From Shared Memory to Message Passing

Stefan Schmid
T-Labs & TU Berlin

Some parts of the lecture (e.g., "Skript" and exercises) are based on the lectures of Prof. Roger Wattenhofer at ETH Zurich and Prof. Christian Scheideler at University Paderborn. Thanks!

Message Passing

Many distributed systems do not have a shared memory, but pass messages along networks: classic social networks...

Message Passing

Many distributed systems do not have a shared memory, but pass messages along networks: ... OSN...

Message Passing

Many distributed systems do not have a shared memory, but pass messages along networks: ... Internet networks...
Message Passing

Many distributed systems do not have a shared memory, but pass messages along networks: ... wireless networks...

Multi-hop sensor networks

Message Passing

Many distributed systems do not have a shared memory, but pass messages along networks: ... but also smart grids...

Small grid of communication nodes and powerline communication

Fundamental Questions: Communication Network?

Sometimes topology can be chosen (e.g., peer-to-peer networks)!

What communication networks / architectures are "good"?

A comparison.

(degree-diameter tradeoff, network expansion, routing, robustness under dynamics, ...).

Line: simple, but long communication delays and not robust?

Butterflies?

De Bruijn graphs?
What is the „best“ degree / diameter tradeoff?

**Line:** diameter n-1, degree 2

**Clique:** diameter 1, degree n-1

**Hypercube:** diameter log n, degree log n

Can we reduce both?

Yes. It must hold that degree > n (why?)

**Pancake graphs:** log n / log log n diameter, log n / log log n degree

How to make these topologies robust and dynamic? (E.g., continuous-discrete approach)

---

Fundamental Questions: Algorithms

Fundamental tasks on networks:

- **Routing:** sending a message from node A to node B?
- **Broadcasting:** sending a message to all nodes?
- **Aggregation:** Finding the most frequent element in the network? The node measuring the hottest temperature? Etc.
- **Election:** of a leader
- **Coloring:** the network (e.g., frequency spectrum allocation in wireless networks), computing independent sets, etc.

Complexity evaluation:

- Distributed runtime: number of communication rounds until task fulfilled?
- Message complexity: number (and size) of messages to be transmitted?
- Local complexity of algorithm: Complexity of algorithm in node?

---

Example: Local Vertex Coloring

**Local Algorithm**

Each processor / node must act based on information about its k-hop neighborhood! (Fast and efficient algorithms, good under dynamics)

Sometimes local algorithms can even approximate NP-hard problems quite well and fast!

**Vertex Coloring**

Nodes should color themselves such that no adjacent nodes have same color – but minimize # color!

Used, e.g., for wireless spectrum allocation...

---

Local Algorithm

Simplified round model, all nodes execute the same protocol, ...

In one round:

1. send messages (message complexity)
2. receive messages
3. process messages (local computation complexity)

Number of rounds until termination: time complexity
Local Algorithm

Simplified round model, all nodes execute the same protocol, ...

In one round:
1. send messages (message complexity)
2. receive messages
3. process messages (local computation complexity)

Number of rounds until termination: time complexity

Local Algorithm

Simplified round model, all nodes execute the same protocol, ...

In one round:
1. send messages (message complexity)
2. receive messages
3. process messages (local computation complexity)

Number of rounds until termination: time complexity

Local Algorithm

Simplified round model, all nodes execute the same protocol, ...

In one round:
1. send messages (message complexity)
2. receive messages
3. process messages (local computation complexity)

Number of rounds until termination: time complexity

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)

Local Vertex Coloring for Rooted Tree?

Idea? (Assume, e.g., arbitrary but unique IDs are given at nodes, ...)}
Local Vertex Coloring for Tree?

Two colors suffice: root sends binary message down...
Local Vertex Coloring for Trees?

Can we do faster than diameter of tree?! Yes! With constant number of colors in $\log^*(n)$ time!!

$log^*$: log = divide by two, $\log_2 = ?$, $\log^* = ?$

One of the fastest non-constant time algo that exist! Besides inverse Ackermann function or so...

$log^*$ (log of log) = log of log of log of log of log... of log (at most 6 colors)

Why is this good? If something happens (dynamic network), back to good state in a sec!

There is a lower bound of log-star too, so that’s optimal!

How does it work?

Initially: each node has unique log(n)-bit ID = legal coloring

(Interpret ID as color => n colors)

Idea: root should have label 0 (fixed)

In each step: send ID to children; receive color from parent and interpret as little-endian bit string $c_2...c_0$

let $i$ be smallest index where $c_i$ and $c_0$ differ

set new $c_i$ as bit string $c_i || c_v(i)$

until $c_i \in \{0,1,2,...,5\}$ (at most 6 colors)
How does it work?

Initially, each node has unique log(n)-bit ID = legal coloring (interpret ID as color => n colors)

Idea:
- root should have label 0 (fixed)
- in each step: send ID to children; receive color from parent and interpret as little-endian bit string: \( c_p = c(k)\ldots c(0) \)
- let \( i \) be smallest index where \( c_v \) and \( c_p \) differ
- set new \( c_v = i \) (as bit string) || \( c_v(i) \)
- until \( c_v \in \{0,1,2,\ldots,5\} \) (at most 6 colors)

Round 2

Idea:
- root should have label 0 (fixed)
- in each step: send ID to children; receive color from parent and interpret as little-endian bit string: \( c_p = c(k)\ldots c(0) \)
- let \( i \) be smallest index where \( c_v \) and \( c_p \) differ
- set new \( c_v = i \) (as bit string) || \( c_v(i) \)
- until \( c_v \in \{0,1,2,\ldots,5\} \) (at most 6 colors)

Round 3, etc.

Why does it work?

Why is this \( \log^* \) time?!

Idea: In each round, the size of the ID (and hence the number of colors) is reduced by a log factor.
- To index the bit where two labels of size \( n \) bits differ, \( \log(n) \) bits are needed!
- Plus the one bit that is appended...

Why is this a valid vertex coloring?!

Idea: During the entire execution, adjacent nodes always have different colors (invariant) because:
- ID always differ as new label is index of difference to parent plus own bit there (if parent would differ at same location as grand parent, at least the last bit would be different).
Lower Bounds

- **r-hop View**
  R-hop view of a node v defined as set of states of all nodes in r-hop neighborhood of v.

- **Neighborhood Graph**
  Vertices of neighborhood graph are r-hop views; vertices are connected if views could result from adjacent nodes.

Observe: Any deterministic vertex coloring algorithm can be seen as mapping r-hop neighborhoods to colors. (Same neighborhood gives same color.)
Observe: Chromatic number of neighborhood graph (classic coloring) implies possible local algorithm coloring.

\[ \Rightarrow \text{Gives us lower bound of possible colorings with } r \text{-neighborhood!} \]