An Application-Level Network Mapper

Arnaud Legrand, Frédéric Mazoit, Martin Quinson

An Application-Level Network Mapper

Arnaud Legrand — Frédéric Mazoit — Martin Quinson

N° 5792

Janvier 2006

Thème NUM

Rapport de recherche

An Application-Level Network Mapper

Arnaud Legrand[*] , Frédéric Mazoit[†] , Martin Quinson[‡]

Abstract: Modern grid platforms present a much more complex interconnection topology than classical super-computers or clusters. Information about this topology is critical to network-aware applications, such as a grid scheduler optimizing communication times, or such as data replication managers.

This paper presents a theoretical framework and the corresponding tool for automatically discovery of network topology. The goal is not to discover the physical layout of machines interconnections, but to construct a synthetic view of the effects of the topology as perceived by an application. Among other things, this requires that performance of concurrent transfers can be assessed.

Our work uncovers an original mathematical model that arises from the formalization of the topology discovery problem, for which we propose an algorithm. We prove that when the underlying graph is a constellation of trees (*i.e.,* a set of trees whose roots are interconnected by a complete clique), our algorithm produces a valid solution. We then extend the algorithm to other cases. We present some preliminary evaluation results obtained with the SimGrid simulator.

Key-words: Grid computing, network topology, communication performance prediction

[*] ID, UMR 5132 (CNRS–INPG–INRIA–UJF), Grenoble, France. `Arnaud.Legrand@imag.fr`
[†] LIP, UMR 5668 (CNRS–ENS-Lyon-UCBL–INRIA), Lyon, France. `Frederic.Mazoit@ens-lyon.fr`
[‡] LORIA, UMR 7503 (UHP–CNRS–INPL–INRIA–Nancy2), Nancy, France. `Martin.Quinson@loria.fr`

Découverte de topologie au niveau applicatif

Résumé : Cet article présente un outil de découverte automatique de la topologie du réseau. Son objectif est d'évaluer les performances de transferts concurrents (par exemple pour améliorer des communications collectives) et non de découvrir le schéma d'interconnexion physique des machines (ce qui pourrait servir à diagnostiquer des problèmes de configuration réseau).

Un modèle mathématique permettant la formalisation du problème est introduit. Grâce aux outils analytiques résulants, nous proposons un premier algorithme pour résoudre ce problème. Nous démontrons sous certaines hypothèses la validité de cet algorithme lorsqu'il est possible de construire une solution formant une constélation d'arbres (c'est-à-dire un ensemble d'arbres dont les racines sont interconnectées par une clique complète). Nous étendons ensuite cet algorithme pour lui permettre de traiter certains cas oú l'introduction d'autres formes de cycles dans la solution est nécessaire. Des résultats préliminaires obtenus sur simulateur sont également proposés.

Mots-clés : Calcul distribué à grand échelle, topologie de réseaux, prédiction de performances de communications

1 Introduction

Grids are a type of parallel and distributed systems that result from the sharing and aggregation of geographically distributed resources between several organizations [4]. Unlike classical parallel machines, Grids present dynamic, heterogeneous and often non-dedicated capacities. Gathering accurate, up to date and relevant informations about them is then a very challenging issue, which has to be addressed before developing network-aware applications on the grid.

Most of the previous work in the grid community such as the NWS [9] or RPS [2] projects focused on acquiring quantitative knowledges like network bandwidth and latency, or CPU load. More qualitative informations such as the network topology are however crucial to achieve tasks such as host placement or collective communications performance prediction. Since our goal is to permit the advent of network-aware applications and not to develop a network administration tool, the most relevant information is not the actual physical interconnection schema but a mean to estimate the performance of concurrent transfers.

The topology reconstruction problem is a classical subject in the networking community and has already received a lot of attention. The most classical tools to that extend (like `ping` or `traceroute`) provide too limited information to detect the network bottlenecks occurring in the case of concurrent data streams. The use of tools providing sufficient information such as `pathchar` or the SNMP protocol are often restricted to trusted users for security reasons since the involved protocols can be used to conduct Deny Of Service attacks or reveal information considered as commercial secret by network owners. This limitation makes those approaches unsuited to the grid context since the platform is most often constituted by several distinct organizations diverging in their security policies. It is thus very difficult or even impossible to be given privileges beyond the simple user ones on the whole platform.

Some recent work focus on the design of new measurement scheme usable without any specific privileges. In [7], the authors presents a methodology based only on regular packet exchanges. Unfortunately, the obtained view of the platform does not fulfill our needs. It focuses on the physical interconnection topology and the description of the path followed by the packets while our goal is to identify the interferences between concurrent streams. These two notions are very close, but do not necessary match. If, using this methodology, two paths are reported to be independent (*i.e.,* they do not share any network element), it is clear that data streams using these paths cannot interfere, but the contrary is not necessary true. Indeed, if the shared section is over-dimensioned and able to carry both streams without impacting on their performance, we want our tool to report the paths as independent. This goal's divergence makes this approach unusable directly in our context.

The closest project to our goal is ENV (Effective Network View [8]). It can reconstruct a view of the network topology focusing on interactions of concurrent data streams without requiring any specific privileges on the platform. In [5], we study the possibility of using this tool to automatically place the classical NWS grid monitoring tool. Unfortunately, for efficiency reasons, ENV only report a hierarchical view of the network whereas classical grid

testbeds are often constituted of wide area constellation of local area networks. The tree-based view offered by ENV may thus lack some lateral connections. For example, the nodes of a clusters will be represented as leafs of the tree, and some intra-cluster communication facilities may be omitted by ENV.

This paper introduces a tool called ALNeM (*Application-Level Network Mapper*), which is designed to gather network topology informations useful to network-aware applications. Its goal is not to help administrators to monitor their network and diagnostic bottlenecks or failure points, but rather to give other applications the ability to predict the network performance of data streams. The rest of this article is organized as follows: Section 2 details our goals and introduces the used model. Thanks to the mathematical tools presented in Section 3, we give an algorithm in Section 4. We prove (under some assumptions) that when the underlying graph is forming a constellation of trees (*i.e.,* a set of trees which roots are interconnected by a complete clique), our algorithm builds an equivalent graph. We then extend this algorithm to handle some of the cases where cycles are mandatory in the solution. Section 5 suggests a solution to collect the informations needed for this reconstruction. Experimental results obtained on simulator are presented in Section 6. The proofs of lemmas and theorems are placed in appendix of this paper.

2   The ALNeM project

Before detailing the ALNeM methodology, we should clarify its goals and the model used. First of all, ALNeM does not try to discover the network topology at a router-level and cannot be used network administration tool. It aims at providing a view of the network that is expressive and accurate enough to enable applications running on grid computing platforms to optimize their performances (communication pattern, load-balancing, deployment, . . .). Knowing whether two concurrent data streams interfere is mandatory to achieve this goal.

Note that obtaining a perfect view including all routers, link capacities and routing information may be possible thanks to some tools like `pathchar` and the SNMP protocol. Nevertheless, even when forgetting about the induced security problems that impede their use in our context, such a low-level view is generally not sufficient to determine the interference of two concurrent data streams. Indeed, interference is a dynamic feature of the network and depends on the load on *each* link which is very hard to monitor. Moreover the graph representing the router-level topology is generally very large and thus hard to handle. A suitable network view for network-aware applications optimization should therefore have moderate size while modeling lifelike interferences. For that reason we aim at proposing a minimalist view of the network.

2.1 Model used

The relevant information is more qualitative than quantitative since other tools like the NWS [9] can be used to acquire the quantitative data. At first, the network view can thus be modeled by a simple non-weighted graph.

Definition 1. *A routed graph $G = (V, E, r)$ is a connected non-oriented graph and a routing function $r : V \times V \to V$. $r(u, v)$ is the node to follow to reach v from u. Therefore foreach $u, v \in V : (u, r(u, v)) \in E$.*

Given $u, v \in V$, the notation $\left(u \xrightarrow{G} v \right)$ represents the ordered set of vertices encountered in the graph G on the route from u to v (u and v are respectively the first and the last element of this set). This set obviously depends on the routing function r used.

The most important machines in our context are the ones on which applications can be executed without specific privileges. This excludes the machines of the network infrastructure such as firewalls and routers, which may be omitted from our representation.

Definition 2. *The set of the platform **nodes** (*i.e., *the hosts on which user applications can be deployed) is noted $\mathcal{H}$.*

As stated previously, the searched information is the possible performance impact that data streams occurring at the same time imply on each other. In other words, we need to predict whether the bandwidth between two given machines is affected by a transfer occurring between two other machines at the same time, or if the two streams can coexist without mutual interference. The most natural way to model this notion of interference between the stream (AB) and (CD) in a routed graph is to consider the intersection of the route between A and B and the one between C and D. If this intersection is non-empty, then (AB) and (CD) do interfere on each other.

