Local Terminations and Distributed Computability in Anonymous Networks

Jérémie Chalopin   Emmanuel Godard   Yves Métivier

LIF, CNRS & Aix-Marseille Université

LaBRI, Université de Bordeaux

Session SHAMAN

Distributed Computability

Question

What can be computed by an arbitrary network, with partial knowledge about its topology?

Distributed Computability

Question

What can be computed by an arbitrary network, with partial knowledge about its topology?

For distributed networks, computability is known to be restricted by

- Is it a ring, a tree? ...

Distributed Computability

Question

What can be computed by an arbitrary network, with partial knowledge about its topology?

For distributed networks, computability is known to be restricted by

- Is it a ring, a tree? ...
- Are there loss of messages, or no? ...

Distributed Computability

Question

What can be computed by an arbitrary network, with partial knowledge about its topology?

For distributed networks, computability is known to be restricted by

- Is it a ring, a tree? ...
- Are there loss of messages, or no? ...
- Do nodes have unique identitier? ...

Distributed Computability

Question

What can be computed by an arbitrary network, with partial knowledge about its topology?

For distributed networks, computability is known to be restricted by

- Is it a ring, a tree? ...
- Are there loss of messages, or no? ...
- Do nodes have unique identitier? ...
- Is the size of the network known? ...

Distributed Computability

Question

What can be computed by an arbitrary network, with partial knowledge about its topology?

For distributed networks, computability is known to be restricted by

- Is it a ring, a tree? ...
- Are there loss of messages, or no? ...
- Do nodes have unique identitier? ...
- Is the size of the network known? ...

Distributed Computability

Question

What can be computed by an arbitrary network, with partial knowledge about its topology?

For distributed networks, computability is known to be restricted by

- Is it a ring, a tree? ...
- Are there loss of messages, or no? ...
- Do nodes have unique identitier? ...
- Is the size of the network known? ...

So the question is

Given a *specification* S, a family of networks $\mathcal{F}$, is there a distributed algorithm $\mathcal{A}$ such that, any execution of $\mathcal{A}$ on any network in $\mathcal{F}$ ends with final state (labels) satisfying S.

Definitions

Is the task $(\mathcal{F}, S)$ computable by a distributed algorithm on $\mathcal{F}$?

- $\mathcal{F}$ is a family of graphs; it represents

Definitions

Is the task $(\mathcal{F}, S)$ computable by a distributed algorithm on $\mathcal{F}$?

- $\mathcal{F}$ is a family of graphs; it represents
  - **topology:** trees, rings, meshes, ...

Definitions

Is the task $(\mathcal{F}, S)$ computable by a distributed algorithm on $\mathcal{F}$?

- $\mathcal{F}$ is a family of labeled graphs; it represents
  - **topology:** trees, rings, meshes, ...
  - any information is encoded in the initial labeling:

Definitions

Is the task $(\mathcal{F}, S)$ computable by a distributed algorithm on $\mathcal{F}$?

- $\mathcal{F}$ is a family of labeled graphs; it represents
  - **topology:** trees, rings, meshes, ...
  - any information is encoded in the initial labeling:
    - **structural information:** identities, sense of direction, ...

Definitions

Is the task $(\mathcal{F}, S)$ computable by a distributed algorithm on $\mathcal{F}$?

- $\mathcal{F}$ is a family of labeled graphs; it represents
  - **topology:** trees, rings, meshes, ...
  - any information is encoded in the initial labeling:
    - **structural information:** identities, sense of direction, ...
    - **structural knowledge:** the size, bound on the diameter, the topology ...

Definitions

Is the task $(\mathcal{F}, S)$ computable by a distributed algorithm on $\mathcal{F}$?

- $\mathcal{F}$ is a family of labeled graphs; it represents
  - **topology:** trees, rings, meshes, ...
  - any information is encoded in the initial labeling:
    - **structural information:** identities, sense of direction, ...
    - **structural knowledge:** the size, bound on the diameter, the topology ...
- the specification S is a graph relabelling relation

Definitions

Is the task $(\mathcal{F}, S)$ computable by a distributed algorithm on $\mathcal{F}$?

- $\mathcal{F}$ is a family of labeled graphs; it represents
  - **topology:** trees, rings, meshes, ...
  - any information is encoded in the initial labeling:
    - **structural information:** identities, sense of direction, ...
    - **structural knowledge:** the size, bound on the diameter, the topology ...
- the specification S is a graph relabelling relation
  - $(G, \texttt{input})\ S\ (G, \texttt{output})$: spanning tree, leader election, ...

Definitions

Is the task $(\mathcal{F}, S)$ computable by a distributed algorithm on $\mathcal{F}$?

- $\mathcal{F}$ is a family of labeled graphs; it represents
  - **topology:** trees, rings, meshes, ...
  - any information is encoded in the initial labeling:
    - **structural information:** identities, sense of direction, ...
    - **structural knowledge:** the size, bound on the diameter, the topology ...
- the specification S is a graph relabelling relation
  - $(G, \texttt{input})\ S\ (G, \texttt{output})$: spanning tree, leader election, ...
- Is that enough to define the problem?

Definitions

Is the task $(\mathcal{F}, S)$ computable by a distributed algorithm on $\mathcal{F}$?

- $\mathcal{F}$ is a family of labeled graphs; it represents
  - **topology:** trees, rings, meshes, ...
  - any information is encoded in the initial labeling:
    - **structural information:** identities, sense of direction, ...
    - **structural knowledge:** the size, bound on the diameter, the topology ...
- the specification S is a graph relabelling relation
  - $(G, \texttt{input})\ S\ (G, \texttt{output})$: spanning tree, leader election, ...
- Is that enough to define the problem?
  - Different kinds of computability, depending on the kind of termination we are interested in.

Definitions

Is the task $(\mathcal{F}, S)$ computable by a distributed algorithm on $\mathcal{F}$?

- $\mathcal{F}$ is a family of labeled graphs; it represents
  - **topology:** trees, rings, meshes, ...
  - any information is encoded in the initial labeling:
    - **structural information:** identities, sense of direction, ...
    - **structural knowledge:** the size, bound on the diameter, the topology ...
- the specification S is a graph relabelling relation
  - $(G, \texttt{input})\ S\ (G, \texttt{output})$: spanning tree, leader election, ...
- Is that enough to define the problem?
  - Different kinds of computability, depending on the kind of termination we are interested in.
  - Reliability?

