



Population protocols (DF01,AADFP'04)

- A population $\mathbf{P}$ is a set of ***agents***
- Every agent has a ***bounded*** memory independent of the size of the system
- Algorithms are ***asynchronous, uniform*** and ***anonymous***

Population protocols

- The agents have no control over their ***mobility*** pattern
- A pair of agents communicate if they get ***close*** to each other (one is the ***initiator***)
- Every interaction that is possible eventually happens (***fairness***)

Population protocol

- Input; Output; State S
- InMap In; OutMap Out
- Transition δ

Example ("or")

- Input = Output = State = {0,1}
- In = Out = identity
- Transitions: any agent with "0" that meets an agent with "1" turns to "1"
- Transitions:
  - $\delta(*,1) = \delta(1,*) = (1,1)$
  - $\delta(0, 0) = (0,0)$

Configuration and execution

- A ***configuration*** is a set of states of all agents
- An ***execution*** is a sequence of configurations

Fairness

- An infinite execution E is ***fair*** if for any transition δ(C,C'), if C appears infinitely in E, then C' appears infinitely in E
- A ***computation*** is a fair execution

Stability and convergence

- A configuration C is ***stable*** if for every C' such as $\delta(C,C')$, we have: $Out(C) = Out(C')$

- An infinite execution ***converges*** if it has a stable configuration

Back to the Penguins

- All devices eventually output “alert” iff at least 5 initial values are 1

Back to the Penguins

- Input = {0,1}
- Output = {ALARM, OK}
- S = {0,1,2,3,4,5}
- In (identity): 0 → 0, 1 → 1
- Out:{0, 1, 2, 3, 4} → OK
  5 → ALARM

Back to the Penguins

- δ: (i, j)→ (i+j, 0) if i+j <5
  (5, 5) if i+j ≥5

Example (“>”)

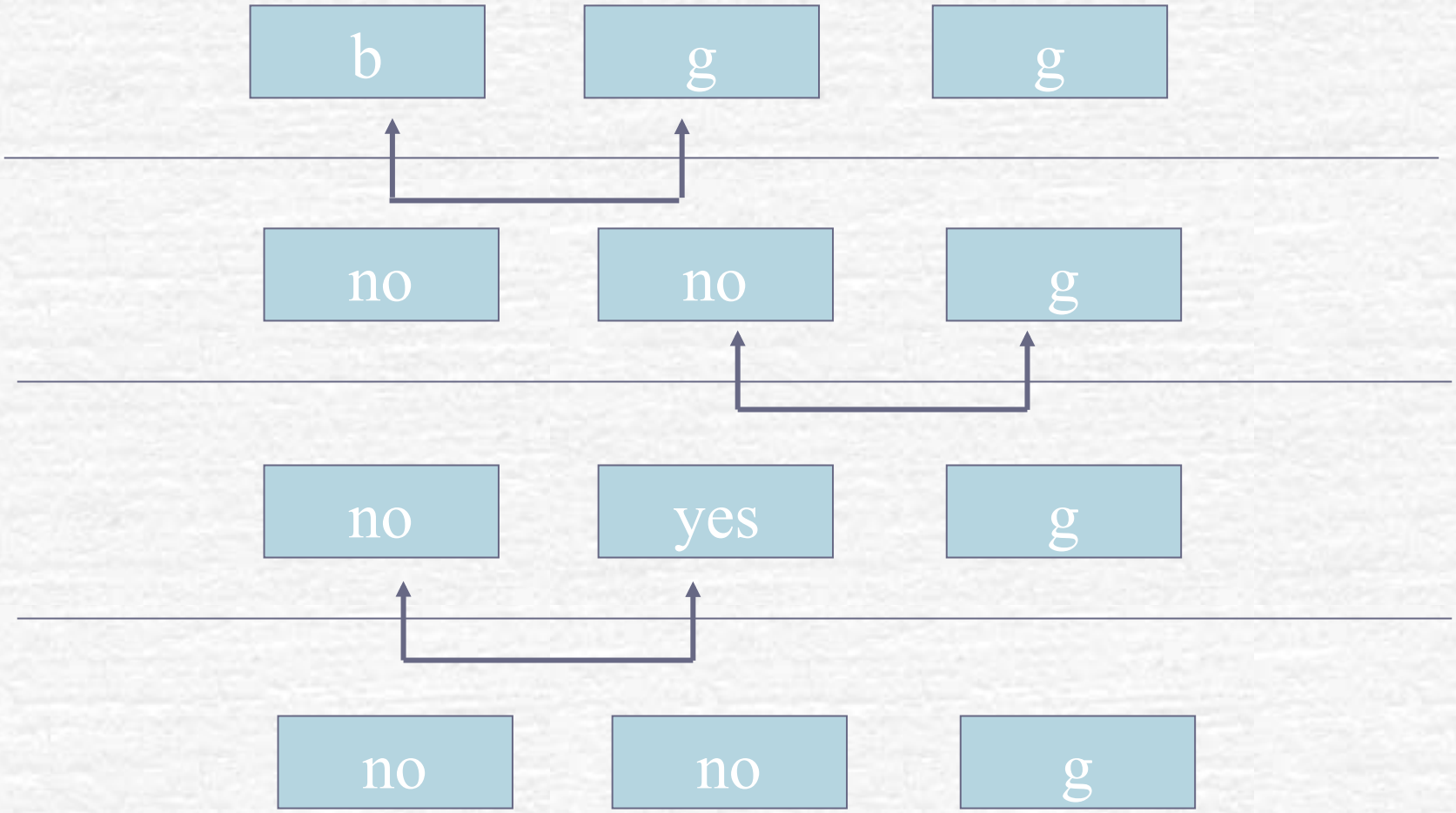
- Determining if there are strictly more ***girls*** than ***boys*** in a party

