MPRI Course 2.18.1
Distributed algorithms on networks

Pierre Fraigniaud

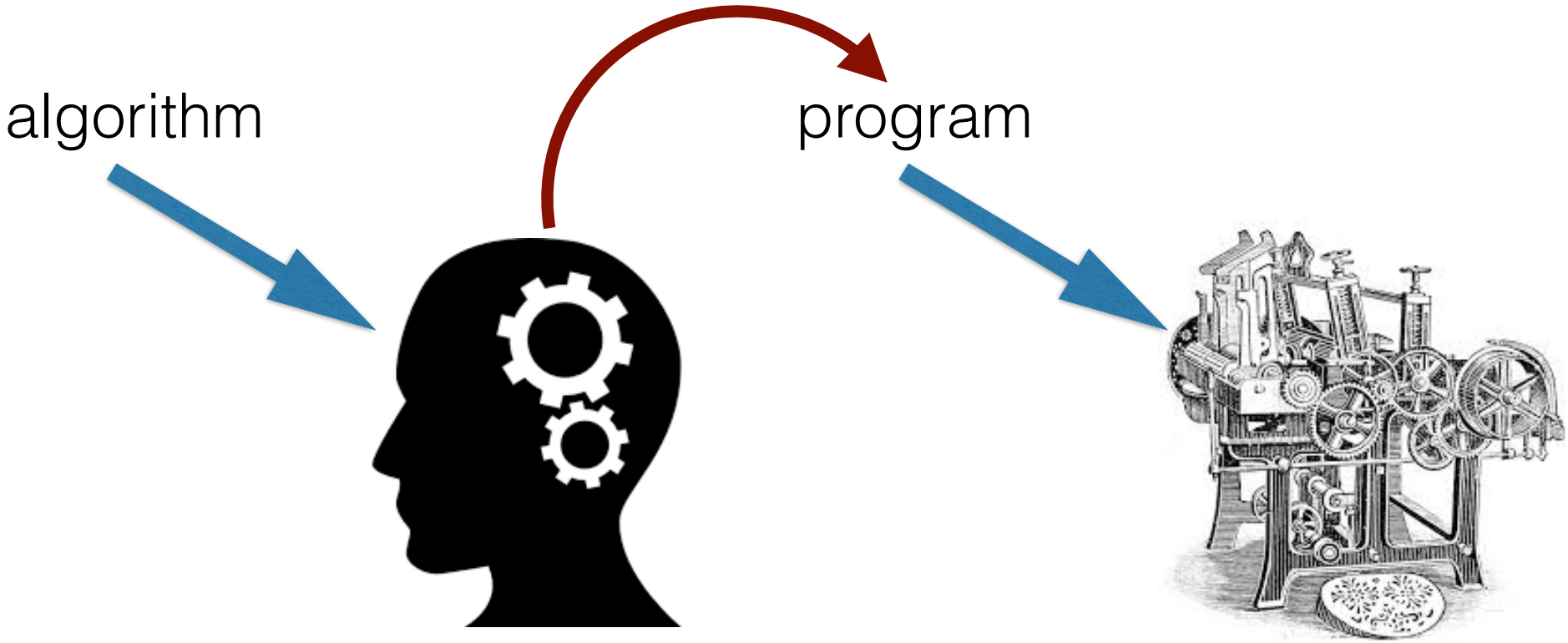
INSTITUT
DE RECHERCHE
EN INFORMATIQUE
FONDAMENTALE

