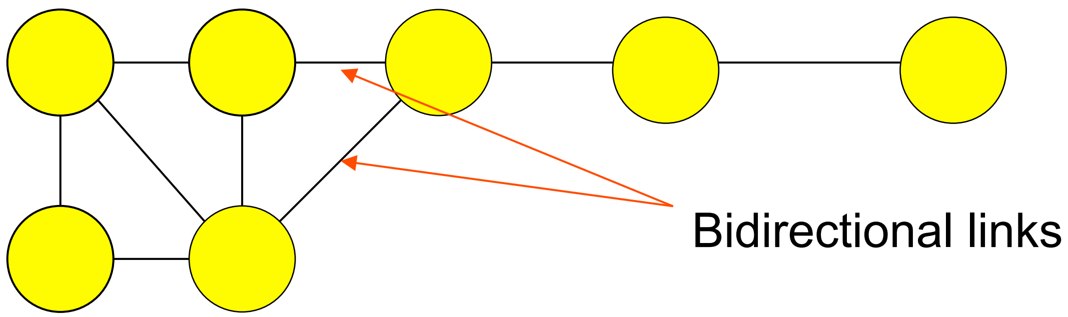
Stabilization and Synchronization of Logical Clocks

Franck Petit

UPMC
Paris

Preliminaries

- $G=(V,E)$ is a connected network
- V: set of n nodes/processes
- E: set of m bidirectional links
- N_p: set of neighboring nodes of p
- Memory shared between neighboring nodes



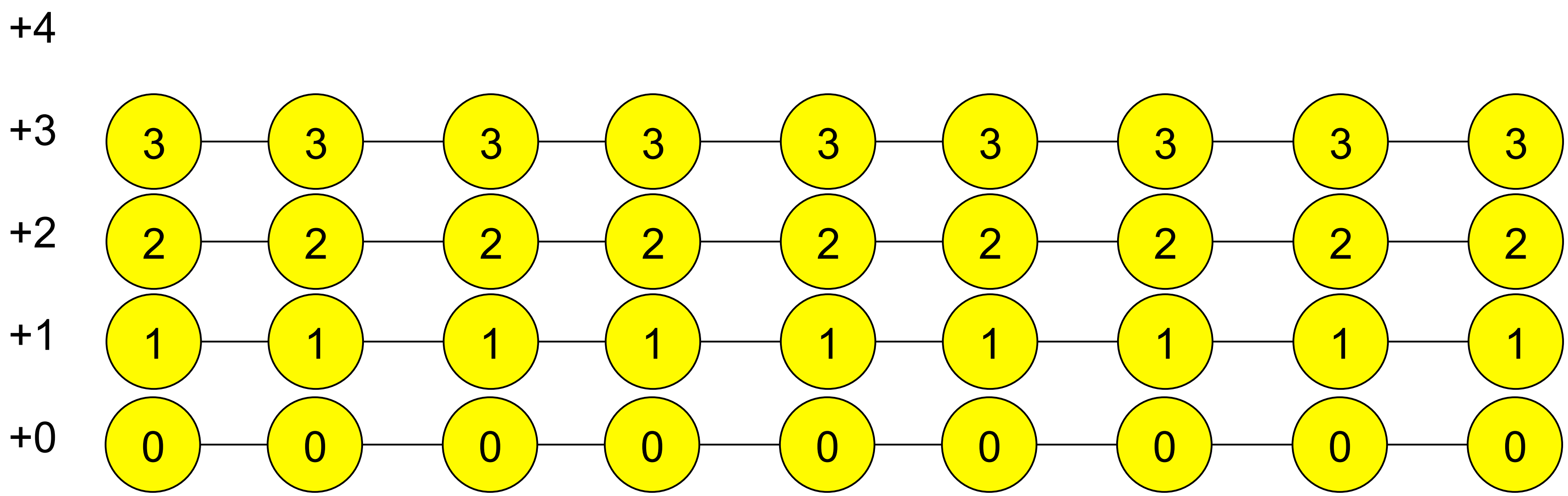
Preliminaries, Distributed Algorithm

- In each computation step:
  - According to its variables and the variables of its neighbors, a node/process is either enabled to execute an action or not
  - Synchronous system
    - Every enabled nodes execute an action atomically
  - Asynchronous system
    - Some enabled nodes are chosen by an unfair adversary
    - The chosen nodes execute an action atomically

Logical Clock Synchronization

- Each node p maintains a logical clock register r_p
- Synchronous/Asynchronous Environment

$$r_p := r_p + 1$$

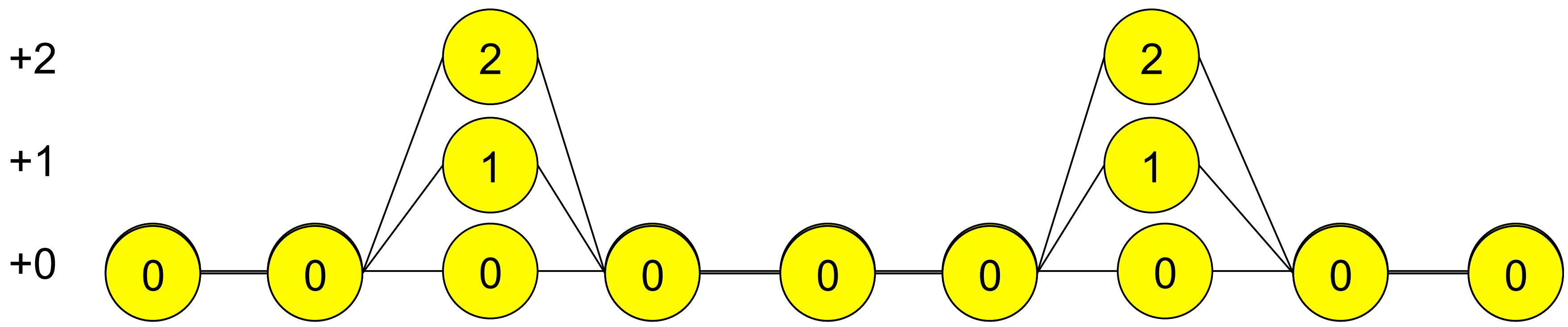


Logical Clock Synchronization

- Each node p maintains a logical clock register r_p
- Synchronous/Asynchronous Environment

$$r_p := r_p + 1$$

+ *Synchronizer*

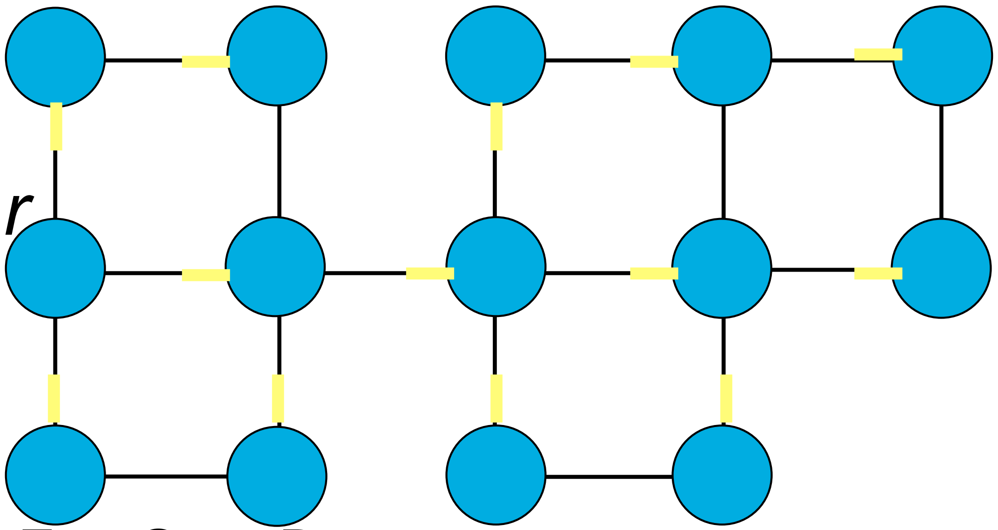


Global Synchronizer (asynchronous) distributed systems

In systems with unique IDs or a particular node (root, leader, main server…)?

