Distributed Computing
16 - LOCAL Variants

Mikaël Rabie
Université de Paris, IRIF
Sequential Complexity
SLOCAL Model

- Each node is activated one after another, to compute its own output
- A node has access to the outputs already computed to produce its own
- Complexity: maximal radius needed among nodes
SLOCAL Model

- Each node is activated one after another, to compute its own output
- A node has access to the outputs already computed to produce its own
- Complexity: maximal radius needed among nodes
- Greedy problems can be solved in radius $O(1)$
SLOCAL Model

- Each node is activated one after another, to compute its own output
- A node has access to the outputs already computed to produce its own
- Complexity: maximal radius needed among nodes
- **Greedy** problems can be solved in radius $O(1)$
SLOCAL Model

- Each node is activated one after another, to compute its own output
- A node has access to the outputs already computed to produce its own
- Complexity: maximal radius needed among nodes
- **Greedy** problems can be solved in radius $O(1)$
SLOCAL Model

- Each node is activated one after another, to compute its own output
- A node has access to the outputs already computed to produce its own
- Complexity: maximal radius needed among nodes
- **Greedy** problems can be solved in radius $O(1)$
SLOCAL Model

- Each node is activated one after another, to compute its own output
- A node has access to the outputs already computed to produce its own
- Complexity: maximal radius needed among nodes
- **Greedy** problems can be solved in radius $O(1)$
- Each node is activated one after another, to compute its own output
- A node has access to the outputs already computed to produce its own
- Complexity: maximal radius needed among nodes
- **Greedy** problems can be solved in radius $O(1)$
- Each node is activated one after another, to compute its own output
- A node has access to the outputs already computed to produce its own
- Complexity: maximal radius needed among nodes
- **Greedy** problems can be solved in radius $O(1)$
SLOCAL Model

- Each node is activated one after another, to compute its own output
- A node has access to the outputs already computed to produce its own output
- Complexity: maximal radius needed among nodes
- **Greedy** problems can be solved in radius $O(1)$
SLOCAL Model

- Each node is activated one after another, to compute its own output
- A node has access to the outputs already computed to produce its own
- Complexity: maximal radius needed among nodes
- **Greedy** problems can be solved in radius $O(1)$
SLOCAL Model

- Each node is activated one after another, to compute its own output
- A node has access to the outputs already computed to produce its own
- Complexity: maximal radius needed among nodes
- **Greedy** problems can be solved in radius $O(1)$
SLOCAL Model

- Each node is activated one after another, to compute its own output
- A node has access to the outputs already computed to produce its own
- Complexity: maximal radius needed among nodes
- Greedy problems can be solved in radius $O(1)$
Volume Complexity
CentLOCAL Model

- In parallel, each node $v$:
  - Knows its own $ld_v$ and degree $d_{ld_v}$
  - At each step, they send a request $(ld_u, k)$, with $k \leq d_{ld_u}$
  - They get $(ld_w, d_{ld_w}, k')$ such that $(u, v) \in E$ are connected by port $k$ from $u$ and $k'$ from $w$
- Complexity: maximal number of requests from a node
In parallel, each node $v$:

- Knows its own $ld_v$ and degree $d_{ld_v}$
- At each step, they send a request $(ld_u, k)$, with $k \leq d_{ld_u}$
- They get $(ld_w, d_{ld_w}, k')$ such that $(u, v) \in E$ are connected by port $k$ from $u$ and $k'$ from $w$

Complexity: maximal number of requests from a node

Request: $(14, 2)$
CentLOCAL Model

- In parallel, each node $v$:
  - Knows its own $Id_v$ and degree $d_{Id_v}$
  - At each step, they send a request $(Id_u, k)$, with $k \leq d_{Id_u}$
  - They get $(Id_w, d_{Id_w}, k')$ such that $(u, v) \in E$ are connected by port $k$ from $u$ and $k'$ from $w$

- Complexity: maximal number of requests from a node

Request: $(14, 2)$
CentLOCAL Model

- In parallel, each node $v$:
  - Knows its own $Id_v$ and degree $d_{Id_v}$
  - At each step, they send a request $(Id_u, k)$, with $k \leq d_{Id_u}$
  - They get $(Id_w, d_{Id_w}, k')$ such that $(u, v) \in E$ are connected by port $k$ from $u$ and $k'$ from $w$