https://www.irif.fr/~pierref

MPRI, Université Paris Cité, 2023-2024

Algorithms vs. programs

Mechanical procedures for solving a given problem

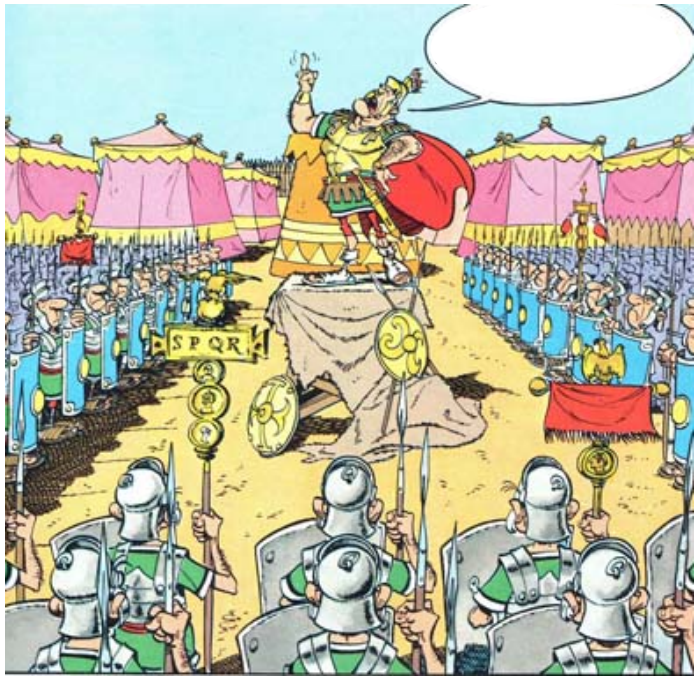


Distributed Algorithm

A collection of autonomous computing entities
collaborating for solving a task
in absence of any coordinator

Parallel vs. Distributed

Parallel computing



Performances

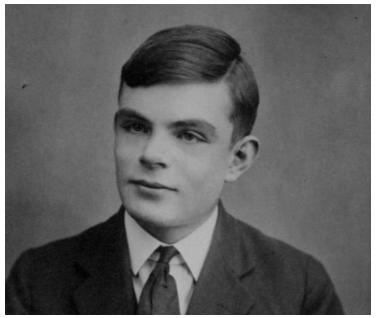
> petaFLOPS (10^{15} op./s)

