Beyond classical chip design lecture 2

Self-stabilization (continued)

Further Reading

Dijkstra, Edsger W.: *Self-stabilization in spite of distributed control.* Selected writings on computing: a personal perspective. Springer New York, 1982. 41-46.

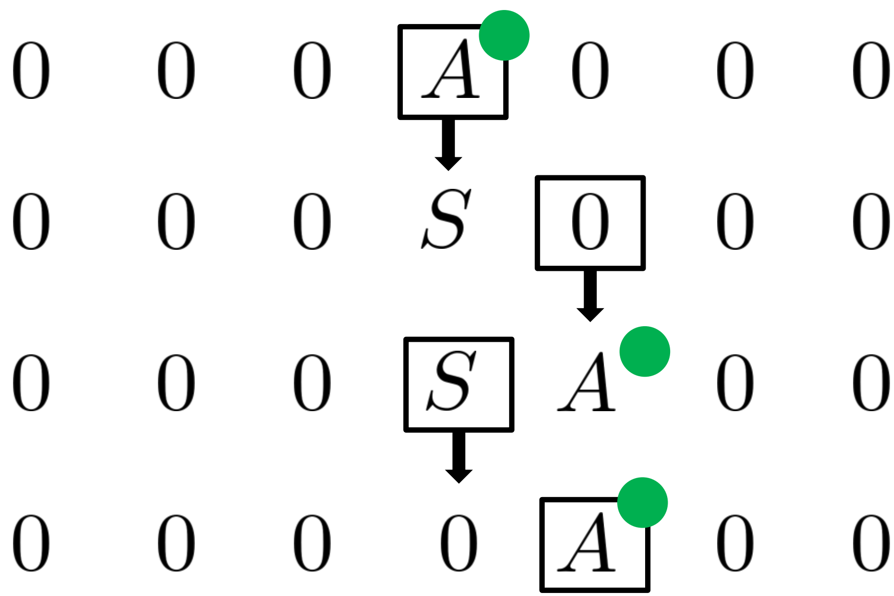
Brown, Geoffrey M., Mohamed G. Gouda, and Chuan-Lin Wu: *Token systems that self-stabilize.* Computers, IEEE Transactions on 38.6 (1989): 845-852.

What we had...

Algorithm:

... enabled = non-trivial transition

... token



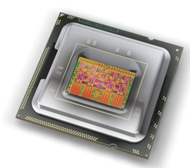
What we wanted...

Algorithm:

3 states,

uniform,

very simple predicate and transition function

-> 

But wait...

... $s(t) \in [n]$

Generally...

Stable transition functions:

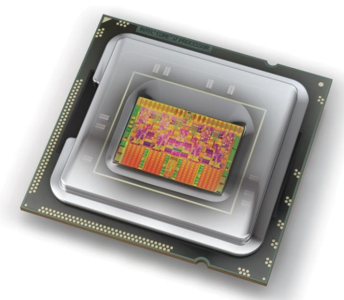
i can make a transition to c at time t &

i **cannot** make a transition to c at time $t+1$

->

i made a transition at time $t+1$

(and thus is in c at time $t+1$)

