# What Can(not) Be Computed in One Round in Interconnection networks?

F. Becker[1] A. Kosowski[2] M. Matamala[3] N. Nisse[4]
I. Rapaport[3] K. Suchan[5] I. Todinca[1]

[1] LIFO, Univ. Orléans, France
[2] Inria, LaBRI, Bordeaux, France
[3] DIM, Universidad de Chile, Santiago, Chile
[4] COATI, Inria, I3S, CNRS, UNS, Sophia Antipolis, France
[5] Universidad Adolfo Ibáñez, Santiago, Chile

ADR Network Science

Paris, Feb. $28^{th}$ 2013

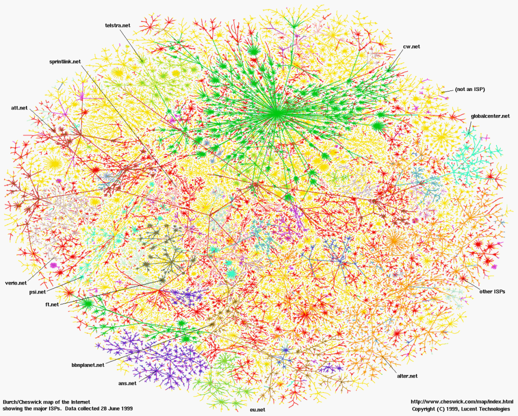
Adding a Referee to an Interconnection Network: What Can(not) Be Computed in One Round. IPDPS 2011
Allowing each node to communicate only once in a distributed system: shared whiteboard models SPAA 2012

# Distributed computation of network properties

## Problem

Huge networks ⇒ generic algorithms (even polynomial) are not efficient

Need to use structural properties



# Distributed computation of network properties

## Problem

Huge networks $\Rightarrow$ generic algorithms (even polynomial) are not efficient

Need to use structural properties

Example of compact routing

| network | mult. stretch | routing table labelled | | routing table name-independent | |
|---|---|---|---|---|---|
| arbitrary $(k \geq 2)$ | shortest path | $O(n \log n)$ | [folk] | $\Theta(n \log n)$ | [Gavoille,Pérennes] (BGP) |
| | $O(k)$ | $O(n^{1/k})$ | [Thorup,Zwick] | $\Theta(n^{1/k})$ | [TZ/Abraham et al.] |
| trees | shortest path | $O(\log n)$ | [TZ/Fraigniaud,Gavoille] | $\Omega(\sqrt{n})$ | [Laing,Rajaraman] |
| | $2^k - 1$ | | | $\Theta(n^{1/k})$ | [Laing/Abraham et al.] |
| doubling-$\alpha$ dimension | $O(1) + \epsilon$ | $O(\log \Delta)$ | [Talwar/Slivkins] | $O(\epsilon^{-\alpha} \log n)$ | [Abraham et al.] |
| | | $O(\log n)$ | [Abraham et al.] | | |
| planar | $1 + \epsilon$ | $O(\log n)$ | [Thorup] | | |
| $H$-minor free | $1 + \epsilon$ | $O(\lvert H\rvert ! \cdot 2^{\lvert H\rvert} \log n)$ | [Abraham,Gavoille] | | |

BGP is generic $\Rightarrow$ large Routing Tables :(
but easy to compute and update :)

other schemes require structural information (e.g., decompositions) on the graph

# Distributed computation of network properties

## Problem

Huge networks $\Rightarrow$ generic algorithms (even polynomial) are not efficient

Need to use structural properties

## Objectives

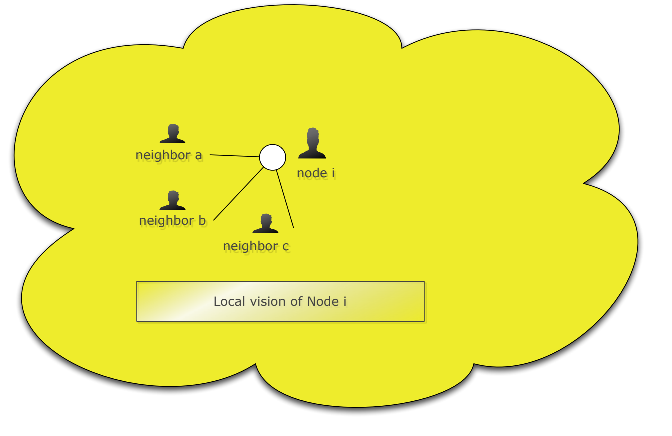
- Understand, compute, discover... structural Properties
- Distributed/Local computation
- Use it for algorithmic purposes (not only routing)
- Model/simulate such networks (static/dynamic behavior)

## Questions

What hypothesis can we adopt for the computation?