- **Wave Algorithms**, *e.g.*,
  - Propagation of Information with Feedback (PIF)
  - Depth-First Token Circulation
- **Global Synchronization**, e.g.,
  - (Group) Mutual Exclusion
  - Leader Election
  - Reset
  - Logical Clock Synchronization
  - Rooted Spanning Tree
  - …

Propagation of Information with Feedback



- For One Puse:
  - Best Case: *O(D)*
  - Worst Case: *O(N)*

Can we provide better complexities?

Logical Clock Synchronization

- Each node p maintains a phase clock register r_p
- Synchronous/$\boxed{\text{Asynchronous}}$ Environment

$$\mathit{If}\ \forall q \in N_p : r_p \leq r_q\ \mathit{then}\ r_p := r_p + 1$$

Logical Clock Synchronization

Unison

- Each node maintains a phase clock register r in $[0,\dots,K-1]=Z_K$

- ***Safety***: *The gap between the phase clock of two neighboring nodes is at most equal to* $1 \pmod K$.

- ***No starvation (Vivacity)***: *Each phase clock* r *is incremented by* $1 \pmod K$ *infinitely often.*

Logical Clock Synchronization

Unison

$K=5$

Minimality of K ?

Logical Clock Synchronization

- K=2?
- No possible order among the clocks

Logical Clock Synchronization

- Successor Function and Predecessor Function possible if $K \geq 3$
- $K=3$
  - Local total order $\leq_l$ over $Z_3=\{0,1,2\}$

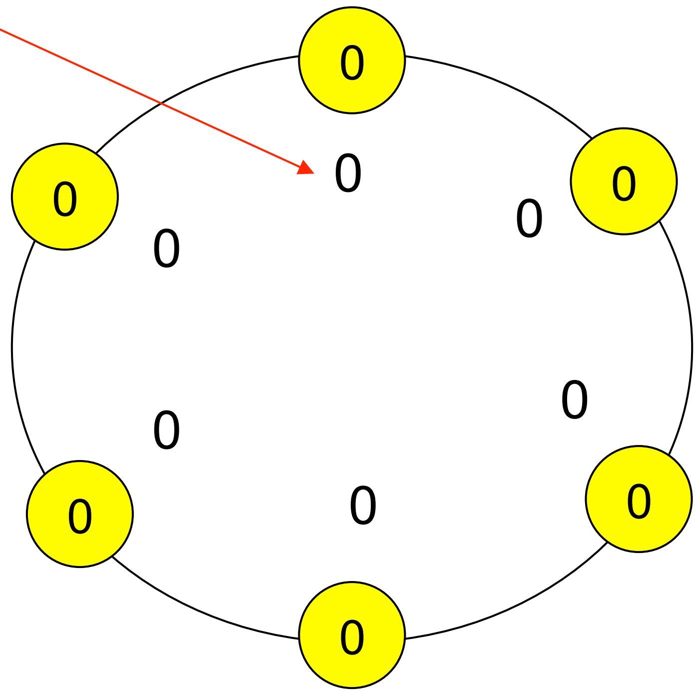
$$a \leq_l b \text{ iff } 0 \leq b-a \bmod 3 \leq 1$$

$$0 \leq_l 1 \;;\; 1 \leq_l 2 \;;\; 2 \leq_l 0 \;;$$

Logical Clock Synchronization

- K=3, Lifting

p.R, virtual register, it counts the number of increments of *p*



δ_0 — *State of the virtual p.R lifted from* γ_0

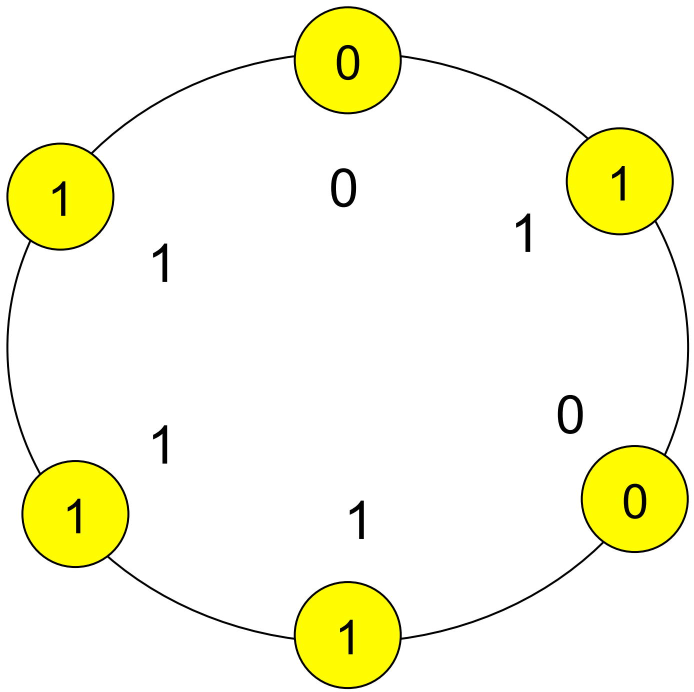
$\downarrow$ — *Projection mod K*

γ_0 — *State of p.r*

Logical Clock Synchronization

- K=3, Lifting

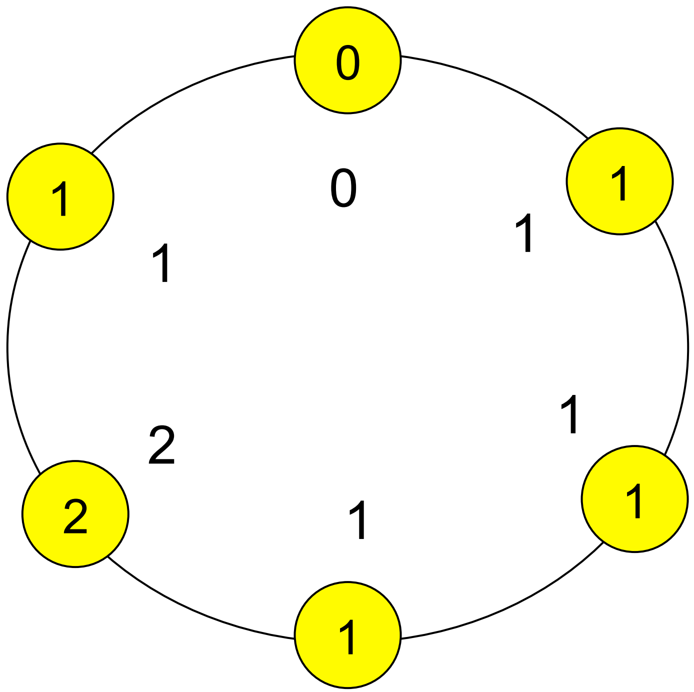
$$
\begin{tikzcd}
\delta_0 \arrow[r, "D_0"] \arrow[d] & \delta_1 \arrow[d] \\
\gamma_0 \arrow[r, "D_0"] & \gamma_1
\end{tikzcd}
$$