- When a boy meets a girl they “***cancel***” each other
- If more girls remain, then output “yes”; else output “no”

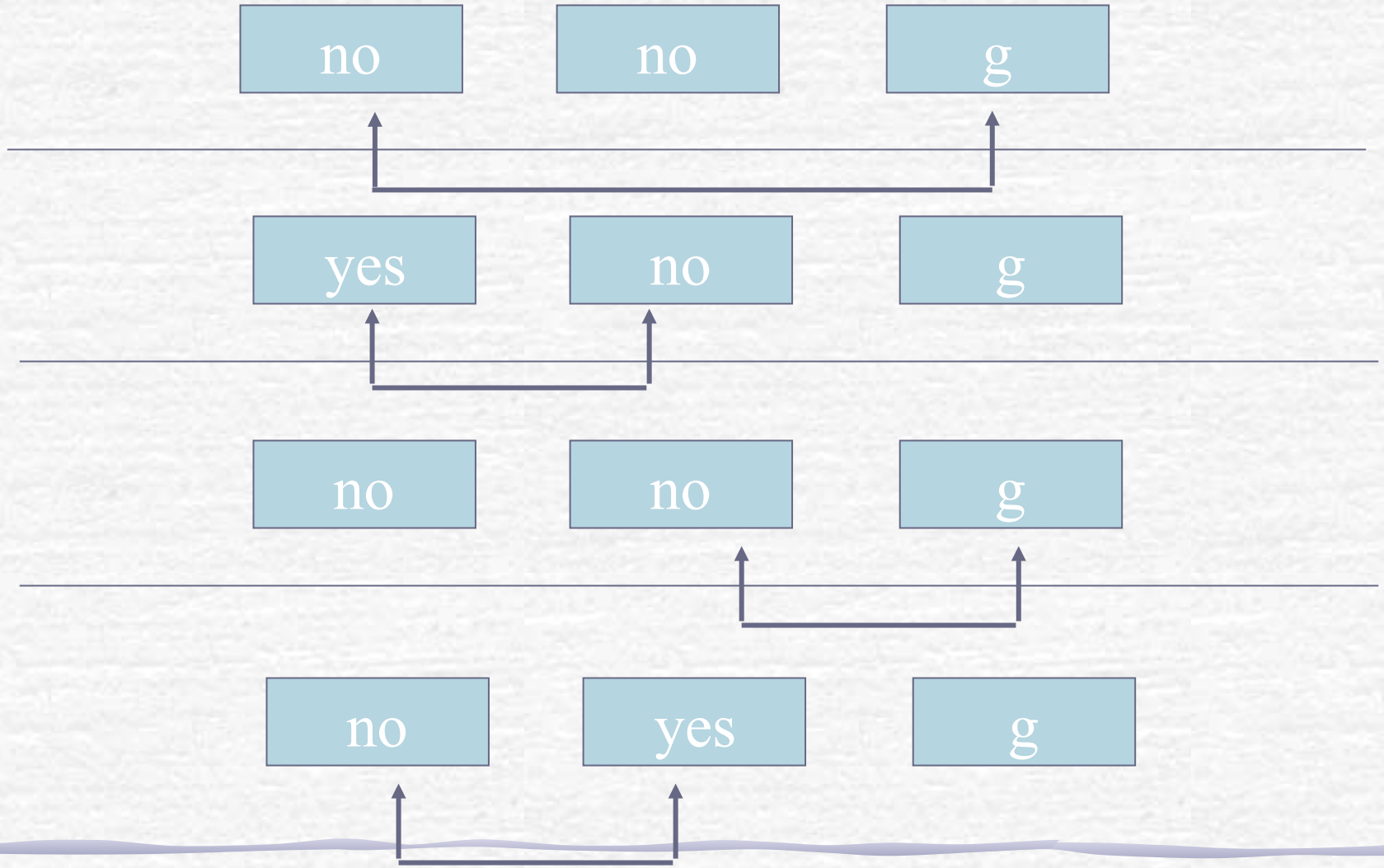
Example (“>”)