Distributed computing

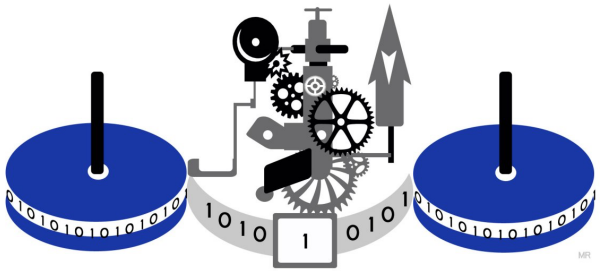
Coping with uncertainty

temporal and spatial

Sequential vs. Distributed

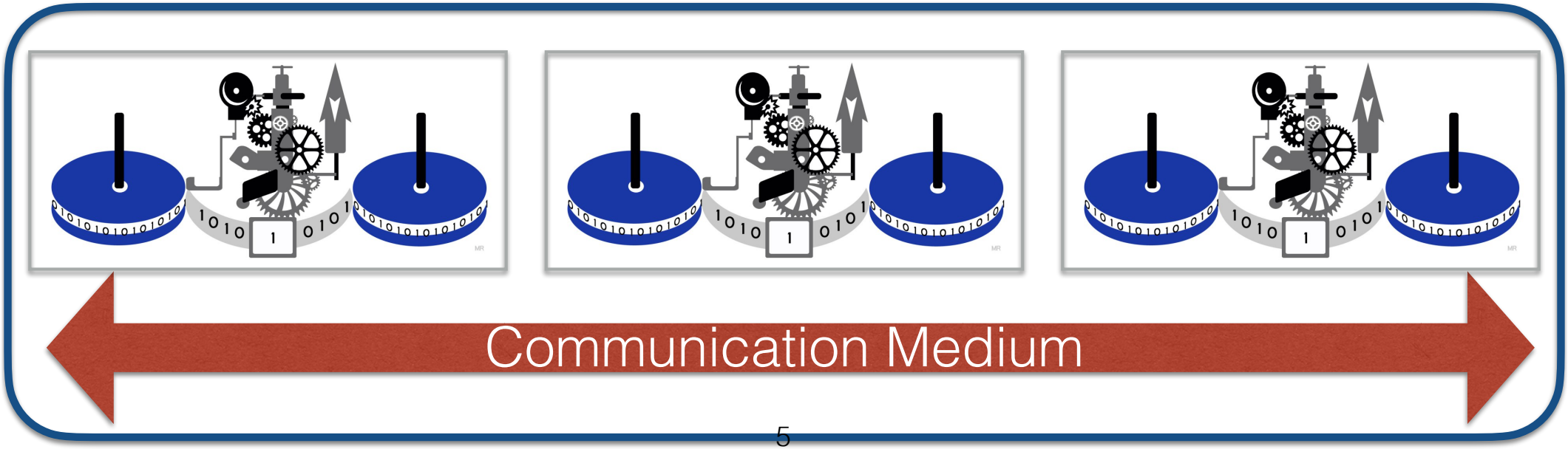


Alan Turing

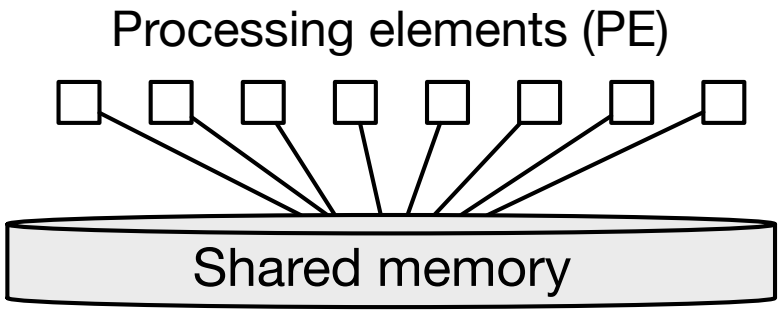


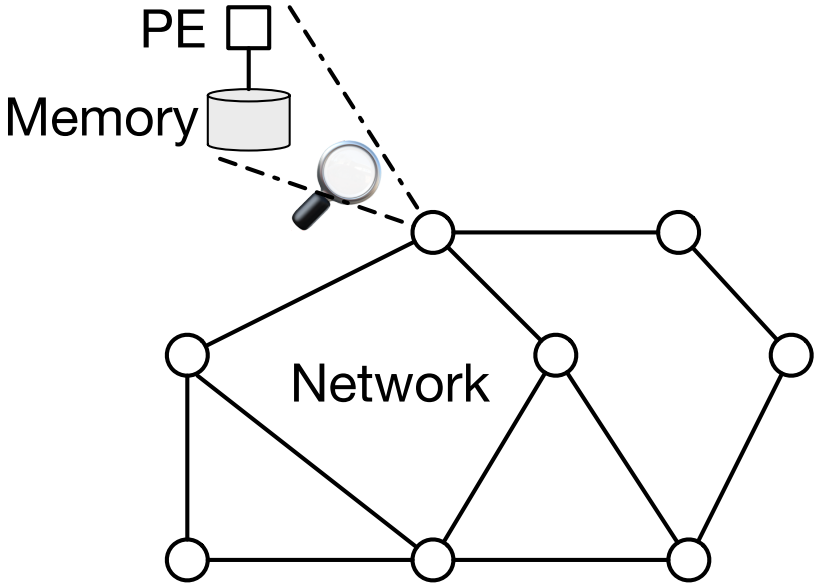
Alonzo Church

Typical model for distributed computing



Examples of Communication Media





Limitations Faced by Distributed Computing: Undecidability  + Uncertainty 

Sources of uncertainties:

- Spatial: communication network
- Temporal: clock drifts (asynchrony, load, etc.)
- Failures (transient, crash, malicious, etc.)
- Selfish behavior (game theory)
- …

Several Turing machines are weaker than one!

2.18.x Courses

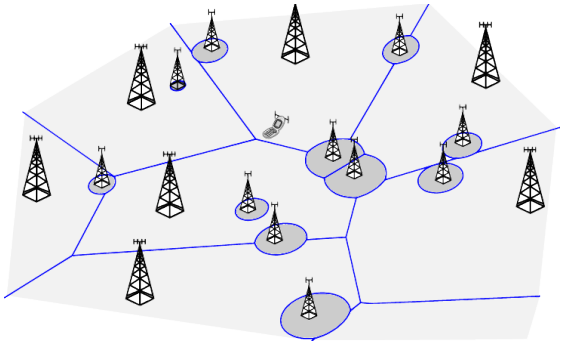
- 2.18.1 Distributed algorithms on networks
  - ✓ Spatial issues: locality

- 2.18.2 Distributed algorithms on shared memory
  - ✓ Temporal issues: asynchrony and failures

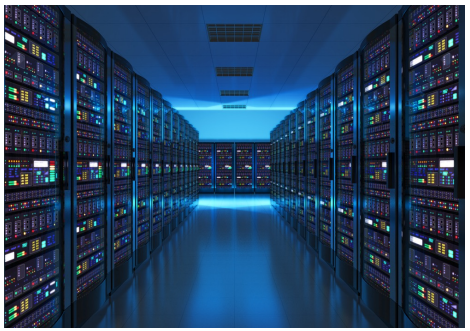
Symmetry Breaking

- Leader election
- Consensus
- Coloring
- Graph problems
- Etc.