Logical Clock Synchronization

- K=3, Lifting

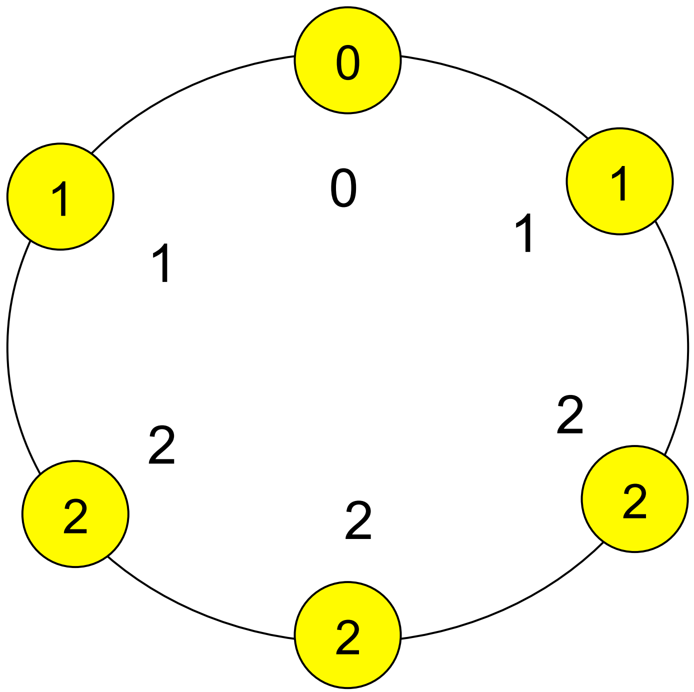
$$\begin{tikzcd} \delta_0 \arrow[r, "D_0"] \arrow[d] & \delta_1 \arrow[r, "D_1"] \arrow[d] & \delta_2 \arrow[d] \\ \gamma_0 \arrow[r, "D_0"] & \gamma_1 \arrow[r, "D_1"] & \gamma_2 \end{tikzcd}$$



Logical Clock Synchronization

- K=3, Lifting

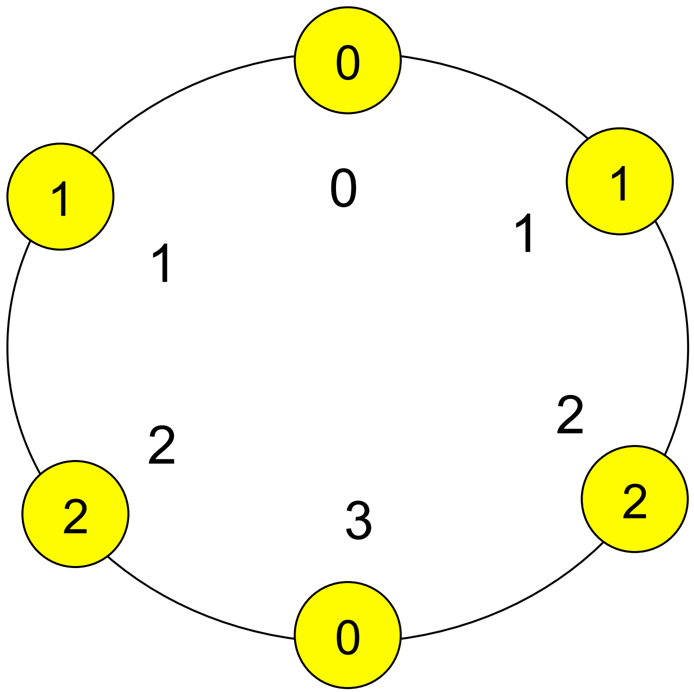
$$
\begin{tikzcd}
\delta_0 \arrow[r, "D_0"] \arrow[d] & \delta_1 \arrow[r, "D_1"] \arrow[d] & \delta_2 \arrow[r, "D_2"] \arrow[d] & \dots \\
\gamma_0 \arrow[r, "D_0"] & \gamma_1 \arrow[r, "D_1"] & \gamma_2 \arrow[r, "D_2"] & \dots
\end{tikzcd}
$$



Logical Clock Synchronization

- K=3, Lifting

$$\begin{tikzcd}
\delta_0 \arrow[r, "D_0"] \arrow[d] & \delta_1 \arrow[r, "D_1"] \arrow[d] & \delta_2 \arrow[r, "D_2"] \arrow[d] & \ldots \\
\gamma_0 \arrow[r, "D_0"] & \gamma_1 \arrow[r, "D_1"] & \gamma_2 \arrow[r, "D_2"] & \ldots
\end{tikzcd}$$



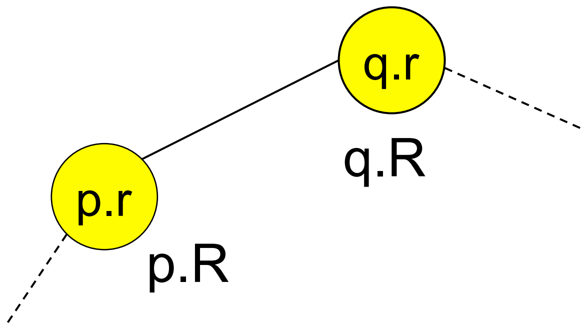
Logical Clock Synchronization

The lifting is compatible with the local ordering of Z_K

Lifted execution

$$\begin{tikzcd}
\delta_0 \arrow[r, "D_0"] \arrow[d] & \delta_1 \arrow[r, "D_1"] \arrow[d] & \delta_2 \arrow[r, "D_2"] \arrow[d] & \cdots \\
\gamma_0 \arrow[r, "D_0"] & \gamma_1 \arrow[r, "D_1"] & \gamma_2 \arrow[r, "D_2"] & \cdots
\end{tikzcd}$$

Execution

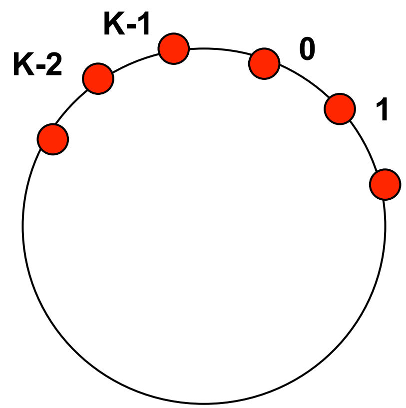


$$\begin{array}{c} p.R \leq q.R \\ \downarrow \\ p.r \leq_l q.r \end{array}$$

The lifting defines a global preorder over the nodes

Generalization over Z_K

- (Local) total order $\leq_l$ over $Z_K=\{0,1,2,\ldots,K\}$
- Let $M \geq 1$ and $K \geq 2M+1$



a $\leq_l$ b iff $0 \leq$ b-a mod K $\leq$ M

With M=2 and K=5, then for 1

$4 \leq_l 1$; $0 \leq_l 1$; $1 \leq_l 1$; $1 \leq_l 2$; $1 \leq_l 3$;

Complexities

- Space : O(1)
- Time, for one pulse:
  - Best Case: *O(1)*
  - Worst Case: *O(D)* ← [DELAY]