What is feasible in a given model?

# Model of distributed computing

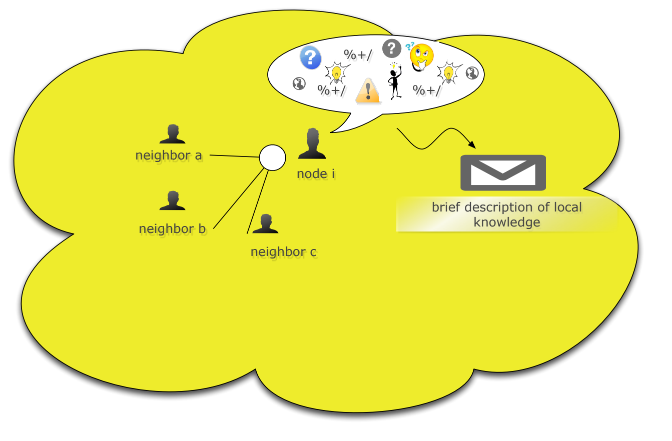



Graph with $n$ nodes.

Nodes have distinct identifiers in $\{1, \cdots, n\}$.

A node knows: its ID and the IDs of its neighbors

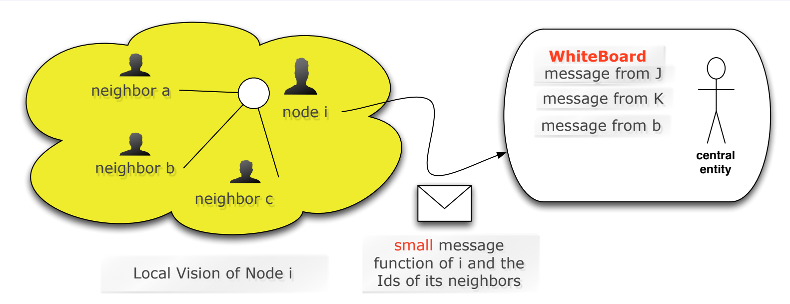
# Model of distributed computing




A node has arbitrary computation power.

Goal: encode its local knowledge in a small message (typically $O(\log n)$).

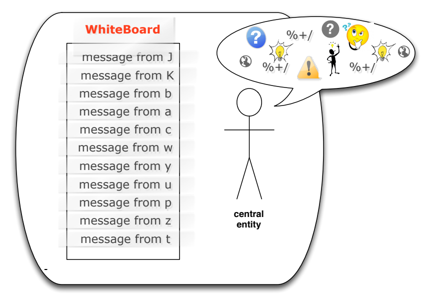
# Model of distributed computing




Each node sends its (unique) message to a central entity

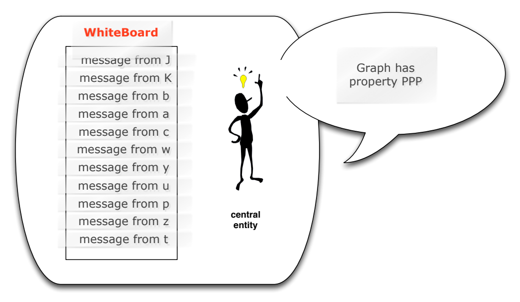
**Remark:** If $|message| = n$ bits, then node gives its whole neighboorhood

# Model of distributed computing




The referee has arbitrary computation power and use the $n$ messages to...

# Model of distributed computing




... answer a question about the graph (typically: "does $G$ has some property?")

# To summerize: Model of distributed computing

### Principle

Does $G$ belongs to $\mathcal{P}$?

- each node encodes its local knowledge

$$message : \text{ID of } v \times \text{IDs of } N(v) \rightarrow message(v), \text{ and}$$

$$|message(v)| = O(\log n) \text{ bits}$$

- the referee decodes the $n$ messages to answer

$$answer : (message(v_i))_{i \leq n} \rightarrow \{true; false\}$$

### Hypothesis

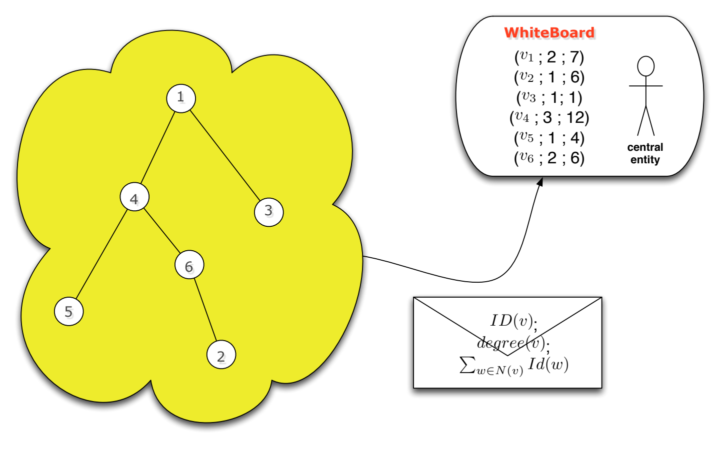
- arbitrary computational power: *message* and *answer* are arbitrary functions
- IDs are distinct in $\{1, \cdots, n\}$