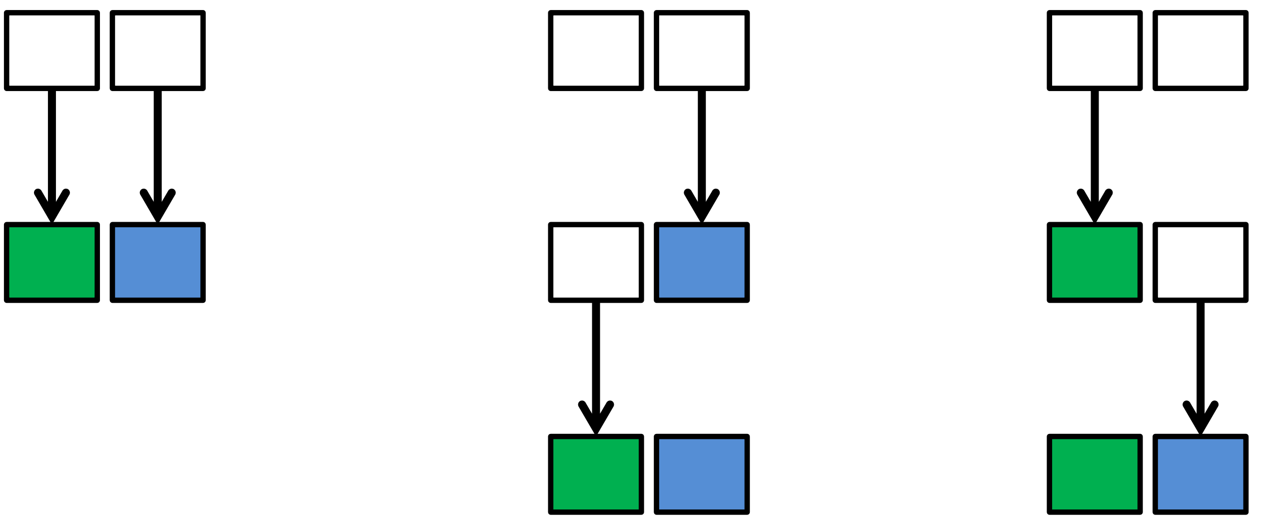


Generally...

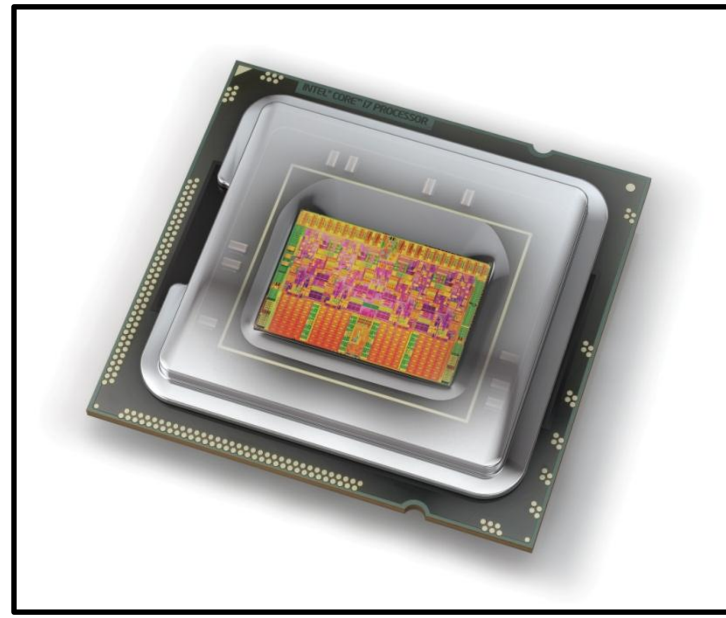
"distributed schedule" $s(t) \subseteq [n]$

stable + distributed schedule ->

"linearizable to" schedule [later]