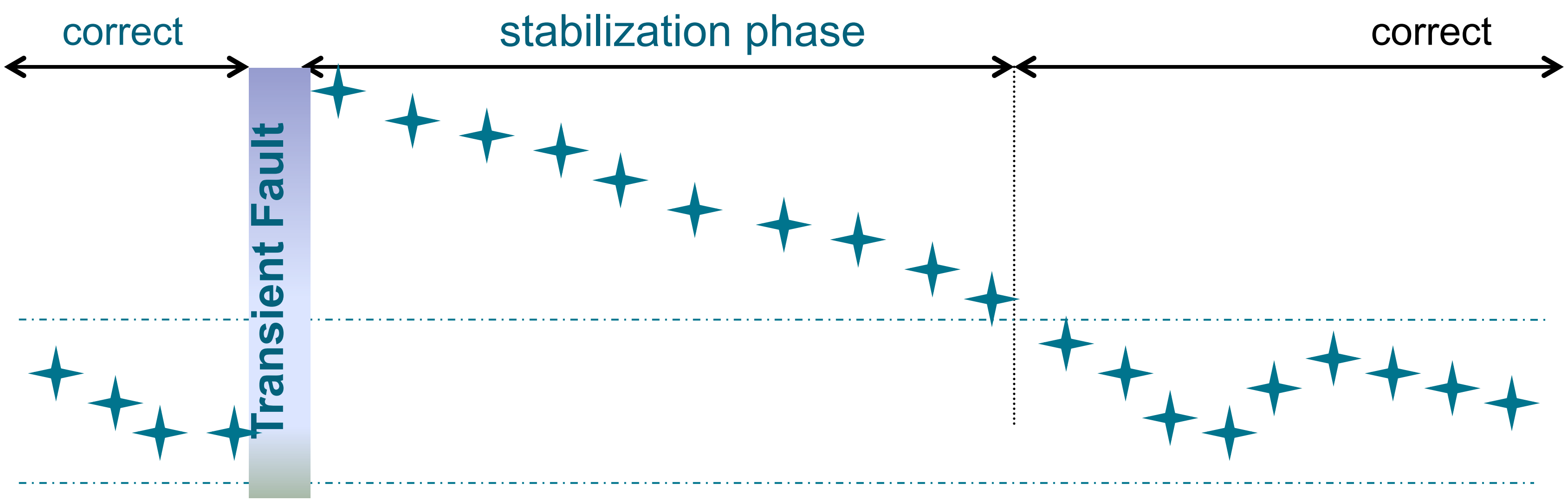
Fault Tolerance

Starting from an arbitrary configuration ?

Self-stabilization

A self-stabilizing system, regardless of its initial state, is guaranteed to converge to the intended behavior in finite time. [Dijkstra 74]

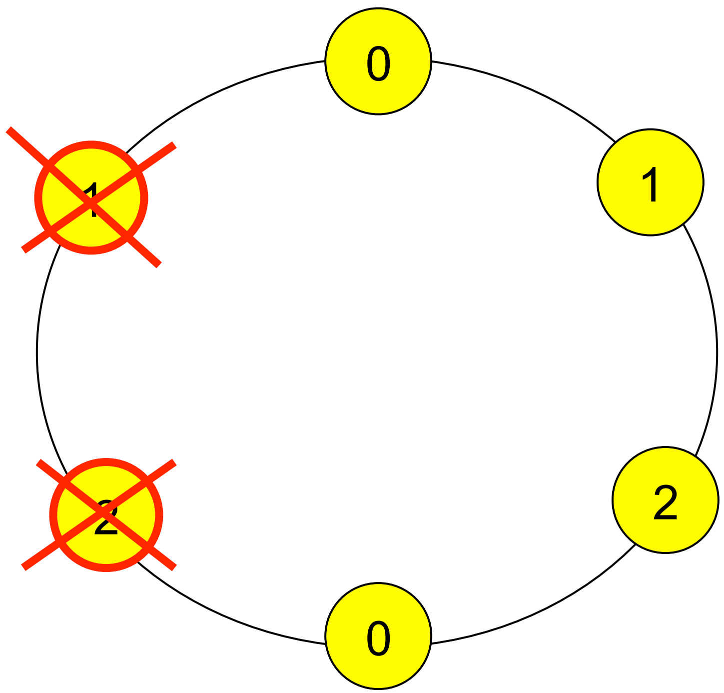
→ Transient Faults



After a fault or an arbitrary initialization

- K=3

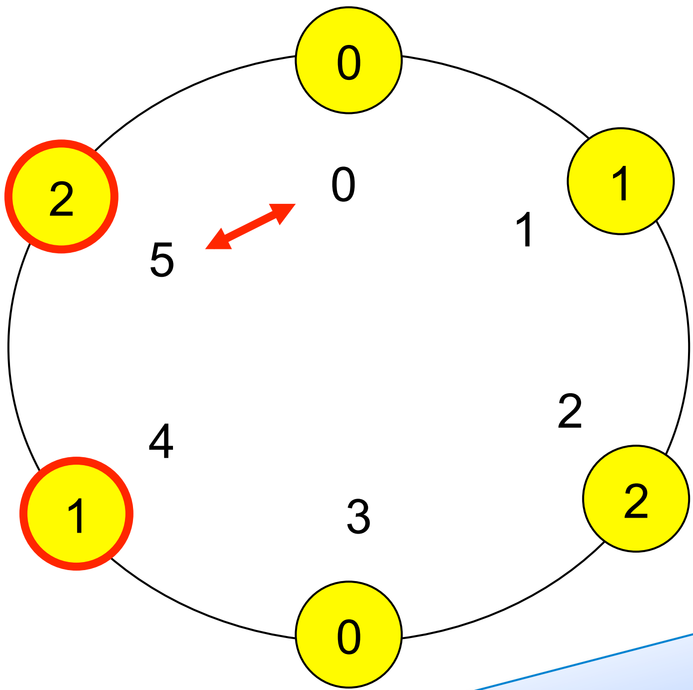
$$\begin{tikzcd}
\delta_0 \arrow[r, "D_0"] \arrow[d] & \delta_1 \arrow[r, "D_1"] \arrow[d] & \delta_2 \arrow[r, "D_2"] \arrow[d] & \ldots \\
\gamma_0 \arrow[r, "D_0"] & \gamma_1 \arrow[r, "D_1"] & \gamma_2 \arrow[r, "D_2"] & \ldots
\end{tikzcd}$$



After a fault or an arbitrary initialization

- K=3

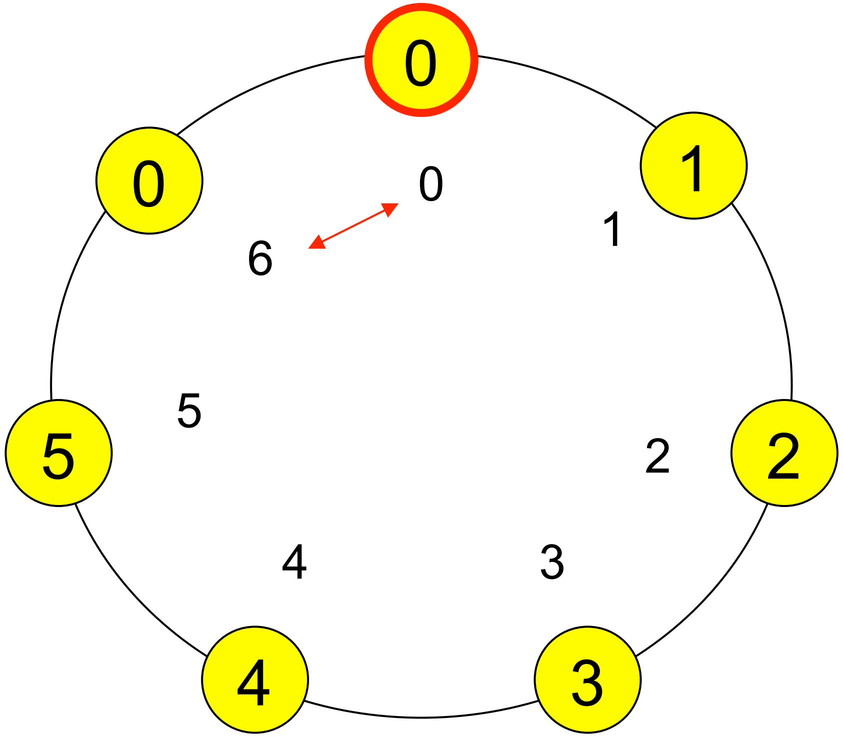
$$\begin{tikzcd} \delta_0 \arrow[r, "D_0"] \arrow[d] & \delta_1 \arrow[r, "D_1"] \arrow[d] & \delta_2 \arrow[r, "D_2"] \arrow[d] & \cdots \\ \gamma_0 \arrow[r, "D_0"] & \gamma_1 \arrow[r, "D_1"] & \gamma_2 \arrow[r, "D_2"] & \cdots \end{tikzcd}$$



DEADLOCK!