Definition 3. *Given $a, b, c, d \in \mathcal{H}$, the fact that the path (ab) interfere with (cd) in the G graph is denoted $(ab) \, \chi_G \, (cd)$. This is defined by the following relation:*

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

The non-interference in the graph G between (ab) and (cd) is denoted $(ab) \, \|_G \, (cd)$, and defined by:

$$(ab) \, \|_G \, (cd) \iff \neg \left((ab) \, \chi_G \, (cd) \right)$$

This defines the so-called theoretical interference (by opposition to the measured interference introduced in the next section). This relation is also noted χ_{th} when the graph on which it applies is given by the context. These definitions trivially lead to the following lemma (since the set intersection is a symmetric relation):

Lemma 1 (χ_{th} is a symmetric relation).

$$\forall a, b, c, d \in \mathcal{H}, \quad (ab) \, \chi_{th} \, (cd) \Leftrightarrow (cd) \, \chi_{th} \, (ab).$$

Remark 1. *The equivalence* $(ab) \, \lambda_{th} \, (cd) \Leftrightarrow (ba) \, \lambda_{th} \, (cd)$ *is not true when the routing is non-symmetric,* i.e., *when* $(ab) \neq (ba)$.

2.2   Measurement methodology

As explained in the previous section, the key notion on which we base our network view is the interference. We now define more precisely how to measure this information.

Definition 4. *Let* $\widetilde{G} = (\widetilde{V}, \widetilde{E}, \widetilde{r})$ *be the routed graph representing the physical topology with* $\widetilde{V}$ *the set of all existing hosts (including the one of the network infrastructure such as routers),* $\widetilde{E}$ *the links connecting them and* $\widetilde{r}$ *the paths followed by data streams.*

The routing $\widetilde{r}$ used in $\widetilde{G}$ depends on the configuration of each element of $\widetilde{V}$ and is thus only locally known. Even if the graph $\widetilde{G}$ reflects a physical reality, it is rarely actually known. Issues like faulty protocol implementations and configuration errors may cause routing inconsistencies such as asymmetric or changing paths, and even loops. According to [6], such situations are quite common in existing wide area networks.

Under those conditions, the simplest way to know whether a transfer impacts another one (without relying on methodology whose use may be restricted to privileged users) is to conduct a direct experiment by comparing the usual bandwidth to the one achieved when the second connection is saturated.

Definition 5. *Given four nodes a, b, c et d, the bandwidth between a and b when no extra traffic is injected between c and d is denoted* $\boldsymbol{bw(ab)}$*. The bandwidth between a and b when the connection between c and d is saturated is denoted* $\boldsymbol{bw_{\parallel cd}(ab)}$*.*

This allows to define an interference notion based on the comparison of these two values: if their ratio equals 0.5, that means that the two data streams fairly share a limiting network facility. If this ratio equals 1, (cd) have no impact on (ab). We use thresholds to account for the measurement errors and the perturbations dues to the external load.

Definition 6. *We consider that* (cd) *impacts on* (ab) *if and only if*

$$\frac{bw_{\parallel cd}(ab)}{bw(ab)} < 0.7$$

We denote that case with the following notation $(ab) \, \lambda_{mes} \, (cd)$ *(mes standing for* measured*).*

Definition 7. *If this ratio is greater than 0.9, we consider that the transfers do not interfere. This is denoted by* $(ab) \, \parallel_{mes} \, (cd)$*.*

A ratio between 0.7 and 0.9 is supposed to result of measurement error, and to imply that the experiment should be conducted again. Those ratio are the same as the ones determined empirically by the authors of the ENV project [8].

Lemma 2 ($\mathcal{X}_{mes}$ **is not a symmetric relation**)**.** *That is to say:*

$$\exists a, b, c, d \in \mathcal{H} \ / \ \Big((ab) \ \mathcal{X}_{mes} \ (cd)\Big) \wedge \neg \Big((cd) \ \mathcal{X}_{mes} \ (ab)\Big)$$

Proof. Figure 1 presents a counter-example to the symmetry of the $\mathcal{X}_{mes}$ relation.

$$a \ \xleftrightarrow{\ 10 \text{ Mo/s} \ } \ c \ \xleftrightarrow{\ 100 \text{ Mo/s} \ } \ b \ \xleftrightarrow{\ 100 \text{ Mo/s} \ } \ d$$

Figure 1: Counter example to the symmetry of the $\mathcal{X}_{mes}$ relation.

In that case, the result of the measurements may be at the same time $\frac{bw_{/\!/cd}(ab)}{bw(ab)} < 0.7$ and $\frac{bw_{/\!/ab}(cd)}{bw(cd)} > 0.9$ since the link (ab) is limited to 10 Mb/s on the (ac) segment. Its impact on (cd) (on which the bandwidth tends to 100 Mb/s) is thus undetectable. $\qquad\square$

We can notice however that the paths (ab) and (cd) in this counter-example do share a common link of $\widetilde{G}$. Moreover, the impact of (ab) on (cd) do exist, even if it remains undetectable to the conducted measurements. This leads to the symetrization of the relation in order to match the $\mathcal{X}_{th}$ notion.

Definition 8. *We consider that* (ab) *and* (cd) *interfere if and only if* (ab) *impacts* (cd) *or* (cd) *impacts* (ab). *This is denoted by* $(ab) \ \mathcal{X}_{rl} \ (cd)$ (rl *standing for* real*). We have:*

$$(ab) \ \mathcal{X}_{rl} \ (cd) \Longleftrightarrow \begin{cases} (ab) \ \mathcal{X}_{mes} \ (cd) \\ (cd) \ \mathcal{X}_{mes} \ (ab) \end{cases} (or) \quad \Longleftrightarrow \quad \begin{cases} \frac{bw_{/\!/cd}(ab)}{bw(ab)} < 0.7 \\ \frac{bw_{/\!/ab}(cd)}{bw(cd)} < 0.7 \end{cases} (or)$$

In the opposite case, we consider that the paths do not interfere with each other, which is denoted by $(ab) \ /\!/_{rl} \ (cd)$.

By construction, the relation $\mathcal{X}_{rl}$ is symmetric, and $(ab) \ \mathcal{X}_{rl} \ (cd) \Longleftrightarrow \neg \ (ab) \ /\!/_{rl} \ (cd)$. It is thus possible to store the value of this relation for all nodes in a four dimensional matrix defined as follows:

Definition 9. *Let* $I(\mathcal{H}, \mathcal{X}_{rl})$ *be the interference matrix between all elements of the* $\mathcal{H}$ *set, as implied by the relation* $\mathcal{X}_{rl}$:

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

2.3   Problem statement

As explained is section 2, we are looking for an easy-to-use view of the network. The relation between the measured interference $\mathcal{X}_{rl}$ and the physical interconnection graph $\widetilde{G}$ is much more complicated than the relation between their theoretical counterparts $\mathcal{X}_{th}$ and G. Therefore, we propose to identify $\mathcal{X}_{rl}$ and $\mathcal{X}_{th}$ as the same notion and to look for a routed graph mimicking the real interferences. Thanks to the notations introduced in the previous two sections, we can now define more formally the problem which ALNEM aims to solve:

Definition 10. INTERFERENCEGRAPH*: Given $\mathcal{H}$ and $I(\mathcal{H}, \mathcal{X}_{\widetilde{G}})$, find a routed graph $G = (V, E, r)$ such that:*

$$\begin{cases} \mathcal{H} \subset V \; ; \\ I(\mathcal{H}, \mathcal{X}_{\widetilde{G}}) = I(\mathcal{H}, \mathcal{X}_{G}) \; ; \\ |V| \; is \; minimal. \end{cases}$$

The idea is to look for an abstract graph G mimicking the effects of $\widetilde{G}$ with regard to the interferences between streams. This allows the users to manipulate G to acquire informations on $\widetilde{G}$. In order to simplify its use, we have to keep the size of G as small as possible.

The first natural question that arises about this optimization problem is the following: "Given an arbitrary interference matrix, is it always possible to find a graph G so that $I(\mathcal{H}, \mathcal{X}_{\widetilde{G}}) = I(\mathcal{H}, \mathcal{X}_{G})$?". The answer to this question is positive as stated in the following theorem.

Theorem 1. *For all $\mathcal{H}$ and $I(\mathcal{H}, \mathcal{X}_{\widetilde{G}})$, there exists a graph $G = (V, E, r)$ such that:*

$$\begin{cases} \mathcal{H} \subset V \; ; \\ I(\mathcal{H}, \mathcal{X}_{\widetilde{G}}) = I(\mathcal{H}, \mathcal{X}_{G}) \; ; \end{cases}$$

Nevertheless the size of the graph built in the proof (detailed in Appendix A) is exponential in the size of $\mathcal{H}$. Therefore, the second question that arises about the hardness of our problem concerns its feasibility in a polynomial size: "Given an arbitrary interference matrix, is there a routed graph of reasonable size (i.e. polynomially bounded) so that $I(\mathcal{H}, \mathcal{X}_{\widetilde{G}}) = I(\mathcal{H}, \mathcal{X}_{G})$?". We still do not know the answer to this question, which prevents us to state properly the decision problem associated to INTERFERENCEGRAPH. In Section 4, we analyze some particular situations where we are able to find a solution such that $\mathcal{H} = V$.

3   Mathematical tools

This section presents some mathematical tools for the theoretical study the INTERFERENCE-GRAPH. They constitute the framework needed to manage the algorithm presented in the next section.