Anonymous Networks and Reliability

A network is *anonymous* when the initial (node) labeling is uniform.

- **Motivations:**

Anonymous Networks and Reliability

A network is *anonymous* when the initial (node) labeling is uniform.

- **Motivations:**
  - Distributed computability in a general setting,

Anonymous Networks and Reliability

A network is *anonymous* when the initial (node) labeling is uniform.

- **Motivations:**
  - Distributed computability in a general setting,
  - Transient Faults: components failure, attacks, ...

Anonymous Networks and Reliability

A network is *anonymous* when the initial (node) labeling is uniform.

- **Motivations:**
  - Distributed computability in a general setting,
  - Transient Faults: components failure, attacks, ...
- **Potentially Anonymous:** arbitrary initial labeling : unique identities, or “partially unique” (pseudonymous) ...

Anonymous Networks and Reliability

A network is *anonymous* when the initial (node) labeling is uniform.

- **Motivations:**
  - Distributed computability in a general setting,
  - Transient Faults: components failure, attacks, ...
- **Potentially Anonymous:** arbitrary initial labeling : unique identities, or "partially unique" (pseudonymous) ...
- **Communication is reliable**

Anonymous Networks and Reliability

A network is *anonymous* when the initial (node) labeling is uniform.

- **Motivations:**
  - Distributed computability in a general setting,
  - Transient Faults: components failure, attacks, ...
- **Potentially Anonymous:** arbitrary initial labeling : unique identities, or “partially unique” (pseudonymous) ...
- **Communication is reliable**

Anonymous Networks and Reliability

A network is *anonymous* when the initial (node) labeling is uniform.

- **Motivations:**
  - Distributed computability in a general setting,
  - Transient Faults: components failure, attacks, ...
- **Potentially Anonymous:** arbitrary initial labeling : unique identities, or “partially unique” (pseudonymous) ...
- **Communication is reliable**

Computability in resource-constrained autonomous large scale systems

- Evolution of the network,
- Impossibility results: properties of the network that have to be “centrally” maintained.
- Possibility results: with efficient algorithms?

Motivations for Local Termination

Different kinds of termination exist for distributed algorithms.

- Implicit termination: processes do not know that the computation is over.
  - [Boldi and Vigna '02] Characterization of tasks computable with implicit termination,
  - based on fibrations and coverings,
  - equivalence with self-stabilizing (terminating) tasks.

Motivations for Local Termination

Different kinds of termination exist for distributed algorithms.

- Implicit termination: processes do not know that the computation is over.
  - [Boldi and Vigna '02] Characterization of tasks computable with implicit termination,
  - based on fibrations and coverings,
  - equivalence with self-stabilizing (terminating) tasks.
- Local termination: at some point, each process knows that it has computed its final value.

Motivations for Local Termination

Different kinds of termination exist for distributed algorithms.

- Implicit termination: processes do not know that the computation is over.
  - [Boldi and Vigna ’02] Characterization of tasks computable with implicit termination,
  - based on fibrations and coverings,
  - equivalence with self-stabilizing (terminating) tasks.
- Local termination: at some point, each process knows that it has computed its final value.
- Global termination detection: processes know that all processes have computed their final value.
  - [C., Godard, Métivier, Tel ’07] Characterization of tasks computable with global termination detection
  - based on coverings and quasi-coverings

Local Terminations

- Local Termination:
  - a process knows when it can stop executing the algorithm.
  - [Boldi and Vigna ’99] Characterization of tasks computable with local termination
  - based on view construction

Local Terminations

- Local Termination:
  - a process knows when it can stop executing the algorithm.
  - [Boldi and Vigna '99] Characterization of tasks computable with local termination
  - based on view construction
- Weak Local Termination:
  - a process knows when it has computed its final value, but can still run the algorithm (processing and forwarding other nodes messages).

Applications

Two distributed algorithms are composed whenever the second uses as inputs the outputs of the first algorithm.

- **weak local termination:** composition of distributed algorithms,
- **local termination:** garbage collection.

Applications

Two distributed algorithms are composed whenever the second uses as inputs the outputs of the first algorithm.

- **weak local termination:** composition of distributed algorithms,
- **local termination:** garbage collection.

A General Framework

Termination appears a key parameter to unify distributed computability results in communication networks.

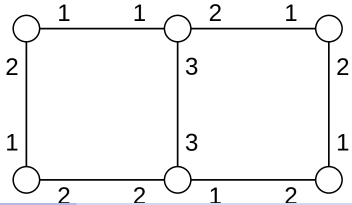
Older Related Works

- Angluin ('80)
- Yamashita & Kameda ('95)
- Boldi & Vigna ('99,'01)
- Mazurkiewicz ('97)
- Metivier et al (90's)
- ...

Model: Reliable Message Passing Systems with Transient Faults

A network is represented as a graph G with a port-numbering δ where each process can

- modify its state,
- send a message via port p,
- receive a message via port q.



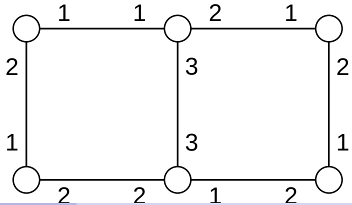
Model: Reliable Message Passing Systems with Transient Faults

A network is represented as a graph G with a port-numbering δ where each process can

- modify its state,
- send a message via port p,
- receive a message via port q.

We consider

- reliable communications,
- anonymous systems.
- asynchronous systems.



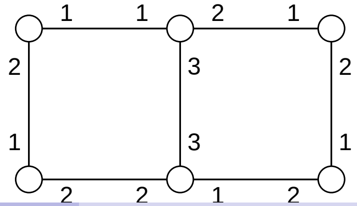
Model: Reliable Message Passing Systems with Transient Faults

A network is represented as a graph G with a port-numbering δ where each process can

- modify its state,
- send a message via port p,
- receive a message via port q.

We consider

- reliable communications,
- (potentially) anonymous systems.
- asynchronous systems.



Examples of Tasks with Different Terminations

Problem

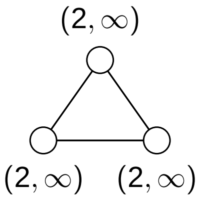
- $v \in V$ is labeled by $(\text{val}(v), \text{dist}(v)) \in \mathbb{N} \times \mathbb{N} \cup \{\infty\}$,
- $v \in V$ should compute $\max\{\text{val}(u) \mid u \text{ is at distance at most dist}(v) \text{ of } v\}$.