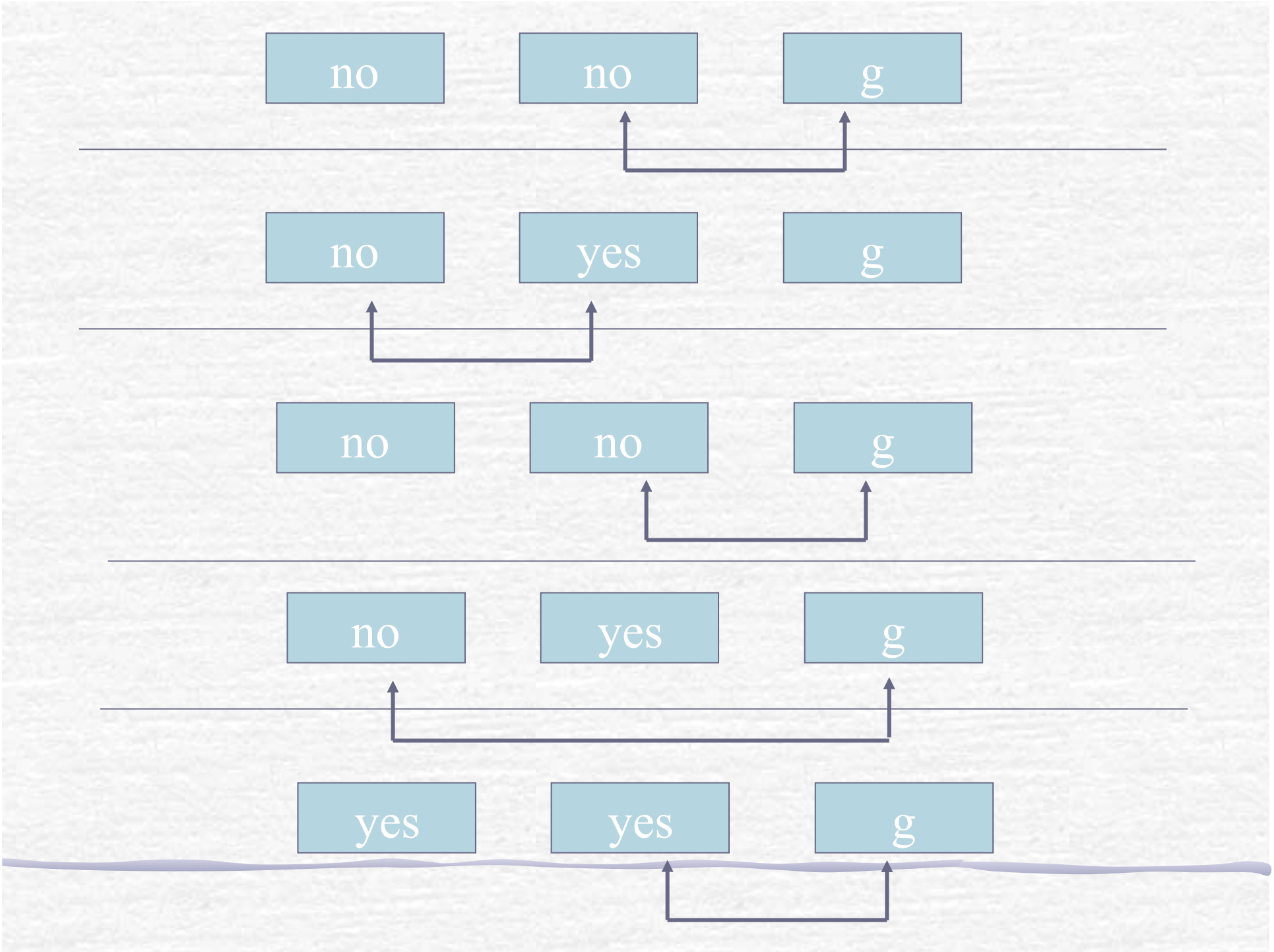
- Input = {g,b}; Output = {yes,no}
- State = {g,b,yes,no}
- Transitions
  - (g,b) -> (no,no)
  - (g,no) -> (g,yes)
  - (b,yes) -> (b,no)
  - (yes,no) -> (no,no)