3.1 Hypotheses

We assume in this paper that the graph $\widetilde{G}$ respects the following hypotheses:

Hypothesis 1 (routing consistency). $\forall (a, b, c) \in \widetilde{V}$:

$$c \in \left(a \underset{\widetilde{G}}{\longrightarrow} b \right) \Longrightarrow \left(a \underset{\widetilde{G}}{\longrightarrow} b \right) \cap \left(a \underset{\widetilde{G}}{\longrightarrow} c \right) = \left(a \underset{\widetilde{G}}{\longrightarrow} c \right)$$

This hypothesis indicates that the routing algorithm used in $\widetilde{G}$ does not present weird inconsistencies. If a vertex c is on the path connecting a to b, the packets transiting from a to b follow the same path on the beginning of their trip than the ones transiting from a to c. The contrary would imply for example that the machine a uses an host ρ as gateway, and that ρ routes the packets for b through c while a can connect to c directly without using ρ.

Remark 2. *Using the routing definition, this hypothesis gives:* $\forall (a, b, c) \in \widetilde{V}$:

$$c \in \left(a \underset{\widetilde{G}}{\longrightarrow} b \right) \Longrightarrow \left(a \underset{\widetilde{G}}{\longrightarrow} b \right) \cap \left(c \underset{\widetilde{G}}{\longrightarrow} b \right) = \left(c \underset{\widetilde{G}}{\longrightarrow} b \right)$$

Hypothesis 2 (routing symmetry). $\forall (a, b) \in \widetilde{V} : \left(a \underset{\widetilde{G}}{\longrightarrow} b \right) = \left(b \underset{\widetilde{G}}{\longrightarrow} a \right)$

This hypothesis indicates that the routing is symmetric.

We know that both of these hypotheses may be violated on Internet since all imaginable routing inconsistency exists in the reality [6]. However, as we would like to get an abstract view of the network where those local difficulties have disappeared, we will try in the following to build a routed graph where these hypotheses hold and will therefore suppose that they hold true when needed.

3.2 Total interference and separators

For sake of clarity, the set $\left(v \underset{\widetilde{G}}{\longrightarrow} w \right)$ is denoted $(v \rightarrow w)$ here.

Definition 11. *Two nodes a and b are said to be in **total interference** if and only if any stream coming out of* a *interfere with any stream coming out from* b. *This is then denoted by $\boldsymbol{a \perp b}$.*

More formally, we have:

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

That is to say that a and b are in total interference if and only if they interfere with all other existing nodes.

Lemma 3 (Separation). *Let a and b in $\mathcal{H}$. We have*

$$a \perp b \Longleftrightarrow \exists \rho \in \widetilde{V} \;\Big/\; \forall z \in \mathcal{H} : \rho \in (a \to z) \cap (b \to z).$$

This express the equivalence between the fact that a and b are in total interference and the existence of a vertex ρ being on all paths connecting a to other nodes as well as all paths coming out of b. The proof of this lemma, detailed in Appendix B, uses the hypothesis 1 (routing consistency), but does not rely on the hypothesis 2 (routing symmetry).

Definition 12. *Given a and $b \in \mathcal{H}$, a vertex ρ such as the one introduced in the lemma 3 (separation) is said to be a **separator** of a and b.*

Theorem 2. *The total interference ($\perp$) is an equivalence relation. Moreover, for each equivalence class of $\perp$, there is a common separator for all pair of element in the class.*

The proof of this theorem, detailed in Appendix C, uses lemma 3 (separation) and hypothesis 2 (symmetric routing).

Theorem 3 (Representativity). *Let $\mathcal{C}$ be an equivalence class for $\perp$ and ρ a separator of its elements.*

$$\forall a \in \mathcal{C}, \; \forall b, u, v \in \mathcal{H}, \;\; (a, u) \,\lambda_{rl}\, (b, v) \Leftrightarrow (\rho, u) \,\lambda_{rl}\, (b, v)$$

That means that the separator of an equivalence class has the same interferences with the external nodes than any element of the class. To rephrase it, the separator constitutes a valid representative for all elements of the class with regard to the interactions with external nodes.

The proof of this theorem, detailed in Appendix D, does not rely explicitly on the hypothesis 2 (symmetric routing). It relies however on the existence of a common separator for all pairs in a given equivalence class of $\perp$ (which in turn relies on the hypothesis 2).

The existence of a common separator representative for any equivalence class of $\perp$ serves as a basis to the algorithm presented in next section. This algorithm consists in searching all equivalence classes of $\perp$, and then replace all of their members by a sole representative: the separator of the class and iterate.

4 Reconstructing algorithm

We now piece together an algorithm reconstructing a graph G which induces the same interference matrix I than $\widetilde{G}$. For that, we proceed in several steps. We first present a version which we prove to give a solution when it is possible to construct G as a tree. The study of the cases where this algorithm fails allow us to generalize it and handle some cases requiring the introduction of cycles in the graph.

4.1 Handling of trees

The general principle is to construct a graph beginning from the extremities and replace the sub-trees in the interference matrix by their separators before iterating.

$\text{TREE}(\mathcal{H}, I(\mathcal{H}, \lambda_{rl}))$

1. *Initialization*

   $i \leftarrow 0$ i: current step
   $\mathcal{C}_i \leftarrow \mathcal{H}$ $\mathcal{C}_i$: set of nodes considered at step i
   $E_i \leftarrow \emptyset \; ; \; V_i \leftarrow \emptyset$ $G_i = (E_i, V_i)$: graph reconstructed after step i

2. *Search of equivalence classes, and separator election*

   Let $h_1, \ldots, h_p$ be the equivalence classes for $\perp$ on $\mathcal{C}_i$. They are easily obtained from $I(\mathcal{C}_i, \lambda_{rl})$ by a greedy algorithm looking for the maximal sets for the inclusion of nodes in total interference.

   Let l_j be the separator of each class h_j.

   $\mathcal{C}_{i+1} \leftarrow \{l_1, \ldots, l_p\}$

3. *Graph update*

   $V_{i+1} \leftarrow V_i \; ; \; E_{i+1} \leftarrow E_i$

   For each $h_j \in \mathcal{C}_i$, for each $v \in h_j$, $\begin{cases} E_{i+1} \leftarrow E_{i+1} \cup \{(v, l_j)\} \\ V_{i+1} \leftarrow V_{i+1} \cup \{v\} \end{cases}$

4. *Interference matrix update*

   Let $l_\alpha, l_\beta, l_\gamma, l_\delta \in \mathcal{C}_{i+1}$ be the representative of respectively $h_\alpha, h_\beta, h_\gamma, h_\delta$.

   Let $m_\alpha, m_\beta, m_\gamma, m_\delta \in \mathcal{C}_i$ be so that $m_\alpha \in h_\alpha$, $m_\beta \in h_\beta$, $m_\gamma \in h_\gamma$ and $m_\delta \in h_\delta$.

   $I(\mathcal{C}_{i+1}, \lambda)\big(l_\alpha, l_\beta, l_\gamma, l_\delta\big) = I\left(\mathcal{C}_i, \lambda\right)\big(m_\alpha, m_\beta, m_\gamma, m_\delta\big)$

5. Iterate the steps 2-4 until $\mathcal{C}_i = \mathcal{C}_{i+1}$.

Theorem 4 (Tree algorithm correction). *When the algorithm terminates with only one leader (i.e., $\exists n \,/\, \left|\mathcal{C}_n\right| = 1$), the shortest path routing on the resulting graph G satisfies $I(\mathcal{H}, \lambda_G) = I(\mathcal{H}, \lambda_{rl})$.*

Theorem 5 (Tree algorithm termination conditions). *Given an instance of* INTER-FERENCEGRAPH, *if it is possible to build a tree G being a solution, then* TREE *terminates with only one leader.*

These two theorems are proved respectively in Appendix E and Appendix F thanks to the theorem 3 (representativity of the separator) as well as both hypotheses (routing consistency and symmetry);

Remark 3. *Any solution found by the* TREE *algorithm is optimal.*

Indeed, the theorem 3 (representativity of the separator) allows to use any element of the class as separator. That way, no new vertex is introduced by the algorithm. The set of vertices in G is thus restricted to $\mathcal{H}$, and G is then clearly optimal.

4.2 Handling of cliques

When, at a given step i of the TREE algorithm, there is no remaining interference between the elements of C_n, it is naturally impossible to find two nodes being in total interference. When this occurs, the algorithm fails to construct the wanted graph.

The intuitive solution in that case, consisting in connecting all the representatives of each class in a clique manner, leads to a valid solution. The following theorem expresses this more formally.

Theorem 6. *(Clique handling validity) If there exists a step i of* TREE *so that* $\forall a_i, u_i, b_i, v_i \in \mathcal{C}_i, (a_i, u_i) \parallel (b_i, v_i)$, *then the graph G connecting all pairs of elements in $\mathcal{C}_i$ satisfies* $I(\mathcal{H}, \lambda_G) = I(\mathcal{H}, \lambda_{rl})$ *when the shortest path routing algorithm is used.*

This extension allows the algorithm to handle tree constellations, *i.e.,* graphs formed by several distinct trees whose roots are all interconnected by a clique.

Remark 4. *The solutions found by this extension, when they exist, are also optimal since no new vertex is introduced.*

4.3 Cycle handling

By application of the theorem 5 (Tree algorithm terminaison condition), we know that if the algorithm fails to end with a connected graph, then no tree mimicing the interferences given by I exists. In particular, it is necessary to introduce cycles in G to find a solution.

