Leader Election

INFORMATIQUE
Sciences
Université Paris Cité

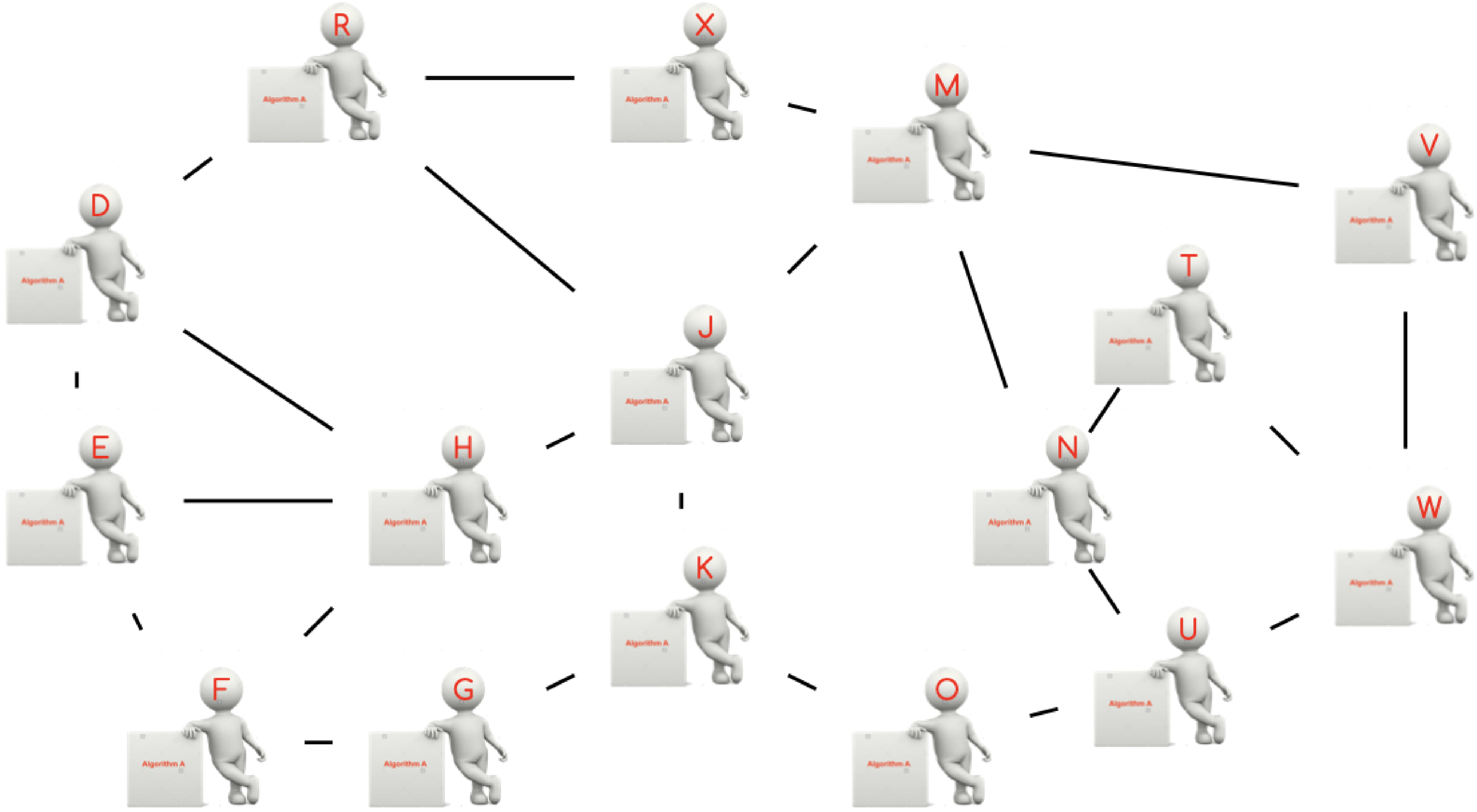
MPRI

Lélia Blin

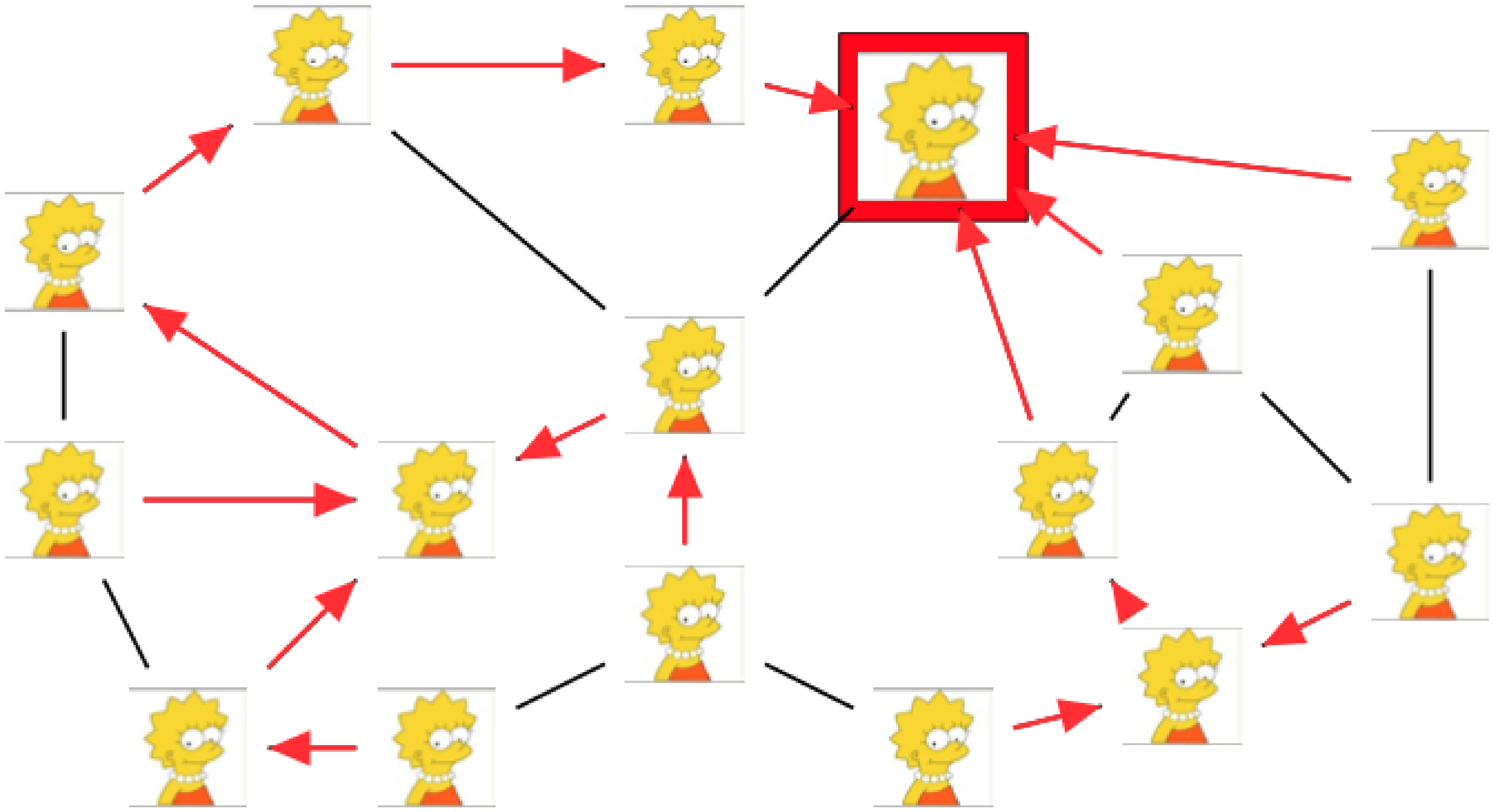
lelia.blin@irif.fr
2023