Execution



Execution




no
no
g
no
yes
g
no
no
g
no
yes
g
yes
yes
g

Sum

- Determining if the numbers of *C*'s is the sum of the number of *A*'s and the number of *B*'s
- Whenever a A sees a B, it cancels a C…

Product

- Determining if the numbers of *C*'s is the product of the number of *A*'s and the number of *B*'s
- Whenever a A sees a new B, it cancels a C
- Problem - what is a ***new*** B?

More generally (AAD06)

- ***Theorem: population protocols compute exactly first order Presburger's arithmetic: +, -, =, >, or, not, and, ...***

- NB. Not as powerful as Peano's arithmetic which also include *

Roadmap

(1) Mobility

(2) Crashes

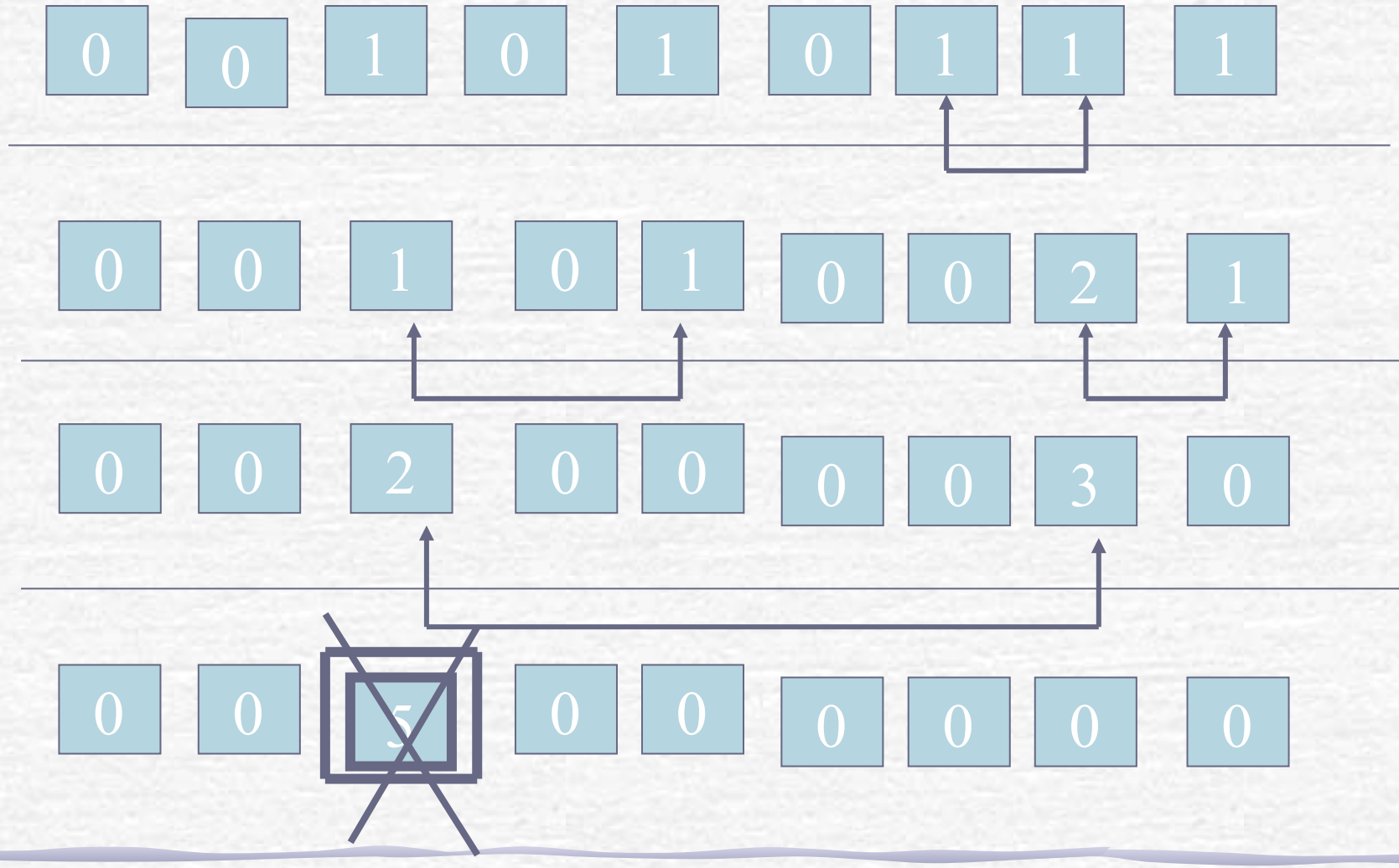
(3) Privacy

(4) Security

But what if?

- One of the agents fail (say by crashing at some inappropriate time)
- An agent might crash exactly when it reaches value 4