We now present an extension of the algorithm handling the interference matrices induced by some graph G containing cycles. Since graphs containing cycles have weakest properties than trees, the chosen approach is more pragmatic and the solution is less generic.

We assume that the TREE algorithm were applied to group in the same connected set all elements in total interference, but that the connectivity of the builded graph is only partial. We are thus at a step i of the algorithm where there is no $l_\alpha, l_\beta \in C_i$ so that $l_\alpha \perp l_\beta$.

The main idea of our algorithm to handle this situation is to find two nodes close to each other on a cycle, cut the cycle between them so that the TREE algorithm can go further, and then reintroduce the cycle by reconnecting those two points afterward.

This approach induces two different difficulties. We first have to detect two such points using only the informations provided by the interference matrix. We then have to find a way to reintroduce the cycle afterward.

In our algorithm, the cycle is cut between the two vertices having the most interference in the matrix, *i.e.,* between the two vertices a and b maximizing the set $\{u, v : au \lambda bv\}$. It is possible to split the vertices set in four sub-sets I_1, I_2, I_3 and I_4 defined as following:

$$\begin{cases} I_1 = \big\{ u \in \mathcal{C}_i : a \in (b \to u) \text{ and } b \notin (a \to u) \big\} \\ I_2 = \big\{ u \in \mathcal{C}_i : a \notin (b \to u) \text{ and } b \in (a \to u) \big\} \\ I_3 = \big\{ u \in \mathcal{C}_i : a \notin (b \to u) \text{ and } b \notin (a \to u) \big\} \\ I_4 = \big\{ u \in \mathcal{C}_i : a \in (b \to u) \text{ and } b \in (a \to u) \big\} \end{cases}$$

The hypothesis 1 implies trivially that $I_4 = \{a, b\}$ because if there existed an element u differing of a and b in I_4, we would have the following contradictory case:
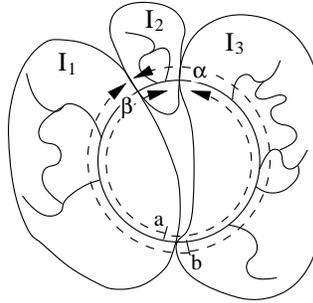
Figure 2: Split of the vertices set to handle cycles.

Using the hypothesis 1 (routing consistency), the general organization of the underlying graph is therefore as depicted in Figure 2. We introduce a point α separating the sets $\{u \in \mathsf{C}_i : b \in (a \to u)\}$ and $\{u \in \mathsf{C}_i : b \notin (a \to u)\}$. Likewise, the point β is constructed with regard to b. Note that these two points may not belong to $\mathcal{H}$ and are difficult to detect but are well-defined.

We now have to determine the three sets I_1, I_2 and I_3 based on the information contained by the interference matrix. If we knew those three sets, we would be able to order the nodes along the main cycle and the corresponding matrix slice on (a, b) would then have a very particular configuration (see Figure 3). Indeed, when $u \in I_1$ and $v \in I_1$ then $(au) \not\asymp (bv)$, explaining that the upper left part of the table is filled of 1.

Finding a nodes permutation so that the matrix slice on (a, b) is organized as in Figure 3 can be done by sorting topologically the strongly connected components of the graph from which this slice is the adjacency matrix. Such a reordering then gives the wanted split of the C_i elements into I_1, I_2 and I_3 based only on the informations contained in the interference matrix.

The next challenge is the reconnection of the graphs obtained by recursion on those subsets. Since we split the cycle on two positions (between a and b, and between α and β), we have to reconnect it on each of them. Reconnecting a and b simply needs the addition of a new edge. In order to reconnect the cycle between α and β, we first have to find the closest vertices from the splitting point.

Figure 3: Interference matrix slice on (a, b) after reordering.

We know that the first step of the TREE recursion on I_1 will find elements being in total interference since we broke the cycle and thus created strings around a and α. Since TREE was unable to find elements in total interference at the previous step, those strings are the only existing subtrees. It is moreover easy to differentiate them since one of them contains a.

By symmetry on I_3 and b, we deduce that the reconnection should occur between one of the point of the connected set created by the first iteration of TREE on I_1 which does not contain a on one side, and one of the point from the connected set resulting of TREE on I_3 on the other side. The choice between the possible multiple candidates is done by selecting the points S_α and S_β having the more interferences with the other points of C_i (using the intuition that points presenting more interferences should be close in the graph).

After identification of S_α and S_β being the sets border where the reconnection should occur, this reconnection is achieved by executing the algorithm recursively on the set $I_2 \cup \{S_\alpha, S_\beta\}$.

This provides us with an algorithm able to handle some cases where cycles are mandatory. This is however not a complete generalization since the graph G by our algorithm does not necessarily respect all the constraints expressed by the interference matrix. Amongst other reasons, this is due to the fact that the decisions are taken using only a slice of the interference matrix to decide how to build G, disregarding the possibly contradictory informations presented elsewhere in the matrix. Theorem 1 suggests that no such greedy algorithm can solve this problem.

5   Collecting the needed data

We now tackle the way the needed data about the interference on the platform may be measured. This is a technically challenging issue which may reveal time-consuming and thus deserve optimizations.

5.1   Intuitive algorithm

The simplest way to collect those informations resides in $|\mathcal{H}|^4$ steps (for each quadruplet $(a, b, c, d) \in \mathcal{H}^4$), each of them consisting in:

1. Measure the bandwidth on (ab) (denoted $bw(ab)$) ;

2. Measure the bandwidth on (ab) when the link (cd) is saturated (denoted $bw_{\parallel cd}(ab)$) ;

3. Compute the ratio.

The steps 1. and 2. must last long enough to let the network stabilize. We first have to wait long enough after an experiment before starting the next one to avoid any perturbation and we also have to wait for the link (cd) to be actually saturated before conducting the second step. Those delays forbid to reasonably test more than ten quadruplets per minute on a possibly wide area network. The execution of this algorithm lasts thus $\frac{|\mathcal{H}|^4}{10}$ minutes, which represents about 10 days when $|\mathcal{H}| = 20$. This solution is thus foredoomed by lack of extensibility.

5.2   Optimizations

In order to speed up the process, ALNEM conducts the experiments in parallel, thanks to independent links. Since they do not interfere with each other, it is possible to saturate several of them at the same time. That way, when a new link is tested, it is not tested against only one link, but against all links currently saturated.

If the saturation of the new link implies a performance decrease on one of the link previously saturated, we report the interference in the matrix and stop the last saturation. If the saturation of the new link does not imply any performance modification of the previously saturated link, we continue by adding another saturation.

The more independent saturation we achieve at the same time, the less the whole measurement process lasts. In order to predict the probably independent links and increase the parallelism, a preliminary step constructs a first guess of the topology using `traceroute`. As discussed before, the result of this step does not perfectly fit our needs since this methodology does not allow to capture all interferences, but those informations reveal precious to guide our measurement algorithm.

The speedup offered by this optimization naturally depends on the $\widetilde{G}$ characteristics. The worst case is when all links interfere with each other, forbidding any parallelization. The best case is when no link interfere with any other, allowing a complete parallelization. In this case, our algorithm needs only $2|\mathcal{H}|^2$ steps to complete (about one hour for 20 nodes). Since typical testbeds are constituted of a wide area constellation of local area networks, which are often trees, we think that this optimization may lead to important speedups.

Furthermore, this algorithm needing a centralized clock, the measurements are synchronized by a specific node called *maestro*. The position of this node in the network does however not impact the results.

6   Mapping example

This section introduces a prototype of ALNEM developed in the SimGrid [1] simulator. The presented results were obtained by feeding the simulator with a topology generated by Tiers [3], and running ALNEM within the simulator without providing it any preliminary information about this topology. The Figure 4 depicts at the same time the "real" topology as provided to SIMGRID (4(a)), the data provided to ALNEM (4(b)) and the G graph reconstructed by ALNEM (4(c)). Figure 5 depicts different steps of ALNEM's reconstruction algorithm in that case.

(a) Topology used ($\widetilde{G}$).   (b) Initial data (plus $I_{\chi_{rl}}$).   (c) ALNEM result (G).
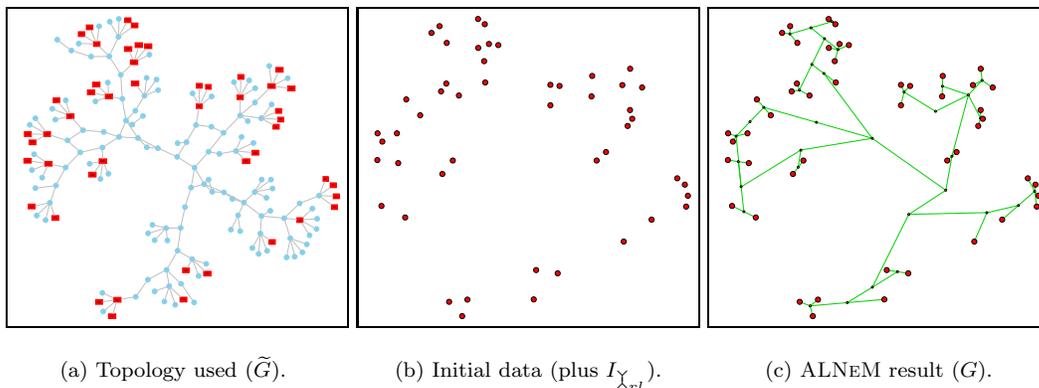
Figure 4: Example of graph reconstruction.