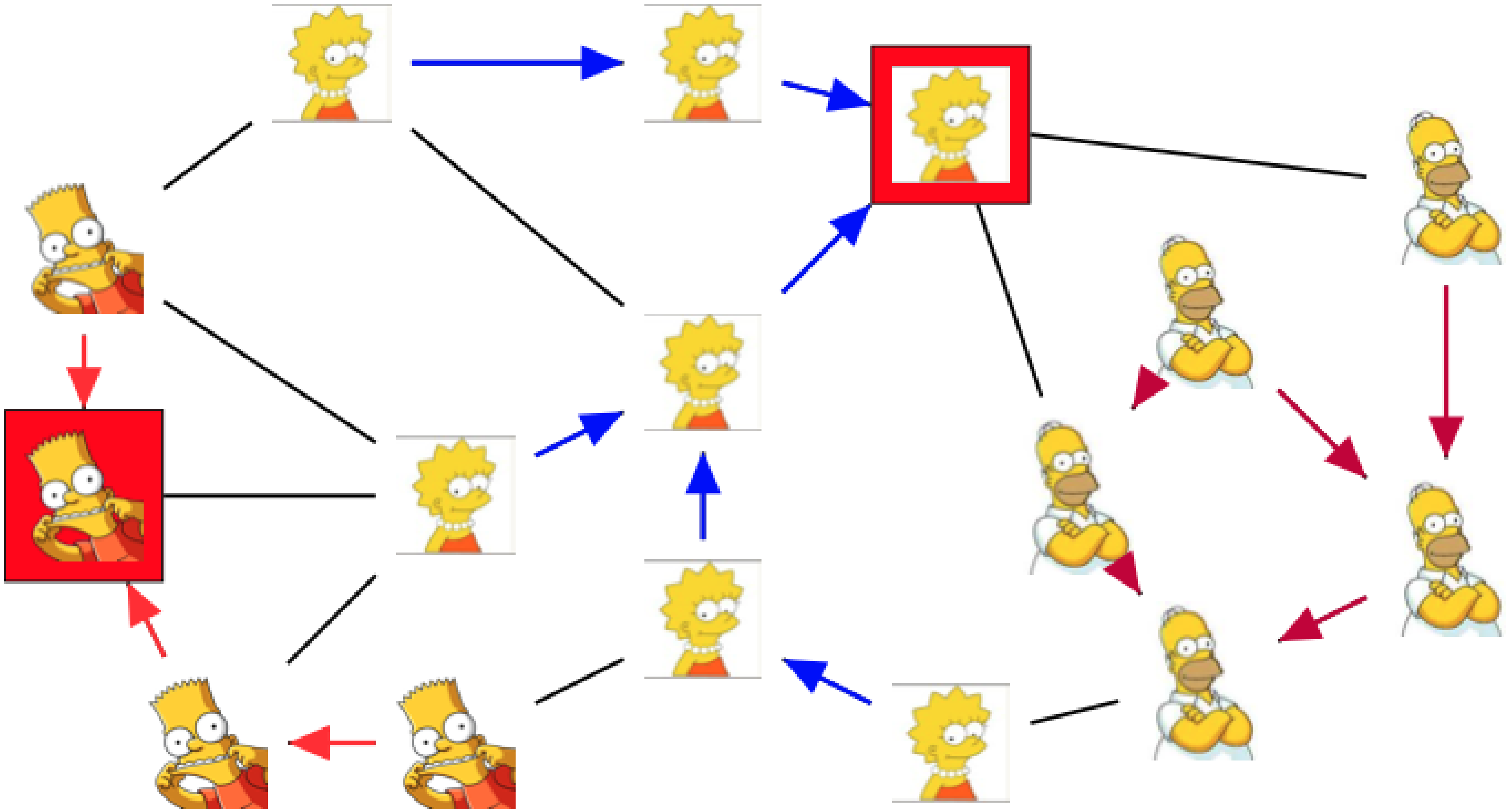
Leader Election (LE)

leader election is the process of designating a single node as the organizer of some task distributed among several nodes.