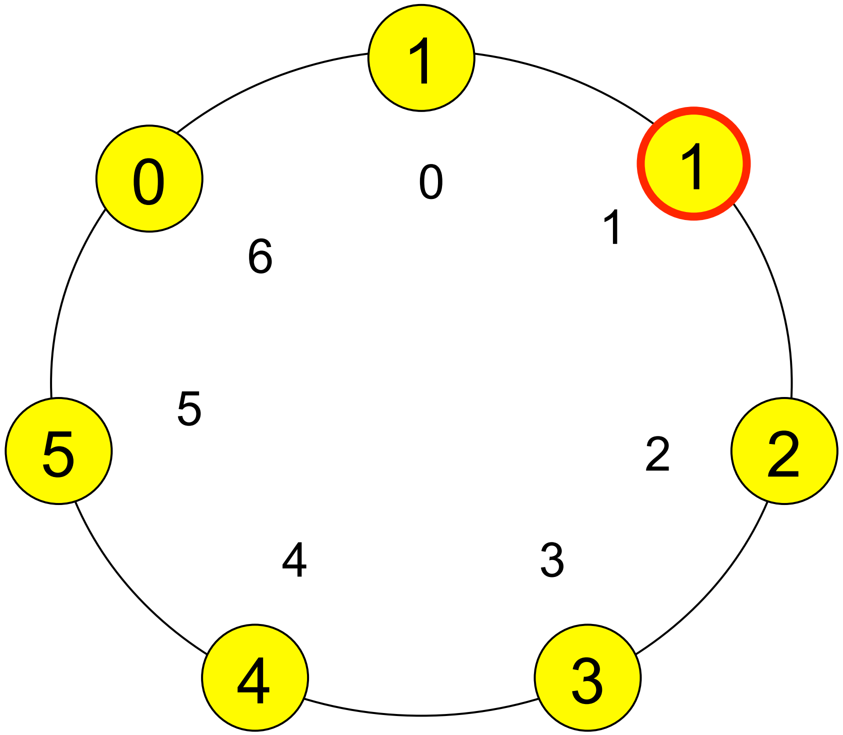
After a fault or an arbitrary initialization

- K=6



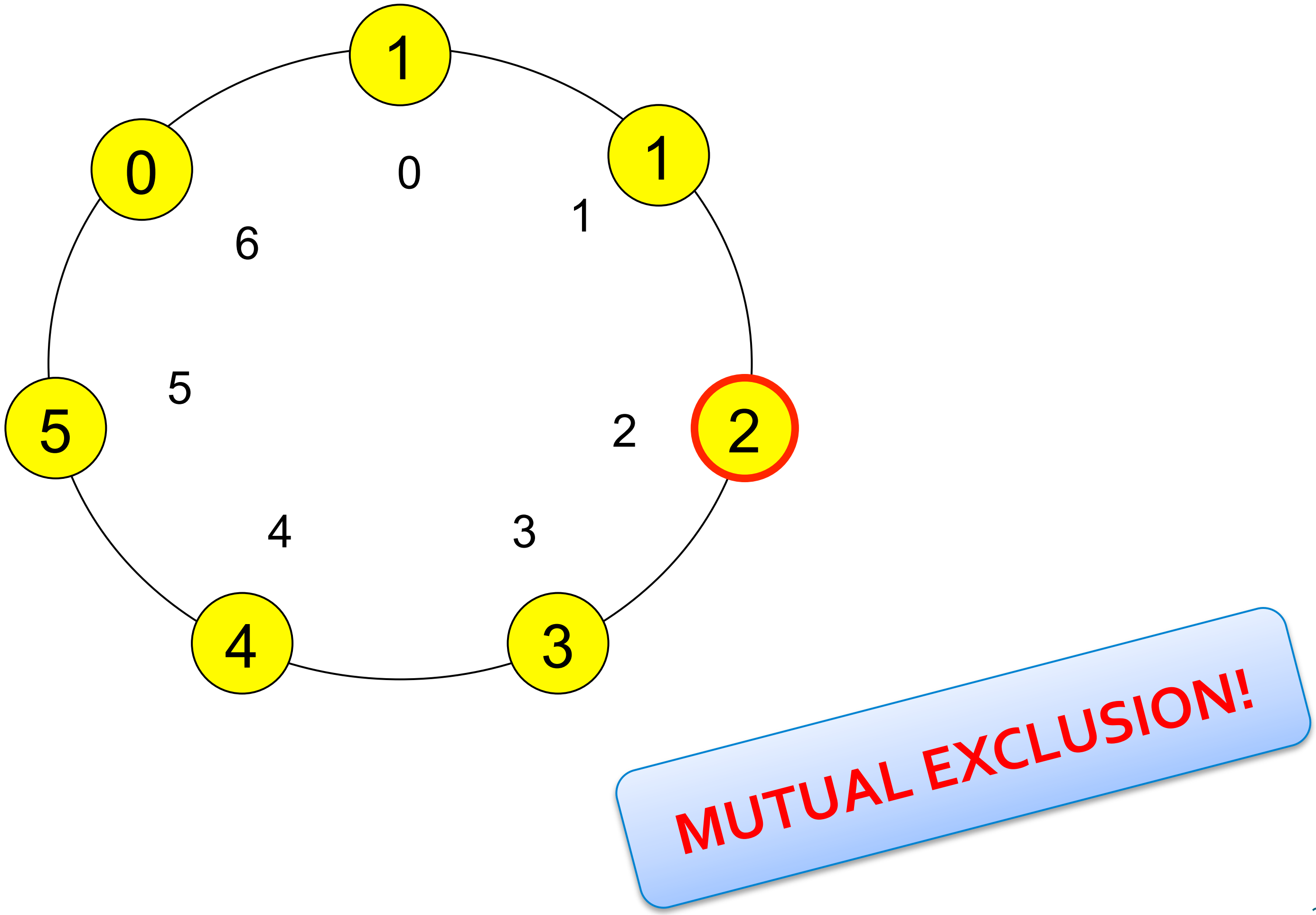
After a fault or an arbitrary initialization

- K=6



After a fault or an arbitrary initialization

- K=6



After a fault or an arbitrary initialization

What are the conditions to be able to (re)-synchronize the system ?