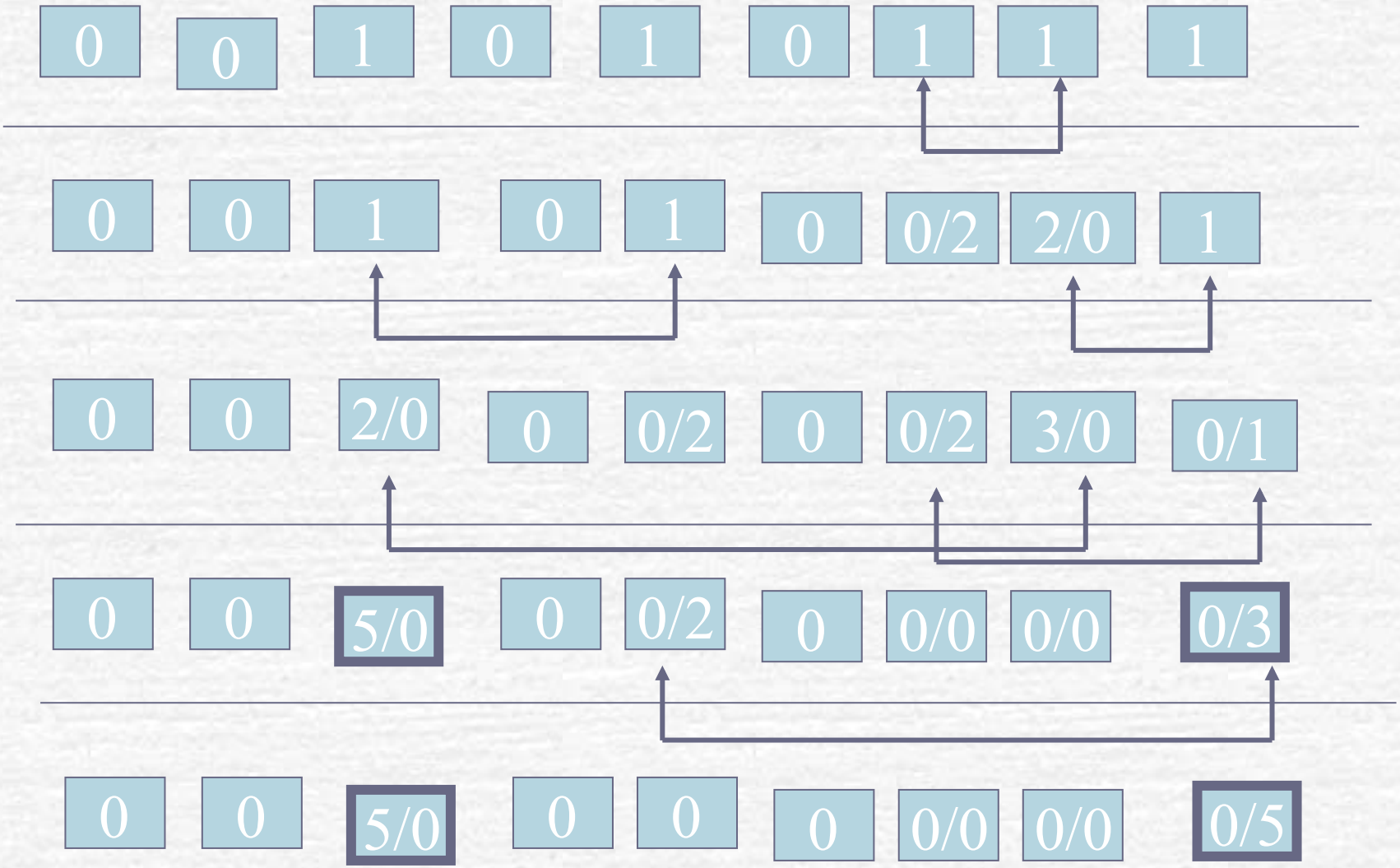
Original algorithm O



Reliable algorithm

- Every agent performs ***twice*** the original algorithm: O1 and O2
- When two agent communicate, one acts as the initiator for O1 and the other as the initiator for O2

Reliable algorithm



More generally (DFGR06)

- ***Theorem: population protocols compute exactly Presburger's arithmetic with a constant number of crashes***

- NB. The notion of "computation" is slightly revised

Roadmap

(1) Mobility

(2) Crashes