7   Conclusion and future work

In this paper, we present an network mapping framework called ALNEM (Application-Level Network Mapper). It is meant to capture a macroscopic and qualitative view of the network topology suited for network-aware applications. It does not try to discover neither the network physical interconnection schema nor the path followed by the packets. This would be interesting for administrators wanting to auscultate their networks, but would be less relevant in our context. This tool does not either try to precisely determine the end-to-end bandwidth since other tools like NWS already retrieve those measurements.

The information captured by ALNEM is a graph accounting for the possible interference (in term of performance) between concurrent data streams. Data movement on (ab) and (cd) can occur at the same time without impacting on the performance *if and only if* the shortest paths on (ab) and (cd) have an empty intersection in this graph. If not, they do impact on each other.
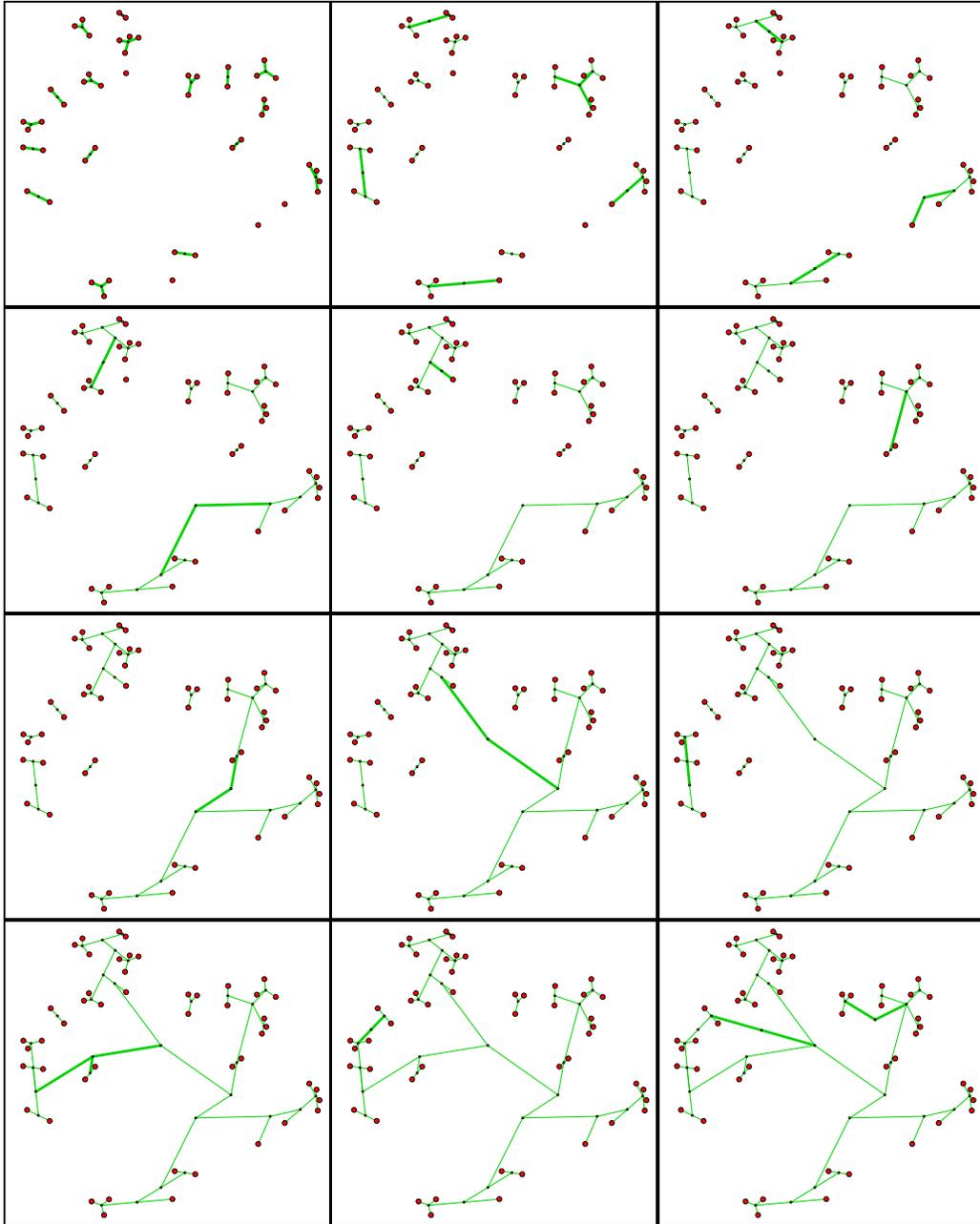
Figure 5: Successive steps of the reconstruction algorithm.

The reconstruction of this representation is a well-defined problem that we call INTERFERENCEGRAPH. Even if we have not been able to prove it yet, we think that this problem is NP–complete. We propose an algorithm that builds the optimal solution whenever the underlying graph is a constellation of trees (*i.e.,* several trees whose roots are interconnected by a clique). We also propose an extension of the algorithm able to handle some cases in which other forms of cycle are mandatory in the builded graph. An accurate study of this extension's limits will be tackled in future work.

We would like to enrich our model in the future to predict the bandwidth threshold above which the performance interference occures. We implicitly assumed that two concurrent streams are enough to detect any bottleneck in the network. This is however not the case, and some resources may become limiting (thus creating an interference) only when shared by three or more streams. We could take this fact into account by introducing a notion of N-interference when N streams are mandatory to reveal the bottleneck. Another solution would be to tag each separator by a threshold representing the bandwidth needed to make the resource it represents limiting.

Finally, we present how ALNEM measure the data needed to the application of this algorithm and some possible optimizations. In order to ensure ALNEM's applicability on large platforms, we plan to increase in future work the parallelism achieved by a finer analysis of the results provided by `traceroute` or `ping`.

In order to limit the number of hosts involved in each mapping (and thus reduce the time needed), we also plan to study how to merge preexisting topology. It may imply to conduct some extra measurements beyond the one needed to get each sub-topology, but would probably allow to come up with a recursive algorithm able to map networks containing a large number of hosts efficiently. Likewise, an iterative algorithm able to add a new node to a previously computed topology is needed to avoid the recomputation of the whole topology when a new node appears or disappears.

More generally, we think that the measurement step may be greatly optimized by a greater interaction with the reconstruction step allowing to schedule the tests depending on the current knowledge about the topology.

To that concern, it may be very interesting to integrate ALNEM to a network monitoring infrastructure such as NWS. This would allow to conduct the more intrusive measurements when regular users do not use the network. It would also ensure the automatic detection of the structural changes by conducting regular tests to verify that previous results still match the current situation. Those results would be beneficial both to NWS itself to improve its configuration and to client applications.

A   Proof of theorem 1: InterferenceGraph is well defined

Theorem 1: For all $\mathcal{H}$ and $I(\mathcal{H}, \mathcal{X}_{\tilde{G}})$, there exists a routed graph $G = (V, E, r)$ such that:

$$\begin{cases} \mathcal{H} \subset V \; ; \\ I(\mathcal{H}, \mathcal{X}_{\tilde{G}}) = I(\mathcal{H}, \mathcal{X}_G) \; ; \end{cases}$$

Proof   We prove the result by induction on the size of $\mathcal{H}$. The result is trivial when $|\mathcal{H}| = 1$. Let us suppose that $|\mathcal{H}| \geq 2$.

Let $w \in \mathcal{H}$ and $\mathcal{H}' = \mathcal{H} \backslash \{w\}$. By induction, we can consider G' a routed graph such that $\mathcal{H}' \subset V'$ and $I(\mathcal{H}', \mathcal{X}_{\tilde{G}}) = I(\mathcal{H}', \mathcal{X}_{G'})$. We build the graph G by iteratively creating a long path from each element $b \in \mathcal{H}'$ to w (see Figure 6(a)).

Let b and $a \in \mathcal{H}'$. To build G, we first apply on a the transformation depicted in Figure 6(b). Therefore, the routing in a has to be redefined. For each element $u \in \mathcal{H}'$, if $(au) \; \mathcal{X}_{\tilde{G}} \; (bw)$ then the route $(a \rightarrow u)$ is not modified, otherwise a light-grey node is added at the beginning. Note that this transformation does not modify the interferences between the nodes belonging to $\mathcal{H}'$. By repeating this operation for each $a \in \mathcal{H}'$ and by building a route from b to w that goes through all the light-grey nodes, we get a graph such that $(au) \; \mathcal{X}_{\tilde{G}} \; (bw)$ if and only if $(au) \; \mathcal{X}_G \; (bw)$. By repeating again this operation for all the remaining nodes $b \in Hn$, we obtain a graph such that $\mathcal{H} \subset V$ and $I(\mathcal{H}, \mathcal{X}_{\tilde{G}}) = I(\mathcal{H}, \mathcal{X}_G)$, hence the result.