R
X
M
V
D
J
T
E
H
N
W
K
U
F
G
O

Lélia Blin



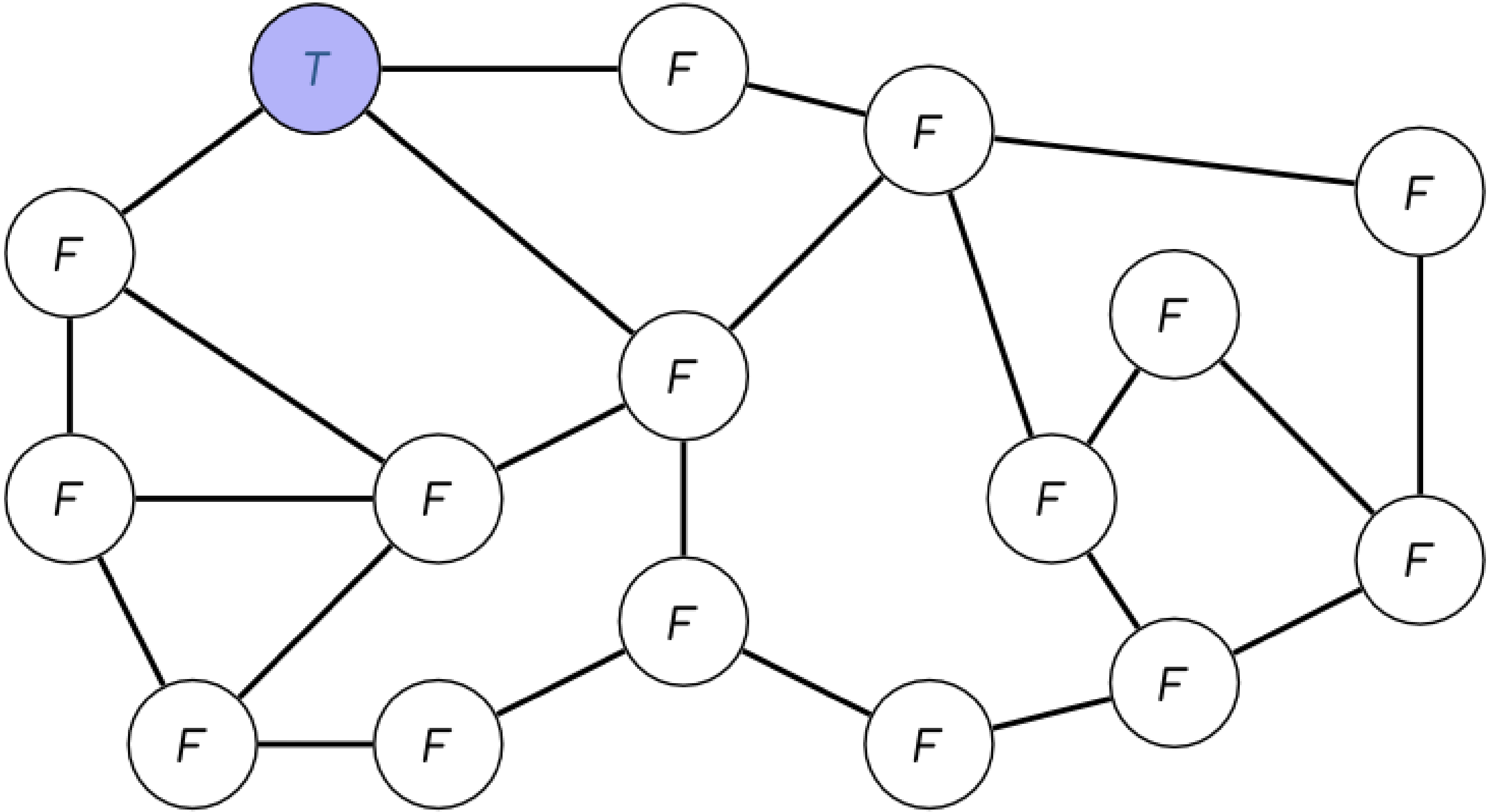


Specifications problem

Small memory specification

Definition problem: The leader election specification sequence consists in a single specification configuration where a unique node maps to $\ell_v = true$, and every other node $u \neq v$ maps to $\ell_u = false$.

<mark style="background: #FF5582A6;">Remark</mark>:
- Nobody knows the identifier of the elected node
- The size of the memory required for satisfying the election is $O(1)$ bits per nodes

