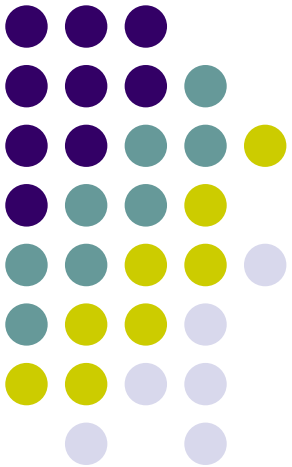
Hydra

Self-stabilizing Population Protocols

Janna Burman



université PARIS-SACLAY

FACULTÉ DES SCIENCES D'ORSAY

LISN LABORATOIRE INTERDISCIPLINAIRE DES SCIENCES DU NUMÉRIQUE

Population Protocols (PP)

[Angluin et al. PODC'04, DC'06]

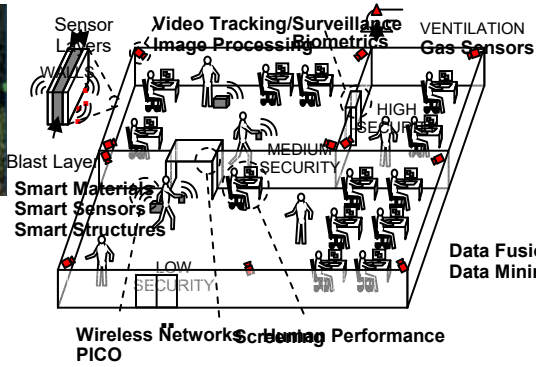
- Collection (*population*) of computational ***agents***
  - **of size *N* (may be unknown)**
  - **indistinguishable**
  - **finite (constant if possible) state memory**
  - **anonymous**
- Interacting
  - in **asynchronous** and **unpredictable** way
  - **in pairs**, while exchanging and updating their states according to a ***transition function***

Transition
(p,q) → (p',q')

- Example of a protocol:
  <u>compute a global property</u> (predicate/function)
  <u>eventually</u> on the input values given to agents
  - E.g., whether 10% of the population has an elevated input value?

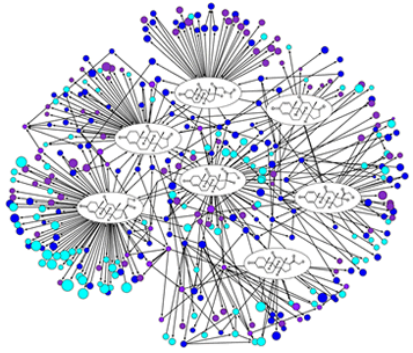
PP Motivating scenarios

- **Passively mobile sensor networks**
  - ZebraNet [ASPLOS'02] (wildlife tracking)
  - EMMA [WCMC'07] (pollution monitoring)



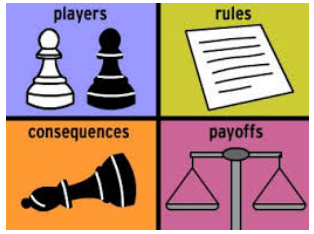
- **Social networks**
  propagation of:
  - trust [Diamadi, Fischer: J.Nat.Sci'01]
  - rumors [Daley, Kendall: J.Inst.Math.Appl'65]
  - epidemics [Bailey:75] [Herbert et al: SIAM'00]

- **Chemical Reaction Networks**
  dynamics of well mixed solutions
  [Gellespie 77], [Soloveichik, Cook, Winfree, Bruck: NC'08], [Doty: SODA'2014]
- **Gene regulatory networks**
  [Bower, Bolouri: MIT press 2004]

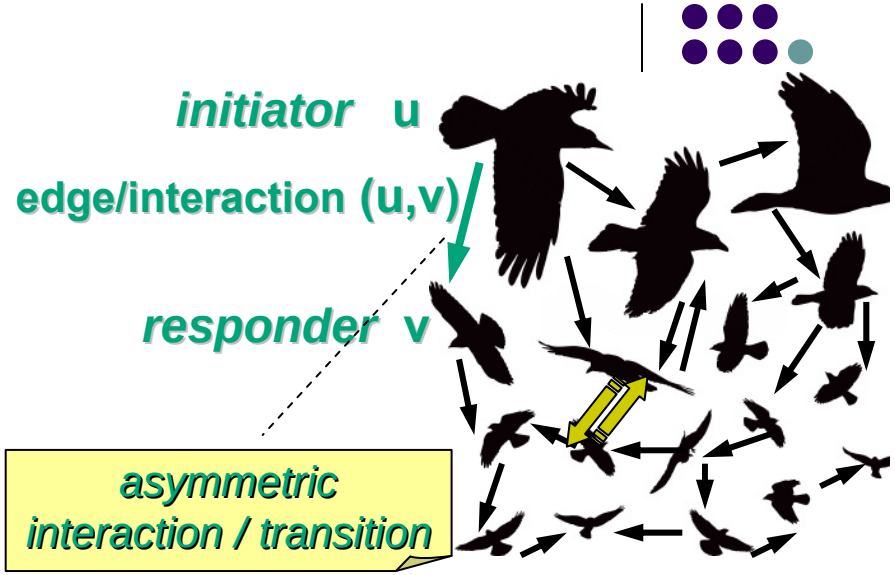


Small fraction of the Organic Chemistry Network (~0.001%). Here, the nodes represent chemical compounds, which are connected by directed arrows representing chemical reactions.

- **Game Theory**
  repetitive games of N-participants
  [Bournez, Chalopin, Cohen, Koegler, Rabie: OPODIS'11]

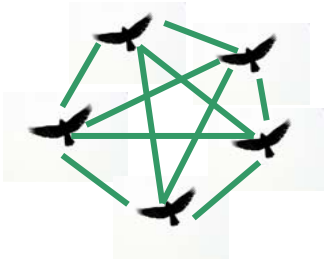


Interaction graph and executions



Interaction Graph

- nodes = agents
- edge (u,v) = possible interaction
- weakly connected
- **Frequently and in this talk - a complete graph**



<u>At each step</u>, a pair of agents to interact is chosen

➔ ***Execution***: infinite sequence of such steps → from one ***configuration*** C_i to another C_{i+1}, updated by transitions of chosen agents' pairs:

$C_0, C_1, C_2, C_3, C_4, \ldots$

Configuration:
a vector of agents' states
or of counts of states

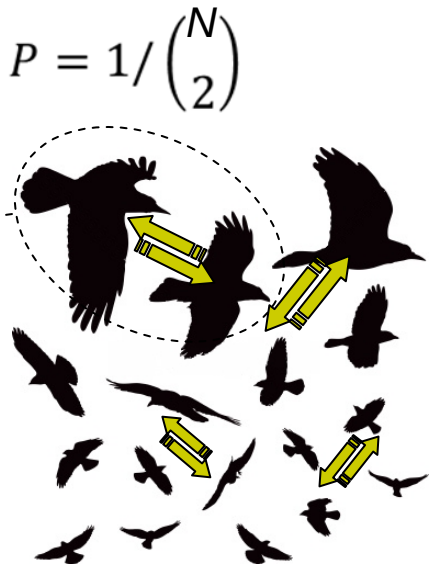
Fair executions

Fairness

- ***Weak***
  every pair of agents interacts infinitely often

- ***Probabilistic (Random Scheduler)***
  at each step, a pair of agents is chosen to interact **uniformly at random**

- ***Global***
  an infinitely often reachable configuration is reached eventually



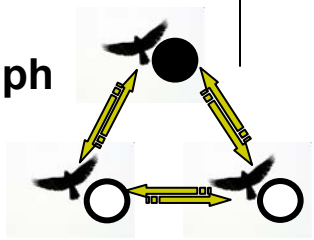
Probabilistic Fairness ➔ Global Fairness w.p.1

Weak vs. Global / Probabilistic Fairness

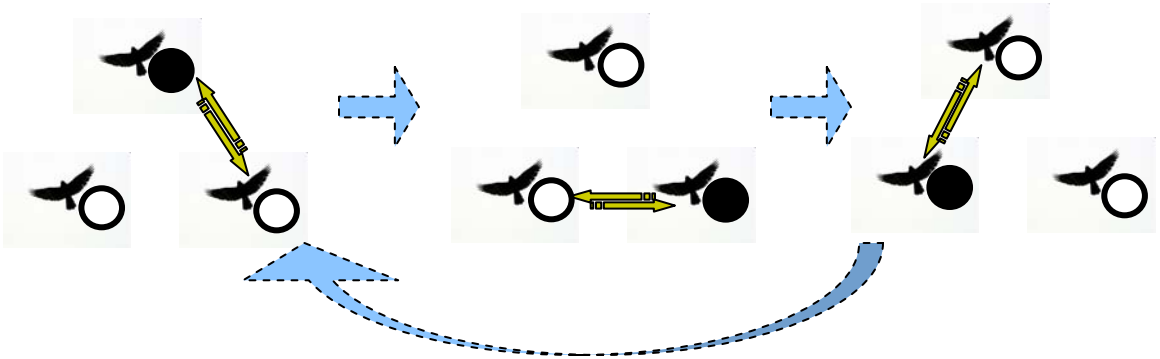
Example:

Protocol (○,○) → (●,●)
(○,●) → (●,○)

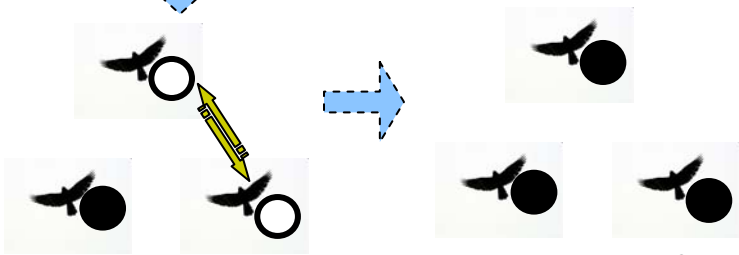
Interaction graph



- *Weak Fairness*
  each pair of agents interact infinitely often



- *Global Fairness*
  infinitely often reachable config. is eventually reached
- *Probabilistic Fairness*
  each pair interacts uniformly at random



Probabilistic Fairness ➔ Global Fairness w.p.1

Main **complexity measures** in PP

Memory Space complexity: in number of different possible **memory states** (or bits) per agent in the protocol

Time complexity

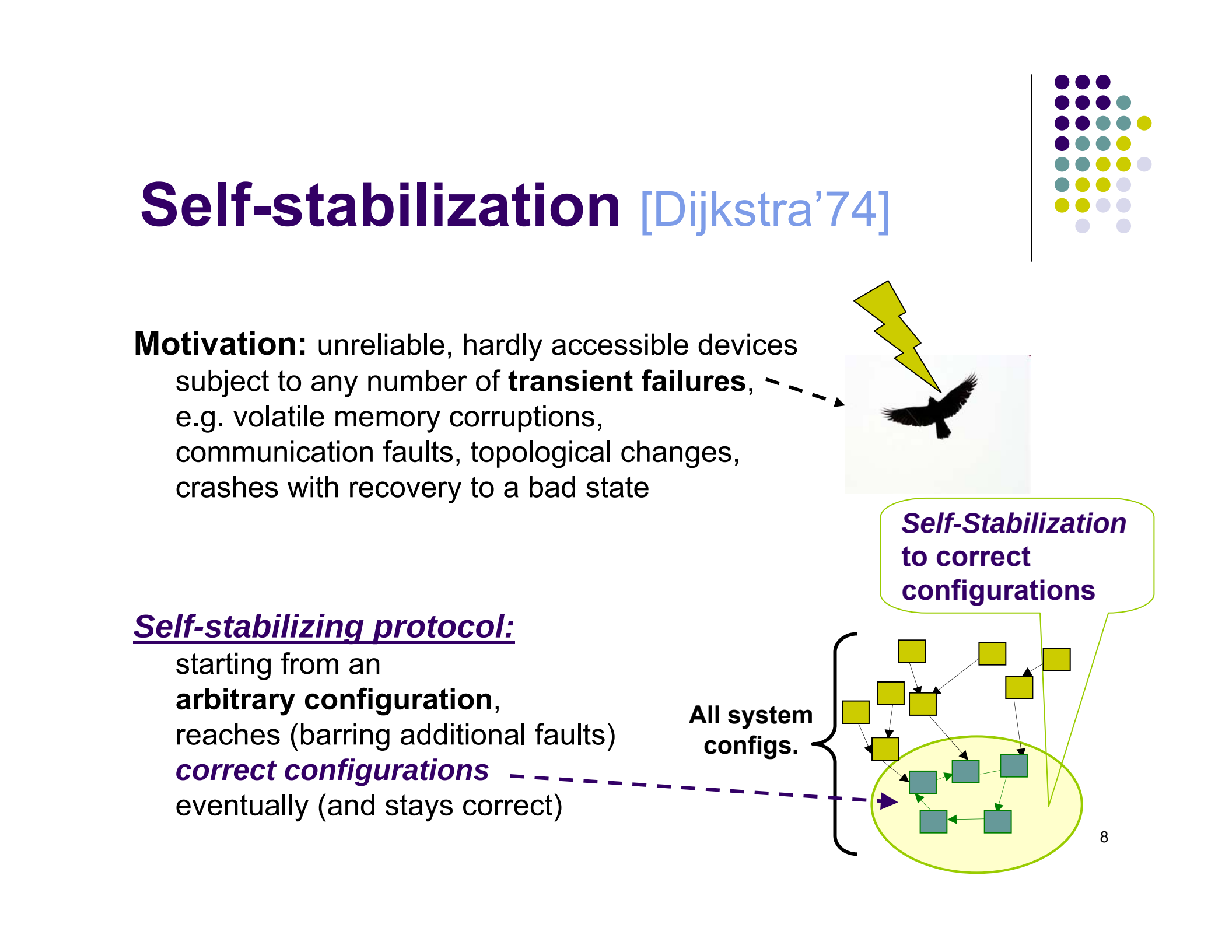
Under random fairness (scheduler):

by ***parallel time*** = the number of interactions (whp or in expectation) until **stabilization** divided by N

