

# An Application-Level Network Mapper

Arnaud Legrand<sup>1</sup>   Frédéric Mazoit<sup>2</sup>   Martin Quinson<sup>1</sup>.

1: ID – UMR 5132 (CNRS – INPG – INRIA – UJF), Grenoble, France.

2: LIP – UMR 5668 (CNRS – ENS-Lyon – UCBL – INRIA), Lyon, France.

# Outline

- Introduction
  - Context
  - Motivation and goals
  - State of the art
- ALNeM
  - Model used
  - Measurement methodology
  - Problem statement
- Mathematical tools, algorithms
  - Total interference and separators
  - Reconstruction trees and cliques of trees
  - Extension for cycles
- Implementation
  - Data collection
- Conclusion
  - Contributions and future work

# Introduction (1/2)

## Motivation

- Modern platforms (Grid, P2P systems) heterogeneous and dynamic.
- Distributed applications have to be reactive and network-aware.
- Quantitative information (bandwidth) well studied [NWS, RPS, ganglia].
- Qualitative information (topology) seldom known, but needed for:
  - Host siting and automatic configuration
  - Group communication

## Definitions of topology

Almost as many as layers in the OSI model.

- Physical interconnexion map (wires in the walls)
- Routing infrastructure (path of network packets, from router to switch)
- Application level (focus on effects – bandwidth, latency – not causes)

# Introduction (2/2)

## Our context is at application level

Grid or P2P systems = multi-organization platforms.

- System heterogeneity  $\Rightarrow$  cannot rely on specific system feature
- Trust issue  $\Rightarrow$  no privileges for grid administrators (“root” or other)

## Our Goal is...

Discover What Applications can Expect from the Platform

Given 4 hosts ( $a, b, c, d$ ), determine whether  $a \rightarrow b$  impact  $c \rightarrow d$  (perfs).

**Intuition:** if they share a link, they share the bandwidth.

## Our goal is not...

- Discover network bottleneck and configuration issues
- Discover packet paths

# Topology discovery methodologies

## State of the art

Method	Restriction	Focus	Routers	Notes
SNMP	authorized	path	all	passive, LAN
traceroute	ICMP	path	all	level 3 of OSI
pathchar	root	path	all	link bandwidth, slow
Other tomography	no	path	$d_{in} \neq d_{out}$	tree bipartite [Rabbat03]
ENV	no	interference	some	tree only
ALNeM	no	interference	?	complete graph

# Outline

- Introduction
  - Context
  - Motivation and goals
  - State of the art
- **ALNeM**
  - **Model used**
  - **Measurement methodology**
  - **Problem statement**
- Mathematical tools, algorithms
  - Total interference and separators
  - Reconstruction trees and cliques of trees
  - Extension for cycles
- Implementation
  - Data collection
- Conclusion
  - Contributions and future work

# Model used

**Definition: routed graph**  $G = (V, E, r)$

Non-oriented graph with *routing function*  $(r : V \times V \rightarrow V)$ .

$(u \xrightarrow{G} v)$  is the path (set of vertices encountered in the graph  $G$ ).

**Definition:  $(ab)$  interfere with  $(cd)$  in  $G$**

$$(ab) \chi_G (cd) \iff \left( a \xrightarrow{G} b \right) \cap \left( c \xrightarrow{G} d \right) \neq \emptyset$$

Symmetric relation:  $(ab) \chi_G (cd) \iff (cd) \chi_G (ab)$

Routing not symmetric:  $(ab) \chi_G (cd) \not\iff (ab) \chi_G (dc)$

**Definition:  $(ab)$  does not interfere with  $(cd)$  in  $G$**

$$(ab) \parallel_G (cd) \iff \neg \left( (ab) \chi_G (cd) \right)$$

# Measurement methodology

## Notation

$bw(ab)$ : bandwidth on  $a \rightarrow b$ .

$bw_{//cd}(ab)$ : bandwidth on  $a \rightarrow b$  when  $c \rightarrow d$  is saturated.

## Definition of the measured interference

$$(ab) \chi_{mes}(cd) \iff \frac{bw_{//cd}(ab)}{bw(ab)} < 0.7 \quad ; \quad (ab) //_{mes}(cd) \text{ if ratio} > 0.9$$

Not symmetric:  $a \xleftrightarrow{10 \text{ Mo/s}} c \xleftrightarrow{100 \text{ Mo/s}} b \xleftrightarrow{100 \text{ Mo/s}} d$   $(ab) \chi_{mes}(cd)$  and  $(cd) //_{mes}(ab)$ .