Applications :



Frequency assignments



Distributed data-bases consistency

2.18.1 Course (2023-24)

1. Pierre Fraigniaud *(CNRS and Université Paris Cité)*

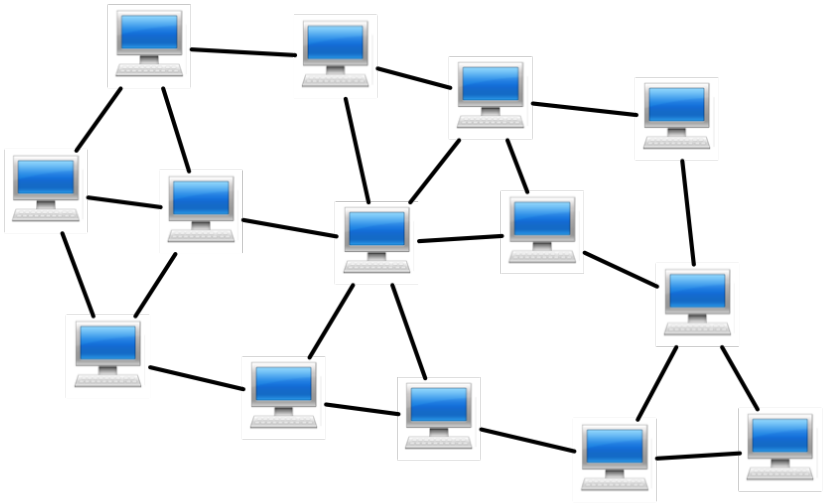
2. Mikaël Rabie *(Université Paris Cité)*

Learning objectives

- Models of distributed computing
- Algorithm design and analysis
- Computability and computational complexity

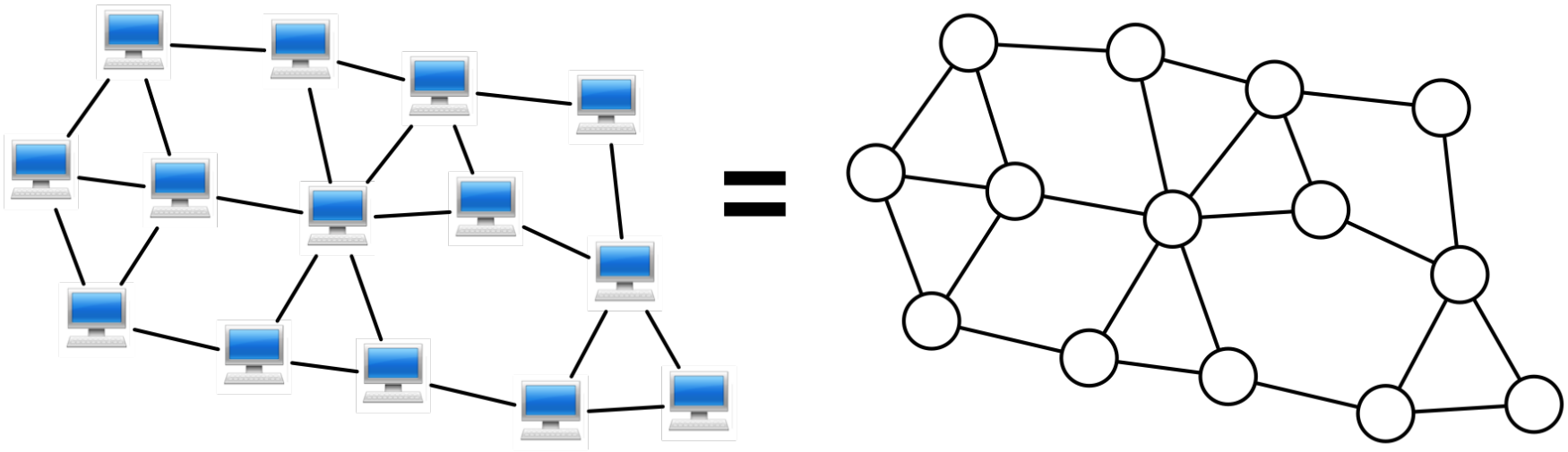
Distributed Algorithms

Identical computers in an unknown network, all running the same algorithm



Distributed Algorithms

Focus on graph problems:
network topology = input graph



Prerequisites

Basic knowledge in:

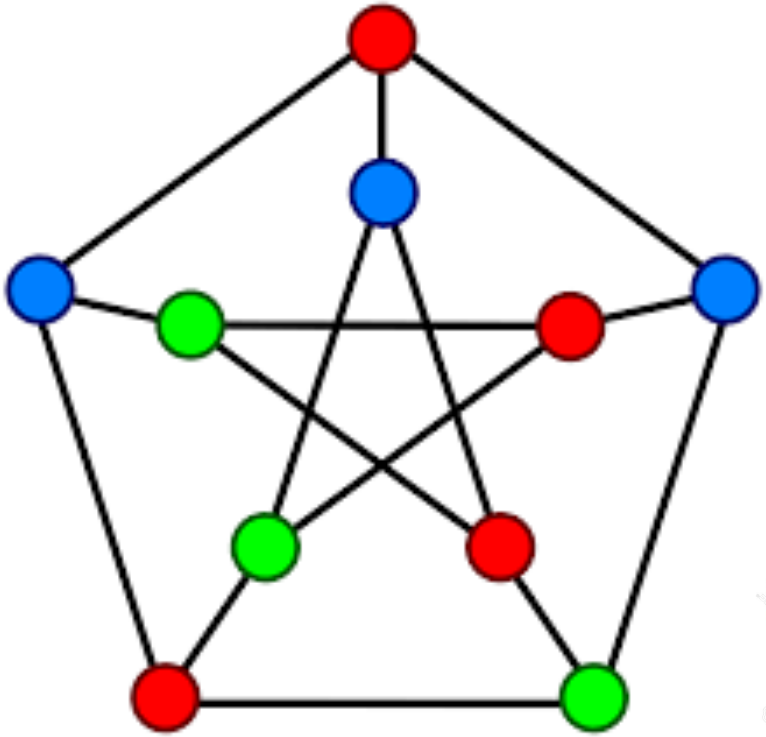
- Graph theory
- (Sequential) algorithm design and analysis
- Elements of probability theory

Warm Up

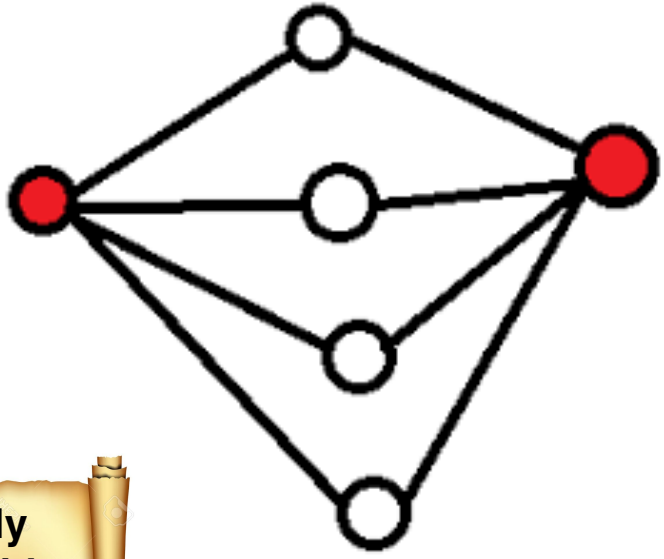
- Deterministic 3-coloring the n-node cycle
- Randomized 3-coloring the n-node cycle