**Remark:** if bounded maximum degree: each node may send its full adjacency list.

### Problem

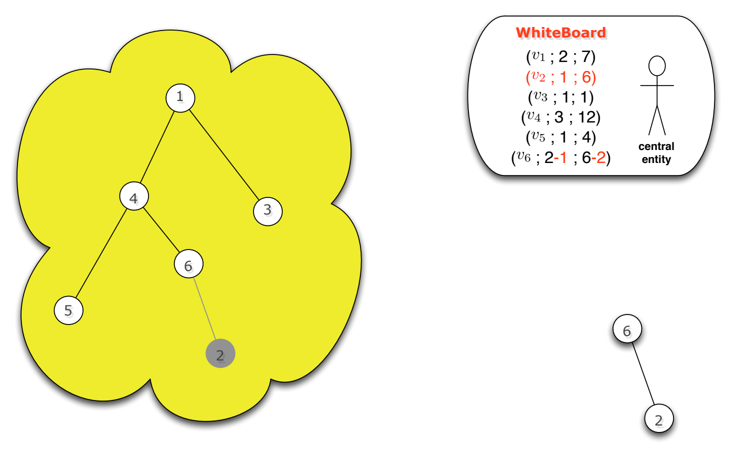
in total: $O(n \log n)$ bits of local information
What kind of question can be answered?

## Example: Does $G$ is a tree? If yes, compute its adjacency matrix.



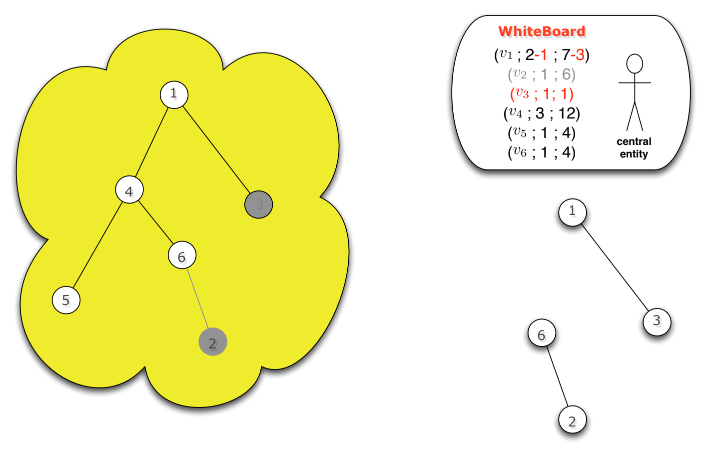

Each node sends its ID, its degree and the sum of the IDs of its neighbors

# Example: Does $G$ is a tree? If yes, compute its adjacency matrix.



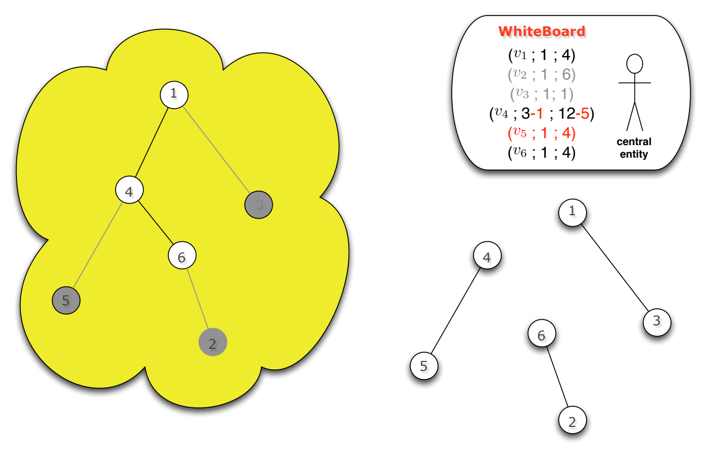

The referee iteratively "prunes" the one-degree nodes in the whiteboard
In parallel, he re-builds the tree.