## Definition of the “real” interference (to reintroduce symmetry)

$$(ab) \chi_r(cd) \iff \begin{cases} (ab) \chi_{mes}(cd) \\ (cd) \chi_{mes}(ab) \end{cases} \text{ (or)} \iff \neg(ab) //_r(cd)$$



# Problem statement

## Notations

$\mathcal{H}$  : set of nodes

Interference matrix  $I(\mathcal{H}, \chi_{rl})$ :

$$I(\mathcal{H}, \chi_{rl})_{(a,b,c,d)} = \begin{cases} 1 & \text{if } (ab) \chi_{rl} (cd) \\ 0 & \text{else} \end{cases}$$

## Definition

INTERFERENCEGRAPH: Given  $\mathcal{H}$  and  $I(\mathcal{H}, \chi_{\bar{G}})$ , find a routed graph  $G = (V, E, r)$  such that:

$$\begin{cases} \mathcal{H} \subset V; \\ I(\mathcal{H}, \chi_{\bar{G}}) = I(\mathcal{H}, \chi_G); \\ |V| \text{ is minimal.} \end{cases}$$

# Outline

- Introduction
  - Context
  - Motivation and goals
  - State of the art
- ALNeM
  - Model used
  - Measurement methodology
  - Problem statement
- **Mathematical tools, algorithms**
  - Total interference and separators
  - Reconstruction trees and cliques of trees
  - Extension for cycles
- Implementation
  - Data collection
- Conclusion
  - Contributions and future work

# Mathematical tools: Total interference and separators

## Definition of the **total interference**

$$a \perp b \iff \forall (u, v) \in \mathcal{H}, (au) \chi_{rl} (bv)$$

## Lemma (Separation)

$$a \perp b \iff \exists \rho \in \tilde{V} \ / \ \forall z \in \mathcal{H} : \rho \in (a \rightarrow z) \cap (b \rightarrow z).$$

Such a  $\rho$  is said to be a *separator* of  $a$  and  $b$ .

**Theorem:  $\perp$  is an equivalence relation (under some assumptions)**

Moreover,  $\forall$  equivalence class,  $\exists$  common separator for all pair of elements.

## Theorem (Representativity)

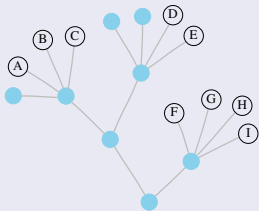
Let  $\mathcal{C}$  be an equivalence class for  $\perp$  and  $\rho$  a separator of its elements.

$$\forall a \in \mathcal{C}, \forall b, u, v \in \mathcal{H}, (a, u) \chi_{rl} (b, v) \iff (\rho, u) \chi_{rl} (b, v)$$

# Reconstructing algorithm

## Handling trees and cliques of trees

Equivalence class  $\Rightarrow$  greedy algorithm *eating* the leaves.

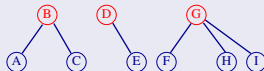
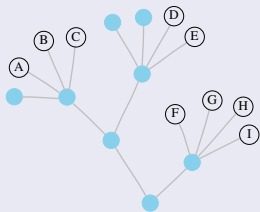


(A) (B) (C) (D) (E) (F) (G) (H) (I)

# Reconstructing algorithm

## Handling trees and cliques of trees

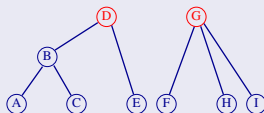
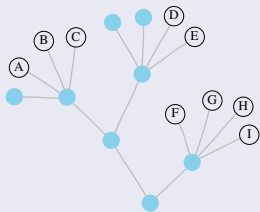
Equivalence class  $\Rightarrow$  greedy algorithm *eating* the leaves.



# Reconstructing algorithm

## Handling trees and cliques of trees

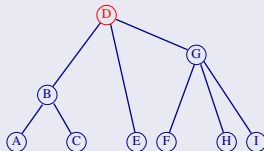
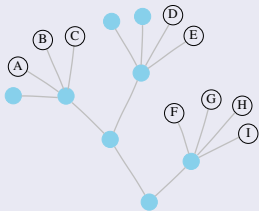
Equivalence class  $\Rightarrow$  greedy algorithm *eating* the leaves.