Basic Graph Problems

Vertex coloring

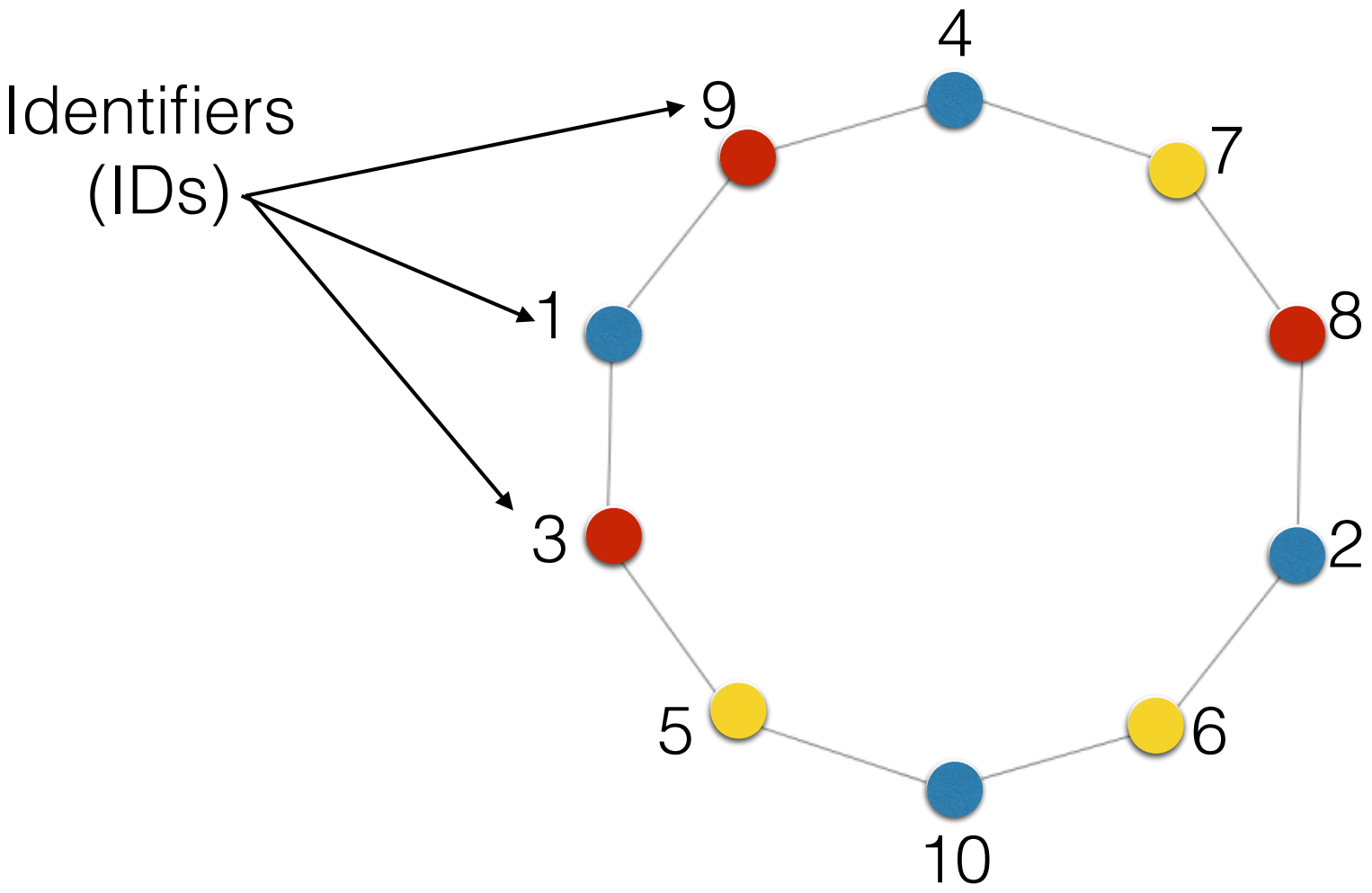


Independent set



Locally Checkable Labeling (LCL)

3-coloring the n-node cycle C_n



Color Reduction
(assuming IDs in $\{1,\ldots,n\}$)

- MyColor ← MyID
- For $i = n$ down to 4 do
  - Send MyColor to neighbors
  - Receive colors from neighbors
  - if Mycolor $= i$ then pick smallest color in $\{1,2,3\}$ distinct from colors of neighbors

Complexity: $O(n)$ rounds

Color Reduction
(assuming arbitrary IDs)

- MyColor ← MyID
- Repeat
    - Send MyColor to neighbors
    - Receive colors from neighbors
    - if Mycolor is locally maximum then
        - pick smallest color in $\{1,2,3\}$ distinct from colors of neighbors
        - send MyColor to neighbors and terminate

Complexity: $O(n)$ rounds

Logarithms

- Unless specified otherwise, all logarithms in these lectures are in base 2

$$\text{For every } x > 0,\ \log_2 x = \frac{\ln x}{\ln 2}$$

- k-bit binary strings encode integers between 0 and $2^k - 1$ (or between a and $a + 2^k - 1$ for any $a \in \mathbb{Z}$)

- For every $n > 0$, encoding all integers in $[0,n-1]$ uses $\lceil \log_2 n \rceil$ bits.

- Encoding $n > 0$ values uses $\lceil \log_2 n \rceil$ bits.

A Super Fast Algorithms

Theorem (Cole and Vishkin, 1986) There exists a distributed algorithm for 3-coloring C_n performing in $O(\log^\star n)$ rounds.