Examples of Tasks with Different Terminations

Problem

- $v \in V$ is labeled by $(\mathrm{val}(v), \mathrm{dist}(v)) \in \mathbb{N} \times \mathbb{N} \cup \{\infty\}$,
- $v \in V$ should compute $\max\{\mathrm{val}(u) \mid u \text{ is at distance at most } \mathrm{dist}(v) \text{ of } v\}$.

- If $\mathrm{dist}(v) = \infty$: implicit termination is OK but weak local termination is impossible

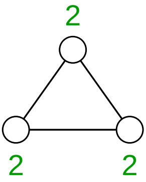


Examples of Tasks with Different Terminations

Problem

- $v \in V$ is labeled by $(\mathrm{val}(v), \mathrm{dist}(v)) \in \mathbb{N} \times \mathbb{N} \cup \{\infty\}$,
- $v \in V$ should compute $\max\{\mathrm{val}(u) \mid u$ is at distance at most $\mathrm{dist}(v)$ of $v\}$.

- If $\mathrm{dist}(v) = \infty$: implicit termination is OK but weak local termination is impossible



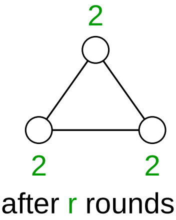
after r rounds

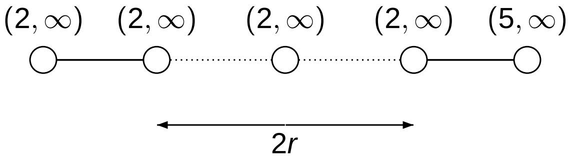
Examples of Tasks with Different Terminations

Problem

- $v \in V$ is labeled by $(\text{val}(v), \text{dist}(v)) \in \mathbb{N} \times \mathbb{N} \cup \{\infty\}$,
- $v \in V$ should compute $\max\{\text{val}(u) \mid u \text{ is at distance at most } \text{dist}(v) \text{ of } v\}$.

- If $\text{dist}(v) = \infty$: implicit termination is OK but weak local termination is impossible



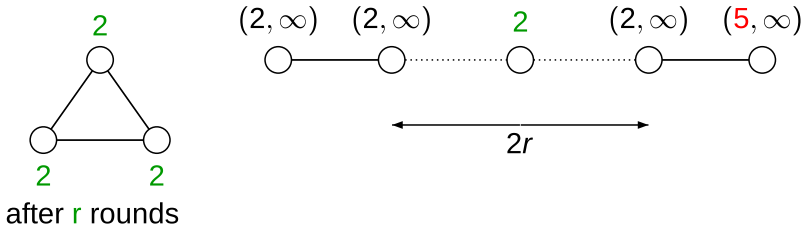


Examples of Tasks with Different Terminations

Problem

- $v \in V$ is labeled by $(\mathrm{val}(v), \mathrm{dist}(v)) \in \mathbb{N} \times \mathbb{N} \cup \{\infty\}$,
- $v \in V$ should compute $\max\{\mathrm{val}(u) \mid u$ is at distance at most $\mathrm{dist}(v)$ of $v\}$.

- If $\mathrm{dist}(v) = \infty$: implicit termination is OK but weak local termination is impossible

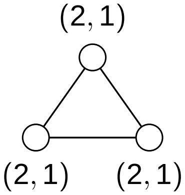


Examples of Tasks with Different Terminations

Problem

- $v \in V$ is labeled by $(\text{val}(v), \text{dist}(v)) \in \mathbb{N} \times \mathbb{N}$,
- $v \in V$ should compute $\max\{\text{val}(u) \mid u \text{ is at distance at most } \text{dist}(v) \text{ of } v\}$.

- If $\text{dist}(v) \in \mathbb{N}$: weak local termination is OK but local termination is impossible

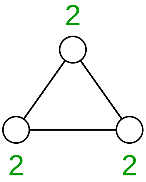


Examples of Tasks with Different Terminations

Problem

- $v \in V$ is labeled by $(\text{val}(v), \text{dist}(v)) \in \mathbb{N} \times \mathbb{N}$,
- $v \in V$ should compute $\max\{\text{val}(u) \mid u \text{ is at distance at most } \text{dist}(v) \text{ of } v\}$.

- If $\text{dist}(v) \in \mathbb{N}$: weak local termination is OK but local termination is impossible



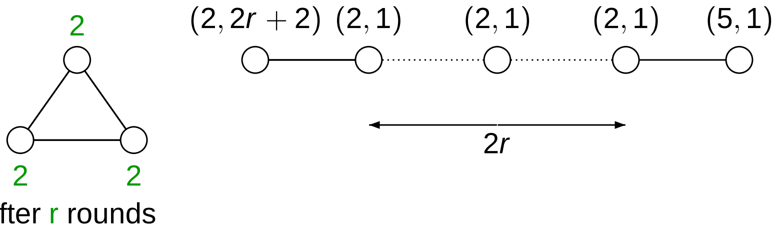
after r rounds

Examples of Tasks with Different Terminations

Problem

- $v \in V$ is labeled by $(\mathrm{val}(v), \mathrm{dist}(v)) \in \mathbb{N} \times \mathbb{N}$,
- $v \in V$ should compute $\max\{\mathrm{val}(u) \mid u \text{ is at distance at most } \mathrm{dist}(v) \text{ of } v\}$.

- If $\mathrm{dist}(v) \in \mathbb{N}$: weak local termination is OK but local termination is impossible

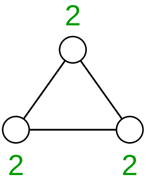


Examples of Tasks with Different Terminations

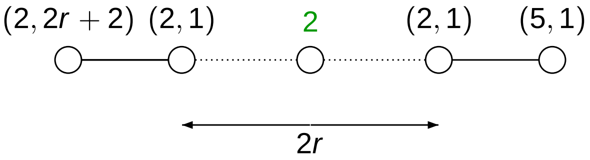
Problem

- $v \in V$ is labeled by $(\mathrm{val}(v), \mathrm{dist}(v)) \in \mathbb{N} \times \mathbb{N}$,
- $v \in V$ should compute $\max\{\mathrm{val}(u) \mid u \text{ is at distance at most } \mathrm{dist}(v) \text{ of } v\}$.