B   Proof of the lemma 3 (separation)

Lemma 3: $\forall a, b \in \mathcal{H}, \; a \perp b \Longleftrightarrow \exists \rho \in \widetilde{V} \; \Big/ \; \forall z \in \mathcal{H} : \rho \in (a \rightarrow z) \cap (b \rightarrow z)$.

Proof of the $\Rightarrow$ part   Suppose that $a \perp b$ and let u be an element of $\mathcal{H} \backslash \{b, a\}$. Let u_a be the first distinct node of a in $(a \rightarrow u) \cap \mathcal{H}$. The construction of those points is depicted by Figure 7(a).

Let ρ be the first vertex of $(a \rightarrow u_a) \cap (b \rightarrow u_a)$. Note that $\rho \in \widetilde{V}$.

1. Let's first prove that u_a is the first node distinct of a and b in $(b \rightarrow u_a) \cap \mathcal{H}$.
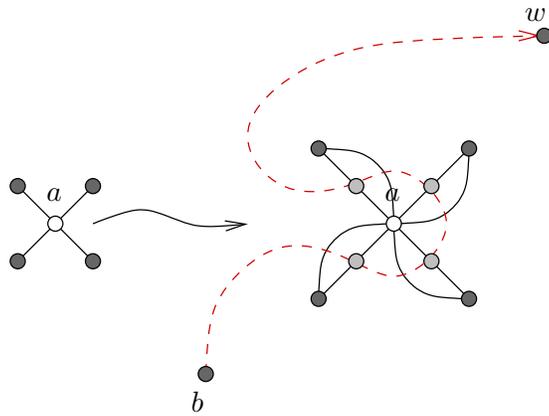
   Proof. Thanks to the hypothesis 1 (routing consistency), we have the following relations:
   $$\begin{cases} (a \rightarrow u_a) = (a \rightarrow \rho) \cup (\rho \rightarrow u_a) \\ (b \rightarrow u_a) = (b \rightarrow \rho) \cup (\rho \rightarrow u_a) \end{cases}$$

   - If $\rho = a$, then there is no $c \in \mathcal{H} \backslash \{b, a\}$ so that $c \in (b \rightarrow a)$ (otherwise, we would have $bc \; /\!/_{rl} \; au_a$, which is absurd since $a \perp b$). Thus, u_a is the first node distinct of a and b in $(b \rightarrow u_a) \cap \mathcal{H}$.
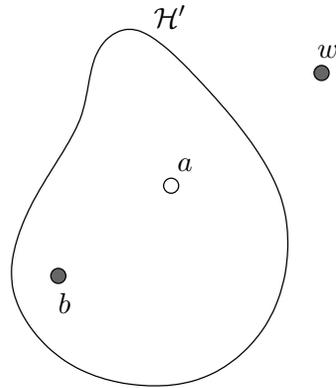
(a) Reasoning by induction on the size of $\mathcal{H}$.

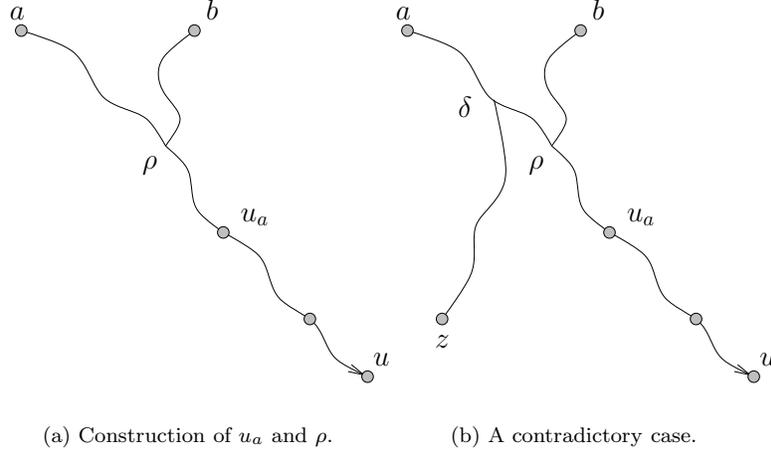(b) Transforming each node.

Figure 6: Sketch of the proof of Theorem 1.

(a) Construction of u_a and ρ.   (b) A contradictory case.

Figure 7: Sketch of the proof of lemma 3.

- If $\rho \neq a$, then $(\rho \to u_a) \cap \mathcal{H} = \{u_a\}$ by definition of u_a. There is no $c \in \mathcal{H}\backslash\{b\}$ so that $c \in (b \to \rho)$. Otherwise, we would have $b \rightsquigarrow c \rightsquigarrow \rho \rightsquigarrow u_a$ and $a \rightsquigarrow \rho \rightsquigarrow u_a$. As the definition of ρ gives $(b \to \rho) \cap (a \to u_a) = \{\rho\}$, we would get $bc /\!\!/_{rl} au_a$, hence a contradiction. Thus, u_a is the first node distinct of a and b in $(b \to u_a) \cap \mathcal{H}$. $\square$

2. Now, let's prove that $\forall z, \ \rho \in (a \to z)$.

   Proof (by contradiction). Let us assume that there exists a z so that $\rho \notin (a \to z)$ exists. Using hypothesis 1 (routing consistency), we have gives the following relation (depicted by Figure 7(b)):

   $$(a \to z) \cap (\rho \to u_a) = \emptyset$$

   Let δ be the first vertex in $(a \to z) \cap (a \to u_a)$. By application of the hypothesis 1 (routing consistency), we know that $\delta \in (a \to \rho)$ et $(\delta \to z) \cap (\rho \to u_a) = \emptyset$.

   ρ being the first element of $(a \to z) \cap (b \to z)$ by definition, we get that $(a \to \rho) \cap (b \to \rho) = \{\rho\}$. Since $\delta \in (a \to \rho)$, we then have that $\delta \notin (b \to \rho)$.

   Using one more time the hypothesis 1 (routing consistency), the relation becomes $(a \to z) \cap (b \to u_a) = \emptyset$. It is the definition of $(ab) /\!\!/_{rl} (bu_a)$, hence a contradiction with the hypothesis $a \perp b$. $\square$

3. By symmetry, we have $\rho \in (b \to z)$.

This leads to the wanted result:

$$\forall z \in \mathcal{H} : \begin{cases} \rho \in (a \to z) \\ \rho \in (b \to z) \end{cases}$$

Proof of the $\Leftarrow$ part   This implication is trivial. Let $\rho \in \widetilde{V}$ be so that:

$$\forall z \in \mathcal{H} : \begin{cases} \rho \in (a \to z) \\ \rho \in (b \to z) \end{cases}$$

This leads to $\forall u, v \in \mathcal{H}, \ \rho \in (a \to u) \cap (b \to v)$, and thus to $(au) \mathrel{\chi} (bv)$.
Thus $a \perp b$.

C   Proof of theorem 2 ($\perp$ is an equivalence relation)

Theorem 2.   *The total interference ($\perp$) is an equivalence relation. Moreover, for each equivalence class of $\perp$, there is a common separator for all pair of element in the class.*

In order to prove that $\perp$ is an equivalence relation, we naturally have to prove its reflexive property (which done in lemma 4), its symmetric property (lemma 5) and its transitive property (lemma 6). This last lemma also prove the existence of a common separator across the whole equivalence class.

Lemma 4 ($\perp$ is reflexive).   *That is to say $\forall a \in \mathcal{H}, a \perp a$.*

Proof. For all $a, u, v \in \mathcal{H}$, $(a \to u) \cap (a \to v) = \{a\} \neq \emptyset$. Thus $(au) \mathrel{\chi} (av)$, which is the definition of $a \perp a$. $\square$

Lemma 5 ($\perp$ is symmetric).   *That is to say $\forall a, b \in \mathcal{H}, a \perp b \Leftrightarrow b \perp a$.*

Proof. This is trivially given by the fact that $\mathrel{\chi}$ is symmetric by construction. $\square$

Lemma 6 ($\perp$ is transitive).   *Given three nodes a, b, $c \in \mathcal{H}$, if $\big(a \perp b\big) \wedge \big(b \perp c\big)$ then this nodes have a common separator and thus $a \perp c$.*

Proof. Let $a, b, c \in \mathcal{H}$ be so that $a \perp b$ and $b \perp c$.
Let $u \in \mathcal{H}$. Let u_a be the first node distinct of a in $(a \to u) \cap \mathcal{H}$. Let ρ be the first node in $(a \to u_a) \cap (b \to u_a)$ and σ be the first node in $(b \to u_a) \cap (c \to u_a)$.
ρ is thus a separator of a and b while σ is a separator of b and c (this, as well as the existence of ρ and σ, is given by application of the lemma 3). These definitions also imply that $\rho \in (b \to c)$ and $\sigma \in (b \to c)$.
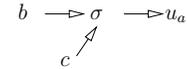
1. Let's proof that $\rho = \sigma$ by contradiction.

Proof. Let's assume that $\rho \neq \sigma$. Two cases are possible: $(b \to \rho \to \sigma \to c)$ or $(b \to \sigma \to \rho \to c)$.