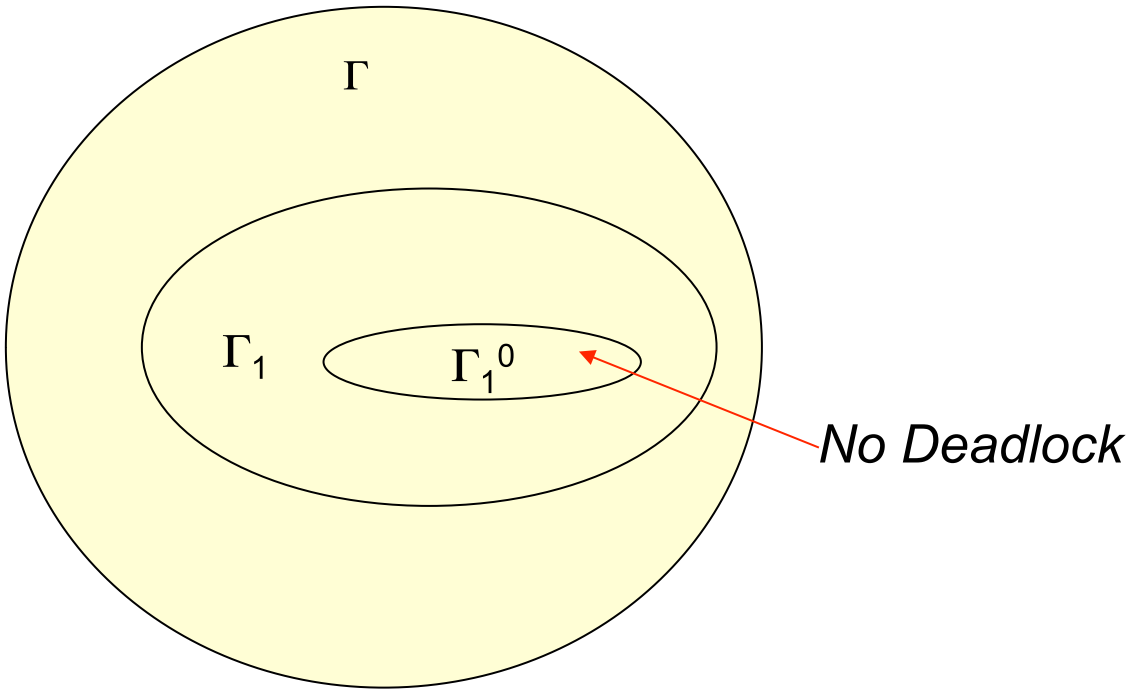
System Configurations

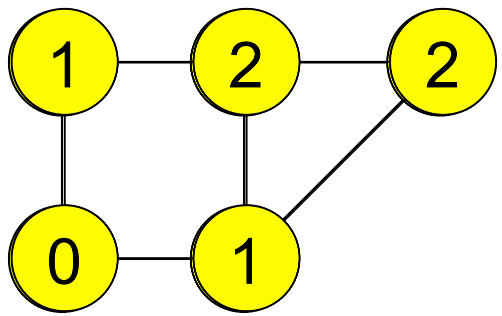
Γ : The register values are arbitrary

Γ_1 : The register of two neighboring process is less than or equal to 1

Γ^0_1 : There is no deadlock (there exists a compatible lifting to the configuration)

K=4





System Configurations

Γ : The register values are arbitrary

Γ_1 : The register of two neighboring process is less than or equal to 1

Γ^0_1 : There is no deadlock (there exists a compatible lifting to the configuration)

THEOREM:

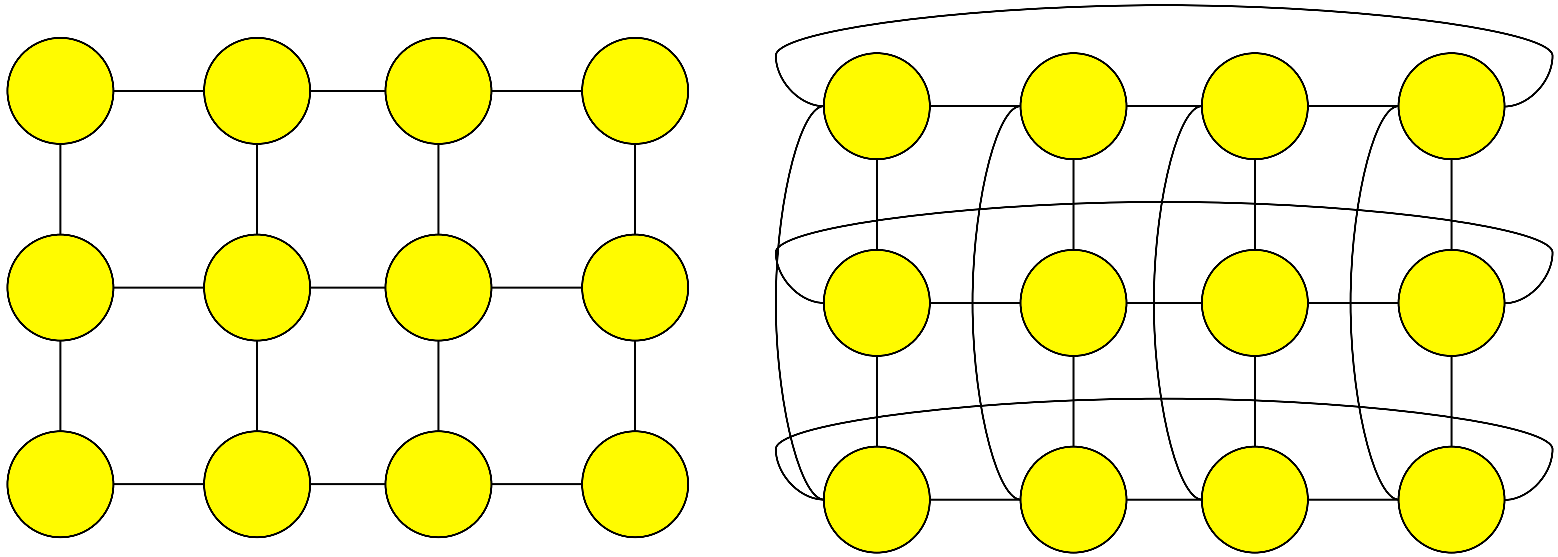
$$K > C_G \Longrightarrow \Gamma_1 = \Gamma_1^0$$

- C_G ($C_G \leq n$), the *cyclomatic characteristic* of the graph:
  Equal to the size of the greatest cycle in one of the cycle basis of G where the size of the greatest cycle is minimum (equal to 2 if G is acyclic)

[Boulinier, Petit, and Villain, PODC 2004]