# Example: Does $G$ is a tree? If yes, compute its adjacency matrix.



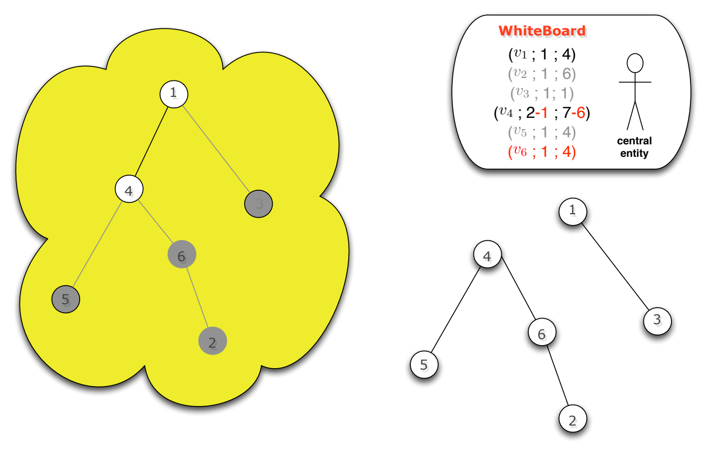

The referee iteratively "prunes" the one-degree nodes in the whiteboard
In parallel, he re-builds the tree.

# Example: Does $G$ is a tree? If yes, compute its adjacency matrix.




The referee iteratively "prunes" the one-degree nodes in the whiteboard
In parallel, he re-builds the tree.

## Example: Does $G$ is a tree? If yes, compute its adjacency matrix.




The referee iteratively "prunes" the one-degree nodes in the whiteboard
In parallel, he re-builds the tree.

# What Can(not) Be Computed in One Round

**Possible**

Decide if a graph has bounded degeneracy (include planar graphs, bounded genus graphs, bounded treewidth graphs...). If yes, build their adjacency matrix.

*proof:* generalization of the "pruning process" of trees.

**Not possible**

Decide if the graph contains a triangle, a (induced or not) square.
Decide if the graph has diameter at most 3

*proof:* Kind of reduction.
If possible $\Rightarrow$ Possible to build adjacency matrix of bipartite graphs.
$2^{\Omega(n^2)}$ such graphs $\Rightarrow$ impossible to distinguish all of them with $O(n \log n)$ bits.
$\Rightarrow$ contradiction

**We don't know????**

Decide if the graph is connected.

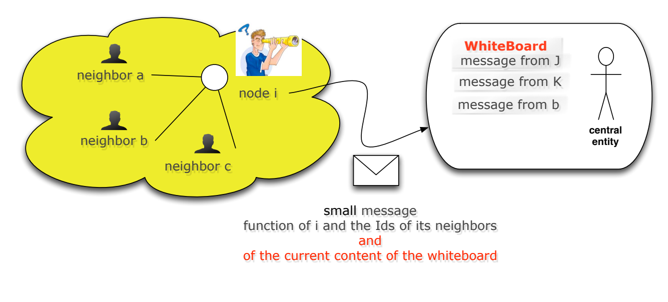
# Generalization

**Until now:**

All nodes write simultaneously on the whiteboard
Don't take advantage of what is written by other nodes.

**Now:**

Nodes can also read the whiteboard.
Can use previous messages to compute their own message



# 4 Models

**SimAsync** — model above

All nodes write simultaneously on the whiteboard

**SimSync**

Nodes write sequentially.
Worst ordering: order chosen by an adversary

**Async:** — model of asynchronicity

Nodes rise hand to speak.
If several nodes rise hand, all write simultaneously.

**Sync** — model of synchronicity

Nodes rise hand to speak.
If several nodes rise hand, they write sequentially in worst odering.

# Results

## Hierarchy of models

$\textsc{SimAsync}(\log n) < \textsc{SimSync}(\log n) < \textsc{Async}(\log n) \leq \textsc{Sync}(\log n)$

| message: $O(\log n)$ bits | SimAsync | SimSync | Async | Sync |
|---|---|---|---|---|
| BUILD k-DEGENERATE | yes | yes | yes | yes |
| ROOTED MIS | no | yes | yes | yes |
| Square | no | no | ? | ? |
| Connectivity | ? | ? | yes | yes |
| Spanning tree | ? | ? | yes | yes |
| Bipartite-BFS | no | no | yes | yes |
| BFS | ? | ? | ? | yes |

## Orthogonal criteria

Let $f(n) = o(n)$ and $g(n) = o(f(n))$.

There exist problems solvable in $\textsc{SimAsync}(f(n))$ and not in $\textsc{Sync}(g(n))$.

# Further works

Several messages per nodes?

Probabilistic algorithms?

What if graph partially known (only few messages)?

Connectivity?

What is a realistic model?

...