- Complexity: maximal number of requests from a node

Request: $(14,2) \Rightarrow (2,4,1)$
In parallel, each node $v$:
- Knows its own $ld_v$ and degree $d_{ld_v}$
- At each step, they send a request $(ld_u, k)$, with $k \leq d_{ld_u}$
- They get $(ld_w, d_{ld_w}, k')$ such that $(u, v) \in E$ are connected by port $k$ from $u$ and $k'$ from $w$

Complexity: maximal number of requests from a node

Request: (14,3)
CentLOCAL Model

- In parallel, each node $v$:
  - Knows its own $ld_v$ and degree $d_{ld_v}$
  - At each step, they send a request $(ld_u, k)$, with $k \leq d_{ld_u}$
  - They get $(ld_w, d_{ld_w}, k')$ such that $(u, v) \in E$ are connected by port $k$ from $u$ and $k'$ from $w$
- Complexity: maximal number of requests from a node

Request: $(14, 3) \Rightarrow (8, 2, 2)$
CentLOCAL Model

- In parallel, each node $v$:
  - Knows its own $Id_v$ and degree $d_{Id_v}$
  - At each step, they send a request $(Id_u, k)$, with $k \leq d_{Id_u}$
  - They get $(Id_w, d_{Id_w}, k')$ such that $(u, v) \in E$ are connected by port $k$ from $u$ and $k'$ from $w$
- Complexity: maximal number of requests from a node

Request: $(2,3)$
CentLOCAL Model

- In parallel, each node $v$:
  - Knows its own $ld_v$ and degree $d_{ld_v}$
  - At each step, they send a request $(ld_u, k)$, with $k \leq d_{ld_u}$
  - They get $(ld_w, d_{ld_w}, k')$ such that $(u, v) \in E$ are connected by port $k$ from $u$ and $k'$ from $w$

- Complexity: maximal number of requests from a node

Request: $(2,3) \Rightarrow (10,4,3)$
Problem $A$ can be solved in time $\Theta(f(n))$ in the LOCAL model.

$\Rightarrow$ $A$ can be solved in time $\Omega(f(n))$ and $O(\Delta f(n))$ in the CentLOCAL model.

Even et al. (2018) showed that there is a CentLOCAL algorithm in time $O(\Delta \log^* n + \Delta^3)$ for $\leq \Delta^2$-coloring a graph.

There is a CentLOCAL algorithm in time $O(\Delta \log^* n + \Delta^3)$ for orienting a graph where the longer oriented path is of length $\leq \Delta^2$. Any greedy problem can be solved in time $O(f(\Delta) \log^* n)$. 

Greedy Problems
Problem $A$ can be solved in time $\Theta(f(n))$ in the LOCAL model

$\Rightarrow$ $A$ can be solved in time $\Omega(f(n))$ and $O\left(\Delta^{f(n)}\right)$ in the CentLOCAL model
Problem $A$ can be solved in time $\Theta(f(n))$ in the LOCAL model

$\Rightarrow$ $A$ can be solved in time $\Omega(f(n))$ and $O\left(\Delta^{f(n)}\right)$ in the CentLOCAL model

**Even et. al (2018)**

There is a CentLOCAL algorithm in time $O(\Delta \times \log^* n + \Delta^3)$ for $\leq \Delta^2$-coloring a graph.

There is a CentLOCAL algorithm in time $O(\Delta \times \log^* n + \Delta^3)$ for orienting a graph where the longer oriented path is of length $\leq \Delta^2$.

Any greedy problem can be solved in time $O(f(\Delta) \times \log^* n)$. 
Rosenbaum and Suomela (2020)
In the CentLOCAL model, if $n$ is not given in advance and identifiers do not require to be polynomial in $n$, there is no problem whose time complexity is in $\omega(\log^* n) \cap o(n)$. 

• Take $N$ such that $T(N) \ll N$
• Do a distance $N$-coloring
• Simulate the algorithm with the new identifiers
Rosenbaum and Suomela (2020)

In the CentLOCAL model, if \( n \) is not given in advance and identifiers do not require to be polynomial in \( n \), there is no problem whose time complexity is in \( \omega(\log^* n) \cap o(n) \).