(3) Privacy

(4) Security

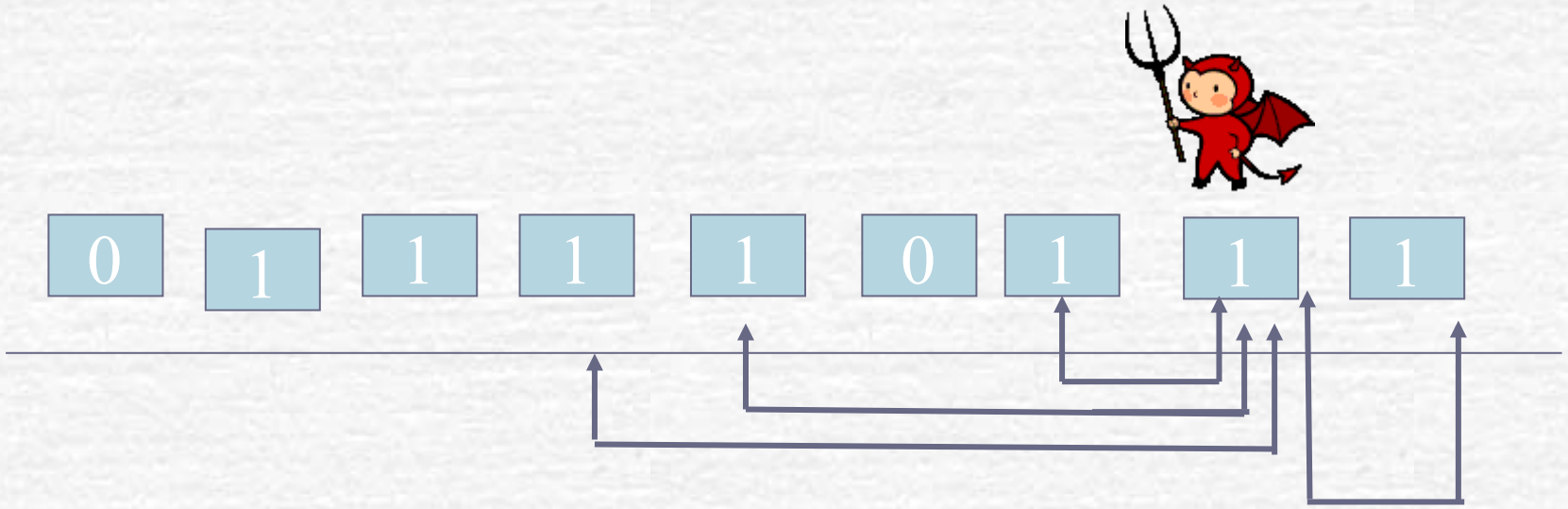
What about privacy?

- How can we ***hide*** the initial values from curious agents?
- How can we compute a result while preventing any agent from figuring out, at any point in time, any information besides its own input and the result of the computation?

What about privacy?

- How can we make it impossible for a curious agent to distinguish the situation where exactly 5 penguins have low temperature from the situation where strictly more do?