- If $\mathrm{dist}(v) \in \mathbb{N}$: weak local termination is OK but local termination is impossible



after r rounds

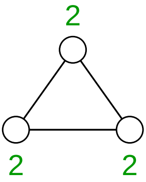


Examples of Tasks with Different Terminations

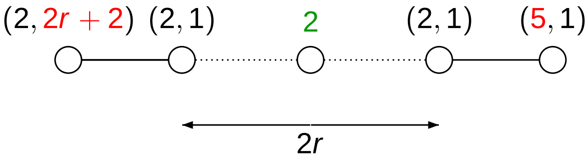
Problem

- $v \in V$ is labeled by $(\text{val}(v), \text{dist}(v)) \in \mathbb{N} \times \mathbb{N}$,
- $v \in V$ should compute
  $\max\{\text{val}(u) \mid u \text{ is at distance at most } \text{dist}(v) \text{ of } v\}$.

- If $\text{dist}(v) \in \mathbb{N}$: weak local termination is OK but local termination is impossible



after r rounds

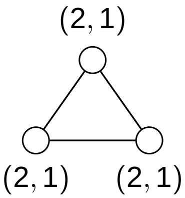


Examples of Tasks with Different Terminations

Problem

- $v \in V$ is labeled by $(\text{val}(v), \text{dist}(v)) \in \mathbb{N} \times \mathbb{N}$,
- $v \in V$ should compute $\max\{\text{val}(u) \mid u \text{ is at distance at most dist}(v) \text{ of } v\}$.

- If for neighbors u, v, $|\text{dist}(u) - \text{dist}(v)| \leq 1$: local termination is OK but global termination is impossible

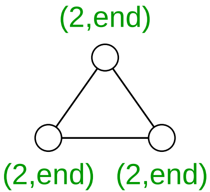


Examples of Tasks with Different Terminations

Problem

- $v \in V$ is labeled by $(\text{val}(v), \text{dist}(v)) \in \mathbb{N} \times \mathbb{N}$,
- $v \in V$ should compute $\max\{\text{val}(u) \mid u \text{ is at distance at most } \text{dist}(v) \text{ of } v\}$.

- If for neighbors u, v, $|\text{dist}(u) - \text{dist}(v)| \leq 1$: local termination is OK but global termination is impossible



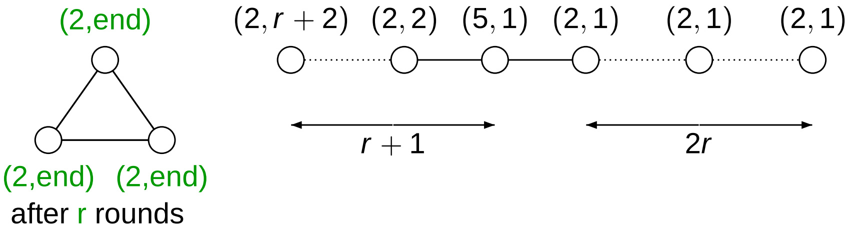
after r rounds

Examples of Tasks with Different Terminations

Problem

- $v \in V$ is labeled by $(\text{val}(v), \text{dist}(v)) \in \mathbb{N} \times \mathbb{N}$,
- $v \in V$ should compute $\max\{\text{val}(u) \mid u \text{ is at distance at most } \text{dist}(v) \text{ of } v\}$.

- If for neighbors u, v, $|\text{dist}(u) - \text{dist}(v)| \leq 1$: local termination is OK but global termination is impossible

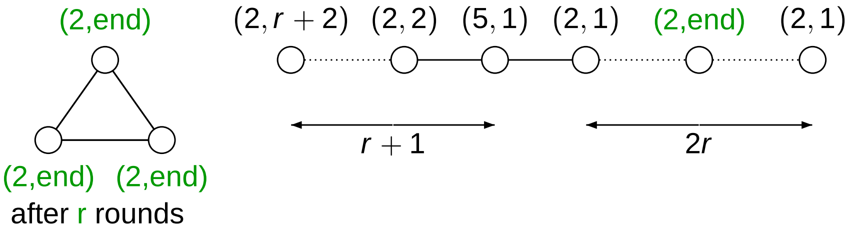


Examples of Tasks with Different Terminations

Problem

- $v \in V$ is labeled by $(\text{val}(v), \text{dist}(v)) \in \mathbb{N} \times \mathbb{N}$,
- $v \in V$ should compute
  $\max\{\text{val}(u) \mid u \text{ is at distance at most dist}(v) \text{ of } v\}$.

- If for neighbors u, v, $|\text{dist}(u) - \text{dist}(v)| \leq 1$: local termination is OK but global termination is impossible

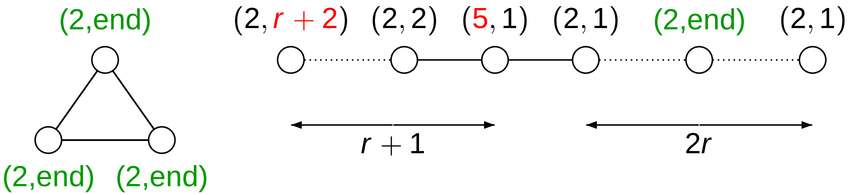


Examples of Tasks with Different Terminations

Problem

- $v \in V$ is labeled by $(\text{val}(v), \text{dist}(v)) \in \mathbb{N} \times \mathbb{N}$,
- $v \in V$ should compute $\max\{\text{val}(u) \mid u \text{ is at distance at most } \text{dist}(v) \text{ of } v\}$.

- If for neighbors u, v, $|\text{dist}(u) - \text{dist}(v)| \leq 1$: local termination is OK but global termination is impossible



after r rounds

Our Contribution

- A characterization of tasks computable with weak local termination
  - based on coverings and quasi-coverings

Our Contribution

- A characterization of tasks computable with weak local termination
  - based on coverings and quasi-coverings
- A characterization of tasks computable with weak local termination by polynomial algorithms
  - algorithms exchanging a polynomial number of bits

Our Contribution

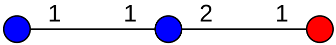
- A characterization of tasks computable with weak local termination
  - based on coverings and quasi-coverings
- A characterization of tasks computable with weak local termination by polynomial algorithms
  - algorithms exchanging a polynomial number of bits
- A new characterization of tasks computable with local termination .
  - based on coverings and quasi-coverings

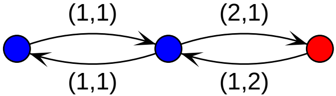
Our Contribution

- A characterization of tasks computable with weak local termination
  - based on coverings and quasi-coverings
- A characterization of tasks computable with weak local termination by polynomial algorithms
  - algorithms exchanging a polynomial number of bits
- A characterization of tasks computable with local termination by polynomial algorithms.
  - based on coverings and quasi-coverings

From graphs to digraphs

We represent the network by a labeled digraph (G, δ, λ) where the state of each process is encoded by its label.