# Reconstructing algorithm

## Handling trees and cliques of trees

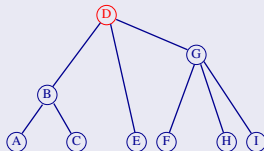
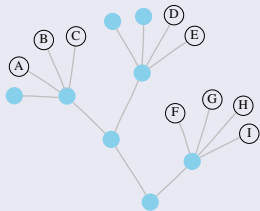
Equivalence class  $\Rightarrow$  greedy algorithm *eating* the leaves.



# Reconstructing algorithm

## Handling trees and cliques of trees

Equivalence class  $\Rightarrow$  greedy algorithm *eating* the leaves.



**Theorem:** When  $|C_{\text{inf}}| = 1$ , the graph built is a solution.

**Theorem:** If a tree being a solution exists,  $|C_{\text{inf}}| = 1$ .

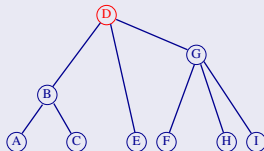
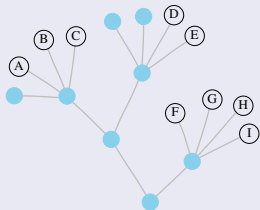
**Remark:** The graph built is optimal (wrt  $|V|$  since  $V = \mathcal{H}$ )



# Reconstructing algorithm

## Handling trees and cliques of trees

Equivalence class  $\Rightarrow$  greedy algorithm *eating* the leaves.



**Theorem:** When  $|C_{\text{inf}}| = 1$ , the graph built is a solution.

**Theorem:** If a tree being a solution exists,  $|C_{\text{inf}}| = 1$ .

**Remark:** The graph built is optimal (wrt  $|V|$  since  $V = \mathcal{H}$ )

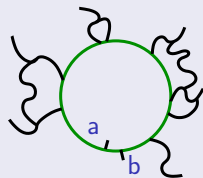
**Theorem:** When no interferences in  $I$ , clique of  $C_i$  is valid solution

**Remark:** It is also optimal

# Reconstructing algorithm: Extension for cycles

## Idea

- Find  $a$  and  $b$  close to each other on a cycle;
- cut the cycle in between;
- iterate previous algorithm;
- reintroduce the cycle: reconnect  $(a, b)$ .



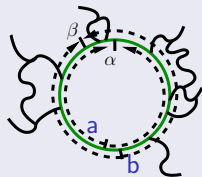
## Finding out how to cut

$a, b$ : nodes with the most interferences (i.e., maximizing  $\{u, v : au \times bv\}$ )

# Reconstructing algorithm: Extension for cycles

## Idea

- Find  $a$  and  $b$  close to each other on a cycle;
- cut the cycle in between;
- iterate previous algorithm;
- reintroduce the cycle: reconnect  $(a, b)$ .



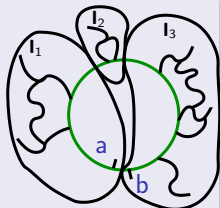
## Finding out how to cut

$a, b$ : nodes with the most interferences (i.e., maximizing  $\{u, v : au \times bv\}$ )

# Reconstructing algorithm: Extension for cycles

## Idea

- Find  $a$  and  $b$  close to each other on a cycle;
- cut the cycle in between;
- iterate previous algorithm;
- reintroduce the cycle: reconnect  $(a, b)$ .



## Finding out how to cut

$a, b$ : nodes with the most interferences (i.e., maximizing  $\{u, v : au \times bv\}$ )

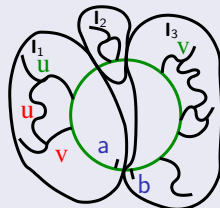
$$\begin{cases} I_1 = \{u \in C_i : a \in (b \rightarrow u) \text{ and } b \notin (a \rightarrow u)\} \\ I_2 = \{u \in C_i : a \notin (b \rightarrow u) \text{ and } b \in (a \rightarrow u)\} \\ I_3 = \{u \in C_i : a \notin (b \rightarrow u) \text{ and } b \notin (a \rightarrow u)\} \\ I_4 = \{u \in C_i : a \in (b \rightarrow u) \text{ and } b \in (a \rightarrow u)\} \end{cases}$$

$$I_4 = \{a; b\} \text{ or } \begin{array}{c} a \\ \curvearrowright \\ \bullet \\ \curvearrowleft \\ b \end{array} \bullet u$$

# Reconstructing algorithm: Extension for cycles

## Idea

- Find  $a$  and  $b$  close to each other on a cycle;
- cut the cycle in between;
- iterate previous algorithm;
- reintroduce the cycle: reconnect  $(a, b)$ .