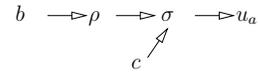(a) Let's assume that $(b \to \rho \to \sigma \to c)$ (Relation denoted Ⓐ).

     i. Let's show that $(a \to \rho) \cap (c \to \sigma) = \emptyset$ is impossible since it would imply $(ab) \parallel (cu_a)$.
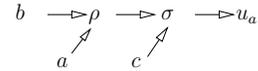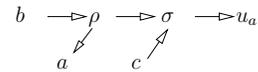
       Proof.

The definition of σ imply:

$$b \dashrightarrow \sigma \dashrightarrow u_a \quad\nearrow c$$

With Ⓐ, this becomes:

$$b \dashrightarrow \rho \dashrightarrow \sigma \dashrightarrow u_a \quad\nearrow c$$

With the definition of σ, this becomes:

$$b \dashrightarrow \rho \dashrightarrow \sigma \dashrightarrow u_a \quad\nearrow a \quad\nearrow c$$

With the hypothesis 2 (symmetry), this leads to:

$$b \dashrightarrow \rho \dashrightarrow \sigma \dashrightarrow u_a \quad\nearrow a \quad\nearrow c$$

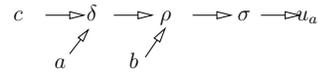Since we supposed $\rho \neq \sigma$, this gives $(b \to \rho \to a) \cap (c \to \sigma \to u_a) = \emptyset$.
A rewriting of each parts of $\cap$ leads to: $(b \to a) \cap (c \to u_a) = \emptyset$.
Thus, $(ab) \parallel (cu_a)$, which is contradictory with $b \perp c$. □

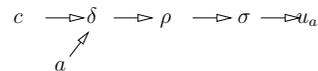   ii. Let's show that $(a \to \rho) \cap (c \to \sigma) \neq \emptyset$ is also impossible in that case.

       Proof. Let δ be the first element of $(a \to \rho) \cap (c \to \sigma)$.

The definition of δ as well as Ⓐ lead to:

$$c \dashrightarrow \delta \dashrightarrow \rho \dashrightarrow \sigma \dashrightarrow u_a \quad\nearrow a \quad\nearrow b$$

By definition of ρ, this leads to:

$$c \dashrightarrow \delta \dashrightarrow \rho \dashrightarrow \sigma \dashrightarrow u_a \quad\nearrow a$$

Thus, ρ is placed before σ on $(b \to u_a) \cap (c \to u_a)$, which is contradictory if $\rho \neq \sigma$ since σ is the first element of this set by definition. □

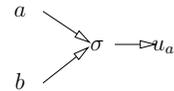   (i) and (ii) being the two only possible cases, the relation Ⓐ is contradictory.

(b) By symmetry, $(b \to \sigma \to \rho \to c)$ is also impossible.
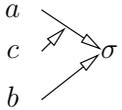
Both cases being impossible, the proposition $\rho \neq \sigma$ is contradictory, and thus $\rho = \sigma$. □

2. σ is thus at the same time the first element of $(a \to u_a) \cap (b \to u_a)$ and of $(b \to u_a) \cap (c \to u_a)$.

Since σ is the first element of $(a \to u_a) \cap (b \to u_a)$, we have:

$$a \searrow \quad \sigma \dashrightarrow u_a \quad b \nearrow$$

Only three cases are possible to add c on this figure while respecting the fact that σ is the first element of $(b \rightarrow u_a) \cap (c \rightarrow u_a)$:

- $\quad$ This situation is contradictory because it would imply (with both hypotheses of routing consistency and symmetry) that $(a \rightarrow c) \cap (b \rightarrow u_a) = \emptyset$ and thus $(ac) /\!\!/ (bu_a)$. This is contradictory since $a \perp b$.
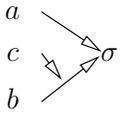
- $\quad$ By symmetry, this situation would imply $(bc) /\!\!/ (au_a)$.
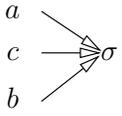
- $\quad$ This situation is thus the only possible one. This gives that σ is the first separator of a and c.

This gives that σ is a common separator for a, b and c.

In particular, this is a separator of a and c, and thus $a \perp c$.

$\square$

Remark 5. *The lemma 6 (transitivity of $\perp$) holds only if the hypothesis 2 (symmetric routing) is met. Figure 8 depicts a situation where this hypothesis is not respected, and where $a \perp b$ and $b \perp c$, but $a \not\perp c$ (because $(ad) /\!\!/ (cb)$).*
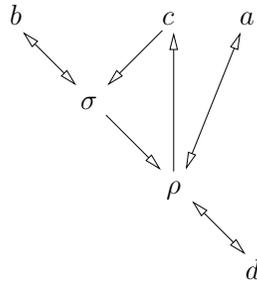

Figure 8: Counter-example to the lemma 6 (transitivity of $\perp$) when hypothesis 2 (symmetric routing) is not verified.

D   Proof of theorem 3 (representativity of the separator)

Theorem 3. *Let $\mathcal{C}$ be an equivalence class for $\bot$ and ρ a separator of its elements.*

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

Proof. Let ρ be a common separator for the nodes of $\mathcal{C}$ (its existence is given by the theorem 2). Let be $a \in \mathcal{C}$ and $b, u, v \in \mathcal{H}$.

$\Rightarrow$ **part (by contradiction):** Let's assume that $(a, u)\ \lambdaslash\ (b, v)$ and $(\rho, u)\ \not\lambdaslash\ (b, v)$.

This leads to $(a \to u) \cap (b \to v) \neq \emptyset$ and $(\rho \to u) \cap (b \to v) = \emptyset$.

With the hypothesis 1 (consistency of routing), this leads to $(a \to \rho) \cap (b \to v) \neq \emptyset$.

Let θ be the first node of $((a \to \rho) \setminus \{\rho\}) \cap (b \to v)$.

The path from a to v is thus $(a \to \theta \to v)$, and ρ is part of it, which is contradictory.

We proved that: $\big((a, u)\ \lambdaslash\ (b, v)\big) \vee \big((\rho, u)\ \not\lambdaslash\ (b, v)\big)$.

I.e., $\big((a, u)\ \lambdaslash\ (b, v)\big) \Rightarrow \big((\rho, u)\ \lambdaslash\ (b, v)\big)$.

$\Leftarrow$ **part:** Let's assume that $(a, u)\ \not\lambdaslash\ (b, v)$, *i.e.,* $(a \to u) \cap (b \to v) = \emptyset$.

ρ being a separator of a, it is placed on all the paths going out of a. In particular, $(a \to u) = (a \to \rho \to u)$.

A trivial rewrite of this relation gives: $(a \to \rho \to u) \cap (b \to v) = \emptyset$.

Thus, $(\rho \to u) \cap (b \to v) = \emptyset$, *i.e.,* $(\rho, u)\ \not\lambdaslash\ (b, v)$.

We proved that: $\big((a, u)\ \not\lambdaslash\ (b, v)\big) \Rightarrow \big((\rho, u)\ \not\lambdaslash\ (b, v)\big)$.

I.e., $\big((\rho, u)\ \lambdaslash\ (b, v)\big) \Rightarrow \big((a, u)\ \lambdaslash\ (b, v)\big)$.

We finally proved that: $(a, u)\ \lambdaslash\ (b, v) \Leftrightarrow (\rho, u)\ \lambdaslash\ (b, v)$. $\square$

E   Proof of the Tree algorithm correction

Proof (by induction). We want to proof the following relation (denoted Ⓐ in this proof):
$I(a, u, b, v) = 1 \Leftrightarrow \left(a \xrightarrow{G} u\right) \cap \left(b \xrightarrow{G} v\right) \neq \emptyset$.

$\{a_i\}$ denotes the succession of the leaders of the connected part containing a at step i.

Induction Hypothesis (IH): At the step i, each node quadruplet a, u, b, v can be in one of the three following cases:

- completely deconnected (*i.e.,* there is no path connecting one of these points).

- completely connected, and Ⓐ is verified.

- partially connected, and the existing links does not contradict Ⓐ.

Initialization: given that no connexion exists in the graph at step 0, the induction hypothesis is trivially true in that case.

Induction: Assuming that (IH) is true for all steps j so that $0 \leq j < i$, let's proof that it is then also true for the step i.

Even if does not introduce any particular difficulty, this proof is rather complex, and needs to study 13 separate cases depending on the fact that a, u, b and v are in the same connected part at the end of step i or not. Figure 9 depicts the several cases, which can be sorted in 4 categories.

1. The case ⑬ is trivial since no point is connected yet.

2. The cases ⑥, ⑦ and ⑧ are respectively symmetric to ③, ④ and ⑤ since $(a, u) \, \chi \, (b, v) \Leftrightarrow (b, v) \, \chi \, (a, u)$ by definition of χ.

3. For the cases ①, ③ and ④, we have to distinguish several sub-cases, most of them implying the existence of an already handled case at a previous step (which, by application of (IH) implies the truth of Ⓐ). Let's for example study ③. Depending of the the degree of a_i, three cases are then possible.

   (a) If the degree of a_i is 1, then a_{i-1} is also in the situation ③. Being handled at a previous step, we know by application of (IH) that Ⓐ is true.

   (b) If the degree of a_i is 2, then only two of the nodes a, u and b where connected at the step $i - 1$. the different possible cases come clearly down to previously studied cases: $(au)(b)(v)$ is the case ⑤, $(ab)(u)(v)$ is the case ⑨ and $(ub)(a)(v)$ is the case ⑫. Those cases arising in previous steps, Ⓐ is respected.