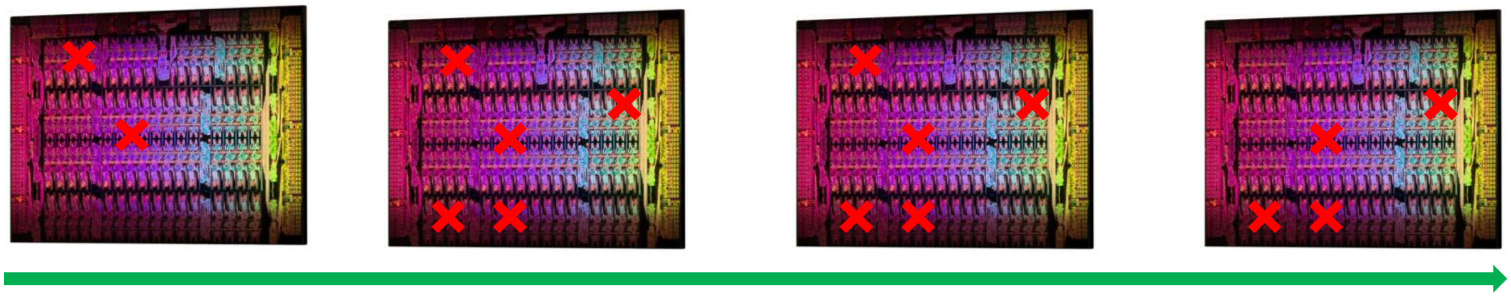


Reliable designs

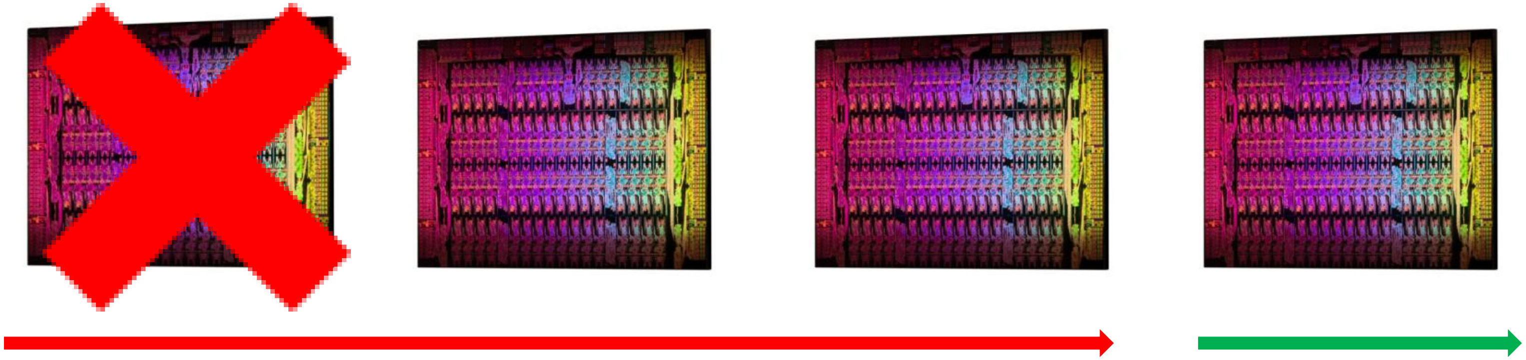


Reliable designs

Fault-tolerance.



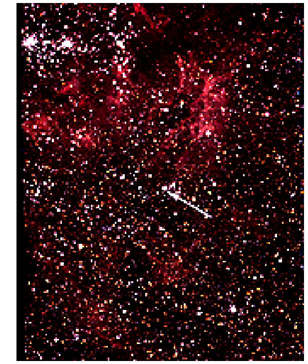
Self-stabilization.



Robustness...

$$0 \quad 0 \quad 0 \quad \boxed{A} \quad 0 \quad 0 \quad 0 \quad 0 \quad 0$$

Robustness...



... no mutex.

Self-stabilization

For all initial states, all executions from this state: exists a time T:

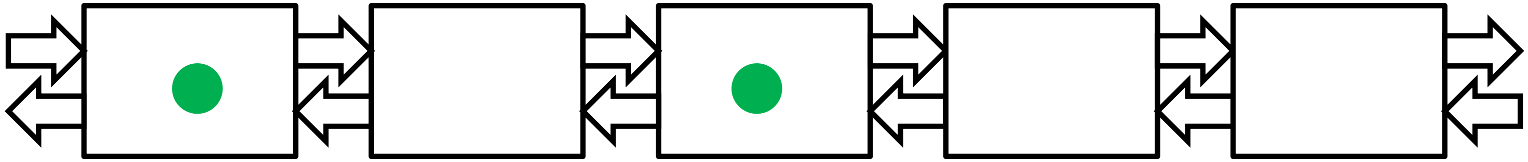
T-postfix fulfils requirements.

Exists a time T: for all initial states, all executions from this state:

T-postfix fulfils requirements.