## Finding out how to cut

$a, b$ : nodes with the most interferences (i.e., maximizing  $\{u, v : au \times bv\}$ )

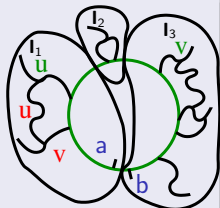
$$\begin{cases} I_1 = \{u \in C_i : a \in (b \rightarrow u) \text{ and } b \notin (a \rightarrow u)\} \\ I_2 = \{u \in C_i : a \notin (b \rightarrow u) \text{ and } b \in (a \rightarrow u)\} \\ I_3 = \{u \in C_i : a \notin (b \rightarrow u) \text{ and } b \notin (a \rightarrow u)\} \\ I_4 = \{u \in C_i : a \in (b \rightarrow u) \text{ and } b \in (a \rightarrow u)\} \end{cases}$$

$\forall u$	$a$	$\alpha$	$\beta$	$b$	
$a$		1	1	1	} $I_1$
$\alpha$		0	0	1	
$\beta$		0	0	1	} $I_2$
$b$		0	0	1	

# Reconstructing algorithm: Extension for cycles

## Idea

- Find  $a$  and  $b$  close to each other on a cycle;
- cut the cycle in between;
- iterate previous algorithm;
- reintroduce the cycle: reconnect  $(a, b)$ .



## Finding out how to cut

$a, b$ : nodes with the most interferences (i.e., maximizing  $\{u, v : au \times bv\}$ )

$$\begin{cases} l_1 = \{u \in C_i : a \in (b \rightarrow u) \text{ and } b \notin (a \rightarrow u)\} \\ l_2 = \{u \in C_i : a \notin (b \rightarrow u) \text{ and } b \in (a \rightarrow u)\} \\ l_3 = \{u \in C_i : a \notin (b \rightarrow u) \text{ and } b \notin (a \rightarrow u)\} \\ l_4 = \{u \in C_i : a \in (b \rightarrow u) \text{ and } b \in (a \rightarrow u)\} \end{cases}$$

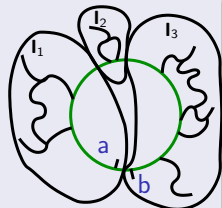
$\begin{matrix} u \\ a \end{matrix}$	$\alpha$	$\beta$	$b$	
$a$	1	1	1	} $l_1$
$\alpha$	0	0	1	
$\beta$	0	0	1	} $l_2$
$b$	0	0	1	

Topological sort on the graph associated to the matrix slice gives  $l_1, l_2, l_3$

# Reconstructing algorithm: Extension for cycles

## Idea

- Find  $a$  and  $b$  close to each other on a cycle;
- cut the cycle in between;
- iterate previous algorithm;
- reintroduce the cycle: reconnect  $(a, b)$ .



## Finding out how to cut

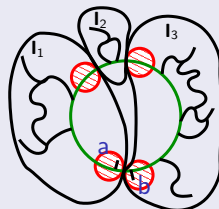
...

## How to connect parts afterward

# Reconstructing algorithm: Extension for cycles

## Idea

- Find  $a$  and  $b$  close to each other on a cycle;
- cut the cycle in between;
- iterate previous algorithm;
- reintroduce the cycle: reconnect  $(a, b)$ .



## Finding out how to cut

...

## How to connect parts afterward

First step on  $I_1 \rightarrow$  Finds 2 classes  $I_{1_a}$  and  $I_{1_\alpha}$ ;  $a \in I_{1_a}$ .

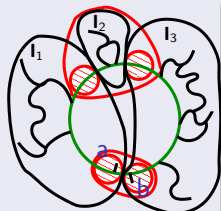
First step on  $I_3 \rightarrow$  Finds 2 classes  $I_{1_b}$  and  $I_{1_\beta}$ ;  $b \in I_{1_b}$ .



# Reconstructing algorithm: Extension for cycles

## Idea

- Find  $a$  and  $b$  close to each other on a cycle;
- cut the cycle in between;
- iterate previous algorithm;
- reintroduce the cycle: reconnect  $(a, b)$ .



## Finding out how to cut

...

## How to connect parts afterward

First step on  $I_1 \rightarrow$  Finds 2 classes  $I_{1_a}$  and  $I_{1_\alpha}$ ;  $a \in I_{1_a}$ .