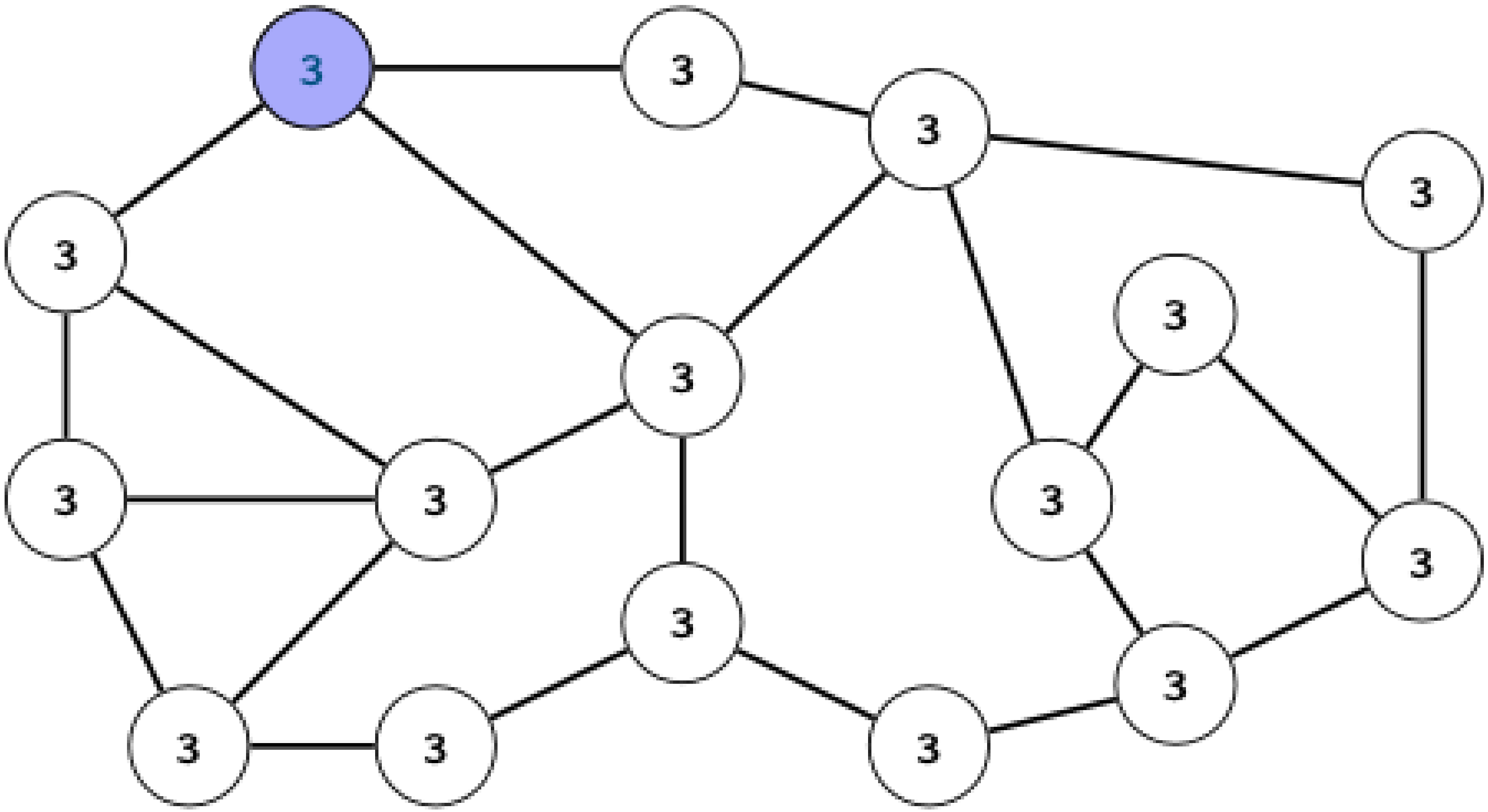

T
F
F
F
F
F
F
F
F
F
F
F
F
F
F
F
F

Best knowledge specification

> **Definition problem**: The leader election specification sequence consists in a single specification configuration where a unique node maps to $\ell_v = Id_v$, and every other node $u \neq v$ maps to $\ell_u = Id_v$.

<mark style="background: #FF5582A6;">Remark</mark>:

- Everybody knows the identifier of the elected node
- The size of the memory required for satisfying the election is $O(\log n)$ bits per nodes


3
3
3
3
3
3
3
3
3
3
3
3
3
3
3
3
3

Impossibility result

LE is not possible in anonymous regular network

State of art : silent

Article	Scheduler	Knowledge	Rounds	Steps	Memory
AG90	weaklyfair	N	$O(N)$	?	$\Theta(\log N)$
DH97	Fair	N	$O(D)$	?	$O(NlogN)$
DLP10	Unfair		$O(n)$	?	unbounded
DLV11X2	Unfair		$O(n)$	?	$\Theta(\log n)$
KK13	Synchrone		$O(D)$	?	$\Theta(\log n)$
ACDDP17	Unfair		$O(n)$	$O(n^3)$	$\theta(\log n)$

- <mark style="background: #FFF3A3A6;">AG90</mark> A. Arora and M. Gouda, Distributed Reset (Extended Abstract): 10th Conference on Foundations of Software Technology and theoretical Computer Science (FSTTCS) 1990.
- <mark style="background: #FFF3A3A6;">DH97</mark> S. Dolev, T. Herman, Superstabilizing protocols for dynamic distributed systems, Chic. J. Theor. Comput. Sci. (1997).
- <mark style="background: #FFF3A3A6;">DLP10</mark> A.K. Datta, L.L. Larmore, H. Piniganti, Self-stabilizing leader election in dynamic networks, in: Stabilization, Safety, and Security of Distributed Systems –12th International Symposium, SSS, 2010, pp. 35–49.

- <mark style="background: #FFF3A3A6;">DLV11X2</mark> A.K. Datta, L.L. Larmore, P. Vemula, Self-stabilizing leader election in optimal space under an arbitrary scheduler, Theor. Comput. Sci. 412 (40) (2011) 5541–5561.
  A.K. Datta, L.L. Larmore, P. Vemula, An O(n)-time self-stabilizing leader election algorithm, J. Parallel Distrib. Comput. 71 (11) (2011) 1532–1544.
- <mark style="background: #FFF3A3A6;">KK13</mark> A. Kravchik, S. Kutten, Time optimal synchronous self stabilizing spanning tree, in: DISC, 2013, pp. 91–105.
- <mark style="background: #FFF3A3A6;">ACDDP17</mark> - Karine Altisen, Alain Cournier, Stéphane Devismes, Anaïs Durand, Franck Petit: Self-stabilizing leader election in polynomial steps. Inf. Comput. 254: 330-366 (2017)

1990

A. Arora and M. Gouda, Distributed Reset (Extended Abstract):

10th Conference on Foundations of Software Technology and theoretical Computer Science (FSTTCS) 1990.