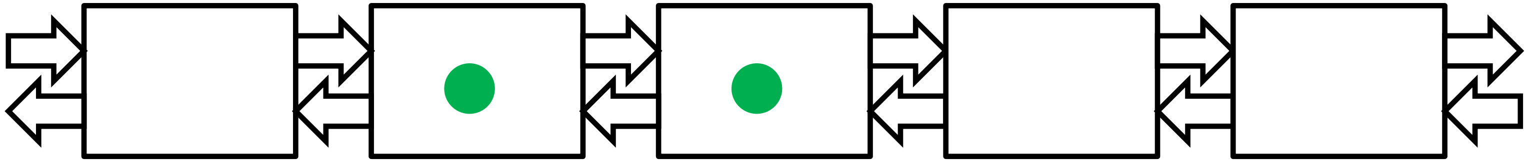
Example problem

Token passing system.



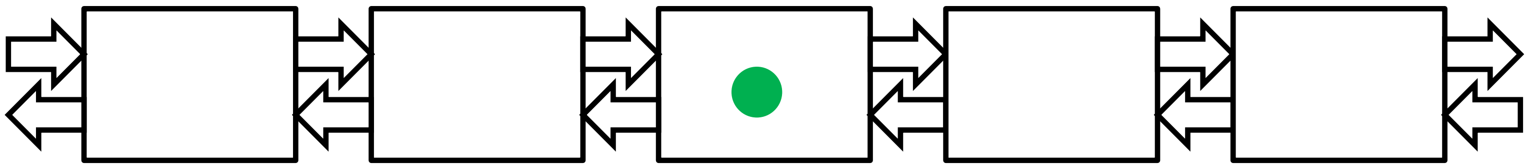
Example problem

Token passing system.



Example problem

Token passing system.



Example problem

Token passing system.

Example problem

Token passing system.

Self-stabilization

"equal speeds are bad"

Self-stabilization

“equal speeds are bad”

Solution 1. Randomness.

- implementation
- fault-free behavior

Self-stabilization

“equal speeds are bad”

Solution 1. Randomness.

- implementation
- fault-free behavior

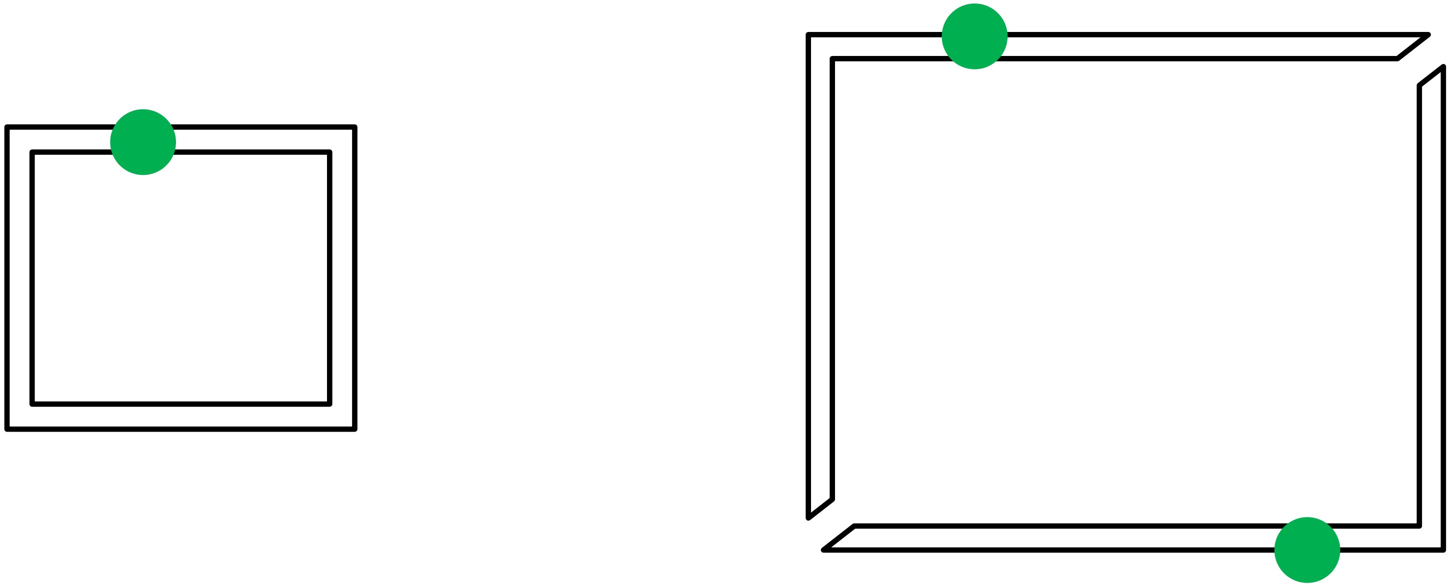
But: ... no token case!