- Per parallel time unit, each agent participates in O(1) interactions in average

until stabilization
to the correct output/behavior

Self-stabilization [Dijkstra'74]

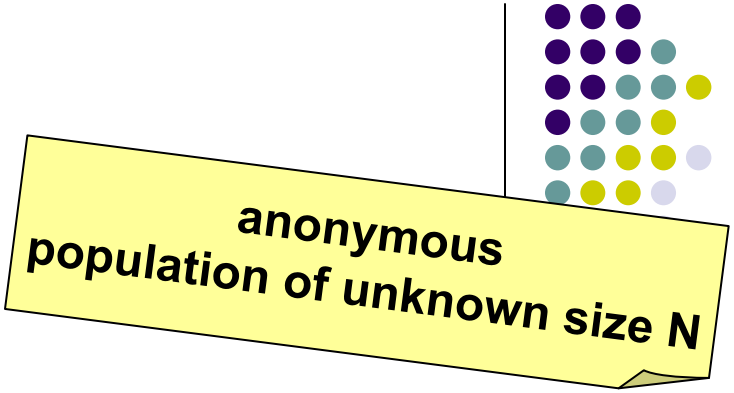


Motivation: unreliable, hardly accessible devices subject to any number of **transient failures**, e.g. volatile memory corruptions, communication faults, topological changes, crashes with recovery to a bad state

Self-stabilizing protocol:

starting from an **arbitrary configuration**, reaches (barring additional faults) ***correct configurations*** eventually (and stays correct)

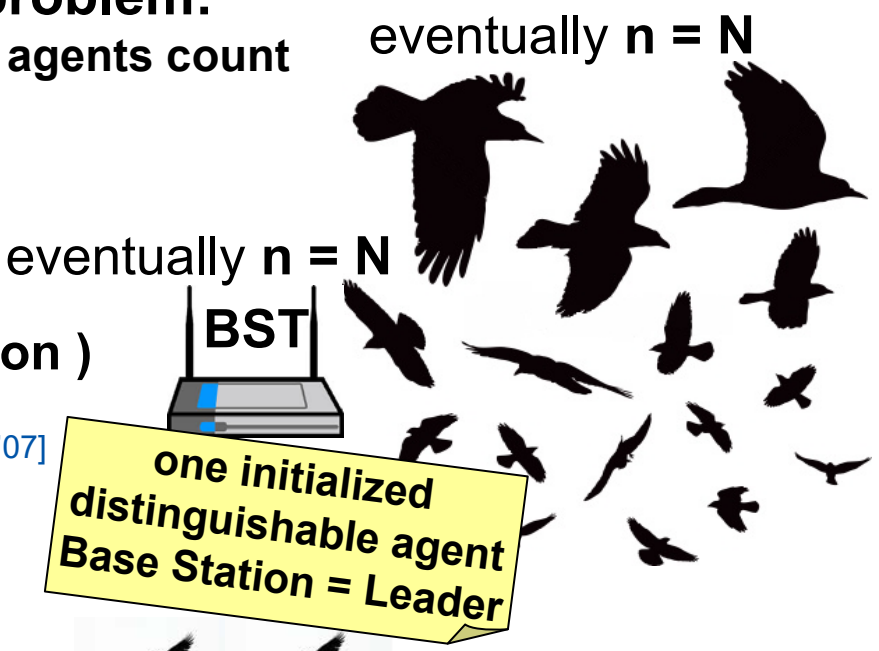
The Counting Problem



Exact Population Size Counting problem:

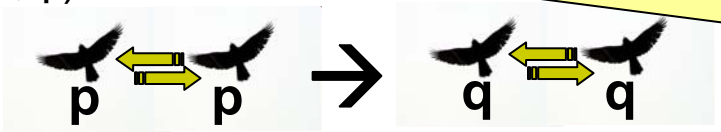
eventually (at the *stabilization*), some agents count all the N agents (in a variable n)*

*** some agents won't have a uniformly finite state**



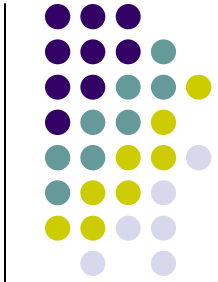
Additional Goals:

- ✓ **no initialization (→ self-stabilization)**
  - ✓ **Impossible**
    [Beauquier, Clément, Messika, Rosaz, Rozoy: DISC'07]
- ✓ ***symmetric* deterministic PP:**
  - ✓ in transition (p,q)→(p',q'):
    if p = q then p' = q'



- ✓ **Optimize (exact) agent state space**
  - ✓ impossible with 1 state → **2 states with global fairness**
    [Beauquier, Burman, Clavier, Sohier: DISC'15]
  - ➢ **Optimize time!**
    [Aspnes, Beauquier, Burman, Sohier: OPODIS'16]

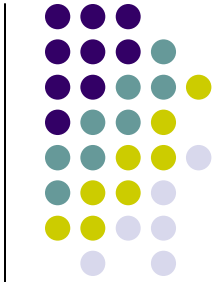
Counting: Motivation



- **Fundamental** and extensively studied task in computer science
- **Useful in many domains**: network traffic monitoring, database query optimization, data mining, …
- **In PP, counting is used to solve other tasks**:
  - uniform bipartition or k-partition
    - [Yasumi, Ooshita, Yamaguchi, Inoue: IEICE Trans.'19]
    - [Yasumi, Kitamura, Ooshita, Izumi, Inoue: Int. J. Netw. Comput.'19]
  - naming
    - e.g. [J. Burman, J. Beauquier, D. Sohier: DISC'19]
  - degree recognition
    - [Sudo, Shibata, Nakamura, Kim, Masuzawa: IEEE TPDS'21]

  …

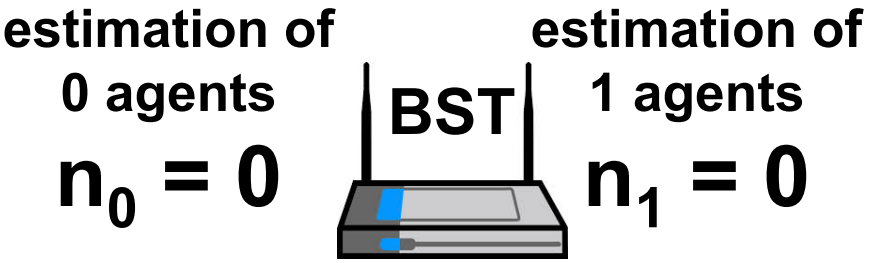
Space-Optimal Counting
Global fairness, 2 states
(Algorithm)



agents are initialized arbitrarily, each is in state b $\in$ {0,1}

Initialization of BST:

n $\leftarrow$ n_0 $\leftarrow$ n_1 $\leftarrow$ 0



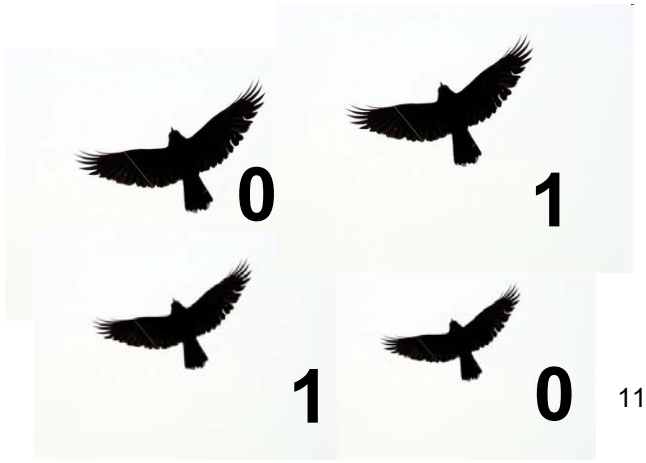
If b-agent interacts with BST:

If (n_b **>** 0) **then** n_b **--**

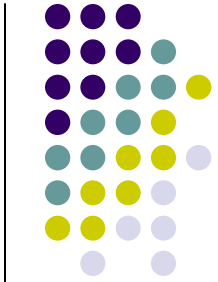
b $\leftarrow$ 1 **-** b

n_b **++**

n $\leftarrow$ n_1 + n_0



Space-Optimal Counting
Global fairness, 2 states
(Algorithm)



agents are initialized arbitrarily, each is in state b $\in$ {0,1}

Initialization of BST:

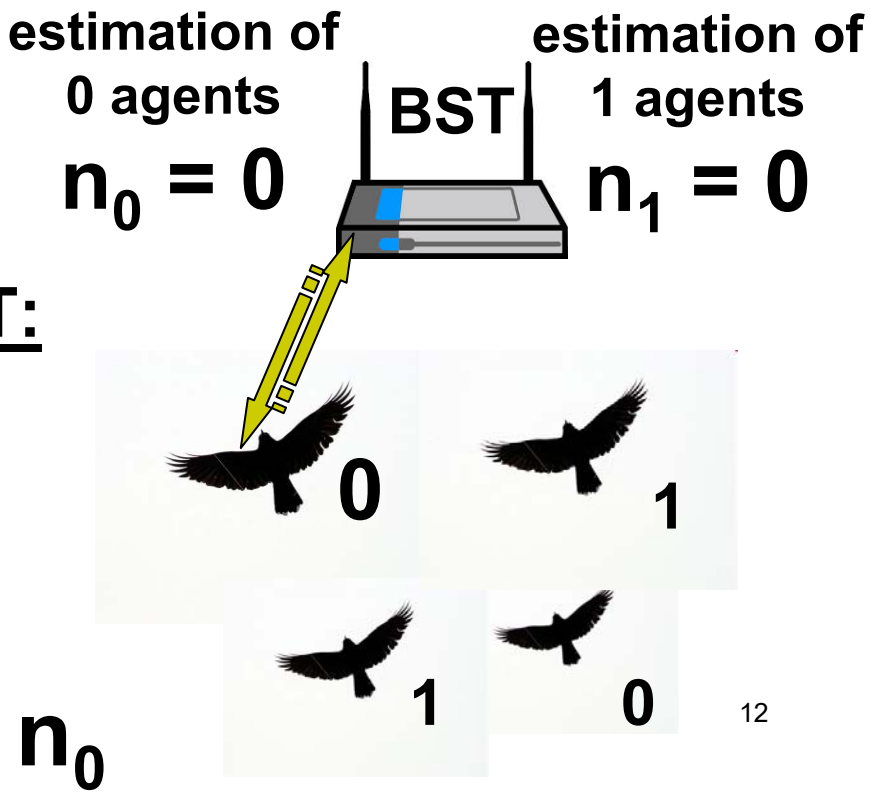
$n \leftarrow n_0 \leftarrow n_1 \leftarrow 0$

If b-agent interacts with BST:

If $(n_b > 0)$ **then** n_b **--**

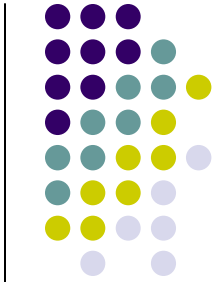
$b \leftarrow 1 - b$

n_b **++**



$$n \leftarrow n_1 + n_0$$

Space-Optimal Counting
Global fairness, 2 states
(Algorithm)



agents are initialized arbitrarily, each is in state b $\in$ {0,1}

Initialization of BST:

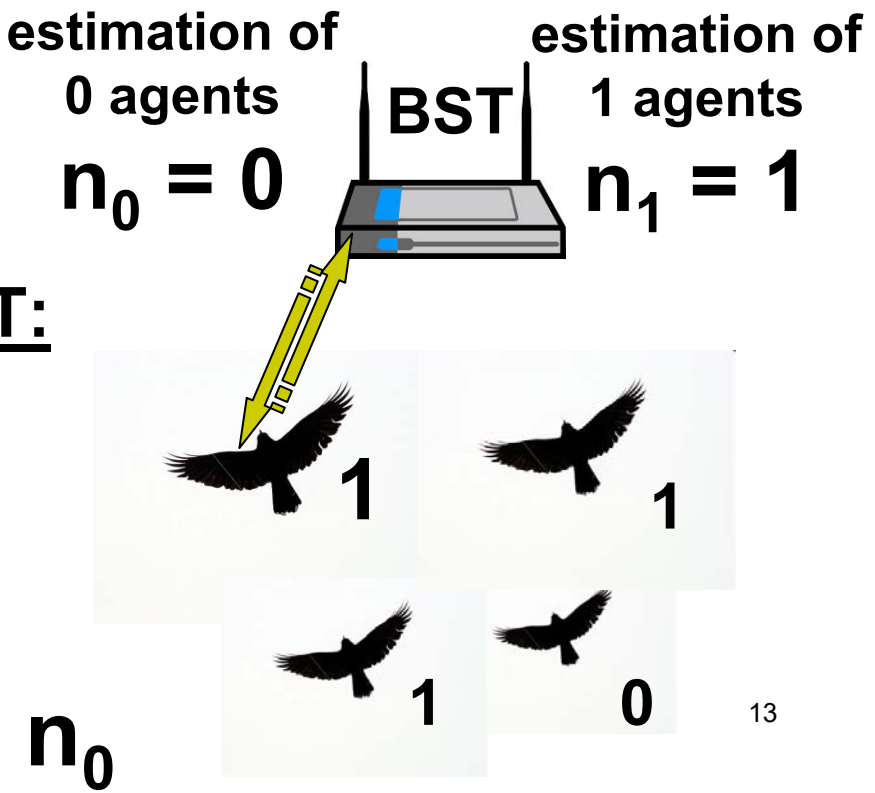
$n \leftarrow n_0 \leftarrow n_1 \leftarrow 0$

If b-agent interacts with BST:

If $(n_b > 0)$ **then** n_b **--**

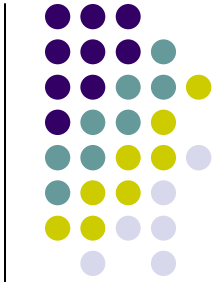
$b \leftarrow 1 - b$

n_b **++**



$$n \leftarrow n_1 + n_0$$

Space-Optimal Counting
Global fairness, 2 states
(Algorithm)



agents are initialized arbitrarily, each is in state b $\in$ {0,1}

Initialization of BST:

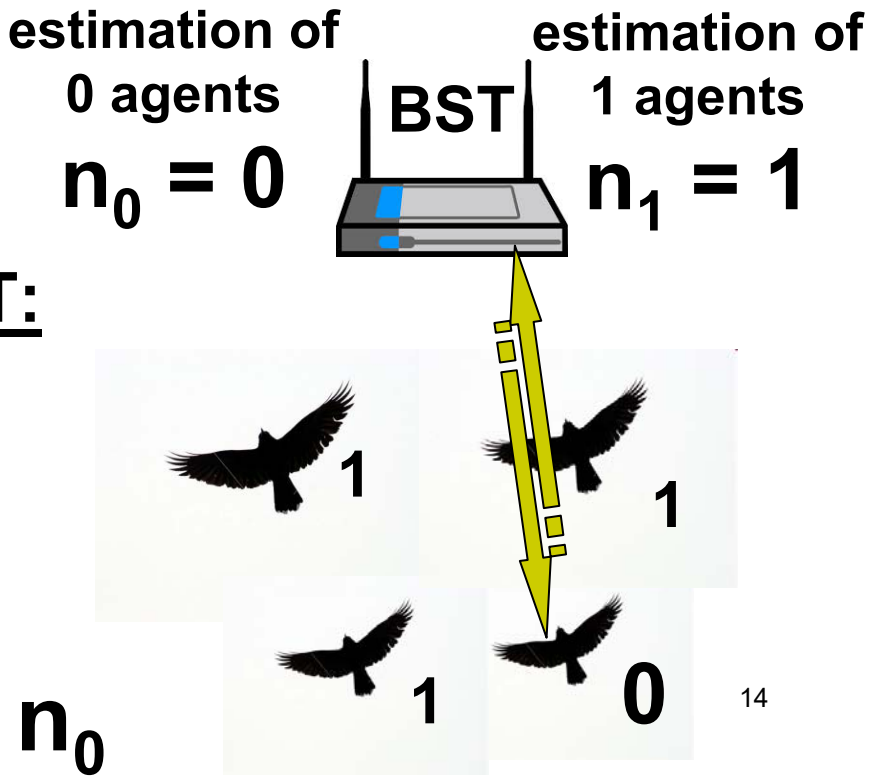
$n \leftarrow n_0 \leftarrow n_1 \leftarrow 0$

If b-agent interacts with BST:

If $(n_b > 0)$ **then** n_b **--**

$b \leftarrow 1 - b$

n_b **++**



$$n \leftarrow n_1 + n_0$$

Space-Optimal Counting
Global fairness, 2 states
(Algorithm)

agents are initialized arbitrarily, each is in state b $\in$ {0,1}

Initialization of BST:

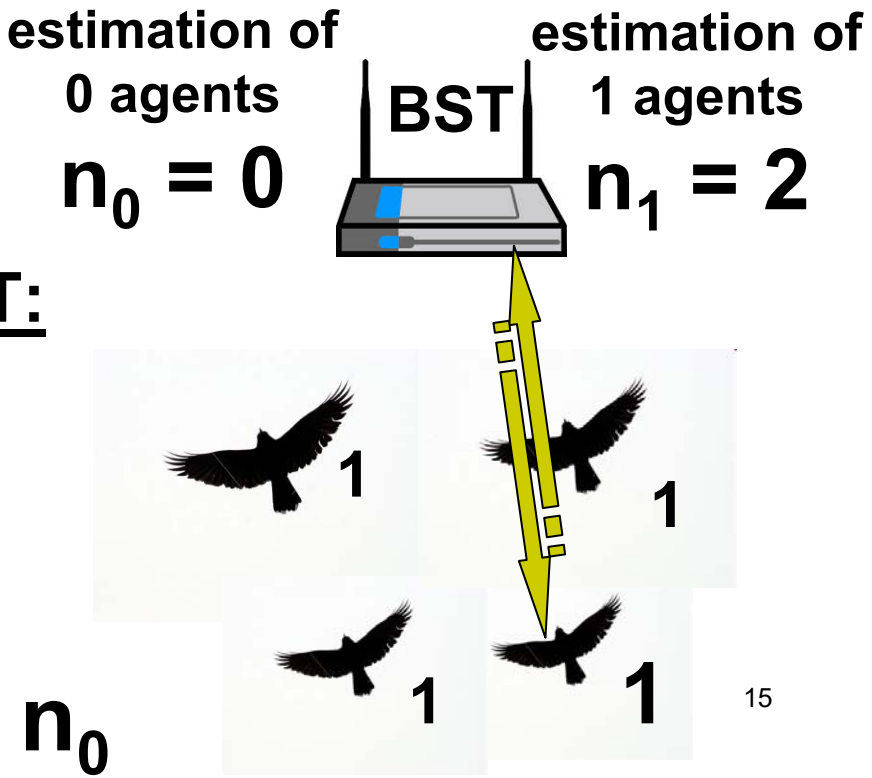
$n \leftarrow n_0 \leftarrow n_1 \leftarrow 0$

If b-agent interacts with BST:

If $(n_b > 0)$ **then** n_b **--**

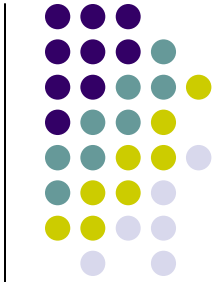
$b \leftarrow 1 - b$

n_b **++**



$$n \leftarrow n_1 + n_0$$

Space-Optimal Counting
Global fairness, 2 states
(Algorithm)



agents are initialized arbitrarily, each is in state b $\in$ {0,1}

Initialization of BST:

$n \leftarrow n_0 \leftarrow n_1 \leftarrow 0$

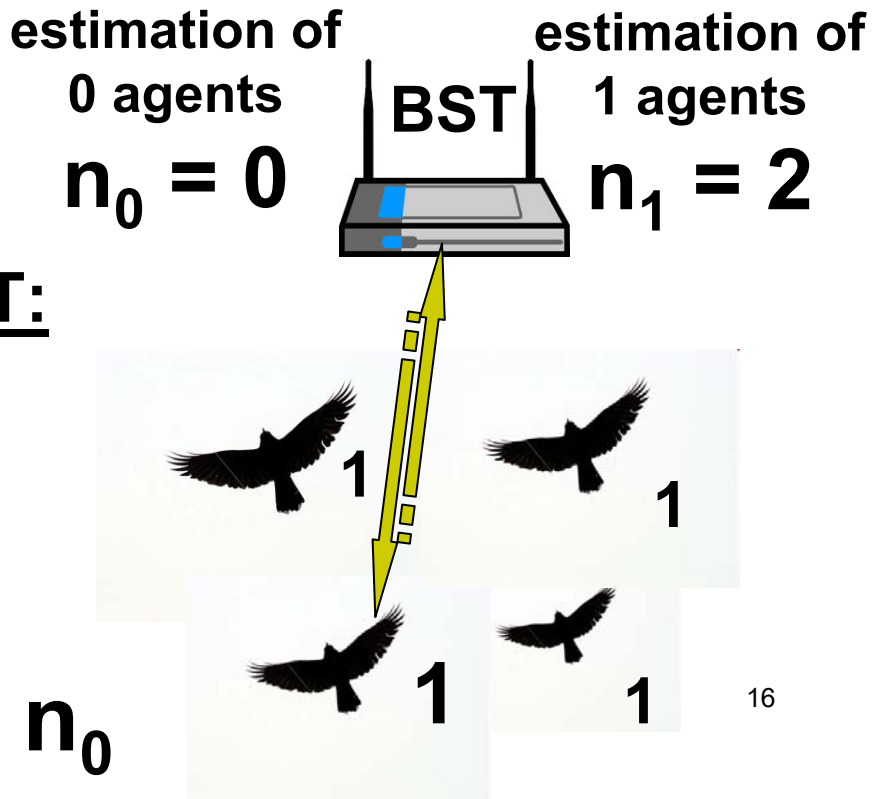
If b-agent interacts with BST:

If $(n_b > 0)$ **then** n_b **--**

$b \leftarrow 1 - b$

n_b **++**

$$n \leftarrow n_1 + n_0$$



Space-Optimal Counting
Global fairness, 2 states
(Algorithm)

agents are initialized arbitrarily, each is in state b $\in$ {0,1}

Initialization of BST:

$n \leftarrow n_0 \leftarrow n_1 \leftarrow 0$

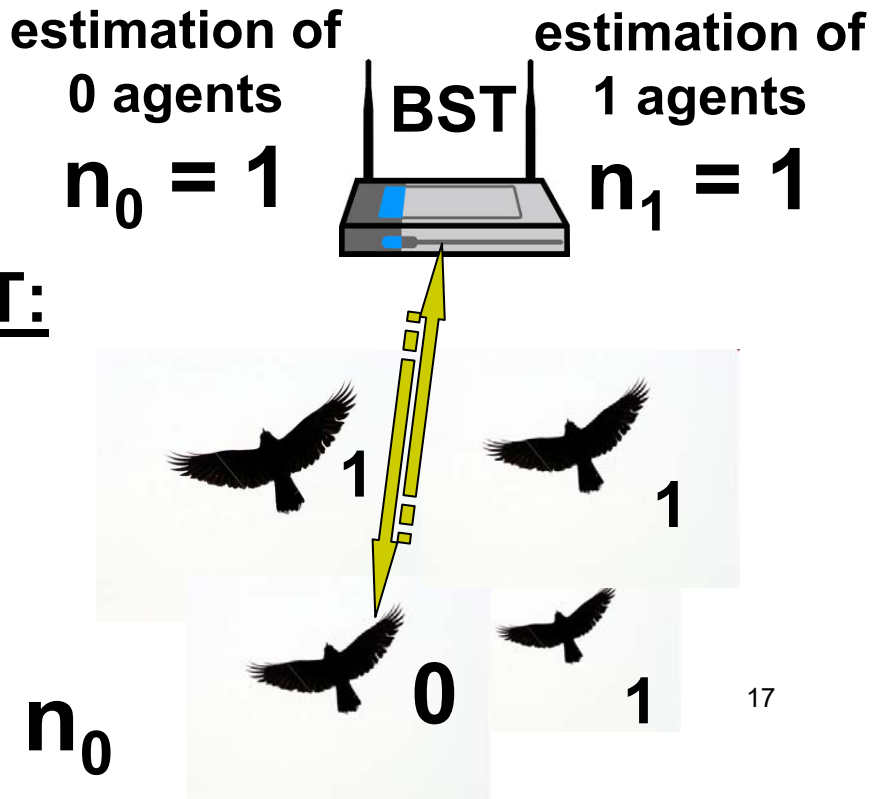
If b-agent interacts with BST:

If $(n_b > 0)$ **then** n_b **--**

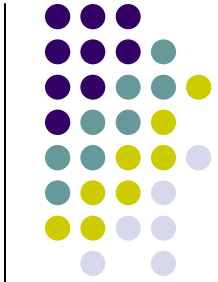
$b \leftarrow 1 - b$

n_b **++**

$$n \leftarrow n_1 + n_0$$



Space-Optimal Counting
Global fairness, 2 states
(Algorithm)



agents are initialized arbitrarily, each is in state b $\in$ {0,1}

Initialization of BST:

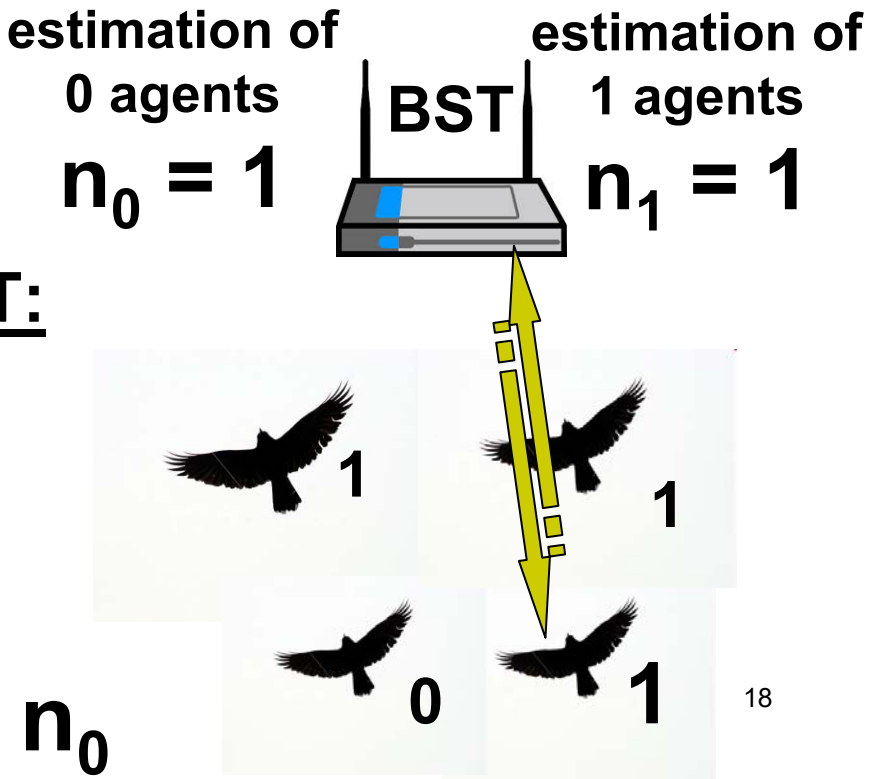
$n \leftarrow n_0 \leftarrow n_1 \leftarrow 0$

If b-agent interacts with BST:

If $(n_b > 0)$ **then** n_b **--**

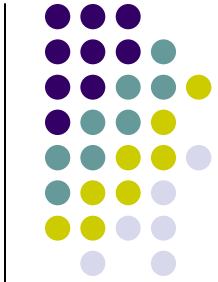
$b \leftarrow 1 - b$

n_b **++**



$$n \leftarrow n_1 + n_0$$

Space-Optimal Counting
Global fairness, 2 states
(Algorithm)



agents are initialized arbitrarily, each is in state b $\in$ {0,1}

Initialization of BST:

$n \leftarrow n_0 \leftarrow n_1 \leftarrow 0$

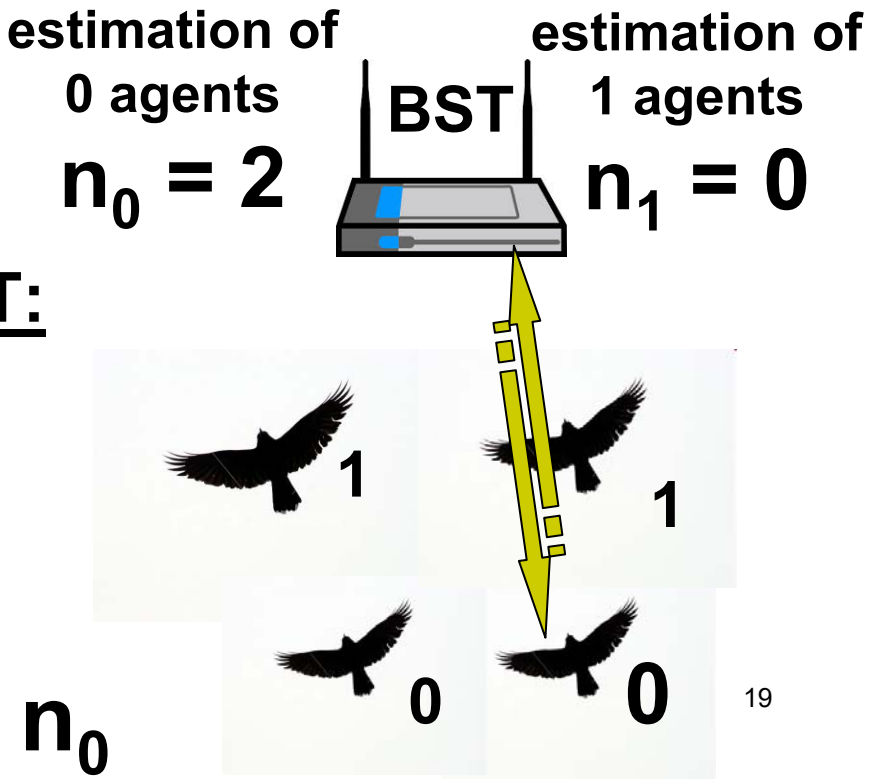
If b-agent interacts with BST:

If $(n_b > 0)$ **then** n_b **--**

$b \leftarrow 1 - b$

n_b **++**

$$n \leftarrow n_1 + n_0$$



Space-Optimal Counting
Global fairness, 2 states
(Algorithm)

agents are initialized arbitrarily, each is in state b $\in$ {0,1}

Initialization of BST:

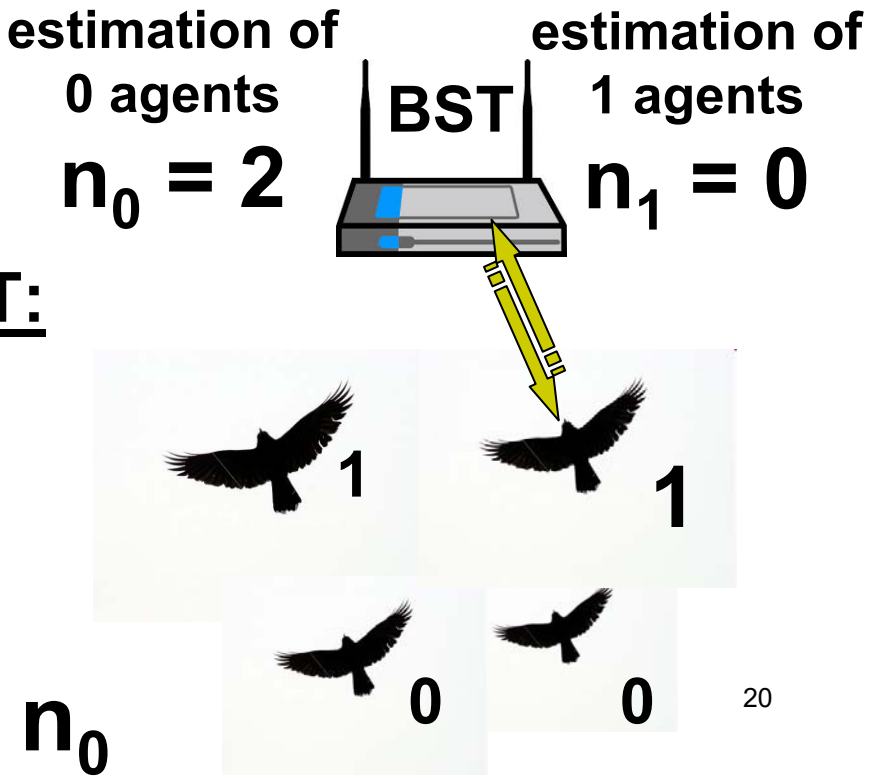
$n \leftarrow n_0 \leftarrow n_1 \leftarrow 0$

If b-agent interacts with BST:

If $(n_b > 0)$ **then** n_b **--**

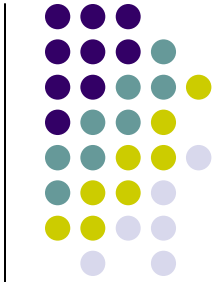
$b \leftarrow 1 - b$

n_b **++**



$$n \leftarrow n_1 + n_0$$

Space-Optimal Counting
Global fairness, 2 states
(Algorithm)



agents are initialized arbitrarily, each is in state b $\in$ {0,1}

Initialization of BST:

$n \leftarrow n_0 \leftarrow n_1 \leftarrow 0$

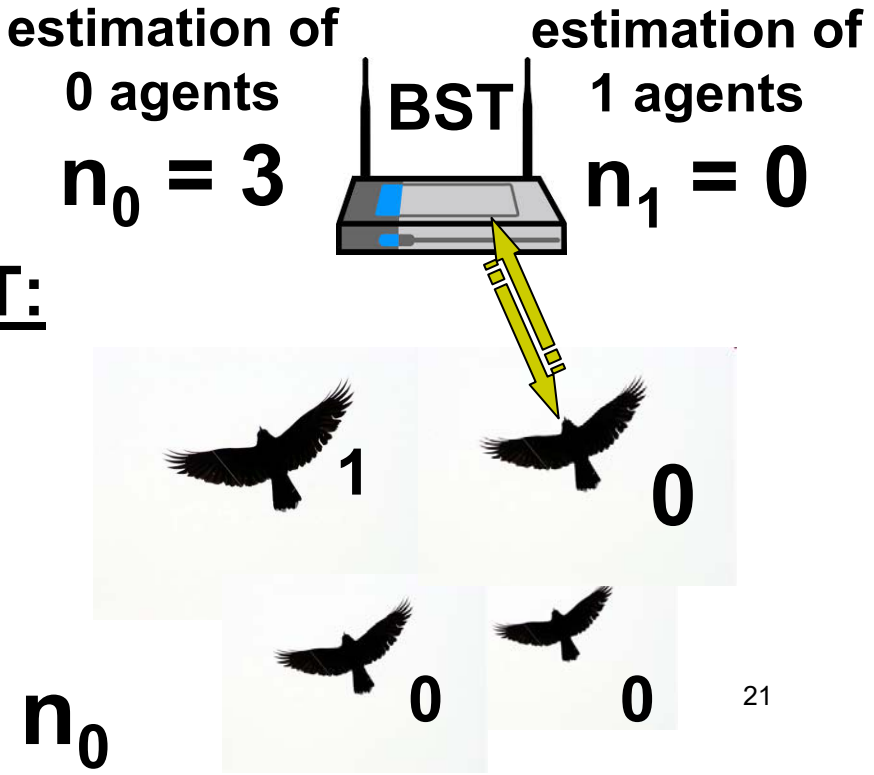
If b-agent interacts with BST:

If $(n_b > 0)$ **then** n_b **--**

$b \leftarrow 1 - b$

n_b **++**

$$n \leftarrow n_1 + n_0$$



Space-Optimal Counting
Global fairness, 2 states
(Algorithm)

agents are initialized arbitrarily, each is in state b $\in$ {0,1}

Initialization of BST:

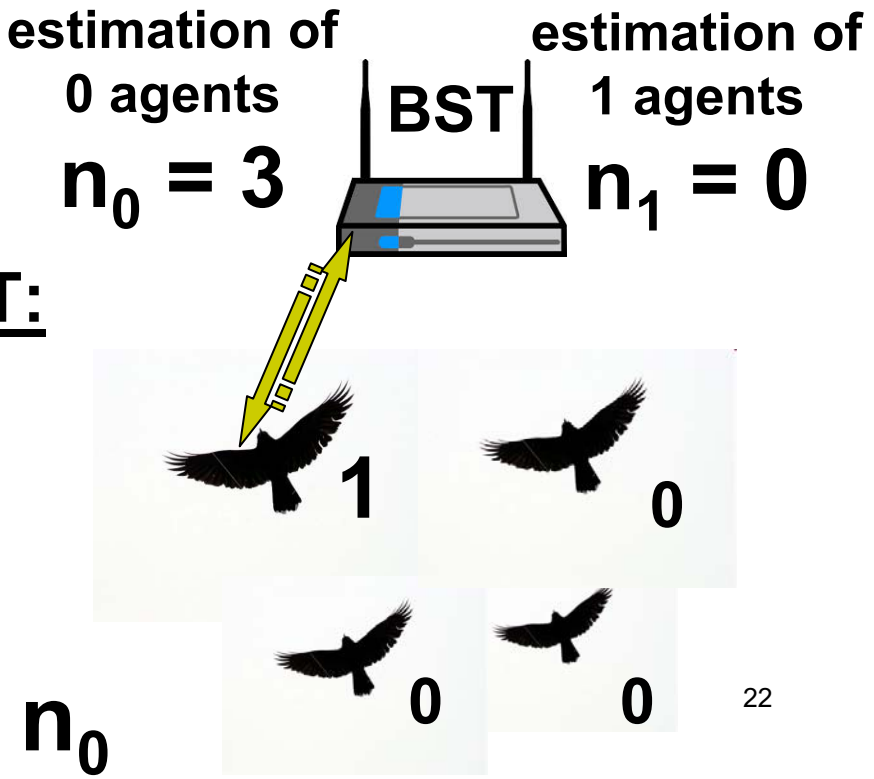
$n \leftarrow n_0 \leftarrow n_1 \leftarrow 0$

If b-agent interacts with BST:

If $(n_b > 0)$ **then** n_b **--**

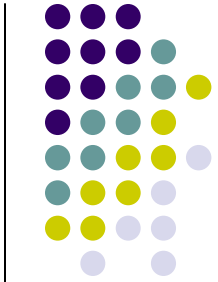
$b \leftarrow 1 - b$

n_b **++**



$$n \leftarrow n_1 + n_0$$

Space-Optimal Counting
Global fairness, 2 states
(Algorithm)



agents are initialized arbitrarily, each is in state b $\in$ {0,1}

Initialization of BST:

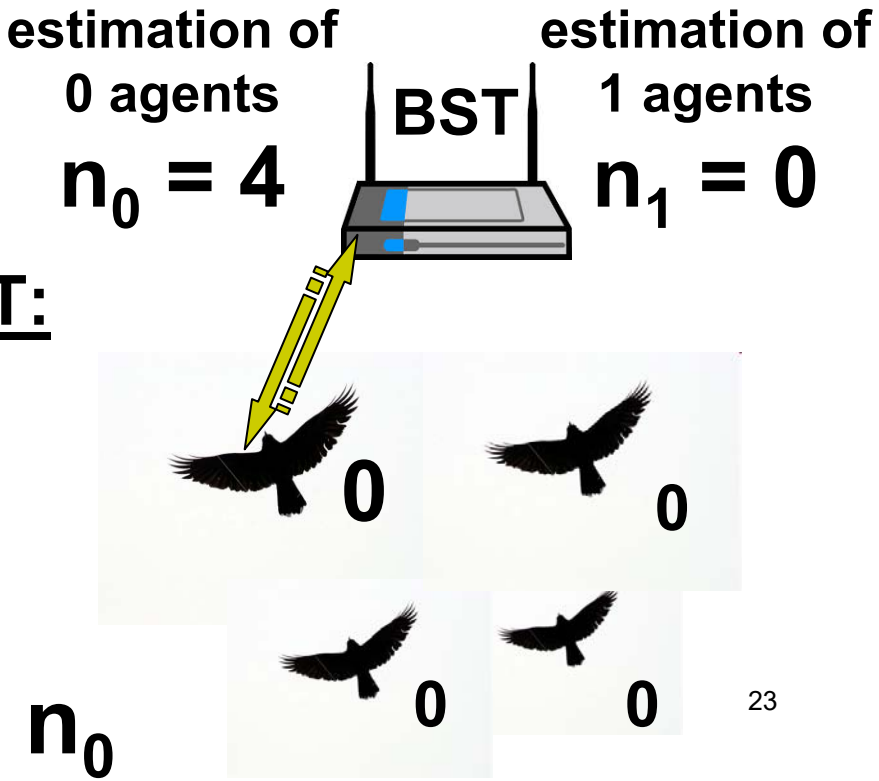
$n \leftarrow n_0 \leftarrow n_1 \leftarrow 0$

If b-agent interacts with BST:

If $(n_b > 0)$ **then** n_b **--**

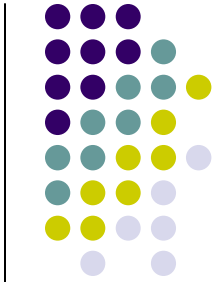
$b \leftarrow 1 - b$

n_b **++**



$$n \leftarrow n_1 + n_0$$

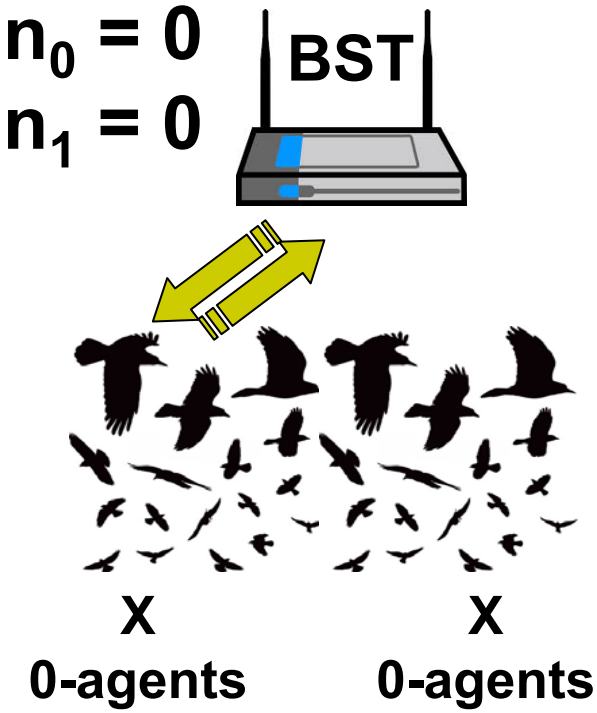
Space-Optimal Counting
Global fairness, 2 states (Proof)



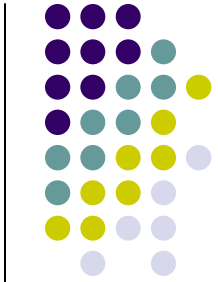
Lemma 1: $n_0 \leq N_0$, $n_1 \leq N_1$, $n = n_0 + n_1 \leq N$;
n can not decrease

Execution cycles are possible:

Make interact X 0-agents, each, once.
They become 1-agents, and $n_1 = X$.