Variables

- r_v identifier of the root (the leader)
- p_v identifier of the parent
- d_v distance from the root

Algorithm

$R_{root} : (r_v < v) \vee (p_v = v \wedge (r_v \neq v \vee d_v \neq 0) \vee (p_v \notin N(v) \cup \{v\} \vee d_v \geq n)$

$\longrightarrow r_v = v, p_v = v, d_v = 0;$

$R_{correct} : p_v \in N(v) \wedge d_v < n \wedge (r_v \neq r_{p_v} \vee d_v \neq d_{p_v} + 1)$

$\longrightarrow r_v = r_{p_v}, d_v = d_{p_v} + 1;$

$R_{parent} : (\exists u \in N(v) | r_v < r_{p_v} \wedge d_v < n) \vee (\exists u \in N(v) | r_v = r_{p_v} \wedge d_{p_v} + 1 < d_v)]$

$\longrightarrow r_v = r_j, p_v = j, d_v = d_j + 1;$

Convergence

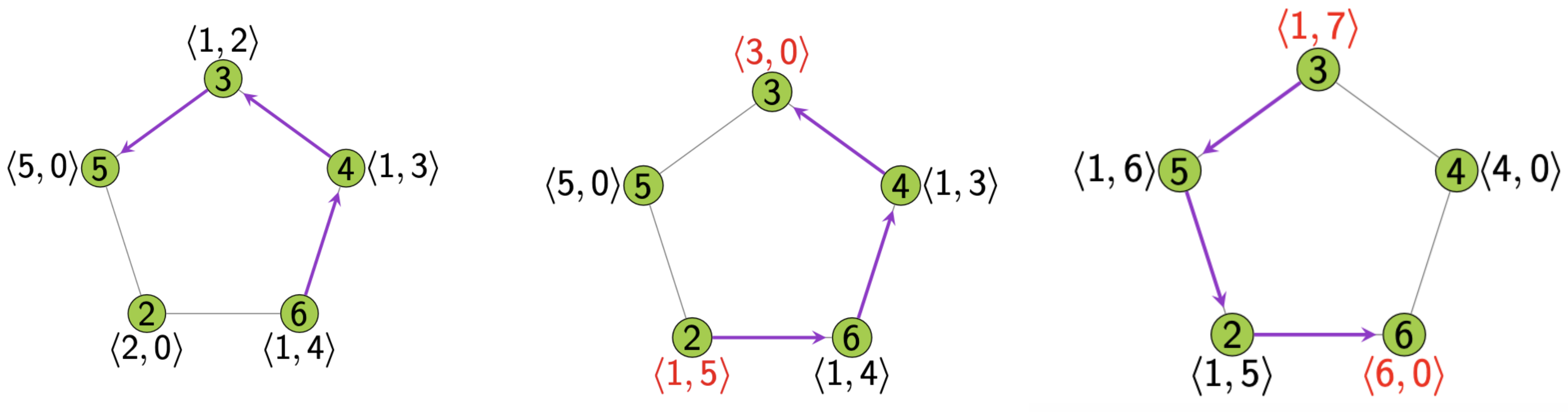
Silent or not ?

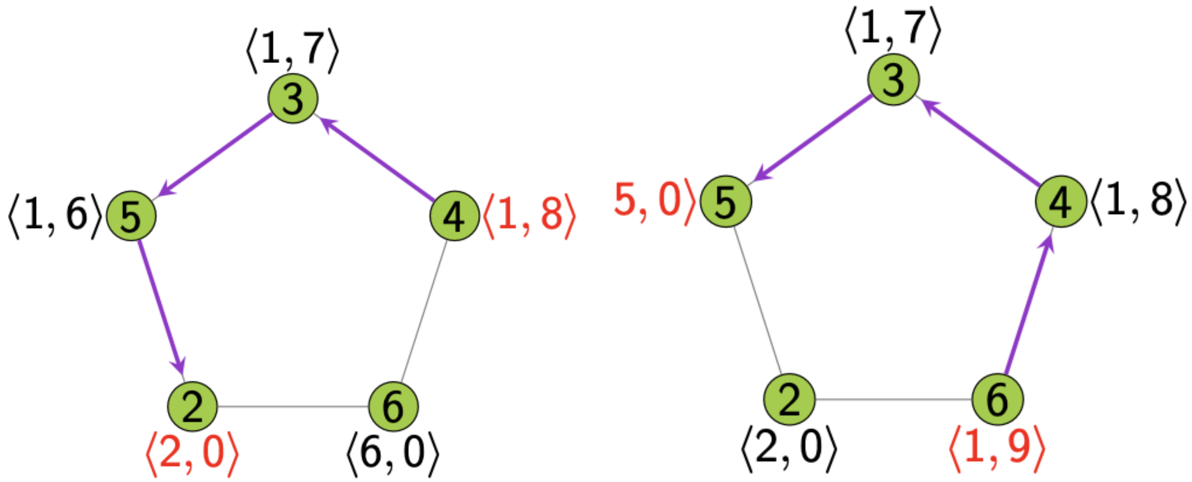
K. Altisen, A. Cournier, S. Devismes, A. Durand, F. Petit,

Self-stabilizing leader election in polynomial steps.

Inf. Comput. 254, 330-366 (2017)

Time Complexity : Analyse




⟨1, 7⟩
3
⟨1, 6⟩
5
4
⟨1, 8⟩
2
6
⟨2, 0⟩
⟨6, 0⟩
⟨1, 7⟩
3
5, 0⟩
5
4
⟨1, 8⟩
2
6
⟨2, 0⟩
⟨1, 9⟩

Solution: Freeze before remove

- additional variable : $State_v \in \{C, EB, EF\}$
  - C means “not involved in a tree removal”
    - Only process of status C can join a tree and
    - only by choosing a process of status C as parent
  - EB: Error Broadcast
  - EF: Error Feedback

Leader Election (Non Silent)

Goal: Sub-logartithmic memory size

Non silent LE

Art.	Topology	P/D	Model	Sched.	Memory
MOOY92	Ring	Proba.	Msg		$O(1)$
AO94	Ring	Proba.	link regist.		$O(\log^* n)$/ed
IL94	Ring	Proba.	State*		$O(1)$
ILS95	Ring**	Deter	State		$O(1)$
BGJ99	Ring++	Deter	State		$O(1)$
BT18	Ring	Deter.	State	unfair	$O(\log \log n)$
BT20	Graph	Deter.	State	unfair	$O(\log \log n + \log \Delta)$

$*$ Augmented State Model ** Size of the ring is prime

++ n-node rings are bounded from above by $n + k$, where k is a small constant.

