First Steps Towards Automatically Building Network Representations

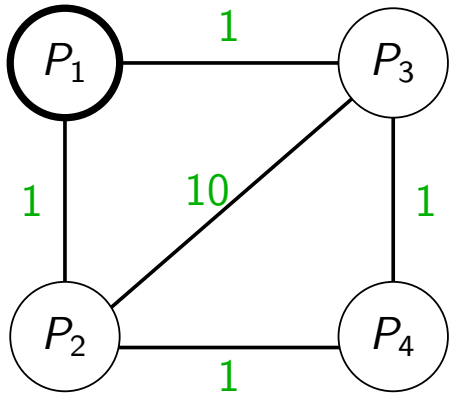
Lionel Eyraud-Dubois	ÉNS-Lyon, France.
Arnaud Legrand	CNRS, Grenoble, France.
Martin Quinson	Nancy University, France.
Frédéric Vivien	INRIA, Lyon, France.

Euro-Par'07

Rennes, August 2007

Scheduling on a large-scale distributed platform

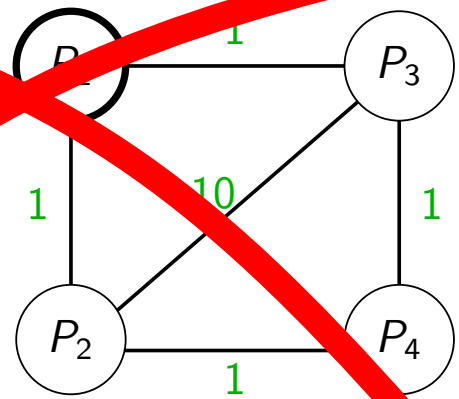
- Let $G_P = (V_P, E_P)$ denote the *platform graph*



- Each edge $P_i \longrightarrow P_j$ is labeled by $c_{i,j}$:
  time necessary to send a unit-size message between P_i and P_j
- Communication model:
  - full-overlap of communications and computations
  - 1-port for incoming communications and 1-port for outgoing communications
- Each node P_i has a processing speed $w_i \in \mathbb{R}$

Scheduling on a large-scale distributed platform

- Let $G_P = (V_P, E_P)$ denote the *platform graph*



- Each edge $P_i \longrightarrow P_j$ is labeled by $c_{i,j}$:
  time necessary to send a unit-size message between P_i and P_j
- Communication model:
  - full-overlap of communications and computations
  - 1-port for incoming communications and 1-port for outgoing communications
- Each node P_i has a processing speed $w_i \in \mathbb{R}$

Eh wait!

How did you get the graph?!

Building a network representation

Motivation

- Modern platforms are **heterogeneous** and **dynamic**
- Distributed applications must be **network-aware** and **reactive**
- Information on the network needed (at least) for:
  - Service and distributed application deployment
  - Communication-aware scheduling
  - Group communication
  - Proximity Neighbor Selection in P2P systems

Several levels of information (depending on the OSI layer)

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

Network mapping process

- Step 1: (End-to-end) measurements
- Step 2: Reconstruct a graph

Classical measurements in a grid environment?

Use of low-level network protocols (like SNMP or BGP)

- Example: Remos
- Use of SNMP restricted for security reasons (DoS or spying)

Use of traceroute or ping (i.e. on ICMP)

- Examples: TopoMon, Lumeta, IDmaps, Global Network Positioning
- Use of ICMP more and more restricted by admins (for security reasons)

> *Over the lifetime of the project, we have noticed that the number of replying destinations in our lists decays at the rate of 2-3% per month.* – Authors of the Skitter project

Pathchar

- Works without privilege on the network, but must be root on hosts

$\Rightarrow$ not adapted to Grid settings

Classical measurements in a grid environment?

Use of low-level network protocols (like SNMP or BGP)

- Example: Remos
- Use of SNMP restricted for security reasons (DoS or spying)

Use of traceroute or ping (i.e. on ICMP)

- Examples: TopoMon, Lumeta, IDmaps, Global Network Positioning
- Use of ICMP more and more restricted by admins (for security reasons)

> *Over the lifetime of the project, we have noticed that the number of replying destinations in our lists decays at the rate of 2-3% per month.* – Authors of the Skitter project

Pathchar

- Works without privilege on the network, but must be root on hosts

$\Rightarrow$ not adapted to Grid settings

Measurements must be at application-level (no privilege)

Solutions relying on application-level measurements

NWS (Network Weather Service – UCSB)