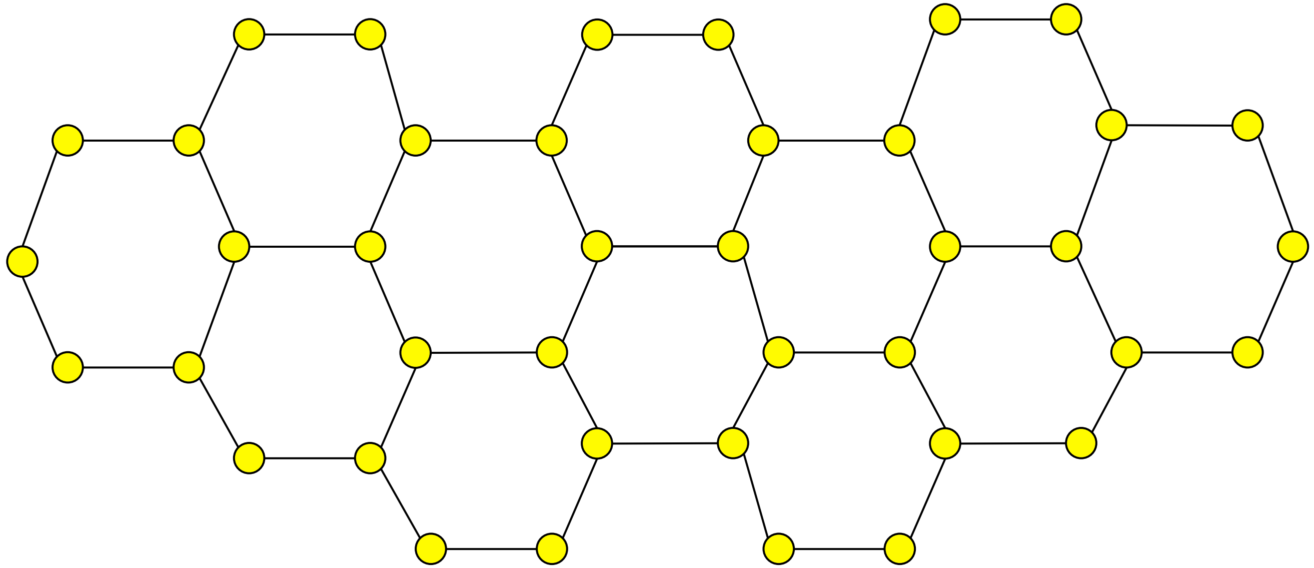
Cyclomatic characteristic of G

C_G=*4* in meches and tories



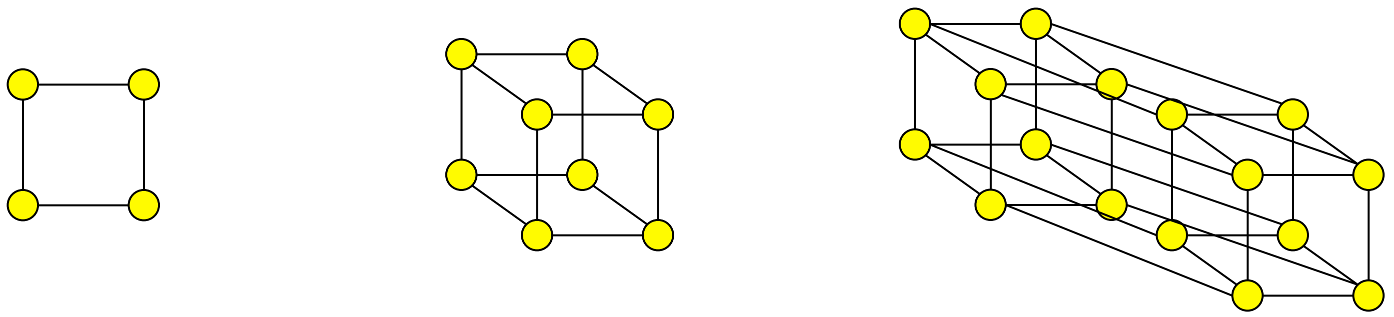
Cyclomatic characteristic of G

C_G=6 in honeycombs



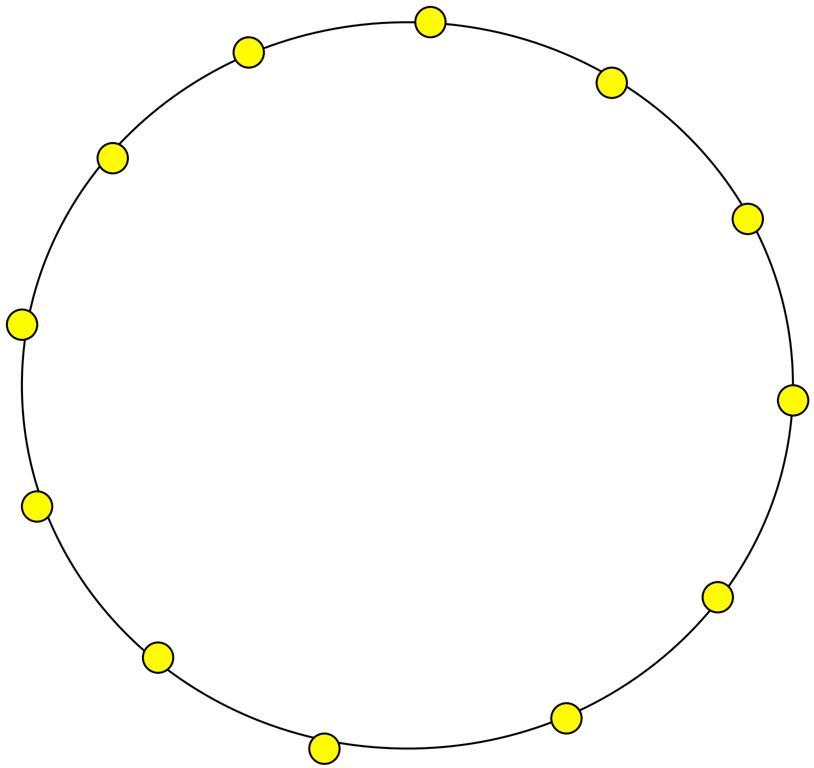
Cyclomatic characteristic of G

$C_G = 4$ in hypercubes



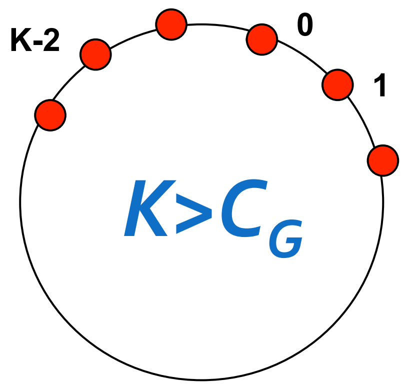
Cyclomatic characteristic of G

C_G=n on rings



Logical Clock Synchronization

- To avoid deadlock due arbitrary initial values, K must be greater than C_G ($C_G \le n$), the *cyclomatic characteristic* of the graph