- <mark style="background: #FFF3A3A6;">MOOY92 </mark> A.J. Mayer, Y. Ofek, R.I Ostrovsky, M. Yung, Self-stabilizing symme- try breaking in constant-space (extended abstract), in: STOC, 1992, pp. 667–678.
- <mark>IL94</mark> G. Itkis, L.A. Levin, Fast and lean self-stabilizing asynchronous protocols, in: FOCS, IEEE Computer Society, 1994, pp. 226–239.
- <mark>ILJ95</mark> G. Itkis, C. Lin, J. Simon, Deterministic, constant space, self-stabilizing leader election on uniform rings, in: WDAG, in: LNCS, Springer, 1995, pp. 288–302.

- <mark style="background: #FFF3A3A6;">BGJ99</mark> J. Beauquier, M. Gradinariu, C. Johnen, Memory space requirements for self-stabilizing leader election protocols, in: Proceedings of PODC 1999, 1999, pp. 199–208.
- L.Blin, S.Tixeuil, Compact deterministic self-stabilizing leader election on a ring: the exponential advantage of being talkative, Distrib. Comput. 31 (2) (2018) 139–166.
- L. Blin, S. Tixeuil, Compact self-stabilizing leader election for general networks. J. Parallel Distributed Comput. 144: 278-294 (2020)

Leader Election Bounds

ILS95: There exists a self-stabilizing leader election algorithm using $O(1)$ bits of memory per node in rings whose size is a prime number.

BGJ99: Any self-stabilizing leader election algorithm requires $\omega(1)$ bits of memory per node in rings whose size is not a prime number.

DGS99: Any silent self-stabilizing leader election algorithm requires at least $\Omega(\log n)$ bits of memory per node.

BT20: There exists a self-stabilizing leader election algorithm in any graph, using O(log log n + log Δ) bits of memory per node.

BFL23: The leader election problem requires Ω(log log n) bits per node

- <mark style="background: #FFF3A3A6;">BFL23</mark> Lélia Blin, Laurent Feuilloley, Gabriel Le Bouder: Optimal Space Lower Bound for Deterministic Self-Stabilizing Leader Election Algorithms.Discret. Math. Theor. Comput. Sci. 25 (2023)

Compact memory [BT18]

- Compact identifier: decomposition bit per bit of the identifier
- Breaking symmetry : Identifier
- $Bit_v(i)$ function:
  - returns the position of the i-th most significant bit equal to 1 in Id_v
  - Exemple: 10→1010

$$\mathtt{Bit}_v(i) := \begin{cases} 4 & \textit{if} \quad i = 1 \\ 2 & \textit{if} \quad i = 2 \\ -1 & \textit{if} \quad i > 2 \end{cases}$$

Bitwise Comparison between identifiers

- u=10 (1010) and v=12 (1100)
- $Bit_u(1) = Bit_v(1) = 4$
- $Bit_u(2) = 2 < Bit_v(2) = 3$
- The identifier of u is greater than that of v

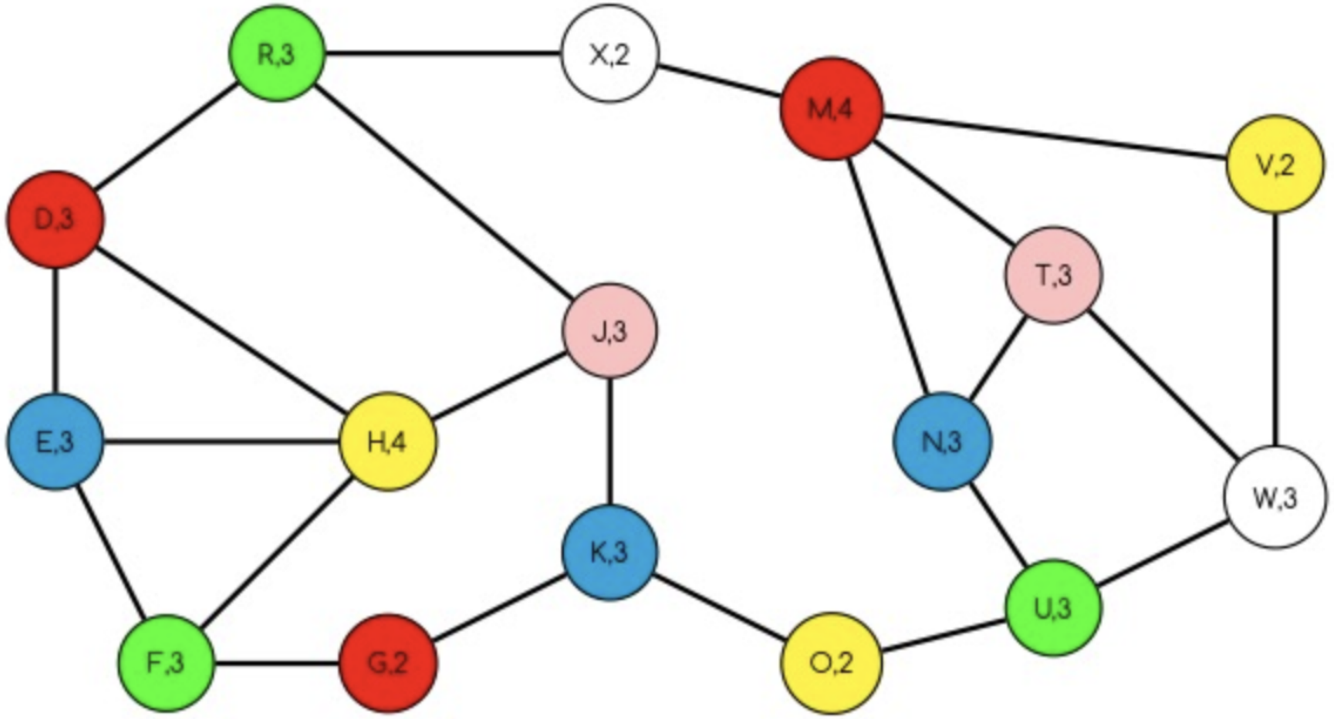
Main ingredients of BT20

1. Pointers: silent self-stabilizing distance-2 coloring
2. Breaking cycle: silent self-stabilizing cycle detection.
3. Fake root: silent self-stabilizing cycle and illegitimate sub spanning tree destruction [10, 13]
4. Rooted spanning tree: talkative self-stabilizing spanning tree-construction.