Self-stabilization

Uniform deterministic solutions for all ring sizes?

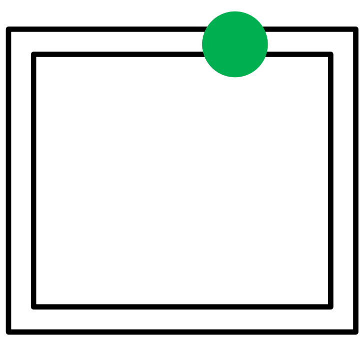
Self-stabilization

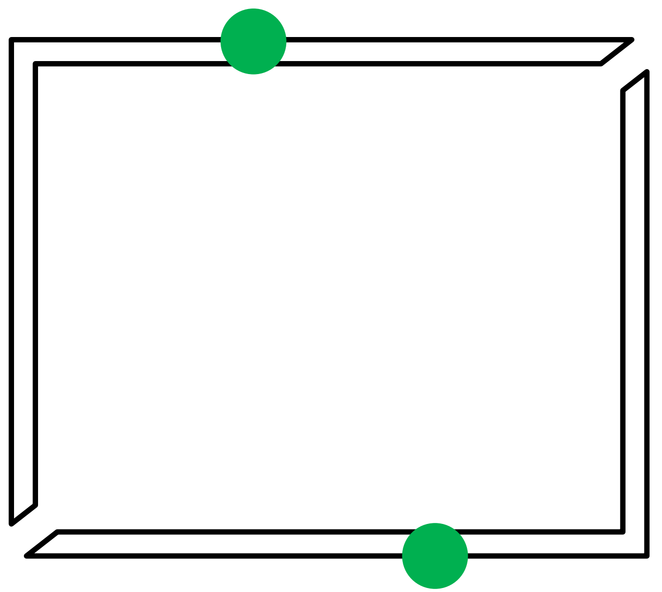
Uniform deterministic solutions for all ring sizes?



Self-stabilization

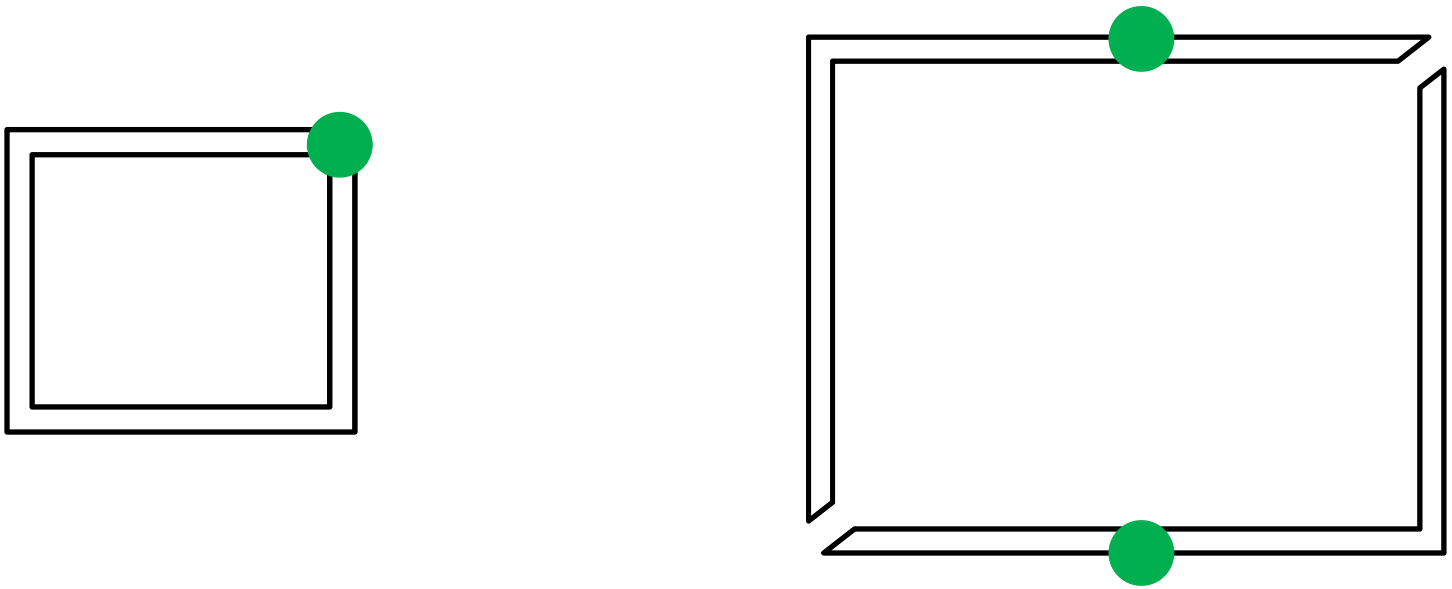
Uniform deterministic solutions for all ring sizes?