Space-Optimal Counting
Global fairness, 2 states (Proof)

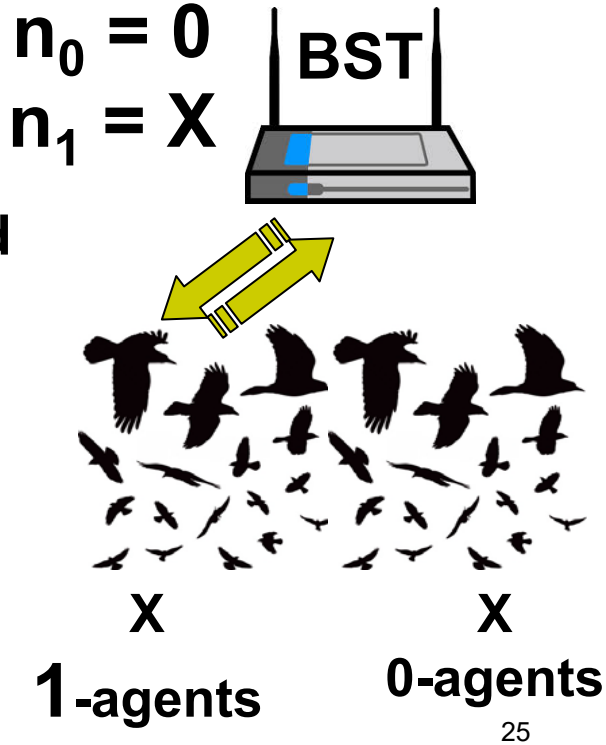


Lemma 1: $n_0 \leq N_0$, $n_1 \leq N_1$, $n = n_0 + n_1 \leq N$;
n_0, n_1, n can not decrease

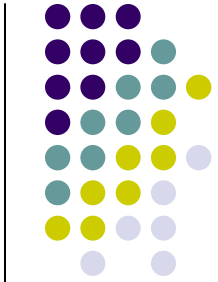
Execution cycles are possible:

Make interact X 0-agents, each, once.
They become 1-agents, and $n_1 = X$.

Repeat. They become 0-agents again and n_0=X.



Space-Optimal Counting
Global fairness, 2 states (Proof)

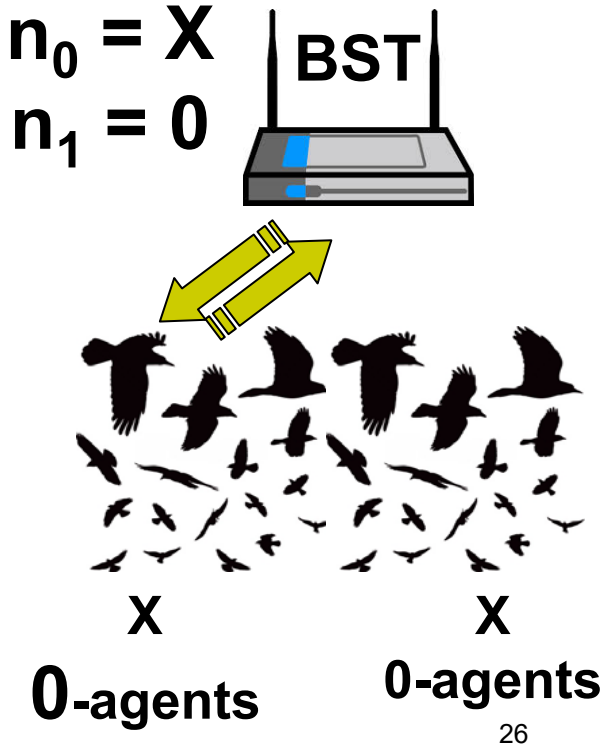


<u>Lemma 1:</u> $n_0 \leq N_0$, $n_1 \leq N_1$, $n = n_0 + n_1 \leq N$;
n_0, n_1, n can not decrease

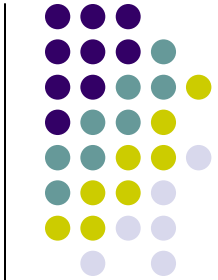
Execution cycles are possible:

Make interact X 0-agents, each, once. They become 1-agents, and $n_1 = X$.

Repeat. They become 0-agents again and n_0=X.



Space-Optimal Counting
Global fairness, 2 states (Proof)



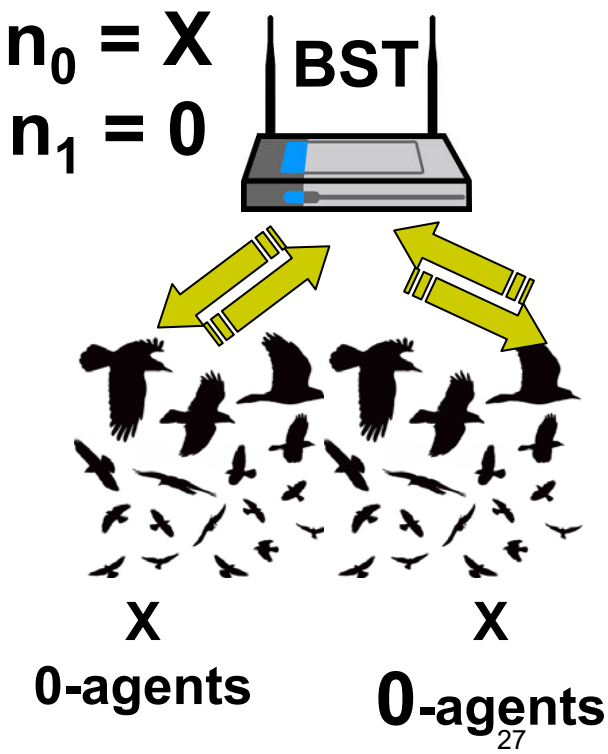
Lemma 1: $n_0 \leq N_0$, $n_1 \leq N_1$, $n = n_0 + n_1 \leq N$;
n_0, n_1, n can not decrease

Execution cycles are possible:

Make interact X 0-agents, each, once.
They become 1-agents, and $n_1 = X$.

Repeat. They become 0-agents again and $n_0 = X$.

Repeat the same with the other X agents.
At the end, n_0 and n_1 are as before.



Space-Optimal Counting
Global fairness, 2 states (Proof)

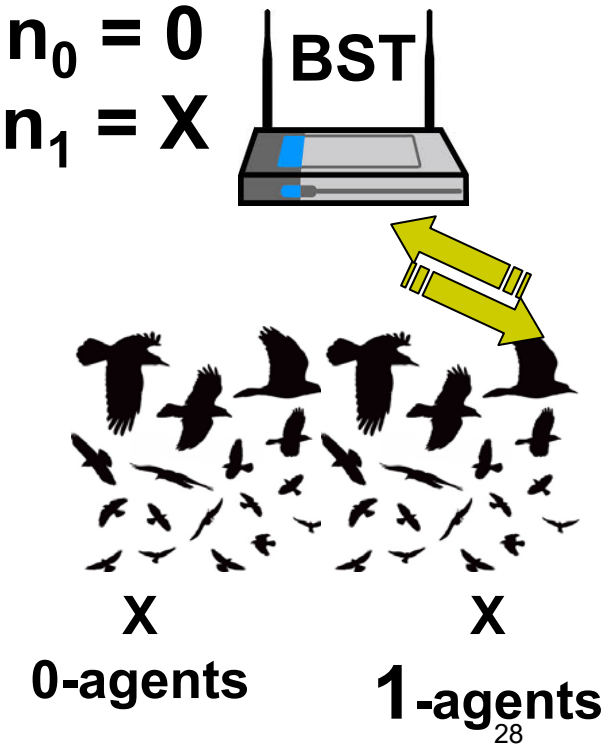
Lemma 1: $n_0 \leq N_0$, $n_1 \leq N_1$, $n = n_0 + n_1 \leq N$;
n_0, n_1, n can not decrease

Execution cycles are possible:

Make interact X 0-agents, each, once.
They become 1-agents, and $n_1 = X$.

Repeat. They become 0-agents again and $n_0 = X$.

Repeat the same with the other X agents.
At the end, n_0 and n_1 are as before.

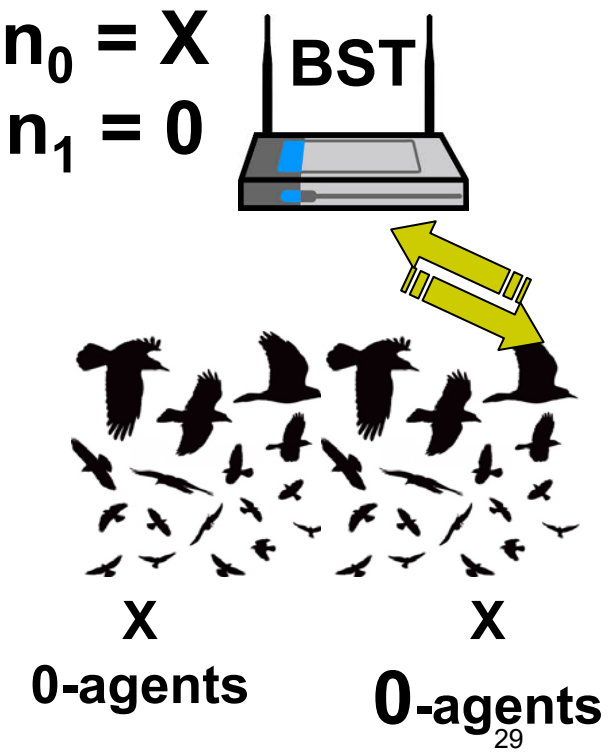


Space-Optimal Counting
Global fairness, 2 states (Proof)

Lemma 1: $n_0 \leq N_0$, $n_1 \leq N_1$, $n = n_0 + n_1 \leq N$;
n_0, n_1, n can not decrease

Execution cycles are possible:

Make interact X 0-agents, each, once.
They become 1-agents, and $n_1 = X$.
Repeat. They become 0-agents again and $n_0 = X$.
Repeat the same with the other X agents.
At the end, n_0 and n_1 are as before.



Space-Optimal Counting
Global fairness, 2 states (Proof)

Lemma 1: $n_0 \le N_0$, $n_1 \le N_1$, $n = n_0 + n_1 \le N$;
n can not decrease

Execution cycles are possible:

Make interact X 0-agents, each, once.
They become 1-agents, and $n_1 = X$.

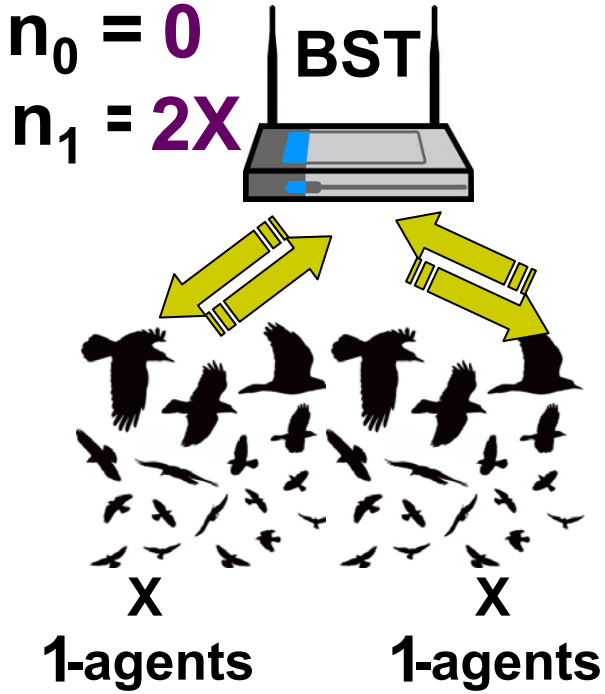
Repeat. They become 0-agents again and $n_0 = X$.

Repeat the same with the other X agents.
At the end, n_0 and n_1 are as before.

Repeat all again.
$n \leftarrow n_0 + n_1 = X$

But, when all agents are in 0-state, make them all interact, one by one, with BST (in a sequence):
$n \leftarrow 2X$

Lemma 2: **From any configuration,**
$\exists$ reachable configuration C with n = N.
By *global fairness*, C is reached.