Lélia Blin

Pointers: silent self-stabilizing distance-2 coloring :

$O(\log \log n + \log \Delta)$



Breaking cycle

Using the uniqueness of the identifiers

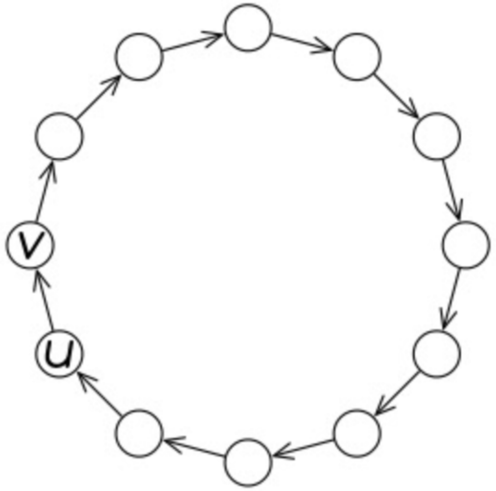


Fig. a

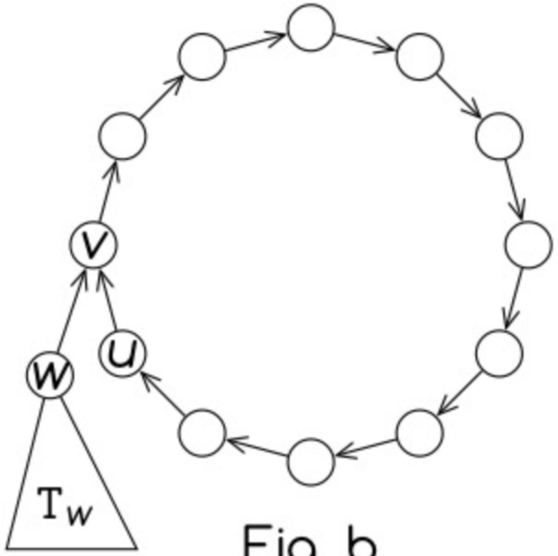


Fig. b

Silent self-stabilizing cycle and illegitimate sub spanning tree destruction

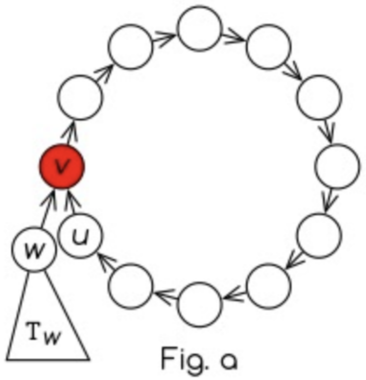


Fig. a

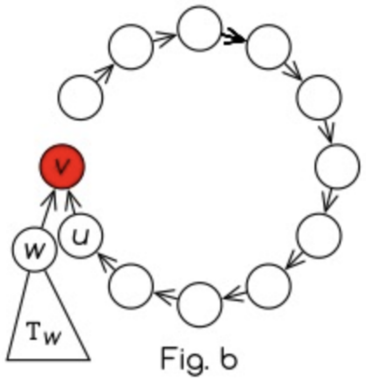


Fig. b

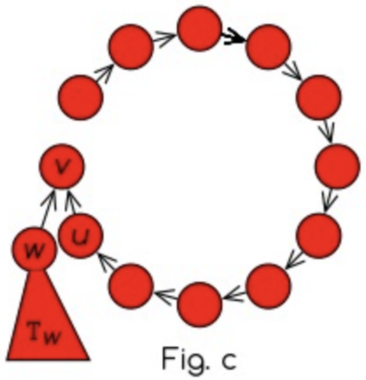
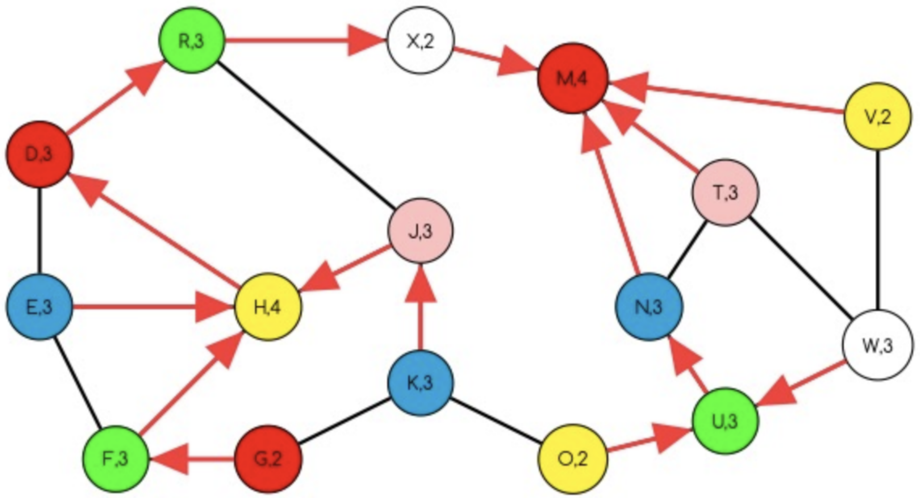