Self-stabilization

Uniform deterministic solutions for all ring sizes?



Double -> contradiction

Self-stabilization

Solution 2. Dijkstra (det, non-uniform, uses size)

machine 0:

$$\text{if } x_{i-1} = x_i \text{ then } x_i' = x_i + 1 \mod (N+1)$$

all others (1..N):

$$\text{if } x_{i-1} \neq x_i \text{ then } x_i' = x_{i-1}$$

Self-stabilization

N “other“ nodes

N+1 states from $V = [N+1] = \{0, \ldots, N\}$

-> say, state N does not occur.

Self-stabilization

Obs 1. node 0 first one to have N.

Obs 2. from N(non-N)....(non-N) eventually reach N...N.

Obs 3. from N...N only 1 execution with mutex & weak fairness.

Self-stabilization

Prop 1. Show $\exists t : x_0(t) = N$

Assume not.

-> 0 makes bounded # non-trivial steps

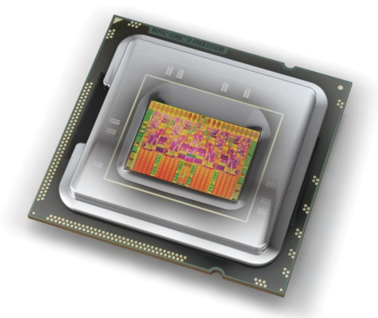
-> last at time t' with $x_0(t') = a$

-> eventually $a \ldots a$

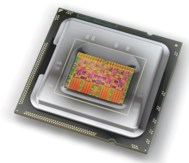
-> eventually 0 makes step

-> contr.

Self-stabilization



->



$V = [N]$: “mod N” instead of “mod N+1”?

Not with distributed scheduler. [hw]

Self-stabilization

not stable, but:

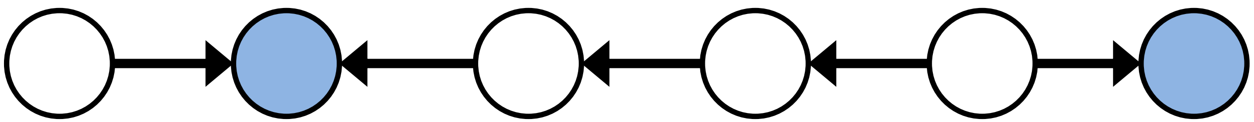
- works with distributed scheduler
- for all ring sizes exists solution

