

Examples of self-stabilization



`alexandre.maurer@epfl.ch`

Outline

- Self-stabilizing coloring
- Self-stabilizing pulse
- Break
- Self-stabilizing broadcast (blackboard)

Self-stabilizing coloring

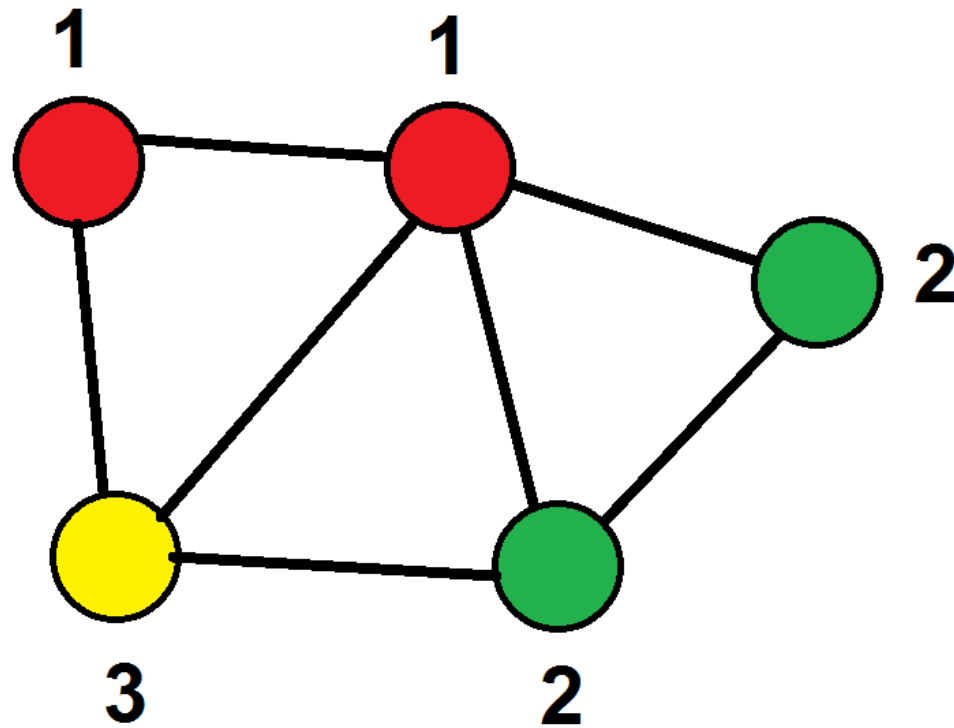


Context

- A graph of degree D
(D = max number of neighbors per node)
- $D + 1$ "colors" $\{1, 2, \dots, D+1\}$
- Each node p has a color $C(p) \in \{1, 2, \dots, D+1\}$

Node coloring problem

Initially, the nodes have any colors:

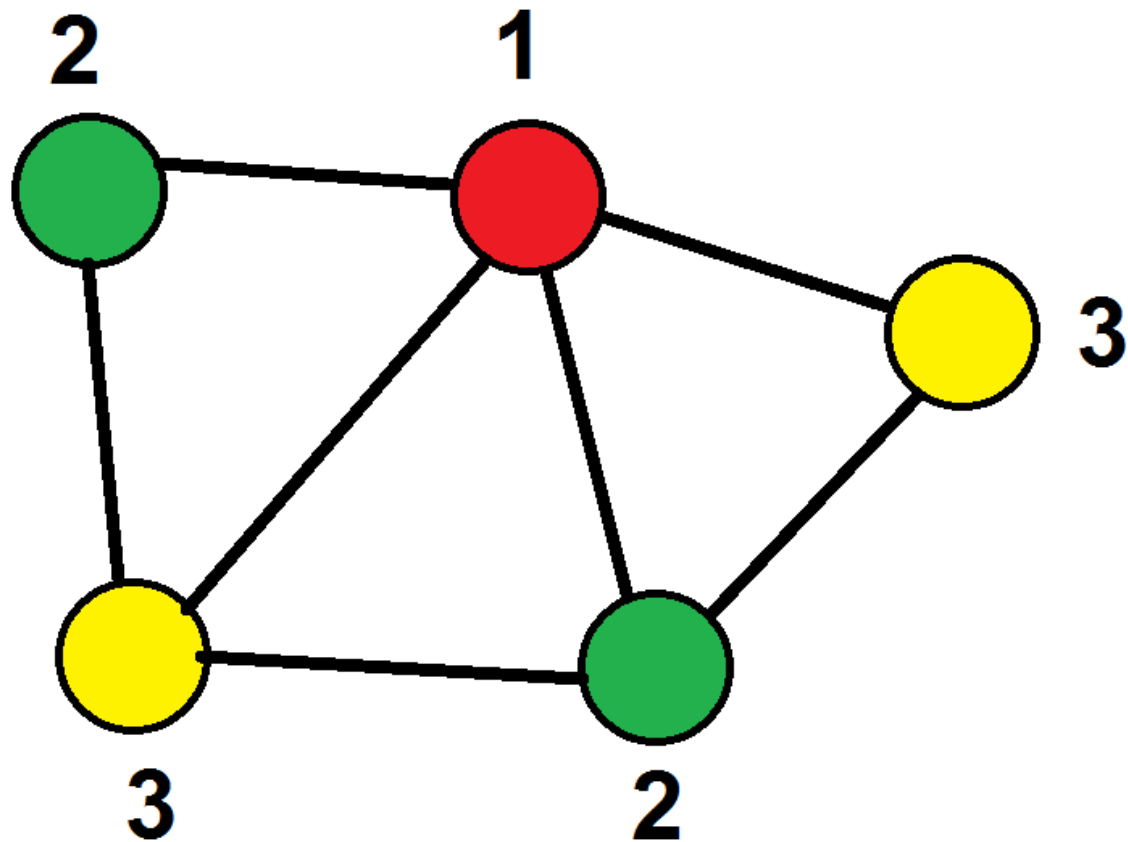


Eventually, we must satisfy the following property:

*For any two neighbor nodes p and q ,
 $C(p) \neq C(q)$*

(the graph is "**well colored**")

Example of "well colored" graph:



Model

- Each node is eventually "activated"
- When a node is "activated", it can execute a given algorithm
- Two neighbor nodes are never activated at the same time

Algorithm

When a node p is activated :

- Let $N(p)$ be the set of neighbors of p
- Let C be a color such that :

$$\forall q \in N(p), C(q) \neq C$$

→ Then, $C(p) := C$

(Such a color C always exists because:

- p has at most D neighbors
- we have $D+1$ colors)

Our goal

Prove that, with this algorithm,
the graph is always eventually
"well colored",
AND remains "well colored"

(In other words, the coloring of the
graph is **self-stabilizing**, because it
works for any initial coloring)

Definition

A node p is "well colored" if:

$$\forall q \in N(p), C(q) \neq C(p)$$

→ If all nodes are "well colored",
then the graph is "well colored"

Lemma 1 (Liveness property)

*Let p be a node that is
"not well colored".*

Then, p is eventually "well colored".

Proof

- p is eventually activated
- When p is activated, no neighbor of p is activated in the same time
- Then, p executes the algorithm, and takes a color different from its neighbors
- Then, p becomes **"well colored"**

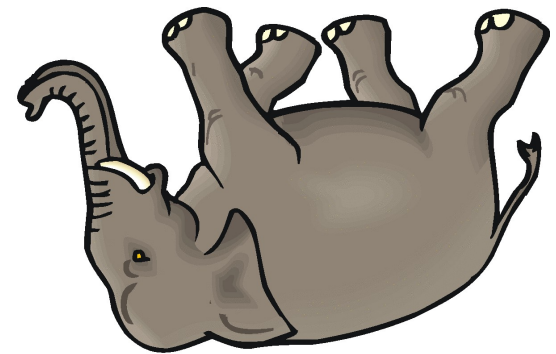
Lemma 2 (Safety property)

*If p is "**well colored**", then p always remains "**well colored**".*

Proof

The proof is by contradiction:

- We suppose the opposite
- We show that this leads to a contradiction



Suppose the opposite: a node p is "**well colored**", then, after a certain time, p is "**not well colored**".



→ Changes only happen when nodes are activated.

Therefore, consider the activation where p goes from "**well colored**" to "**not well colored**".

- 2 cases (mutually exclusive) :
- p is activated
 - at least one neighbor of p is activated

Case 1: p is activated

By hypothesis, no neighbor of p is activated at the same time.

Then, it implies that p takes the same color as one of its neighbors.

→ **contradiction with the algorithm!**



Case 2: at least one neighbor q of p is activated

By hypothesis, p is not activated at the same time.

Then, it implies that q takes the same color as p .