Fig. c

Rooted Spanning tree

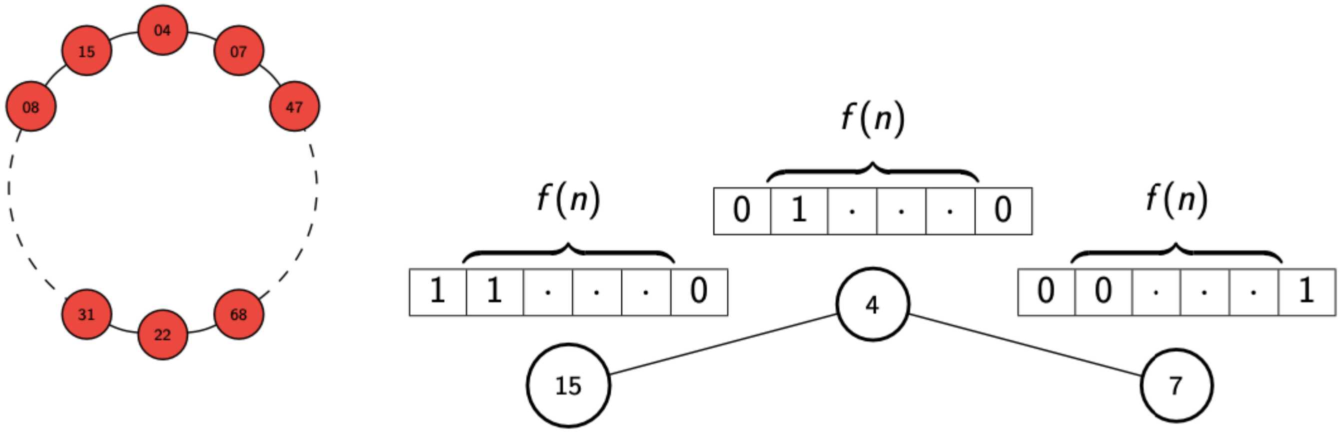
Leader: Node with the maximum degree, the maximum color, the maximum identifier.



Memory Lower bound

BFL23: The leader election problem requires $\Omega(\log \log n)$ bits per node

Main Idea of BLF23



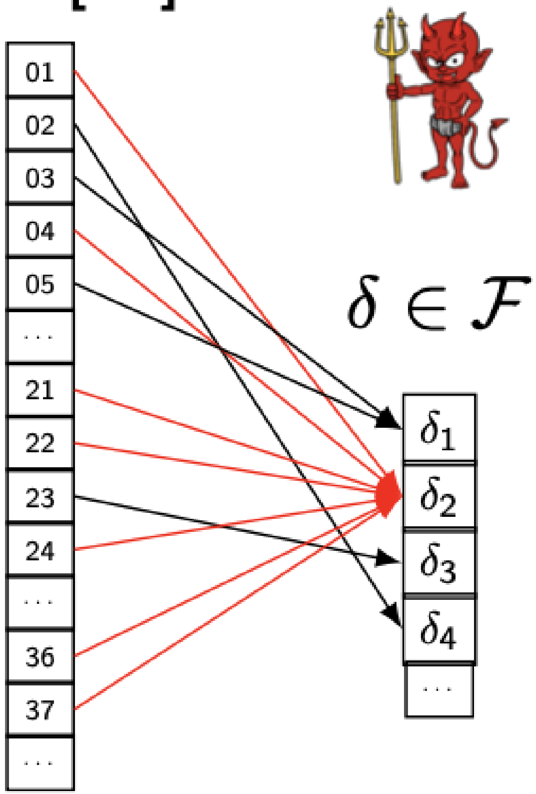
Idea

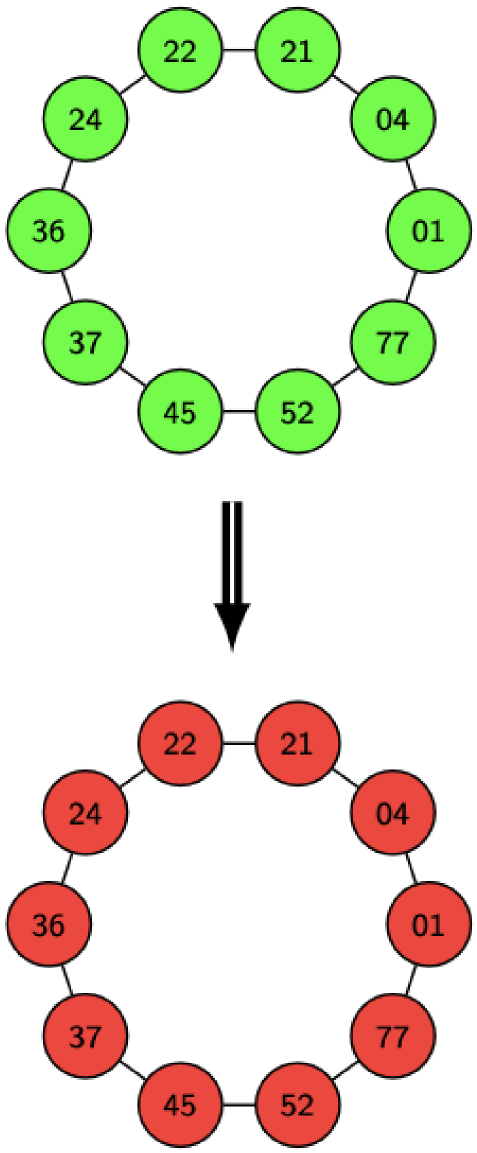
- Algorithme $\mathcal{A} : [n^c] \times \{0,1\}^{3f(n)} \to \{0,1\}^{f(n)}$
- $\forall id \in [1, n^c], \exists \delta_{id} : \{0,1\}^{3f(n)} \to \{0,1\}^{f(n)}$
- $|\mathcal{F}| = |\{\delta_{id}\}| = 2^{f(n)2^{3f(n)}}$

$$f(n) \in o(\log \log n)$$

$$ID \in [n^c]$$

$$\delta \in \mathcal{F}$$





Challenges

Silent LE: Improve the number of steps

LE: Achieve the lower bound memory for graph with a non contant degree.

Open question : It is possible to achieved LE without constructing a spanning tree?