Self-stabilization

Solution 3. [Brown, Gouda]

not stable <-> two neighbours try to make a step at the same time

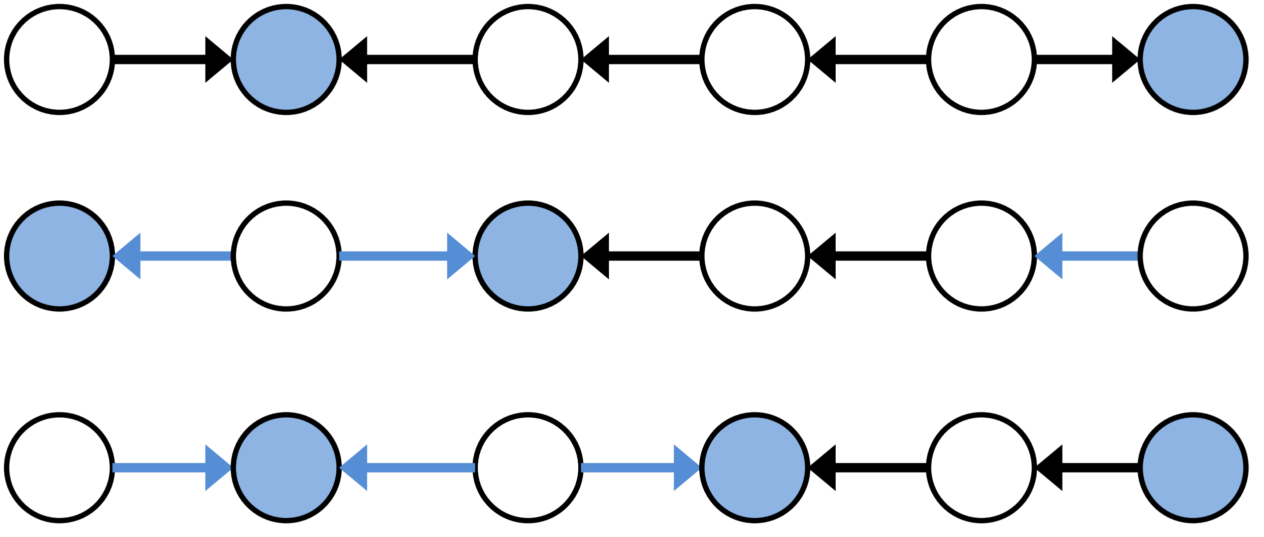
Self-stabilization



Prop 1. neighbour-mutex holds.

Self-stabilization

link-reversal e.g. full/partial reversal



…

Self-stabilization

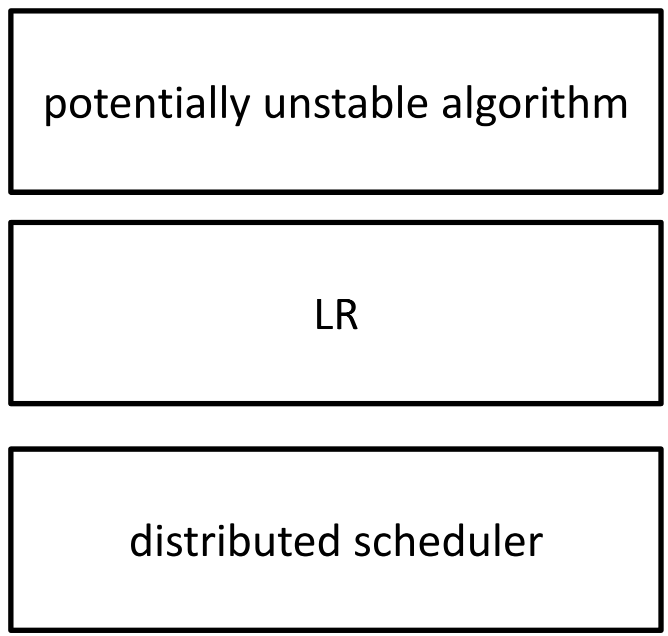
Ring cut...

Prop 2. No deadlock. [hw]

Prop 3. Weak fairness. [hw]

What we obtain...

... link reversal gives a neighbour-mutex, weak fair scheduler.



Simulating scheduler

Distributed, weak-fair scheduler ->

Distributed, **neighbour-mutex**, weak fair scheduler.