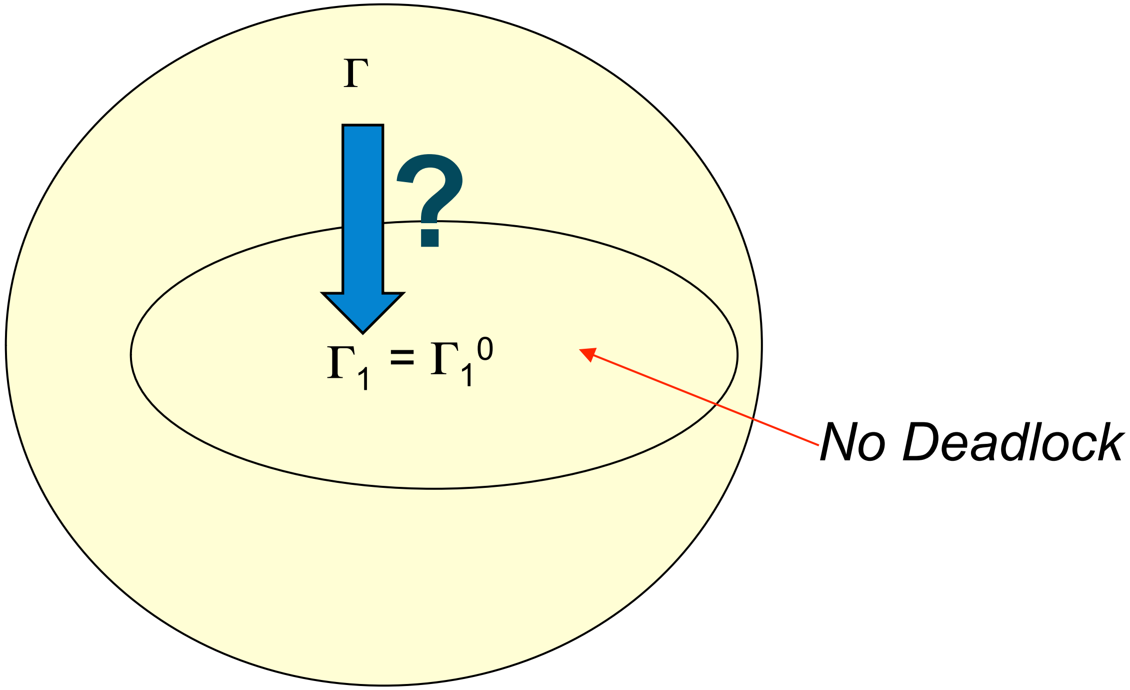
Values of Register r

System Configurations

Γ : The register values are arbitrary

Γ_1 : The register of two neighboring process is less than or equal to 1

Γ^0_1 : There is no deadlock (there exists a compatible lifting to the configuration)



Stabilization

- Global Reset (Stabilizing PIF), implies a root or IDs
- Local Reset (also woks in anonymous networks)

QUESTION:
*What is the motivation behind **anonymity**?*

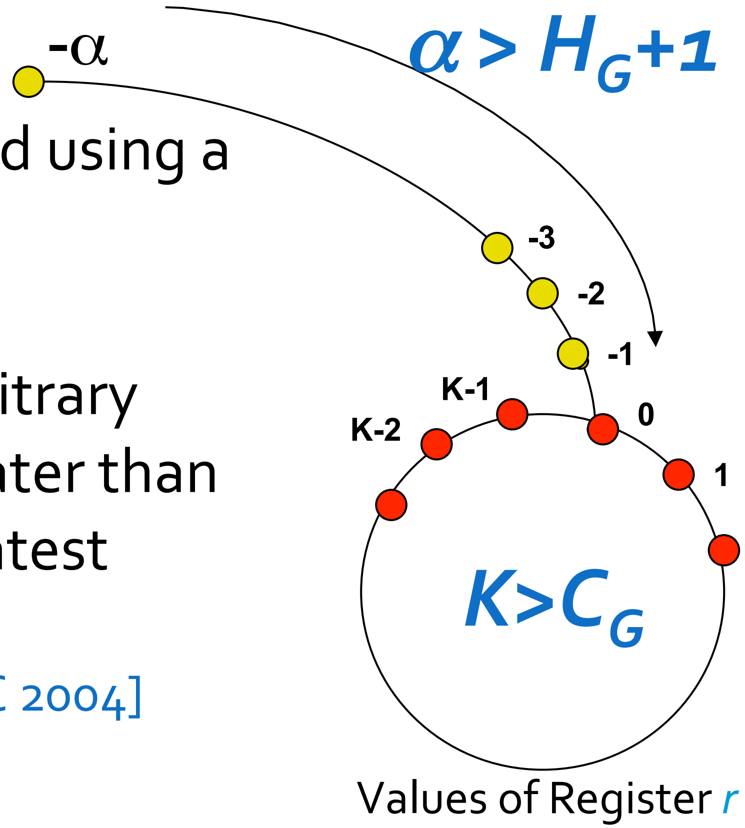
"Only a few amount of bits allows to distinguish a huge number of nodes!"

Advantages of Anonymous Solutions

- Lack of (underlying) infrastructure
  - No unique identifier assignation or no central process
  - No maintenance of any distributed structure
- Economic advantages
- User privacy preserved
  [Delporte-Gallet, Fauconnier, Guerraoui, and Ruppert, OPODIS 2007]
- No one-to-one routing
- Very suitable for sensor networks

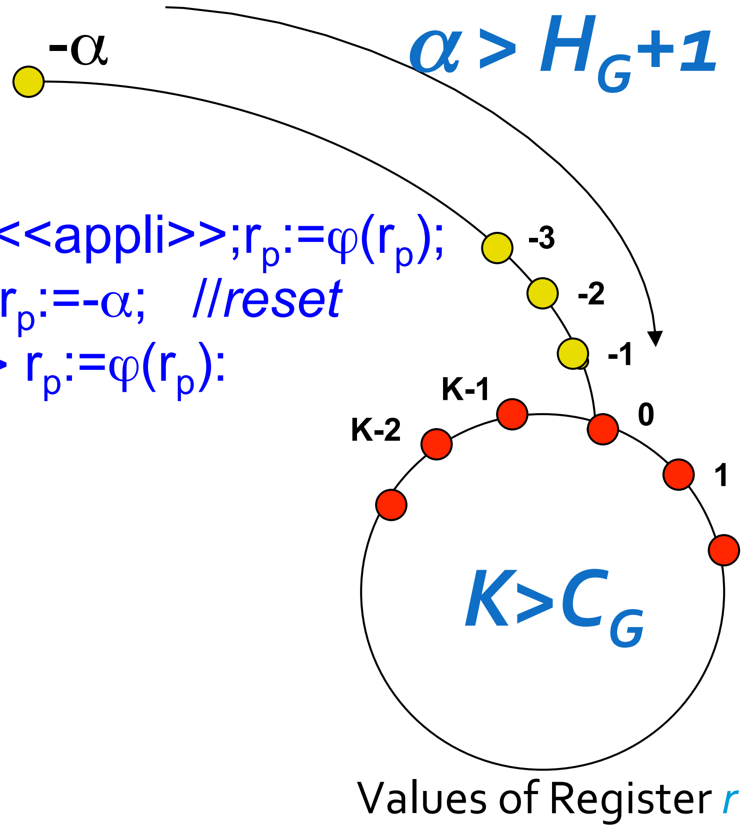
Self-Stabilizing Logical Clock Synchronization

- To avoid starvations due arbitrary initial values, K must be greater than C_G ($C_G \leq n$), the *cyclomatic characteristic* of the graph



- Safety eventually guaranteed using a reset mechanism:
  Register r is set in $[-\alpha,\ldots,0]$
- To avoid starvations due arbitrary initial values, α must be greater than than H_G+1 ($H_G \leq n$), the greatest chordless cycle of the graph
  [Boulinier, Petit, and Villain, PODC 2004]

Self-Stabilizing Logical Clock Synchronization



SSSync

AN: $\forall q \in N_p$: $Correct_p(q) \wedge (NormalStep_p) \rightarrow$ <<appli>>;$r_p := \varphi(r_p)$;

AR: $\neg\ (\forall q \in N_p: Correct_p(q)\ ) \wedge (r_p \notin tail\varphi) \rightarrow r_p := -\alpha$; *//reset*

AC: $r_p \in init\varphi^* \wedge (\ \forall q \in N_p: r_q \in init\varphi^* \wedge r_p \leq \varphi(r_q)) \rightarrow r_p := \varphi(r_p)$:

Asynchronous, Anonymous Logical Clock, Related Works

	# of states	Stabilizing Time
Gouda, Couvreur, Francez, 1992	$O(n^2)$	$O(nd)$
Dolev, 2000	$O(n^2)$	$O(d)$
○ SSSync(K,α,M)		
$\alpha=0$, $K>M.C_G$, $M>H_G-2$	$O(nd)$	$O(nd)$
$\alpha>H_G-2$, $M=1$, $K>C_G$	$O(n)$	$O(n)$
Tree networks	$O(1)$	$O(d)$