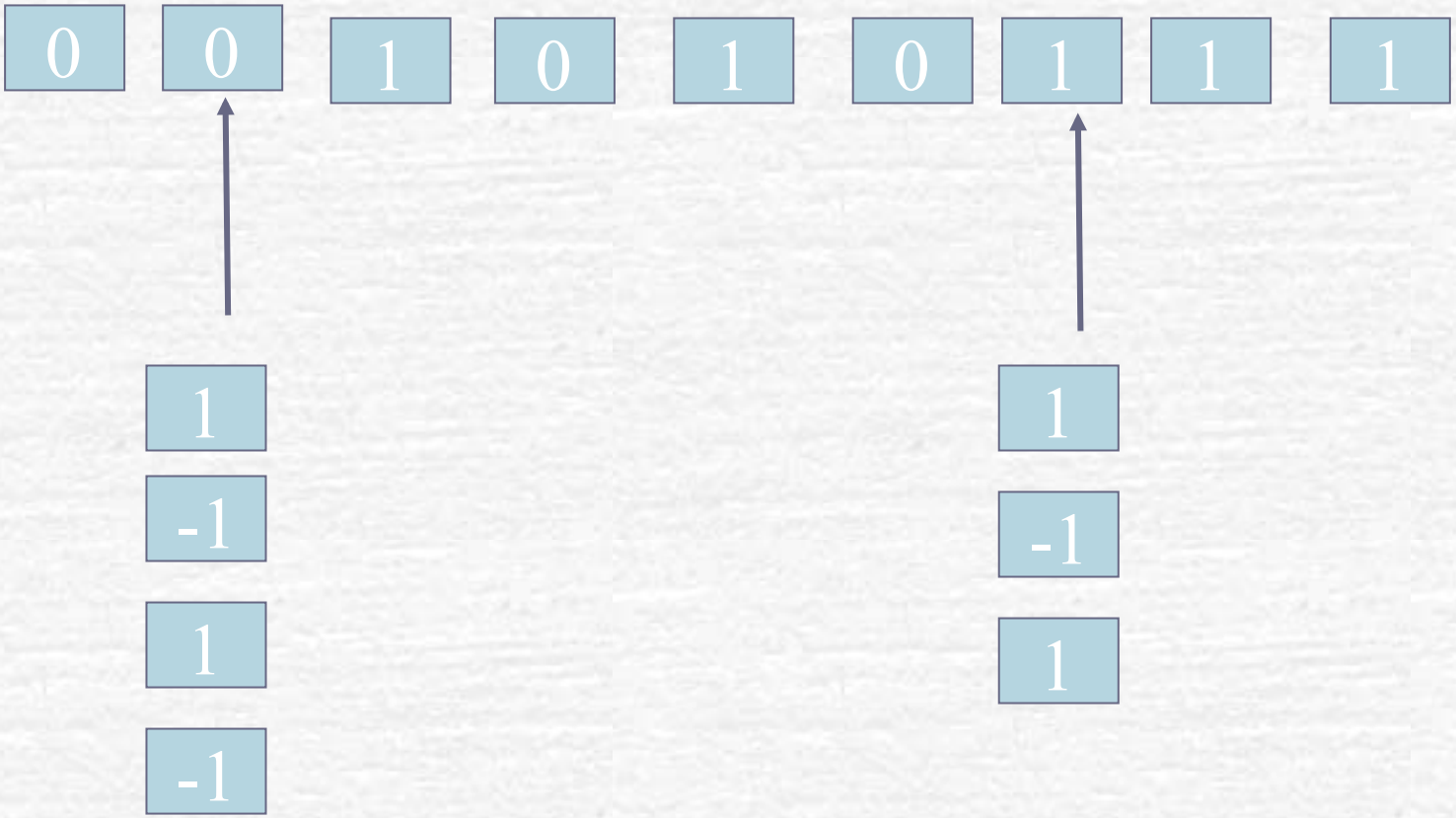
Original algorithm



How to ensure privacy?