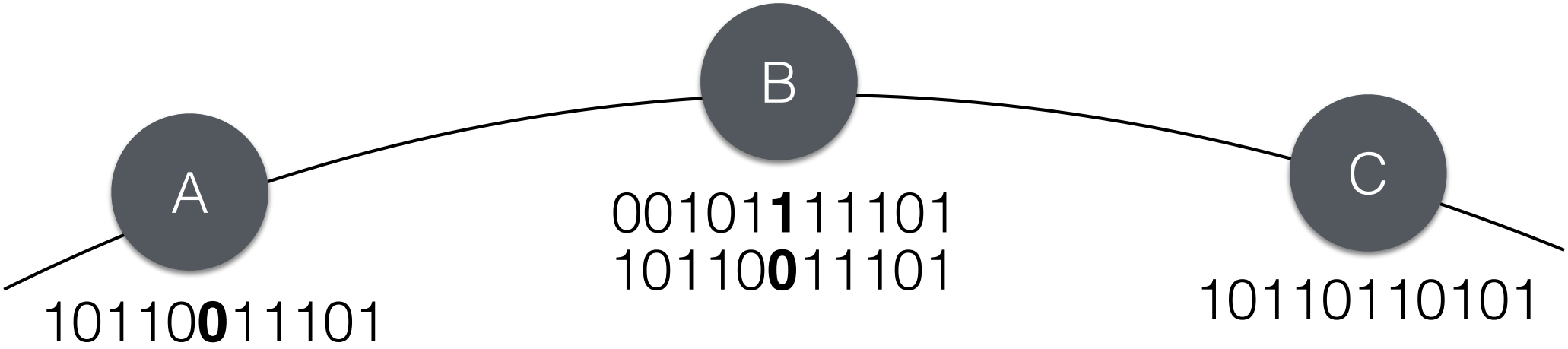
Iterated logarithms:

- $\log^{(0)} x = x$ $\quad\quad$ $\log^{(k+1)} x = \log \log^{(k)} x$
- $\log^* x$ = smallest k such that $\log^{(k)} x < 1$
- $\log^* 10^{100} = 5$

Cole-Vishkin Algorithm

Initial color = ID
Express colors in binary

Assume: n is known, and
consistent sens of direction



new = (position,bit) = (5,1) = 1011

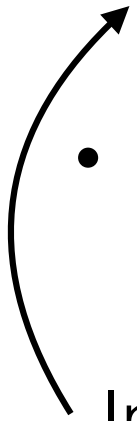
(p',b') (p,b)

p ≠ p' ⇒ proper coloring

p = p' ⇒ b ≠ b' ⇒ proper coloring

Number of iterations

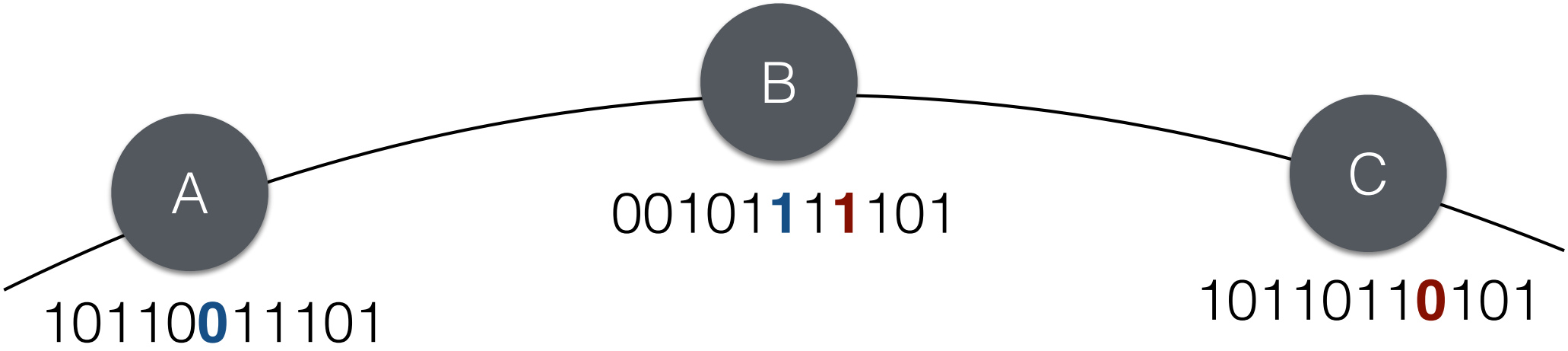
- k-bit colors ⇒ new colors on $\lceil \log_2 k \rceil + 1$ bits
- $\log^\star n + O(1)$ rounds to reach colors on 3 bits
- 8 colors down to 3 colors in 5 rounds
- Total number of rounds = $\log^\star n + O(1)$