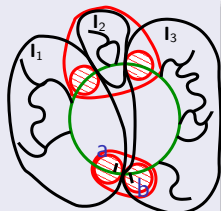
First step on  $I_3 \rightarrow$  Finds 2 classes  $I_{1_b}$  and  $I_{1_\beta}$ ;  $b \in I_{1_b}$ .

Reconnect  $I_{1_a}$  and  $I_{1_b}$ ; Reconnect  $I_{1_\alpha}$  and  $I_{1_\beta}$ .

# Reconstructing algorithm: Extension for cycles

## Idea

- Find  $a$  and  $b$  close to each other on a cycle;
- cut the cycle in between;
- iterate previous algorithm;
- reintroduce the cycle: reconnect  $(a, b)$ .



## Finding out how to cut

...

## How to connect parts afterward

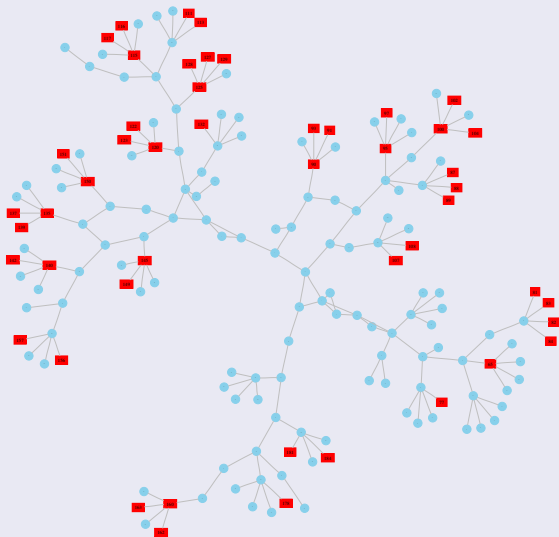
First step on  $I_1 \rightarrow$  Finds 2 classes  $I_{1_a}$  and  $I_{1_\alpha}$ ;  $a \in I_{1_a}$ .

First step on  $I_3 \rightarrow$  Finds 2 classes  $I_{1_b}$  and  $I_{1_\beta}$ ;  $b \in I_{1_b}$ .

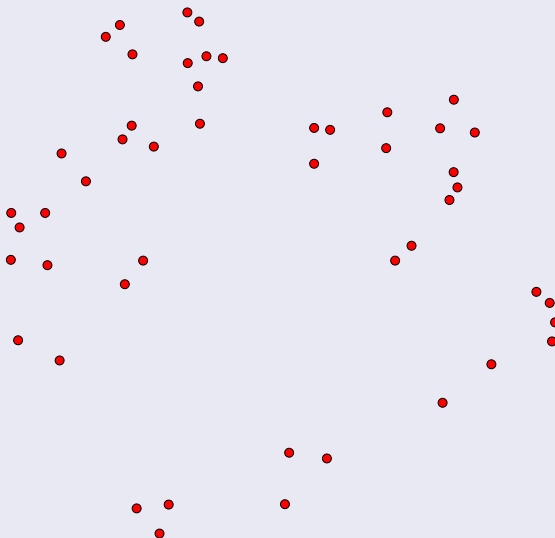
Reconnect  $I_{1_a}$  and  $I_{1_b}$ ; Reconnect  $I_{1_\alpha}$  and  $I_{1_\beta}$ .

No demonstration of this...