- Take \( N \) such that \( T(N) \ll N \)
- Do a distance \( N \)-coloring
- Simulate the algorithm with the new identifiers
Waking Up Complexity
Sleeping LOCAL Model

- At each round, a node decides if it is active or not
- A communicates only with its active neighbors
- Complexity: maximal number of active rounds for a single node
Sleeping LOCAL Model

- At each round, a node decides if it is active or not
- A communicates only with its active neighbors
- Complexity: maximal number of active rounds for a single node
Sleeping LOCAL Model

- At each round, a node decides if it is active or not
- A communicates only with its active neighbors
- Complexity: maximal number of active rounds for a single node
Δ + 1-coloring can be solved in $O(\Delta)$ rounds
Δ + 1-coloring can be solved in $O(\Delta)$ rounds:

- Round 1: all nodes are activated. Know their identifiers and their neighbours’.
- Node of Identifier $Id$ wakes up at round $Id + 1$ to know their neighbours’ colors.
- Neighbours of node of identifier $Id$ also wakes up at that round.
Δ + 1-coloring can be solved in $O(\Delta)$ rounds:

- Round 1: all nodes are activated. Know their identifiers and their neighbours’.
- Node of Identifier $Id$ wakes up at round $Id + 1$ to know their neighbours’ colors.
- Neighbours of node of identifier $Id$ also wakes up at that round.

Problem A can be solved in time $f(n)$ in the SLOCAL model.

$
\Rightarrow A$ can be solved in time $f(n)$ in the Sleeping LOCAL model.
A Link with SLOCAL

$\Delta + 1$-coloring can be solved in $O(\Delta)$ rounds:

- Round 1: all nodes are activated. Know their identifiers and their neighbours’.
- Node of Identifier $Id$ wakes up at round $Id + 1$ to know their neighbours’ colors.
- Neighbours of node of identifier $Id$ also wakes up at that round.

Problem A can be solved in time $f(n)$ in the SLOCAL model

$\Rightarrow A$ can be solved in time $O(f(n)\Delta^{f(n)})$ in the Sleeping LOCAL model.
Any graph problem can be solved in $O(\log n)$ rounds in the Sleeping LOCAL model.
Barenboim and Maimon (2021)

Any graph problem can be solved in $O(\log n)$ rounds in the Sleeping LOCAL model.

Distributed Layered Tree (DLT) - Oriented Spanning Tree such as:

- Each vertex has a label
- The label of a vertex is smaller than its parent’s
- Each vertex knows the label of its neighbours in the tree
**Full Knowledge of the Graph**

**Barenboim and Maimon (2021)**
Any graph problem can be solved in $O(\log n)$ rounds in the Sleeping LOCAL model.

Distributed Layered Tree (DLT) - Oriented Spanning Tree such as:

- Each vertex has a label
- The label of a vertex is smaller than its parent’s
- Each vertex knows the label of its neighbours in the tree

**Constant Coordination**
Broadcast and Convergecast can be done in $O(1)$ rounds in a DLT.
Building a DLT

**Barenboim and Maimon (2021)**
A DLT can be built in $O(\log n)$ rounds in the Sleeping LOCAL model.

- Labels are of the form $(a, b)$, ordered lexicographically.
- At the beginning, all nodes have label $(ld(u), 0)$.
- At the beginning of each expand step, all nodes of a subtree $T$ are of the form $(L(T), b)$.
- Repeat $\log n$ times:
  1. Select a neighbour Tree $T'$ with smaller label $(L(T) > L(T'))$.
  2. Merge $T$ and $T'$, using an edge $(u, v)$.
  3. If $T$ could not choose a neighbour and was not selected
     $T$ chooses a tree $T'$ to join using an edge $(u, v)$.
  4. All nodes learn their new neighbours in the tree.
  5. Change the labels in $T$ such as the second part is the distance to $v$.
  6. Convergecast to gather the new structure of the component $C$ to the root $r$.
  7. Broadcast a new labelling $(L(r), \text{dist}(r))$. 
• Alkida Balliu, Juho Hirvonen, Darya Melnyk, Dennis Olivetti, Joel Rybicki, Jukka Suomela. **Local Mending**. arXiv 2021.
• Leonid Barenboim, Tzalik Maimon. **Deterministic Logarithmic Completeness in the Distributed Sleeping Model**. In DISC 2021.
• Mohsen Ghaffari, Fabian Kuhn, Yannic Maus. **On the complexity of local distributed graph problems**. In STOC 2017.
• Rosenbaum, Suomela. **Seeing far vs. seeing wide** In PODC 2020.