In fact: 6 colors down to 3 colors in 3 rounds

❑

Speeding up CV's Algorithm



- Every node can simulate 2 rounds in just 1 round
- left round + right round ➡ implemented in 1 round
- Total number of rounds = $\frac{1}{2}\log^{\star} n + O(1)$ ❑

Question

Is there a distributed algorithm for 3-coloring the n-node ring performing in less that $\Omega(\log^\star n)$ rounds?

Theorem (Linial, 1992) Any distributed algorithm for 3-coloring C_n performs in at least $\frac{1}{2}\log^\star n - O(1)$ rounds.

Dijkstra Prize 2013

This will be established later in the course

Randomized Coloring of C_n

- MyFinalColor ← $\perp$
- Repeat
  - MyProposedColor ← color in $\{1,2,3\}$ uniformly at random
  - **Send** MyProposedColor to neighbors
  - **Receive** ProposedColors from neighbors
  - if MyProposedColor is different from the FinalColors and ProposedColors of both neighbors then MyFinalColor ← MyProposedColor
  - **Send** MyFinalColor to neighbors
  - **Receive** FinalColors from neighbors
- Until MyFinalColor $\neq \perp$

Claim This (Las Vegas) algorithm runs in $O(\log n)$ rounds w.h.p.

- A **Las Vegas algorithm** is a randomized algorithm that always gives the correct output but whose running time is a random variable.

$$\Pr[\text{running time} \leq T] \geq 1 - \epsilon$$

- A **Monte Carlo algorithm** is a randomized algorithms whose running time is deterministic, but whose output may be incorrect with a certain, typically small, probability.

$$\Pr[\text{error after time } T] \leq \epsilon$$

Definition A sequence $(\mathcal{E}_n)_{n\geq 1}$ of events holds with high probability (w.h.p.) whenever $\Pr[\mathcal{E}_n] = 1 - O(1/n^c)$ for some constant $c > 0$ (typically $c = 1$).

Elements of probability:

Recall:

« A given B holds » or
« A conditioned to B »

- $\Pr[A|B] = \Pr[A \wedge B] / \Pr[B] \Rightarrow \Pr[A \wedge B] = \Pr[A|B] \cdot \Pr[B]$

and

A and B independent
$\Leftrightarrow \Pr[A \wedge B] = \Pr[A] \cdot \Pr[B]$

- $\Pr[A] = \Pr[A|B] \cdot \Pr[B] + \Pr[A|\neg B] \cdot \Pr[\neg B]$

- Union bound: $\Pr[A \vee B] \leq \Pr[A] + \Pr[B]$

$$\Pr[\exists\ s \in S : s \vDash \mathcal{P}] = \Pr[(s_1 \vDash \mathcal{P}) \underset{\text{or}}{\vee} (s_2 \vDash \mathcal{P}) \vee \ldots \vee (s_m \vDash \mathcal{P})]$$

Claim This (Las Vegas) algorithm runs in $O(\log n)$ rounds w.h.p.

Proof At every execution of the repeat loop, for every fixed node u,

$$\Pr[u \text{ terminates}] = \Pr[X \notin \{X_{-1}, X_{+1}\}] \geq \frac{1}{3}$$

Note: At first execution of the repeat loop:

$$\Pr[u \text{ terminates}] = \sum_{x \in \{1,2,3\}} \Pr[(X_{-1} \neq x) \wedge (X_{+1} \neq x)] \cdot \Pr[X = x] \geq \frac{4}{9}$$

- $\Pr[u \text{ does not terminates after } k \text{ rounds}] \leq \left(\frac{2}{3}\right)^k$
- $\Pr[u \text{ does not terminates after } c \log_{3/2} n \text{ rounds}] \leq \frac{1}{n^c}$
- $\Pr[\text{some } u \text{ does not terminate after } c \log_{3/2} n \text{ rounds} \leq \frac{1}{n^{c-1}}$
- $\Pr[\text{every node } u \text{ terminates after } c \log_{3/2} n \text{ rounds} \geq 1 - \frac{1}{n^{c-1}}$ ❑

End Lecture 1