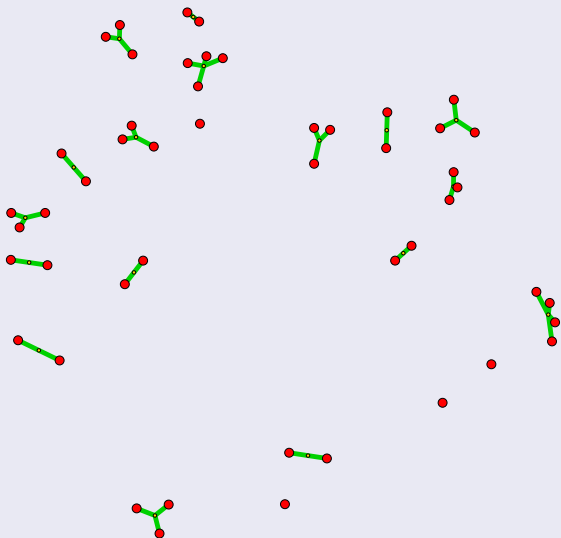
# Example of reconstruction



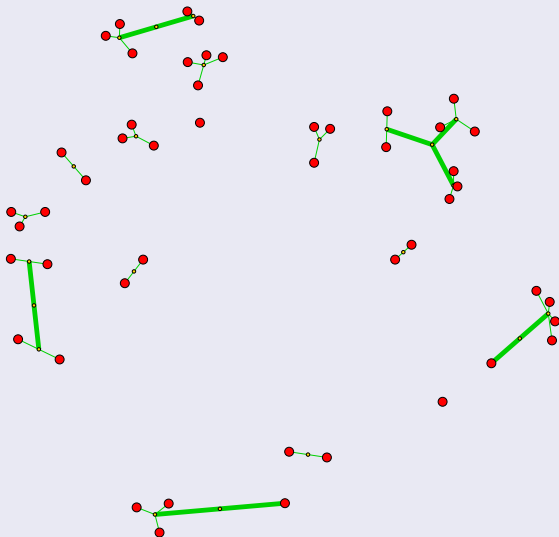
# Example of reconstruction



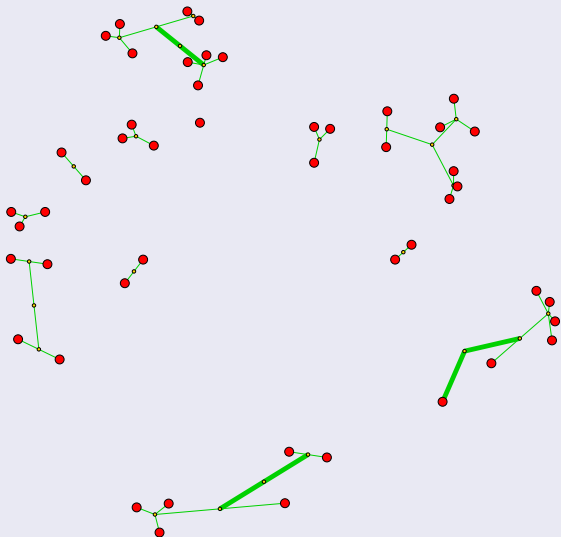
# Example of reconstruction



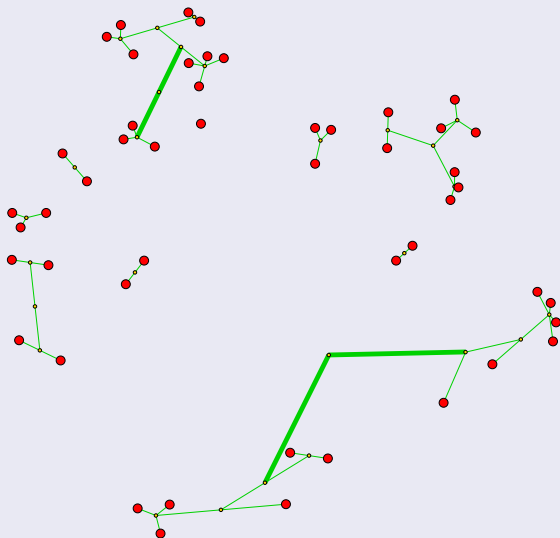
# Example of reconstruction



# Example of reconstruction

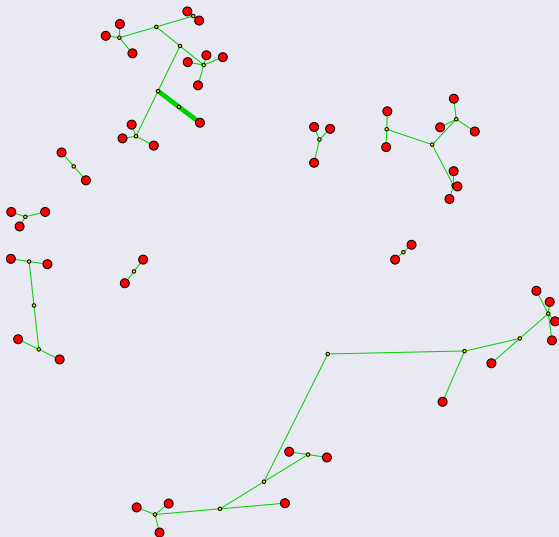


# Example of reconstruction

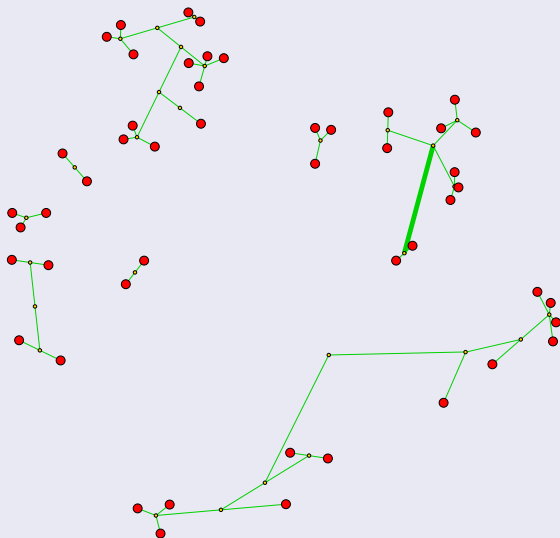




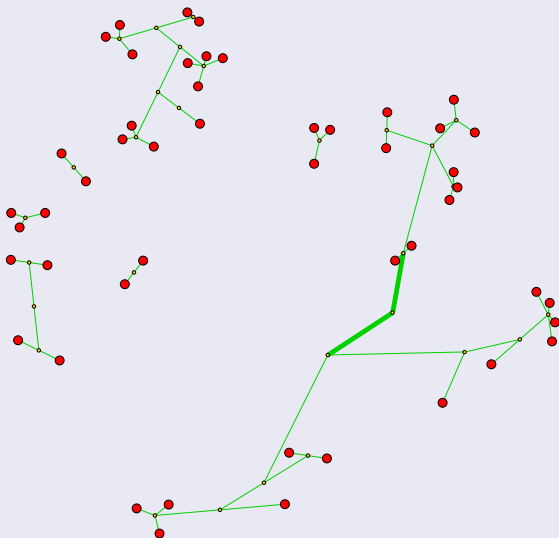
# Example of reconstruction



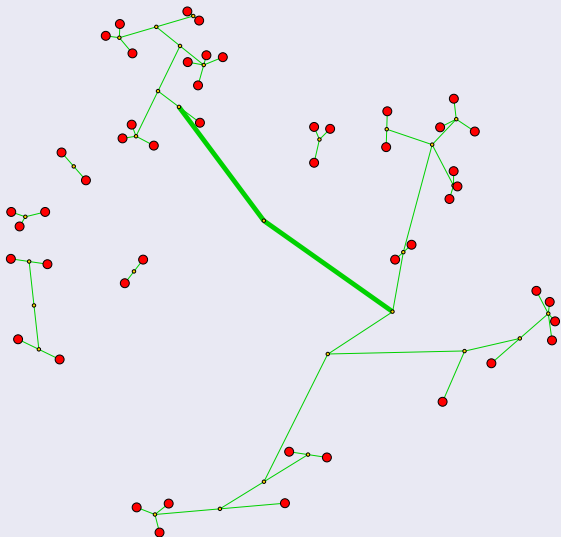
# Example of reconstruction



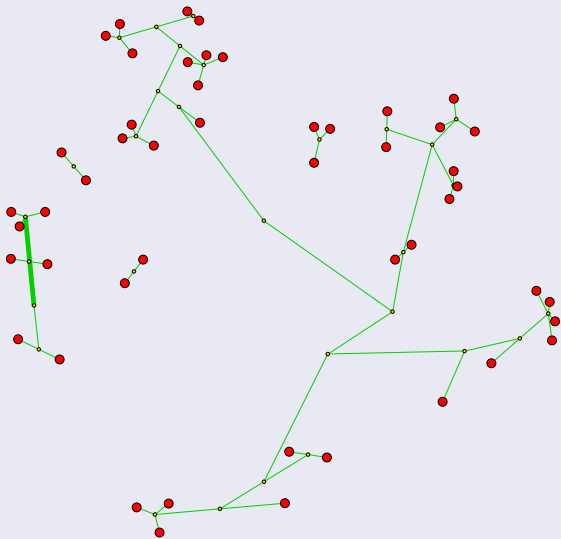
# Example of reconstruction



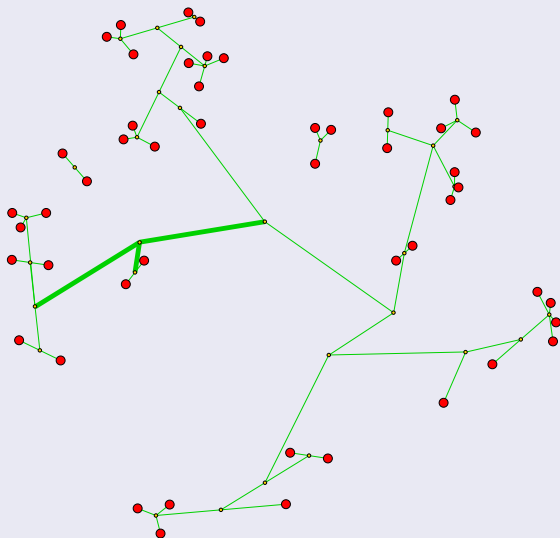
# Example of reconstruction



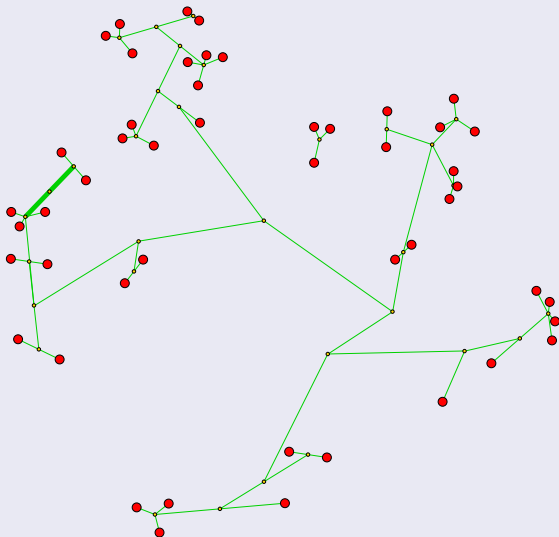
# Example of reconstruction



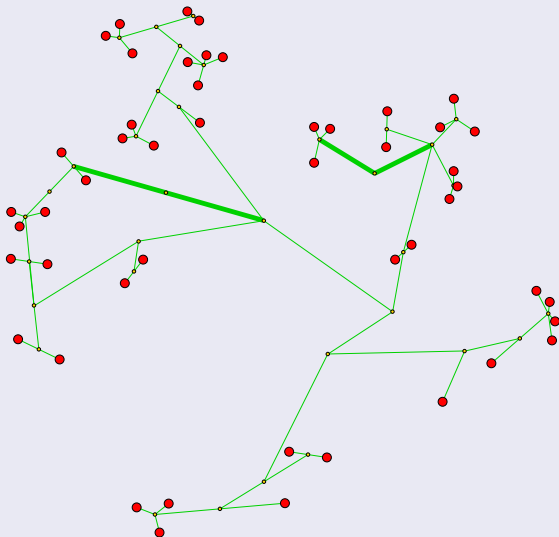