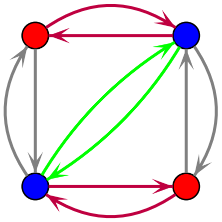




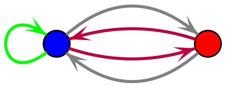
Coverings

Definition

D is a covering of D' via φ if φ is a locally bijective homomorphism.



D



D'

Coverings

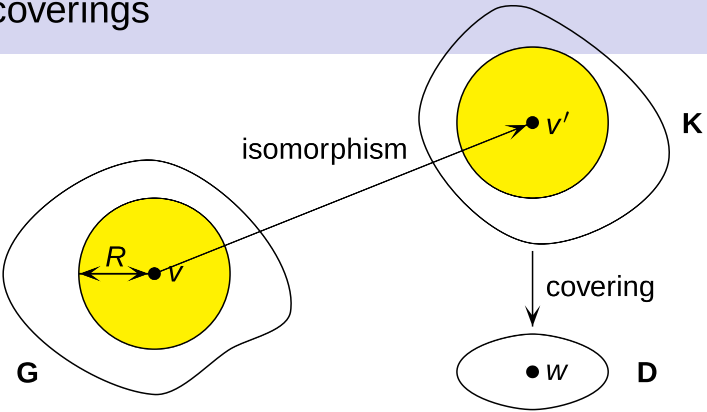
Definition

D is a covering of D' via φ if φ is a locally bijective homomorphism.

Lifting Lemma

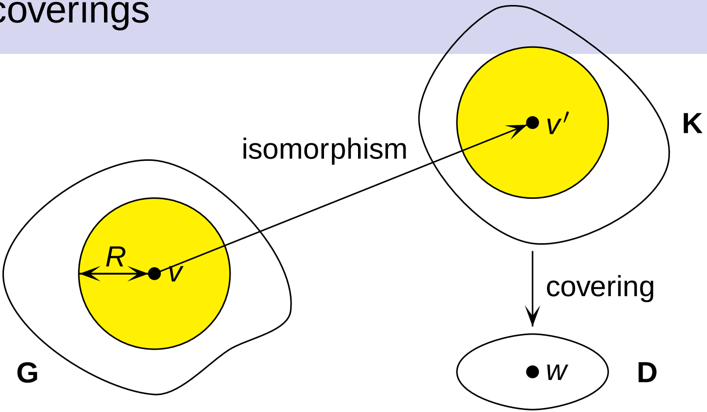
If **D** is a covering of **D**′ via φ, in the synchronous execution of any algorithm $\mathcal{A}$, v and $\varphi(v)$ behave in the same way.

Quasi-coverings



Definition: **G** is a quasi-covering of **D** of radius *R* of center *v*.

Quasi-coverings



Definition: **G** is a quasi-covering of **D** of radius R of center v.

Quasi-Lifting Lemma

In the synchronous execution of any algorithm $\mathcal{A}$, v and w behave in the same way during the first R rounds.

Necessary Condition

Suppose there exists an algorithm $\mathcal{A}$ that computes $(\mathcal{F}, S)$.

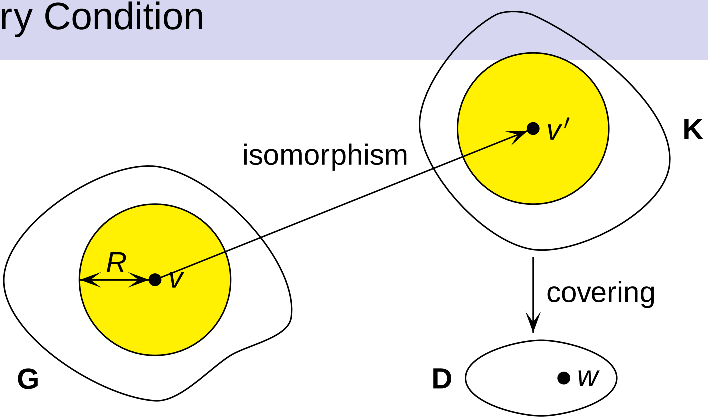
Given any digraph **G**, consider the synchronous execution of $\mathcal{A}$ over **G**.

For any $v \in V(G)$, let i be the first round (if it exists) where $\texttt{term}_i(v) = \textsc{Term}$, then we define:

- $\text{result}(\mathbf{G}, v) = \texttt{output}_i(v)$,
- $\text{time}(\mathbf{G}, v) = i$.

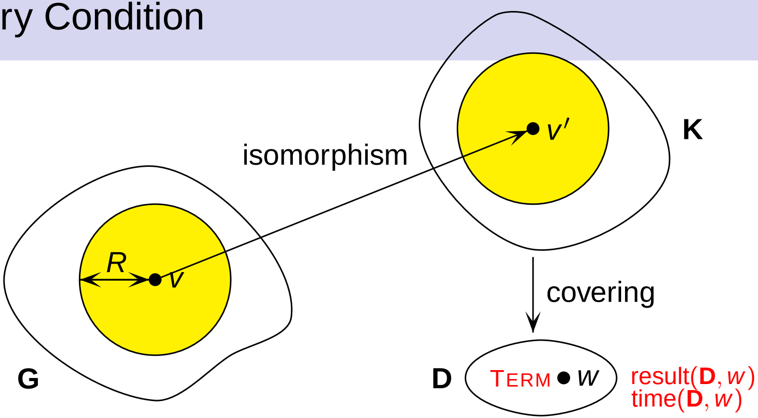
Otherwise, we let $\text{result}(\mathbf{G}, v) = \perp$ and $\text{time}(\mathbf{G}, v) = \infty$.