→ **contradiction with the algorithm!**



Liveness property:

*A node **"not well colored"** eventually becomes **"well colored"**.*

Safety property:

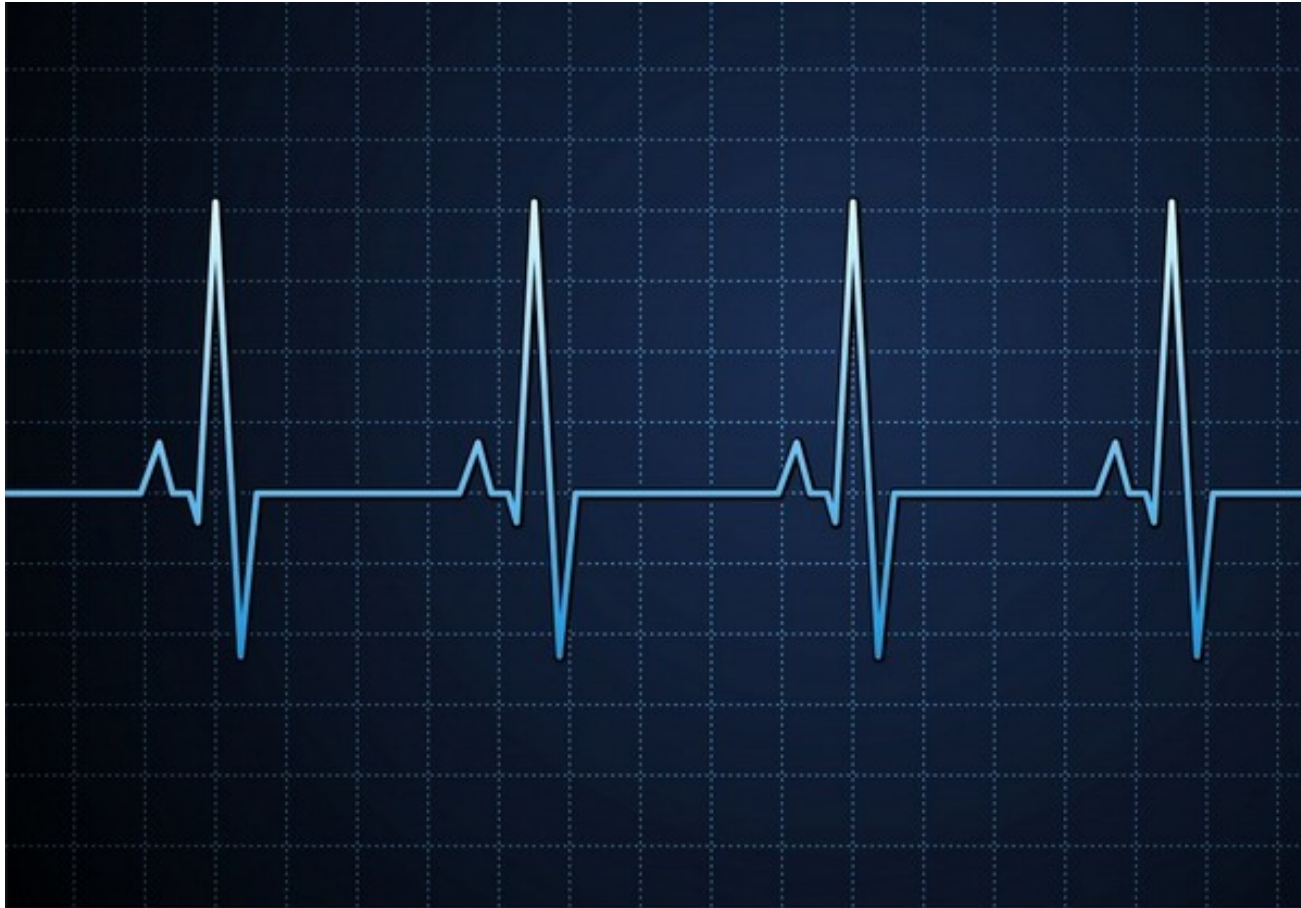
*A node **"well colored"** always remains **"well colored"**.*

- **Each node** is eventually **"well colored"**, and remains **"well colored"**
- **The graph** is eventually **"well colored"**, and remains **"well colored"**

We proved that this (simple) algorithm
is **self-stabilizing** for the
node coloring problem.