# Example of reconstruction



# Example of reconstruction



# Example of reconstruction





# Outline

- Introduction
  - Context
  - Motivation and goals
  - State of the art
- ALNeM
  - Model used
  - Measurement methodology
  - Problem statement
- Mathematical tools, algorithms
  - Total interference and separators
  - Reconstruction trees and cliques of trees
  - Extension for cycles
- **Implementation**
  - **Data collection**
- Conclusion
  - Contributions and future work

# Data collection

## Intuitive algorithm

- 1 Measure the bandwidth on  $(ab)$  ;
- 2 Measure the bandwidth on  $(ab)$  when the link  $(cd)$  is saturated ;
- 3 Compute the ratio.

$N^4$ , 30s. per step  $\Rightarrow$  50 days for 20 hosts.

## Speeding things up

Using traceroute or other tomography

- Independent tests in parallel
- Validation of information sets

Refinement of existing graph?

# Outline

- Introduction
  - Context
  - Motivation and goals
  - State of the art
- ALNeM
  - Model used
  - Measurement methodology
  - Problem statement
- Mathematical tools, algorithms
  - Total interference and separators
  - Reconstruction trees and cliques of trees
  - Extension for cycles
- Implementation
  - Data collection
- Conclusion
  - Contributions and future work

# Conclusion

## Contributions

- Retrieve the interference-based topology from direct measurements
- Strong mathematical basements (optimal for cliques of trees)
- More generic than ENV (partial cycle handling)
- Based on GRAS (development of distributed applications on simulator)

## Future work

- NP-hardness
- Experimentation on real platform (measurements optimization)
- Iterative algorithm (modification detection)
- Couple measurement and reconstruction phases
- Integration within the NWS (auto-configuration; provide information)