Necessary Condition



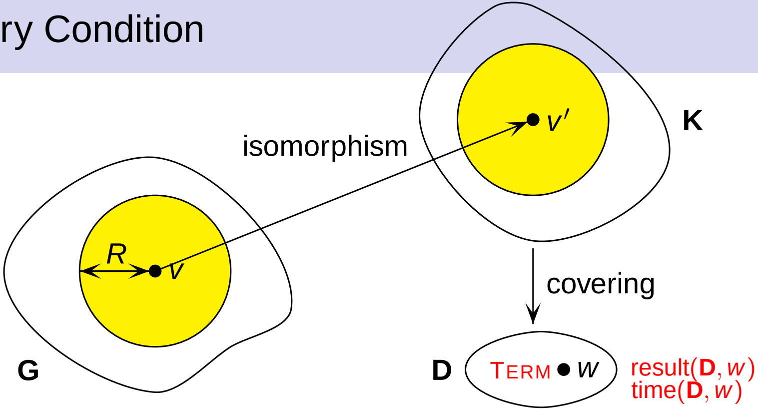
- During the first R rounds of the execution, v and w remain in the same state.

Necessary Condition



- During the first R rounds of the execution, v and w remain in the same state.
- If $R \geq \text{time}(\mathbf{D}, w)$ or $R \geq \text{time}(\mathbf{G}, w)$),
  $\text{result}(\mathbf{D}, w) = \text{result}(\mathbf{G}, v)$ and $\text{time}(\mathbf{D}, w) = \text{time}(\mathbf{G}, v)$.

Necessary Condition



Key Property

Two functions time and result have property (P) if for any $\mathbf{G}, \mathbf{D}$,

- if **G** is a quasi-covering of **D** of radius R
- if $R \geq \min\{\text{time}(\mathbf{G}, v), \textit{time}(\mathbf{D}, w)\}$

then $\text{time}(\mathbf{G}, v) = \text{time}(\mathbf{D}, v)$ and $\text{result}(\mathbf{G}, v) = \text{result}(\mathbf{D}, v)$

Weak Local Termination

Theorem

A task $(\mathcal{F}, S)$ can be computed on $\mathcal{F}$ with weak local termination

$$\Longleftrightarrow$$

there exists some functions time and result computable on $\mathcal{F}$ such that

- **G** *S result*(**G**)
- *time and result have Property (P)*

Weak Local Termination

Theorem

A task $(\mathcal{F}, S)$ *can be computed on* $\mathcal{F}$ *with weak local termination by a polynomial algorithm*

$$\Longleftrightarrow$$

there exists some functions time and result computable on $\mathcal{F}$ *such that*

- $\mathbf{G}\ S\ result(\mathbf{G})$
- *time and result have Property* (P)
- *there exists a polynomial p such that*
  - $\forall \mathbf{G} \in \mathcal{F}$, $\max\{time(\mathbf{G}, v) \mid v \in V(G)\} \leq p(|\mathbf{G}|)$

An Algorithm for the Sufficient Condition

A Mazurkiewicz-like Algorithm $\mathcal{M}$

When executed on a graph **G**,

- \+ it reconstructs **D** such that **G** is a covering of **D**
- \+ it is a polynomial algorithm

An Algorithm for the Sufficient Condition

A Mazurkiewicz-like Algorithm $\mathcal{M}$

When executed on a graph **G**,

- + it reconstructs **D** such that **G** is a covering of **D**
- + it is a polynomial algorithm
- – it terminates only implicitly

An Algorithm for the Sufficient Condition

A Mazurkiewicz-like Algorithm $\mathcal{M}$

When executed on a graph **G**,

+ it reconstructs **D** such that **G** is a covering of **D**
+ it is a polynomial algorithm
– it terminates only implicitly
+ at any step i, it enables each process v to reconstruct $\mathbf{D}_i(v)$ such that **G** is a quasi-covering of $\mathbf{D}_i(v)$ of center v

An Algorithm for the Sufficient Condition

A Mazurkiewicz-like Algorithm $\mathcal{M}$

When executed on a graph **G**,

+ it reconstructs **D** such that **G** is a covering of **D**
+ it is a polynomial algorithm

– it terminates only implicitly

+ at any step i, it enables each process v to reconstruct $\mathbf{D}_i(v)$ such that **G** is a quasi-covering of $\mathbf{D}_i(v)$ of center v

– v does not not know the radius of the quasi-covering

An Algorithm for the Sufficient Condition

Szymanski, Shy and Prywes Algorithm

Combined with our algorithm,

+ enables each v to compute a lower bound $\mathrm{lb}(v)$ on the radius of the quasi-covering
+ once $\mathcal{M}$ has implicitly terminated, this lower bound tends to ∞

An Algorithm for the Sufficient Condition

Szymanski, Shy and Prywes Algorithm

Combined with our algorithm,

+ enables each v to compute a lower bound $\text{lb}(v)$ on the radius of the quasi-covering
+ once $\mathcal{M}$ has implicitly terminated, this lower bound tends to ∞

– we have to be careful to avoid infinite computations

Computing the output

At each time step i, each v knows

- a graph $\mathbf{D}(v)$ such that $\mathbf{G}$ is a quasi-covering of $\mathbf{D}(v)$ of center v
- a vertex $w(v)$: the image of v in $\mathbf{D}(v)$
- a lower bound $\text{lb}(v)$ on the radius of the quasi-covering

Computing the output

At each time step i, each v knows

- a graph $\mathbf{D}(v)$ such that $\mathbf{G}$ is a quasi-covering of $\mathbf{D}(v)$ of center v
- a vertex $w(v)$: the image of v in $\mathbf{D}(v)$
- a lower bound $\text{lb}(v)$ on the radius of the quasi-covering

If $\text{lb}(v) \geq \text{time}(\mathbf{G}, v)$, then by Property (P)

- $\text{time}(\mathbf{G}, v) = \text{time}(\mathbf{D}(v), w(v))$
- $\text{result}(\mathbf{G}, v) = \text{result}(\mathbf{D}(v), w(v))$

Computing the output

At each time step i, each v knows

- a graph $\mathbf{D}(v)$ such that $\mathbf{G}$ is a quasi-covering of $\mathbf{D}(v)$ of center v
- a vertex $w(v)$: the image of v in $\mathbf{D}(v)$
- a lower bound $\mathrm{lb}(v)$ on the radius of the quasi-covering

If $\mathrm{lb}(v) \geq \mathrm{time}(\mathbf{G}, v)$, then by Property (P)

- $\mathrm{time}(\mathbf{G}, v) = \mathrm{time}(\mathbf{D}(v), w(v))$
- $\mathrm{result}(\mathbf{G}, v) = \mathrm{result}(\mathbf{D}(v), w(v))$