Self-stabilizing pulse



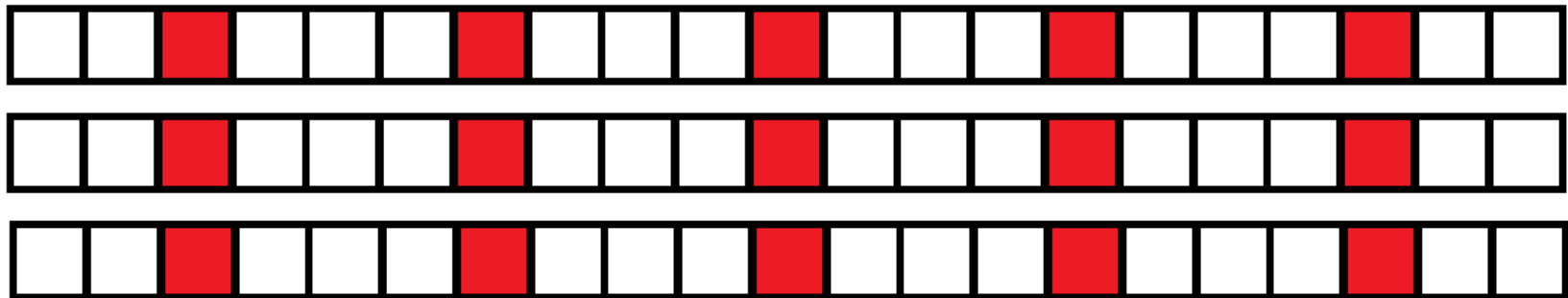
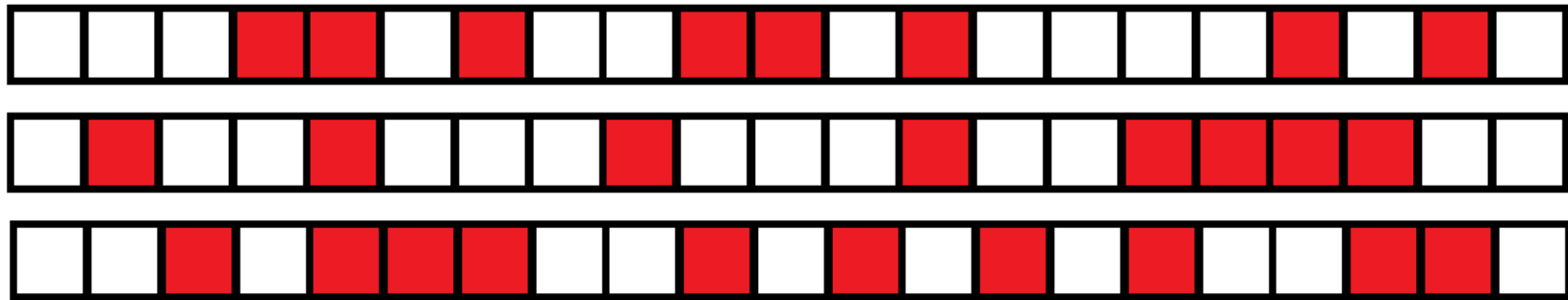


Clapping



Fireflies

Problem



$$T = 4$$

Model

- Discrete time
- n processes
- At any time, each process can "beep"
- All processes know T (the desired period)

$\text{Beep}(t) = 1$

if at least one process beeps at time t

...and 0 otherwise

Algorithm

For each process at time t :

If $\forall t' \in \{t - 1, t - 2, \dots, t - T - 1\}$

Beep(t') = 0

...then, beep.

Otherwise, do not beep.

Lemma 1

Eventually, at least one process beeps.

Proof

Let t be any time.

- If one process beeps at a time $\{ t, t+1, \dots, t+T-1 \} \rightarrow \text{WIN!}$
- Otherwise, according to the algorithm, all processes beep at $t+T \rightarrow \text{WIN!}$

Lemma 2

If at least one process beeps at time t ,
then all processes beep at time $t+T$
(...and do not beep between t and $t+T$)

Proof

- At time $\{t+1, t+2, \dots, t+T-1\}$, according to the algorithm, no one beeps.
- Therefore, at time $t+T$, according to the algorithm, everybody beeps.

Synthesis

- Eventually, someone beeps at some time t
- Then, everybody beeps at $t + T$.
- Therefore, by induction, everyone beeps at $t+2T$, $t+3T$, $t+4T$, ...

→ **Synchronous beeping**