- An agent cannot use crypto (not even signatures because of anonymity)
- An agent can see the entire state of the agent it is interacting with: hence no secret keys are possible

Obfuscation



More generally (DFGR07)

- Theorem: population protocols can privately compute exactly first order Presburger's arithmetic

Roadmap

(1) Mobility

(2) Crashes

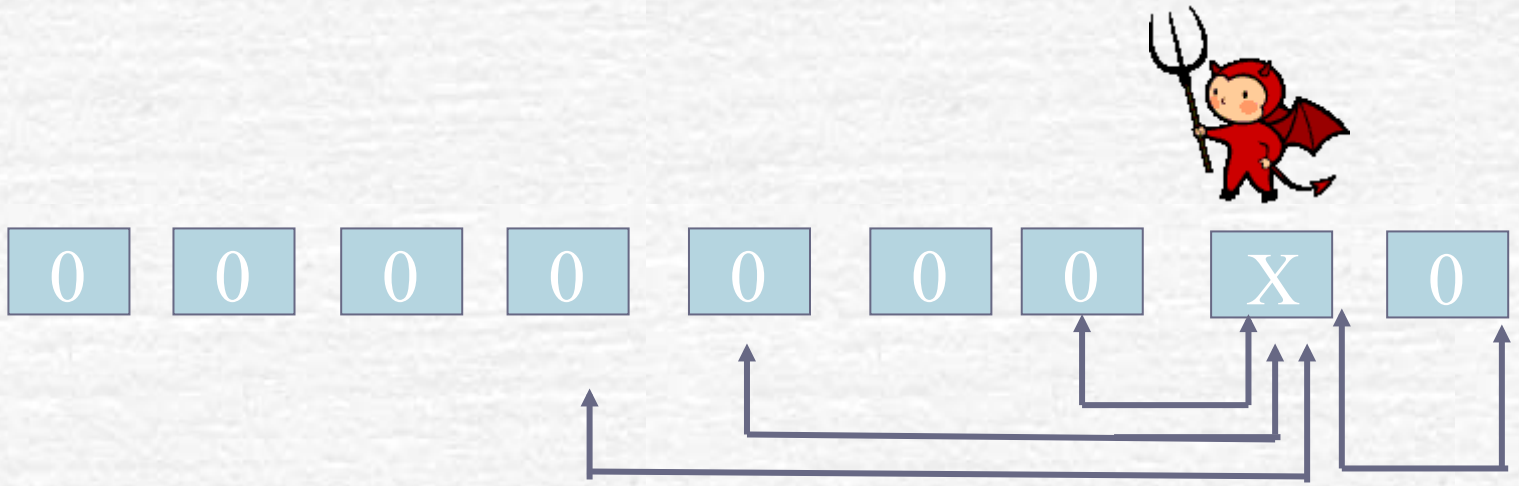
(3) Privacy

(4) Security

What if agents can be malicious?

- What can we compute with one malicious agent? (arbitrary transitions)

Malicious agent



More generally (GR07)

- ***Theorem: population protocols cannot compute any non-trivial predicate with a single malicious agent***

- NB. A predicate is ***trivial*** if every agent can determine its output based only on its input

Limitations of population protocols

- (1) Only ***Presburger's*** arithmetics
- (2) No way to tolerate ***malicious*** agents
- NB. These impossibilities are related to the ***anonymity*** of the model

Identities?

- In practice, few bits are enough to store a very large number of identifiers

Identities?

- Population protocol + one identity per agent
- Identities are useless if agents are amnesic

Identities?

- Population protocol + identities + ***operations***

- Identities can be used to encode any other value: we are back to a distributed system model of ***servers***

Community protocols (GR07)

- Every agent has:
  - A bounded memory where it can execute arithmetics (as in the original population model)

  +

  - A bounded set of slots to store identities and test (only) for their equality

Community protocols (GR07)

- ***Theorem***: ***community protocols*** can exactly compute every ***symmetric*** predicate that can be (Turing) computed in ***n log n*** space

Community protocols (GR07)

- The proof is by reduction to ***Schonhage's pointer machine***: a sequential machine that runs a program using a directed graph structure as its memory

- With a community protocol, we represent each ***node*** by an ***agent*** - The technical difficulty is to have the agents simulate a sequential machine

Community protocols (GR07)

- ***Peter Boas 1989***: a pointer machine with O(n) nodes can simulate a Turing machine with O(n log(n)) space

Community protocols (GR07)

- The Theorem extends to a constant number of ***crash failures*** (using condition-based approach and a technique from DFGR06) as well as ***malicious*** agents
- NB. We assume identifiers cannot be forged (otherwise we are back essentially to population protocols)

The one slide to remember

We can precisely model tiny mobile agents and reason about what they can compute

Complexity?