Problem

If $\mathrm{lb}(v) < \mathrm{time}(\mathbf{G}, v)$, $\mathrm{result}(\mathbf{D}(v), w(v))$ may be undefined if $\mathbf{D}(v)$ is not in $\mathcal{F}$

Computing the output

At each time step i, each v knows

- a graph $\mathbf{D}(v)$ such that $\mathbf{G}$ is a quasi-covering of $\mathbf{D}(v)$ of center v
- a vertex $w(v)$: the image of v in $\mathbf{D}(v)$
- a lower bound $\mathrm{lb}(v)$ on the radius of the quasi-covering

Solution

- v enumerates the graphs of $\mathcal{F}$ until it finds a graph $\mathbf{H}$ that is a quasi-covering of $\mathbf{D}(v)$ of radius $\mathrm{lb}(v)$ of center u via γ such that $\gamma(u) = w(v)$

Computing the output

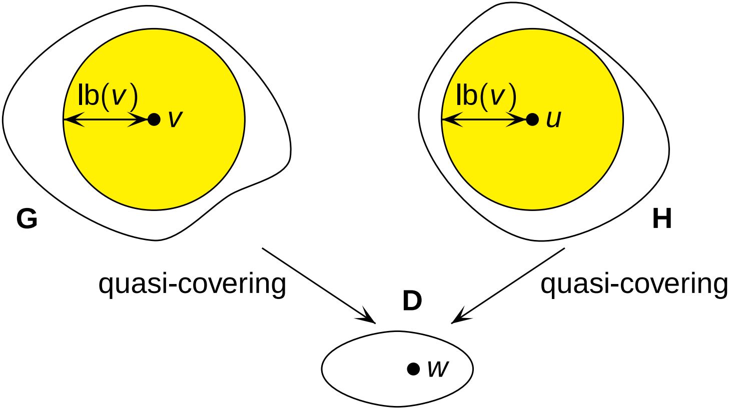
At each time step i, each v knows

- a graph $\mathbf{D}(v)$ such that $\mathbf{G}$ is a quasi-covering of $\mathbf{D}(v)$ of center v
- a vertex $w(v)$: the image of v in $\mathbf{D}(v)$
- a lower bound $\mathrm{lb}(v)$ on the radius of the quasi-covering

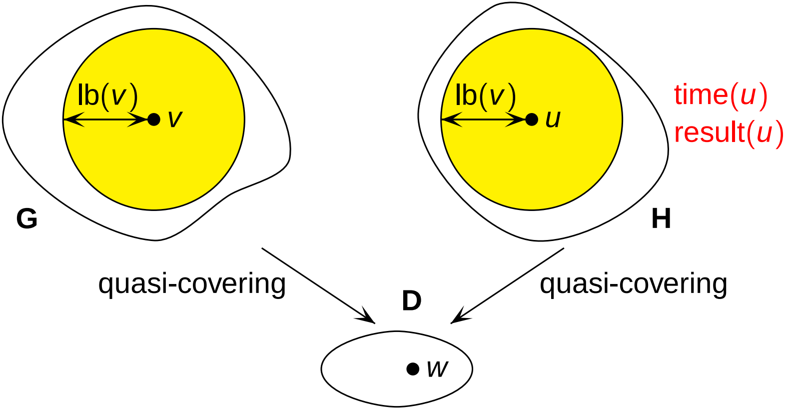
Solution

- v enumerates the graphs of $\mathcal{F}$ until it finds a graph $\mathbf{H}$ that is a quasi-covering of $\mathbf{D}(v)$ of radius $\mathrm{lb}(v)$ of center u via γ such that $\gamma(u) = w(v)$
- If $\mathrm{time}(\mathbf{H}, u) \leq \mathrm{lb}(v)$,
  - $\texttt{output}(v) := \mathrm{result}(\mathbf{H}, u)$

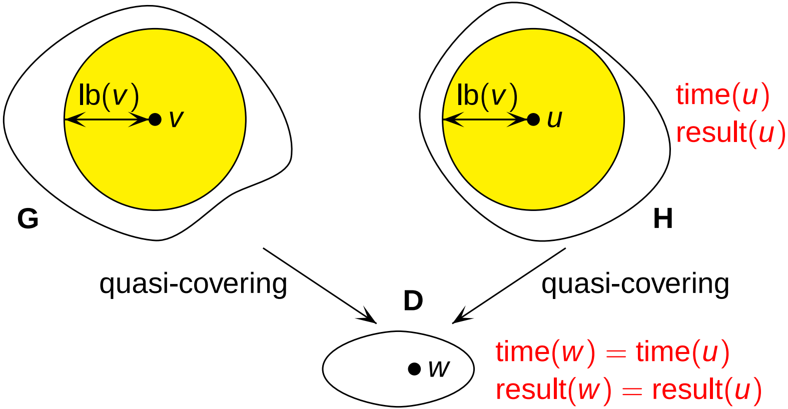
Computing the output



Computing the output



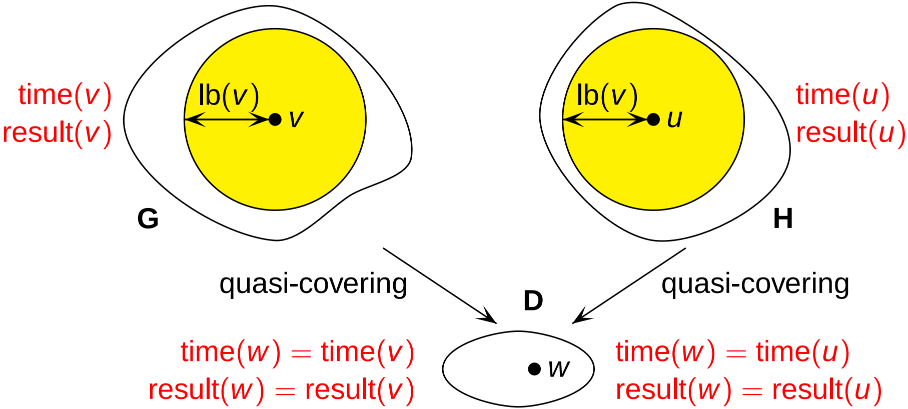
Computing the output



Since time(u) $\leq$ $lb(v)$, by Property (P)

- time(w) = time(u) and result(w) = result(v)

Computing the output



Since $\text{time}(u) \leq lb(v)$, by Property (P)

- $\text{time}(w) = \text{time}(u)$ and $\text{result}(w) = \text{result}(v)$
- $\text{time}(v) = \text{time}(u)$ and $\text{result}(v) = \text{result}(u)$

Complexity

Let $T(\mathbf{G}) = \max\{\text{time}(\mathbf{G}, v) \mid v \in V(G)\}$.