Time and Space Optimal Counting

1. Time lower bound:
space-optimal (2 states) counting stabilizes in
Ω(N log N) expected parallel time

2. This is a tight bound:
there is space-optimal (2 states) counting that stabilizes in **O(N log N)** expected parallel time

[Aspnes, Beauquier, Burman, Sohier: OPODIS'16]

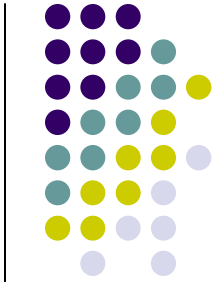
Leader Election (LE) Problem

- Agents have **leader states L** and others called **followers - F**
- Eventually (with probability 1), **only one agent** is in **a leader state**, forever

➔ **Then, the execution is said to have stabilized**

No safety property is asked

Leader Election: Motivation



- **Fundamental** and extensively studied distributed computing primitive
- **Useful for coordination and efficiency**
- For example **in PP, given a leader**:
  - **protocols can become exponentially faster**
    - e.g. [D. Angluin, J. Aspnes, D. Eisenstat: DC'08], [A. Bellevile, D. Doty, D. Soloveichik: ICALP'17]
  - compact – **state space efficient**
    - e.g. [M. Blondin, J. Esparza, S. Jaax: STACS'18]
  - **some problems become feasible**
    - e.g. [J. Burman, J. Beauquier, D. Sohier: DISC'19]

Uniformly initialized LE

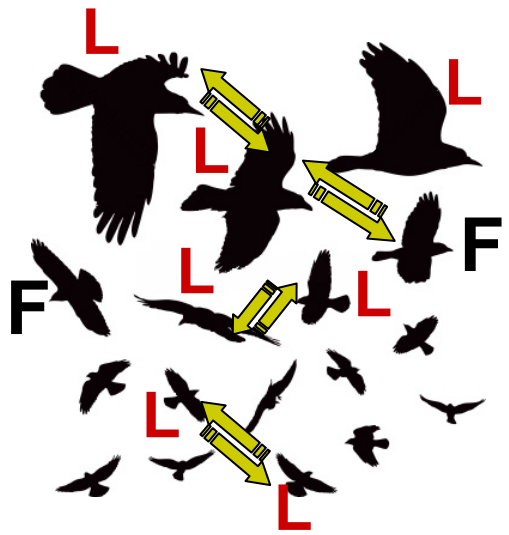
- Simple solution:
  1. **all agents start in state L**
  2. Transition func.: **(L, L) → (L, F)**

Uniformly initialized LE

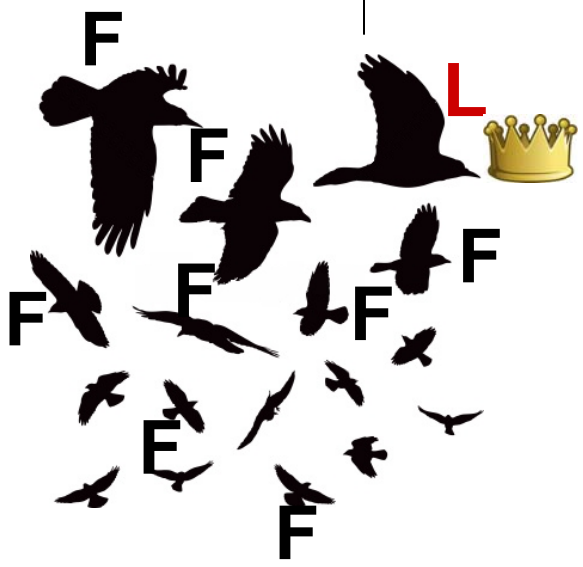
- Simple solution:
  1. all agents start in state L
  2. Transition func.: **(L, L) → (L, F)**

Uniformly initialized LE

- Simple solution:
  1. all agents start in state L
  2. Transition func.: **(L, L) → (L, F)**



Uniformly initialized LE



- **Simple solution:**
  1. all agents start in state L
  2. Transition func.: (L, L) → (L, F)
  3. **stabilize** to a unique leader in L, **in O(N) time** (in expectation)

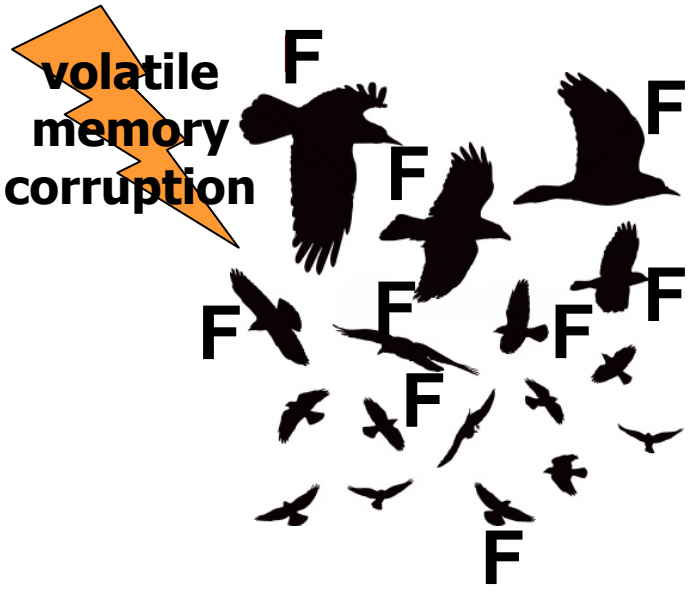
- **With constant state space (and $\leq \frac{1}{2}$ log log N), provably takes Ω(N) time**
  - [D. Doty, D. Soloveichik: DISC'15], [D. Alistarh, J. Aspnes, D. Eisenstat, R. Gelashvili: R.L. Rivest: SODA'17]
- **Solvable with O(log log N) states in optimal O(log N) time**
  - [P. Berenbrink, G. Giakkoupis, P. Kling: STOC'20]

Self-stabilizing LE (SSLE)

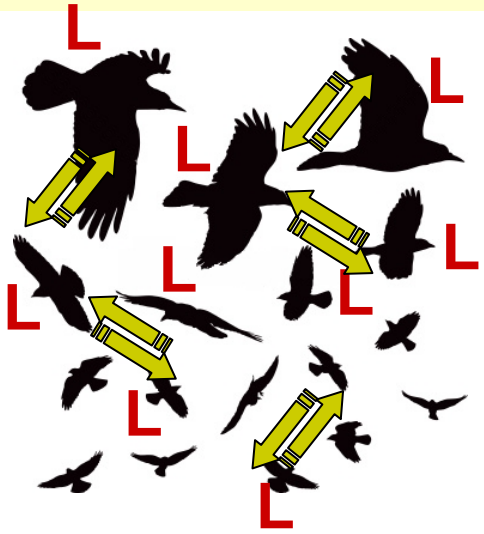
Motivation: unreliable, hardly accessible devices, subject to any number of transient failures

Self-stabilizing protocol:
starting from an **arbitrary configuration**, **stabilizes** (barring additional faults) with probability 1 to *correct configurations*

➔ Simple protocol **(L, L) → (L, F) fails** from a conf. without L



➔Requires Ω(log N) time whp from



N-1 agents have to interact and become F
(Ω(log N) time, by the coupon collector argument)

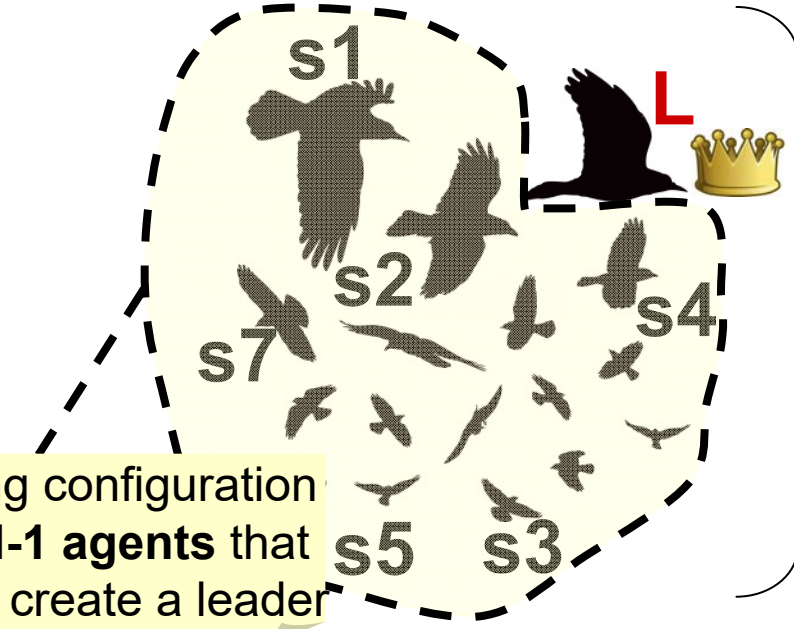
Self-stabilizing LE (SSLE)

Motivation: unreliable, hardly accessible devices, subject to any number of transient failures

Self-stabilizing protocol:
starting from an **arbitrary configuration**,
stabilizes (barring additional faults) with probability 1 to *correct configurations*

➔ **Different protocol for each N (non-uniform)**

➔ **Requires ≥ N states and *exact* knowledge of N**



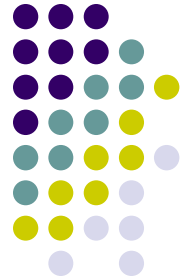
Starting configuration with **N-1 agents** that cannot create a leader

Stable configuration with N agents

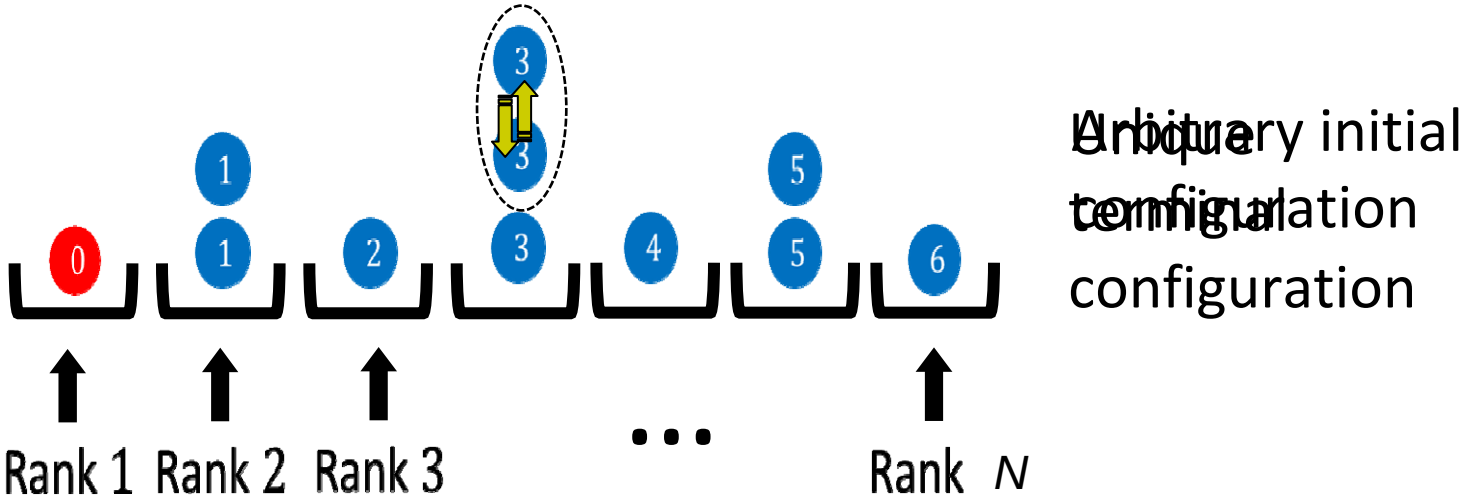
[S. Cai, T. Izumi, and K. Wada: TCS, 2012.]

Simple Self-stabilizing LE

[S. Cai, T. Izumi, and K. Wada: TCS, 2012]



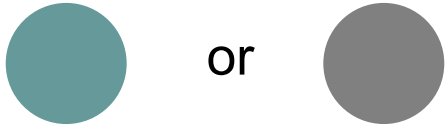
- **Agents' states** ∈ **{0, …, N-1}**, **0** is a **leader** state
- **Transition** func.: **(s, s) → (s, (s + 1) mod N)** for each state s



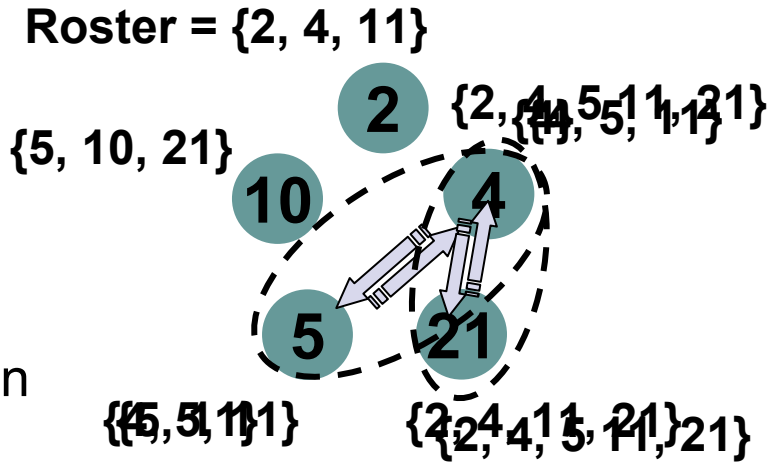
Solves strictly harder **Ranking problem** in **$\Theta(N^2)$ parallel time**

Time-optimal (O(log n)) SSLE/Ranking

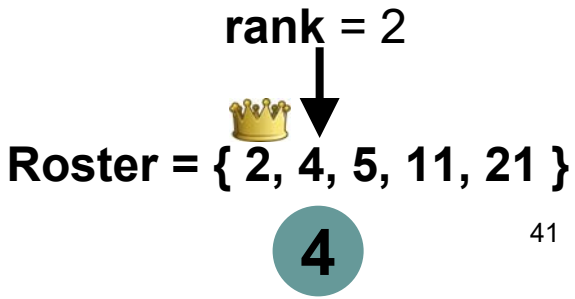
- An agent is either **Collecting** or **Resetting**.