   (c) If the degree of a_i is 3, then the three points where placed in distinct sub-trees at step $i - 1$ and thus $a_i \in (a \rightarrow u) \cap (b \rightarrow v)$. Furthermore, can only be grouped that way in the same subtree by TREE when $a_{i-1} \perp u_{i-1} \perp b_{i-1}$. In particular, $(a_{i-1}, u_{i-1}) \, \chi \, (b_{i-1}, v_{i-1})$. By application of theorem 3 (separator representativity), this leads to $(a, u) \, \chi \, (b, v)$.

      We thus proved that $(a \rightarrow u) \cap (b \rightarrow v) \neq \emptyset$ only if $(a, u) \, \chi \, (b, v)$ in that case, *i.e.*, that Ⓐ is respected in that case.

   Ⓐ is thus respected for all sub-cases of ③.

4. The arguments in all other cases are very comparable to the sub-case (3c). For ② and ⑤, we show at the same time that $(a \rightarrow u) \cap (b \rightarrow v) = \emptyset$ and that $(a, u) \, /\!\!/ \, (b, v)$. In cases ⑨, ⑩, ⑪ and ⑫, we have at the same time $(a \rightarrow u) \cap (b \rightarrow v) \neq \emptyset$ and $(a, u) \, \chi \, (b, v)$.

Thus, in all cases possible at step i, (A) is satisfied if it were respected in previous steps. By induction, the correction of the TREE algorithm is thus proved. □

F Proof of the Tree algorithm termination conditions

Theorem 5. *Given an instance of* INTERFERENCEGRAPH, *if it is possible to build a tree graph G being a solution, then* TREE *terminates with only one leader.*

This proof is trivial. If it is possible to construct a tree being solution of the problem, each sub-tree of this solution is in total interference with the external nodes. The algorithm is thus certain to group those nodes together in the same connected set once it did handle all the sub-trees of the studied sub-tree. TREE will thus continue until all nodes are placed in a connected set.

References

[1] Henri Casanova, Arnaud Legand, and Loris Marchal. Scheduling Distributed Applications: the SimGrid Simulation Framework. In *Proceedings of the third IEEE International Symposium on Cluster Computing and the Grid (CCGrid'03)*, may 2003.

[2] Peter A. Dinda and David R. O'Hallaron. An extensible toolkit for resource prediction in distributed systems. Technical Report CMU-CS-99-138, School of Computer Science, Carnegie Mellon University, July 1999.

[3] Matthew B. Doar. A better model for generating test networks. In *Globecom '96*, Nov 1996. Available at http://citeseer.nj.nec.com/doar96better.html.

[4] Ian Foster and Carl Kesselman (Eds.). *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 1999.

[5] Arnaud Legrand and Martin Quinson. Automatic deployment of the network weather service using the effective network view. In *Workshop on Grid Benchmarking, associated to IPDPS'04*, 2004.

[6] Vern Paxson. *Measurements and Analysis of End-to-End Internet Dynamics*. PhD thesis, University of California, Berkeley, 1997.

[7] Michael Rabbat, Robert D. Nowak, and Mark Coates. Multiple Source Network Tomography. In *IEEE InfoComm*, Honk-Hong, March 7-11 2004.

[8] Gary Shao, Francine Berman, and Rich Wolski. Using effective network views to promote distributed application performance. In *International Conference on Parallel and Distributed Processing Techniques and Applications*, June 1999. Available at http://apples.ucsd.edu/pubs/pdpta99.ps.
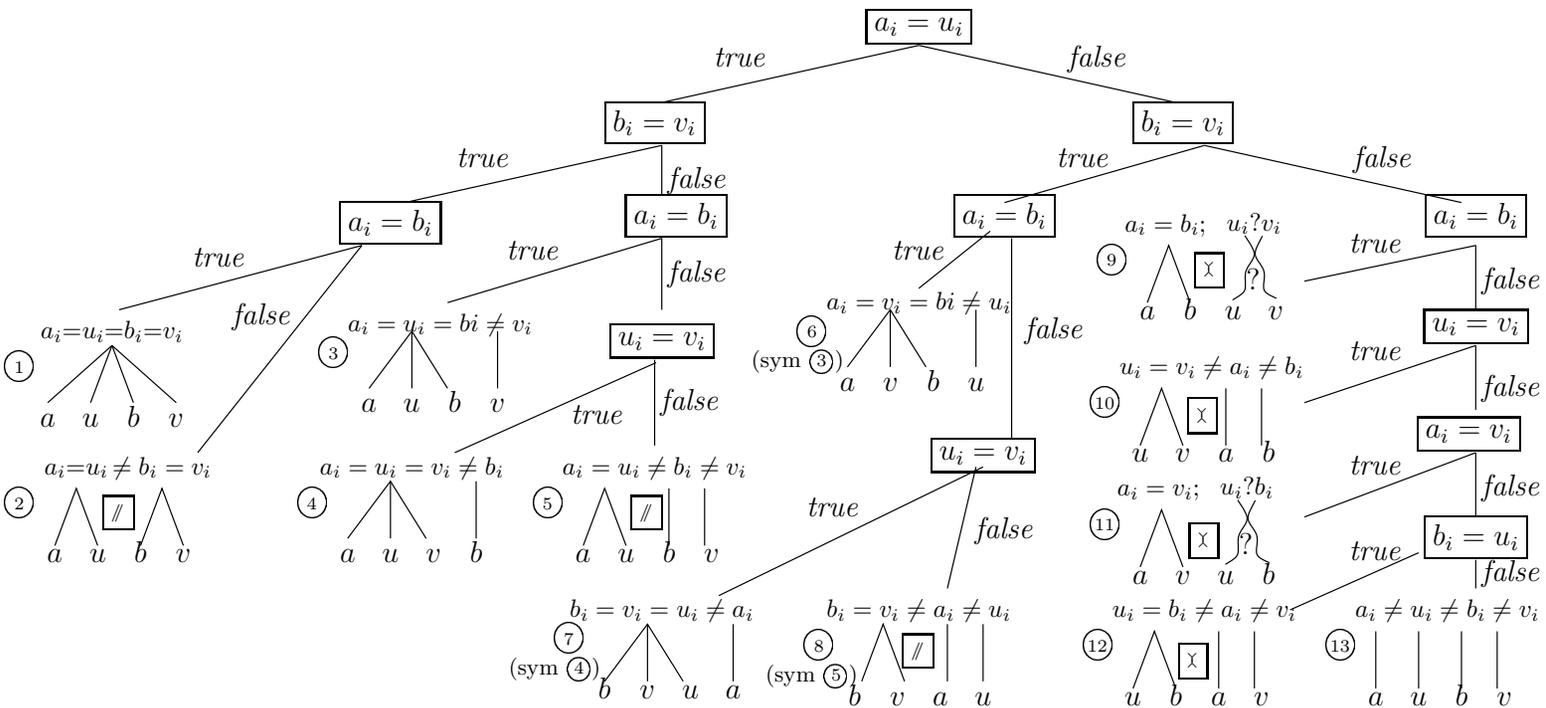
$a_i = u_i$

true   *false*

$b_i = v_i$ $b_i = v_i$

true   *false*   *true*   *false*

$a_i = b_i$ $a_i = b_i$ $a_i = b_i$ $a_i = b_i$

true   *false*   *true*   *false*   *true*   *false*   *true*   *false*

① $a_i = u_i = b_i = v_i$ $a\ u\ b\ v$

② $a_i = u_i \neq b_i = v_i$ $a\ u\ $ // $\ b\ v$

③ $a_i = u_i = b_i \neq v_i$ $a\ u\ b\ v$

④ $a_i = u_i = v_i \neq b_i$ $a\ u\ v\ b$

$u_i = v_i$

true   *false*

⑤ $a_i = u_i \neq b_i \neq v_i$ $a\ u\ $ // $\ b\ v$

$u_i = v_i$

true   *false*

⑦ (sym ④) $b_i = v_i = u_i \neq a_i$ $b\ v\ u\ a$

⑧ (sym ⑤) $b_i = v_i \neq a_i \neq u_i$ $b\ v\ $ // $\ a\ u$

⑥ (sym ③) $a_i = v_i = b_i \neq u_i$ $a\ v\ b\ u$

⑨ $a_i = b_i;\quad u_i ? v_i$ $a\ b\ u\ v$

⑩ $u_i = v_i \neq a_i \neq b_i$ $u\ v\ a\ b$

⑪ $a_i = v_i;\quad u_i ? b_i$ $a\ v\ u\ b$

⑫ $u_i = b_i \neq a_i \neq v_i$ $u\ b\ a\ v$

$u_i = v_i$

true   *false*

$a_i = v_i$

true   *false*

$b_i = u_i$

true   *false*

⑬ $a_i \neq u_i \neq b_i \neq v_i$ $a\ u\ b\ v$

Figure 9: The several cases of the induction proving the correction of the TREE algorithm.

[9] Rich Wolski, Neil Spring, and Jim Hayes. The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing. *Future Generation Computing Systems, Metacomputing Issue*, 15(5–6):757–768, Oct. 1999.