Proposition

Complexity of our Algorithm:

- $O(Dn + T(\mathbf{G}))$ *rounds*
- $O(m^2n + mT(\mathbf{G}))$ *messages*
- $O(\Delta \log n + \log T(\mathbf{G}))$ *bits per message*

Complexity

Let $T(\mathbf{G}) = \max\{\text{time}(\mathbf{G}, v) \mid v \in V(G)\}$.

Proposition

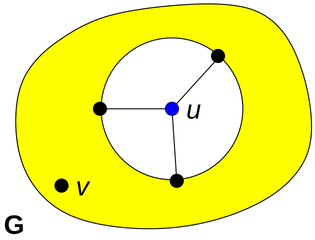
Complexity of our Algorithm:

- $O(Dn + T(\mathbf{G}))$ *rounds*
- $O(m^2 n + mT(\mathbf{G}))$ *messages*
- $O(\Delta \log n + \log T(\mathbf{G}))$ *bits per message*

Corollary

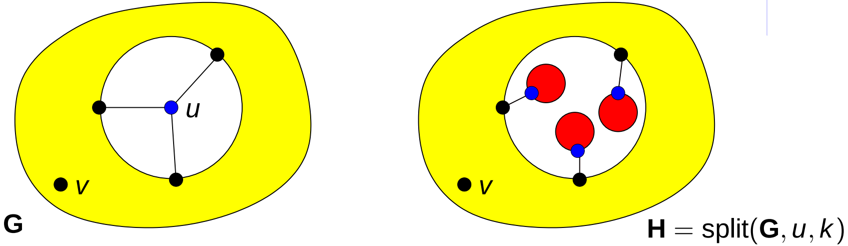
If there exists an algorithm $\mathcal{A}$ that computes a task $(S, \mathcal{F})$ in a polynomial number of rounds (even with big messages), there exists a polynomial algorithm $\mathcal{A}'$ that computes $(S, \mathcal{F})$

Necessary Condition for Local Termination



We define an operator split such that

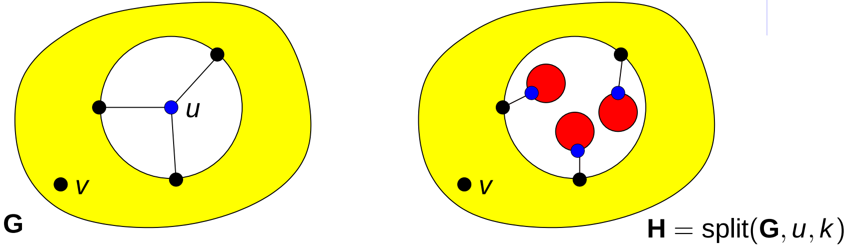
Necessary Condition for Local Termination



We define an operator split such that

- the blue vertices in split($\mathbf{G}, u, k$) behave like u in **G** during k steps

Necessary Condition for Local Termination



Property

Two functions time and result have property (P') if for any $\mathbf{G}$

- if $k = \text{time}(\mathbf{G}, u)$
- then, for any $v \neq u$, $\text{result}(\mathbf{G}, v) = \text{result}(\mathbf{H}, v)$ and $\text{time}(\mathbf{G}, v) = \text{time}(\mathbf{H}, v)$

Local Termination

Theorem

A task $(\mathcal{F}, S)$ can be computed on $\mathcal{F}$ with local termination

$$\iff$$

there exists some functions time and result computable on $\mathcal{F}$ such that

- $\mathbf{G}$ *S result*$(\mathbf{G})$
- *time and result have Property* (P) *and* (P')

Local Termination

Theorem

A task $(\mathcal{F}, S)$ can be computed on $\mathcal{F}$ with local termination by a polynomial algorithm

$$\Longleftrightarrow$$

there exists some functions time and result computable on $\mathcal{F}$ such that

- $\mathbf{G}$ *S result*$(\mathbf{G})$
- *time and result have Property* (P) *and* (P')
- *there exists a polynomial p such that*
  - $\forall \mathbf{G} \in \mathcal{F}$, $\max\{time(\mathbf{G}, v) \mid v \in V(G)\} \leq p(|\mathbf{G}|)$

Conclusion

- Modeling arbitrary networks with reliable communications and “transient” faults.

Conclusion

- Modeling arbitrary networks with reliable communications and “transient” faults.
- Unifying questions: via termination detection,

Conclusion

- Modeling arbitrary networks with reliable communications and “transient” faults.
- Unifying questions: via termination detection,
- Unifying results of characterizations:

Conclusion

- Modeling arbitrary networks with reliable communications and "transient" faults.
- Unifying questions: via termination detection,
- Unifying results of characterizations:
  - coverings, quasi-coverings,

Conclusion

- Modeling arbitrary networks with reliable communications and “transient” faults.
- Unifying questions: via termination detection,
- Unifying results of characterizations:
  - coverings, quasi-coverings,
  - polynomial algorithms.

Perspectives

- Exploiting characterization results,

Perspectives

- Exploiting characterization results,
- Extensions to self-stabilizing systems (and other self-$*$),

Perspectives

- Exploiting characterization results,
- Extensions to self-stabilizing systems (and other self-$*$),
- Others constraints:

Perspectives

- Exploiting characterization results,
- Extensions to self-stabilizing systems (and other self-$*$),
- Others constraints:
  - Communications,

Perspectives

- Exploiting characterization results,
- Extensions to self-stabilizing systems (and other self-$*$),
- Others constraints:
  - Communications,
  - Interactions/scheduling,

Perspectives

- Exploiting characterization results,
- Extensions to self-stabilizing systems (and other self-$*$),
- Others constraints:
  - Communications,
  - Interactions/scheduling,
  - Faults, dynamic networks,

Perspectives

- Exploiting characterization results,
- Extensions to self-stabilizing systems (and other self-$*$),
- Others constraints:
  - Communications,
  - Interactions/scheduling,
  - Faults, dynamic networks,
  - ...

Thank you