- **When Collecting**, it has a **name** $\in \{1, \ldots, N^3\}$ and a **roster** array to collect the other names
  - two agents merge their rosters during an interaction
  - the names are propagated by epidemics in O(log N) time



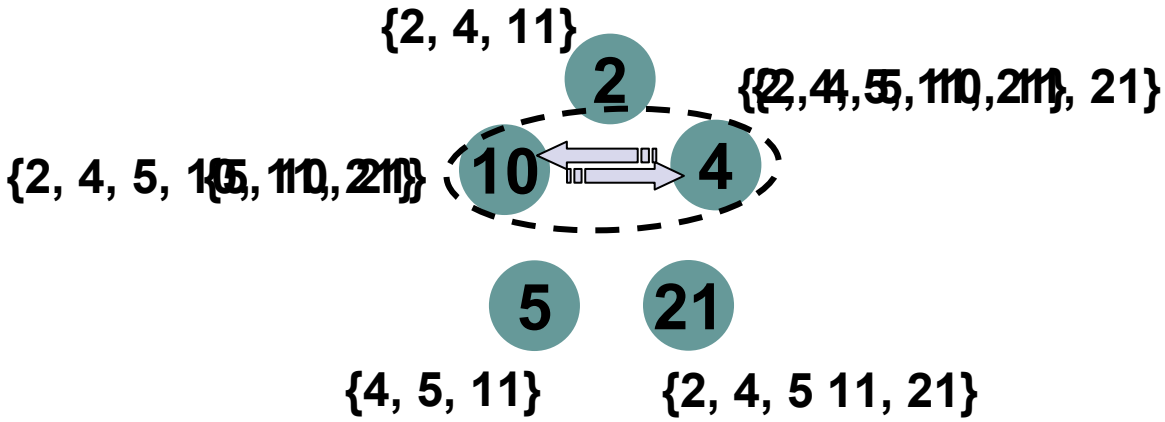
When **|roster| = N**, agent's **rank** is determined by the lexicographical order of its name in the roster



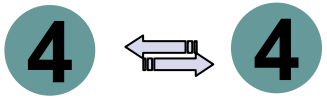
Reset (1)

Reset is triggered **if** an agent detects:

1. **"Ghost name" error**
   a name in a roster belonging to no agent.
   - If no homonyms, detected whenever **|roster| > N**
     (in O(log N) time)

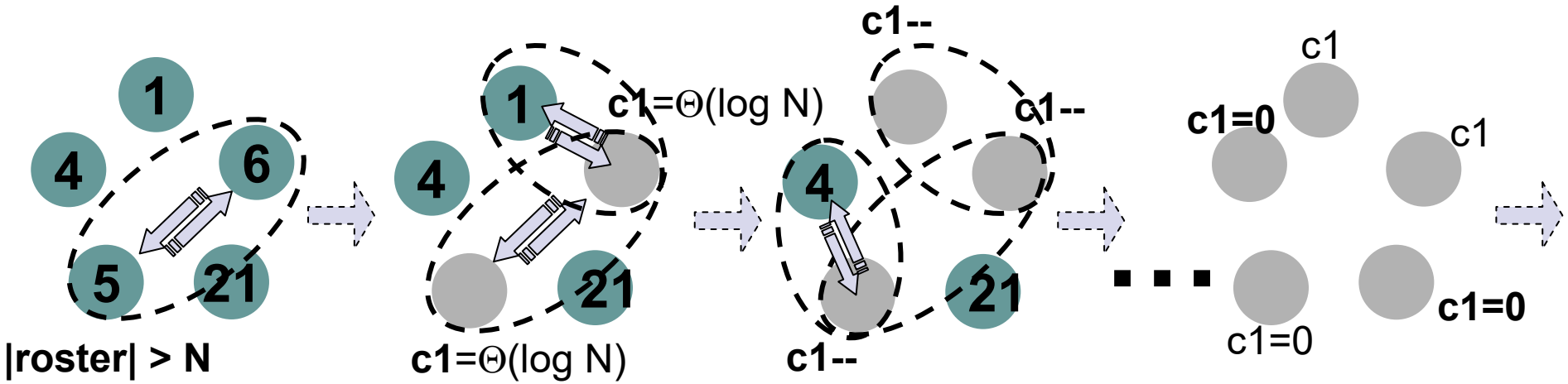


2. **Name collision error**



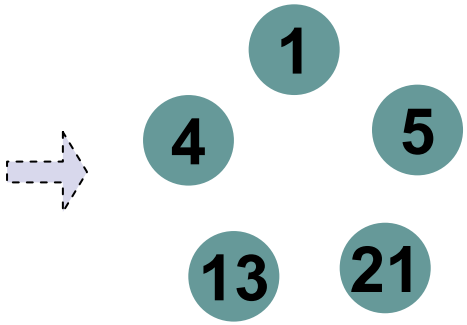
Reset (2)

➔ agents become **Resetting** spreading **Reset in O(log N) time**
("by epidemic", with counters set to Θ(log N))



Reset is triggered

Reset is spread until c1 reaches 0



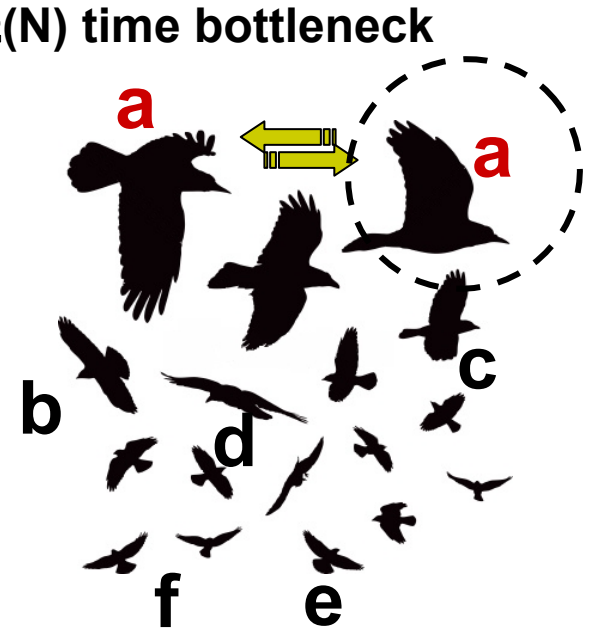
➔ **at Reset exit (when c1 is set to 0), each agent moves to Collecting and picks randomly a name out of N^3 names, with no collisions whp.**

(by the birthday paradox argument)

Name collision detection

- What if there are homonyms, and |roster| is never > N?

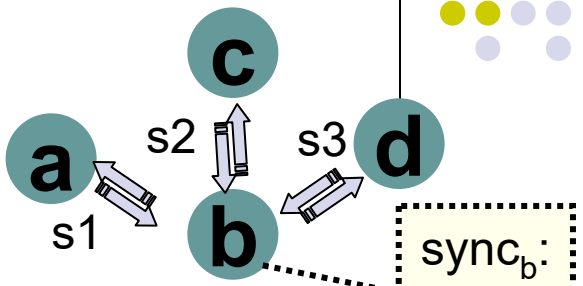
- A simple ***direct detection*** **takes Θ(N) time.**
  - requires direct interaction of two homonyms



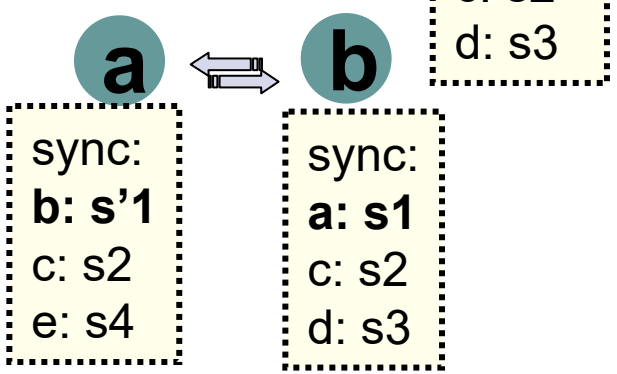
➔ **We use an *indirect* detection**

O(√n) name collision detection

- When two agents meet they generate a *shared random value* **sync** in {1, …, k}
  - They store this value in the dictionary near the name of the other.

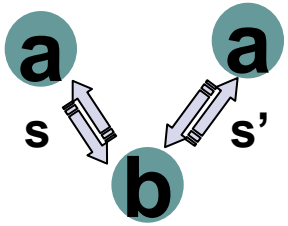


- **If** in the beginning of this interaction **they disagree on this sync value**, they declare a collision



If s'1 ≠ s1, Reset is triggered

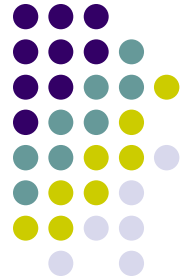
➔ **If there are homonyms,**
***some* agent interacting with both**
will detect it
with prob. 1-1/k in O(√N) time



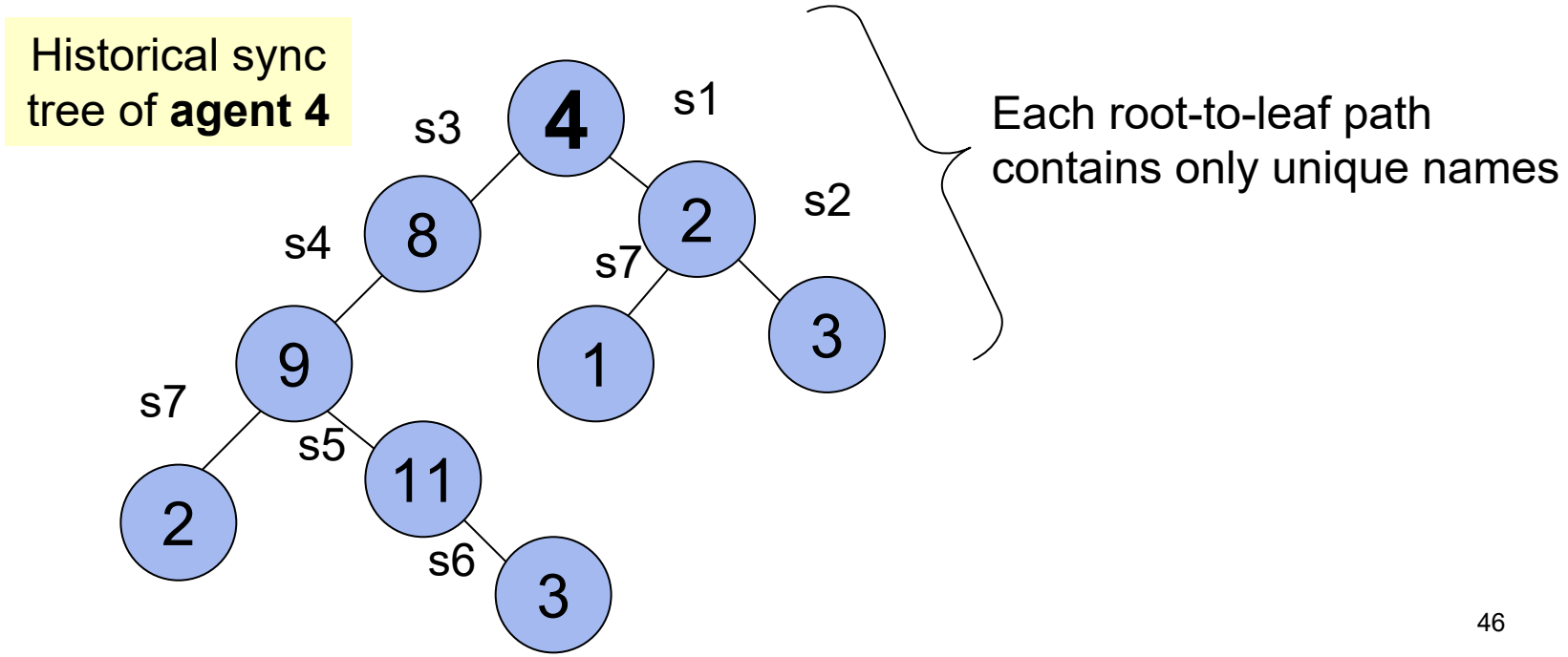
P(s≠s') = 1-1/k

➔ Safety: when all names are unique, all agents agree on the corresponding synch values

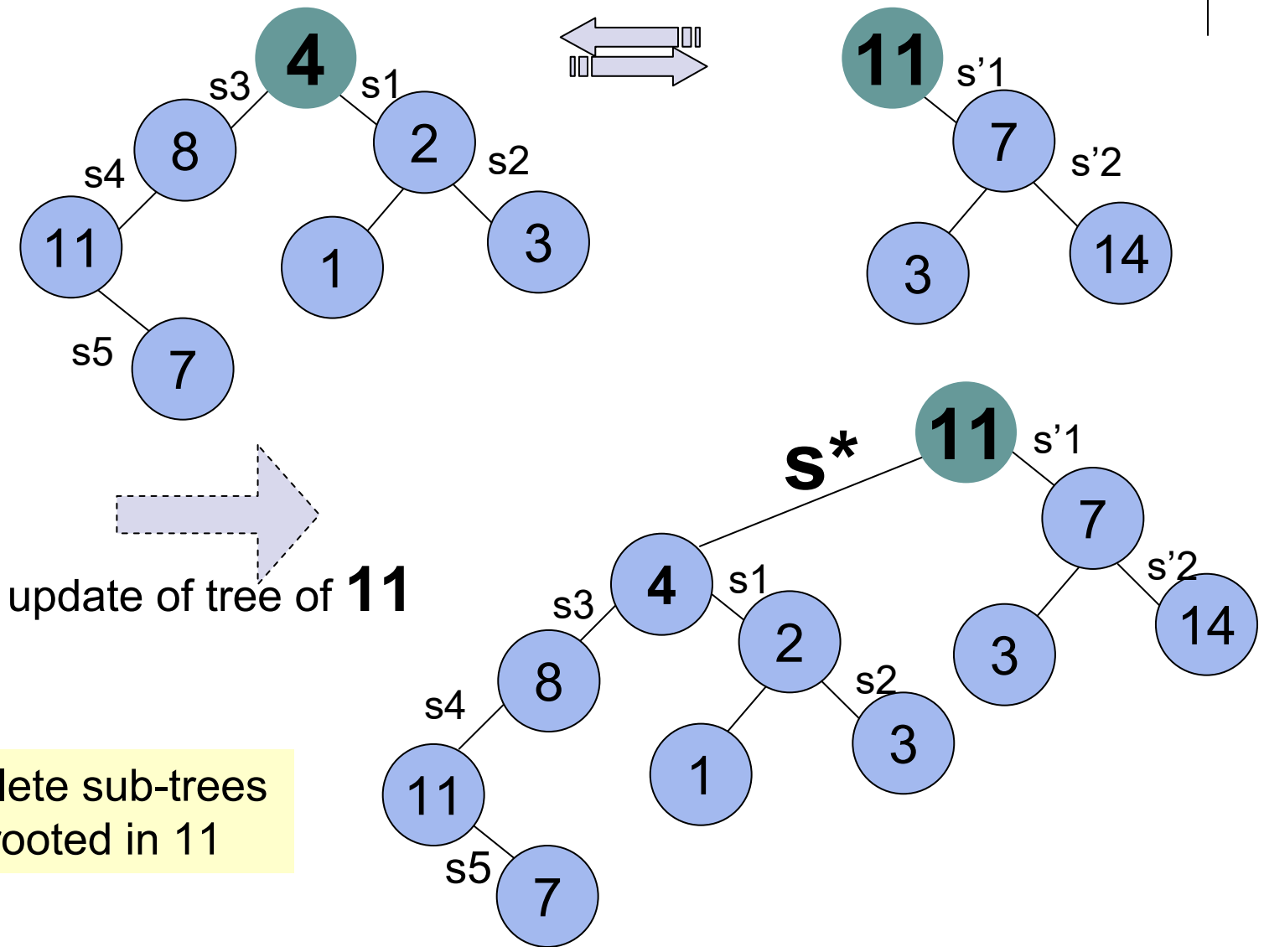
O(log n) name collision detection



- Each agent x keeps **historical sync values** of other agents in a tree structure rooted in x and s.t. each path contains only unique names.
  - **these root-to-leaf paths are historical chains of interactions with correspondingly generated sync values**

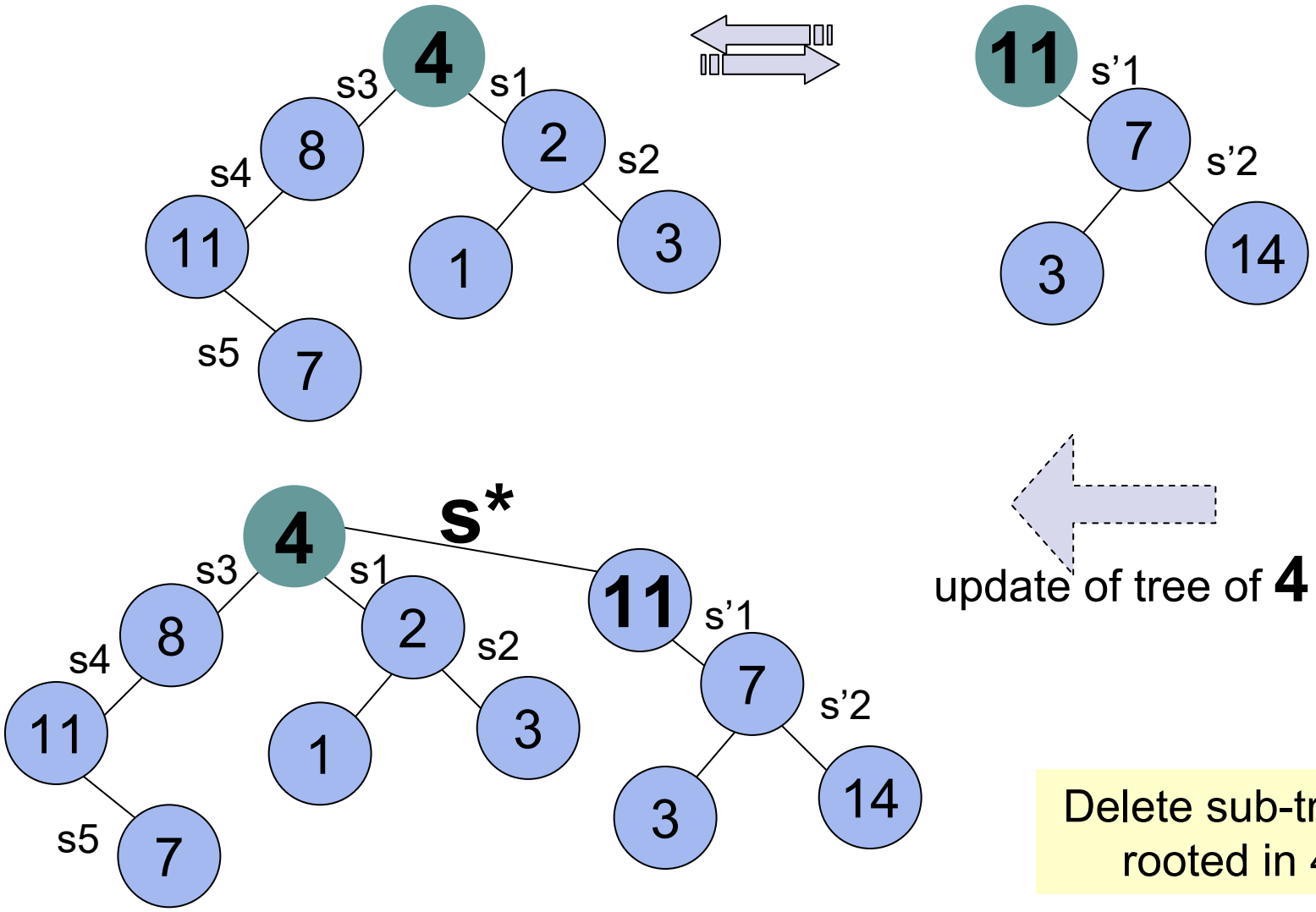


History trees update (1)



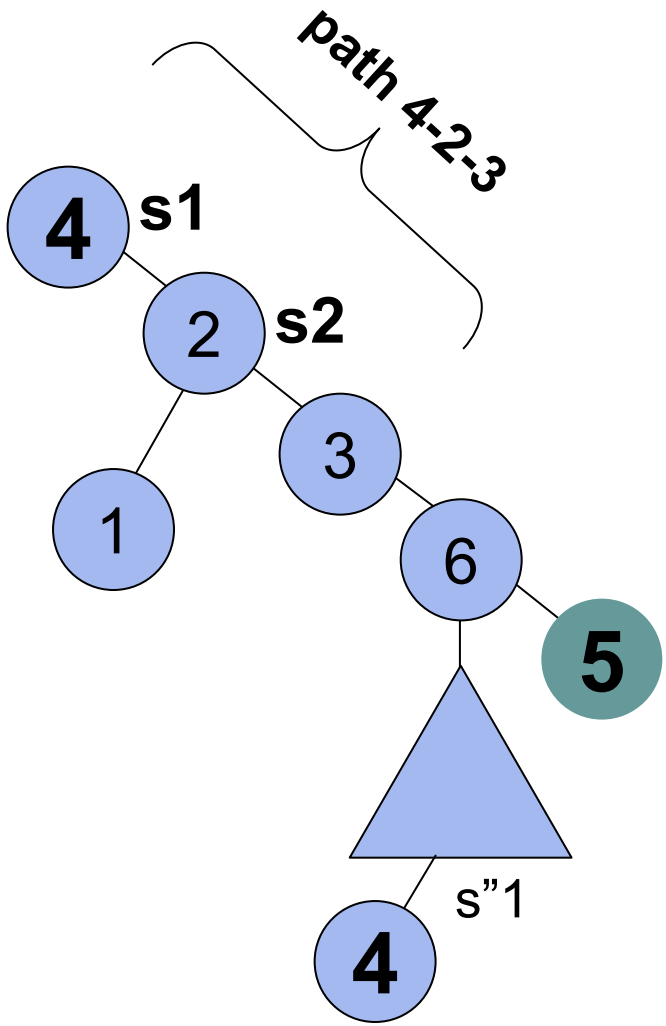
Delete sub-trees rooted in 11

History trees update (2)



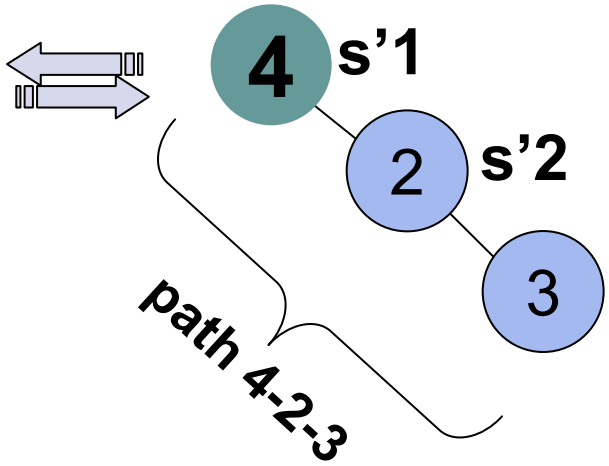
Delete sub-trees rooted in 4

Collision detection check



In an interaction (a,b),
a (resp. b) checks every path p from the root that ends up by b (resp. a) and appears in a's (resp. b's) tree.

If at least for some i, si = s'i, there is no name collision in p



Time and space analysis
of the logarithmic time SSLE/Ranking

Time:

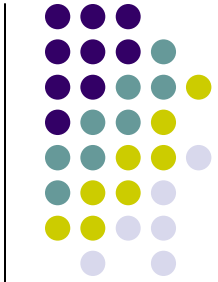
- "Ghost names" are detected in O(log N), via roster ("epidemic") merges, when |roster|>N
- Name collision detection takes O(log N) time:
  - It takes O(log N) time whp for some agent x to hear about two homonyms with corresponding history paths.
  - An homonym will not have any common sync value on a common path, with constant probability.
- Reset takes O(log N)

Space:

History tree is the most costly

$O(N^{\log N} \log N)$ bits = exp $O(N^{\log N} \log N)$ states

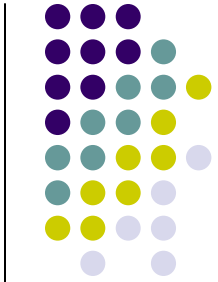
Conclusion



- **Time and space optimal counting**
- **Time optimal SSLE/ranking**

- **Self-stabilization is difficult in PP:**
  - In complete graphs with constant memory, only constant (or some very particular) **functions** are computable
    [S. Mathur, R. Ostrovsky: SSS'21]
  - **SSLE** is impossible in complete graphs with constant memory and without N
    [Angluin, Aspnes, Fischer: ACMJ'08], [S. Cai, T. Izumi, and K. Wada: TCS'12]
  - **SS counting and naming** under weak fairness are impossible without a leader (or other assumptions)
    [Beauquier, Clément, Messika, Rosaz, Rozoy: DISC'07]
    [Beauquier, Burman, Sohier: DISC'19]
  - **SS degree recognition** is impossible without the knowledge of N and of the # of possible interactions
    [Sudo, Shibata, Nakamura, Kim, Masuzawa: IEEE TPDS'21]
  -

PP Enhancements for Self-stabilization



- **Limit the interaction topologie**: bounded degree graphs, simple families of graphs:
  - coloring, orientation, spanning-tree [Angluin, Aspnes, Fischer: ACMJ'08]
  - uniform Leader Election (LE) in rings, tori and regular graphs [Chen, Chen: PODC 19, PODC`20] [Yokota, Sudo, Masuzawa: IEICE Trans. Fundam. Electron. Commu... nput. Sci.'21]

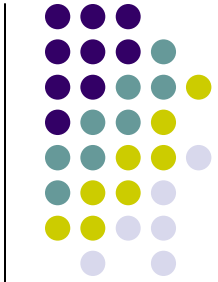
- **Partial synchrony**
  - **every agent meets all the others after a bounded number of int...**
    - mutual exclusion, phase clock, self-stabilizing transformer [BB C... OCOSS'10, BBK TCS'11]
  - **synchronous handshakes** (all agents interact in rando... synchronously):
    - periodic functions [Lamani, Yamashita: TPNC'16]
  - **k-wise interaction PP** [A... amauchi, Kijima, Yamashita: SSS'13]

adding synchronization to interaction scheduling

- **Reducing Uniformity: adding memory dependent on N, leader**
  - **Ω(N) states and knowledge of N**: SSLE (complete graphs) [Cai, Izumi, ...]
  - **Ω(N) states and/or leader:** deterministic oscillators, uniform bipartition, counting, naming [Yasumi, Ooshita, Yamaguchi, Inoue: IEICE Trans.'19] [Cooper, L... Yamashita, Yamauchi: IC'17], [BBCS DISC'15; ABBS ...5; BBS DISC'19]
  - **Mediated PP** (add shared memory for every pair of agents): SSLE [Mizoguchi, Ono, Kijima, Yamashita: DC'12]

reducing uniformity: more memory, leader

Relaxed self-stabilization

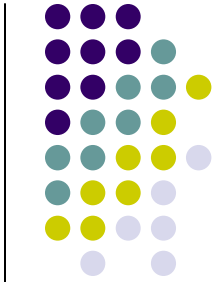


Loosely-stabilizing LE

with upper bound U on N and

- stabilizing in polylog(N) time, but not forever - with poly(N) holding time [Sudo, Ooshita, Kakugawa, Masuzawa, Datta, Larmore: TCS'20]
- with exp(N) holding time: stabilization $\Omega(UN)$ and $\Omega(U)$ states are necessary and sufficient [Izumi: SIROCCO'15]

Perspectives



- Study **more time-space trade-offs**
  - more on leader election (improve state space), majority, consensus, counting, naming, topology constructions, …
  - **under different model parameters** (fairness, existence of leader, symmetry in transitions, type of interaction graph …)
- **General characterization of the computational power**
  - with $\Omega(N)$ states, for non-simple family of graphs, bounded degree graphs, for other stronger models …

Thank you for attention!