- Reports bandwidth, latency, CPU availability, and future trends
- Only quantitative values, no topological information
  (but one can label a big clique with NWS-provided values)

ENV (Effective Network View – UCSD)

- Use interference measurements to build a tree representation

ECO (Efficient Collective Communication – CMU)

- Use application-level measurements to optimize collective communications
- Should be generalized

Existing reconstruction algorithms

- Cliques (NWS, ECO) or trees (ENV, Classical latency clustering)

Solutions relying on application-level measurements

NWS (Network Weather Service – UCSB)

- Reports bandwidth, latency, CPU availability, and future trends
- Only quantitative values, no topological information
  (but one can label a big clique with NWS-provided values)

ENV (Effective Network View – UCSD)

- Use interference measurements to build a tree representation

ECO (Efficient Collective Communication – CMU)

- Use application-level measurements to optimize collective communications
- Should be generalized

Existing reconstruction algorithms

- Cliques (NWS, ECO) or trees (ENV, Classical latency clustering)

Our goal

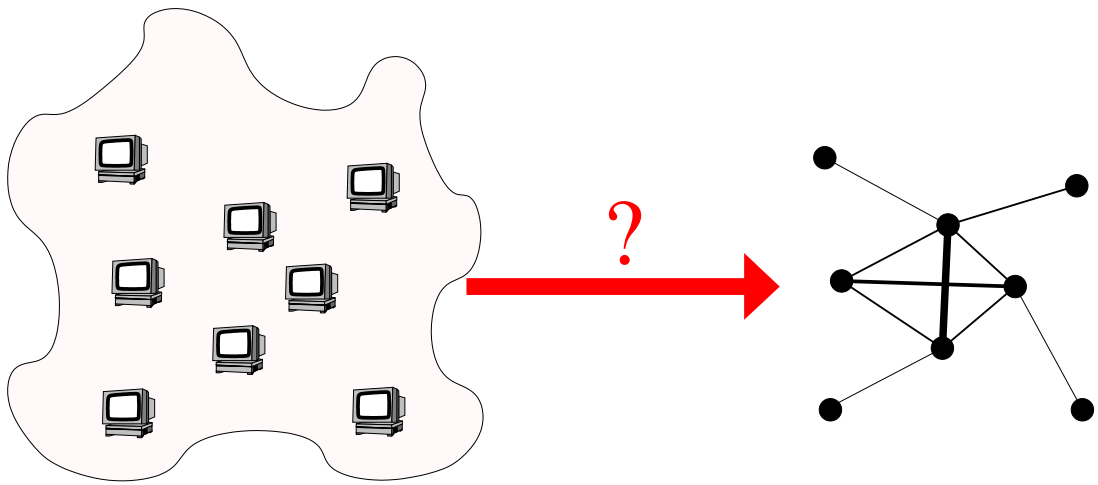
- Assess quality of clique and spanning tree algorithms
- Propose original approaches

Outline

ALNeM (Application-Level Network Mapper)

Presentation

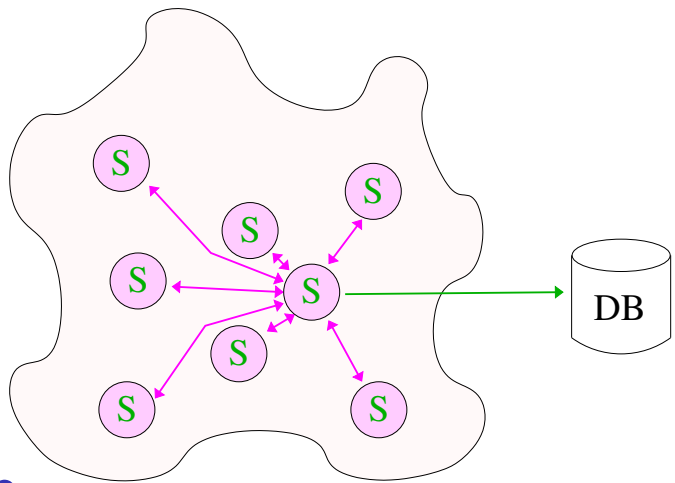
- Long-term goal: be a tool providing topology to network-aware applications
- Short-term goal: allow the study of network mapping algorithms



ALNeM (Application-Level Network Mapper)

Presentation

- Long-term goal: be a tool providing topology to network-aware applications
- Short-term goal: allow the study of network mapping algorithms



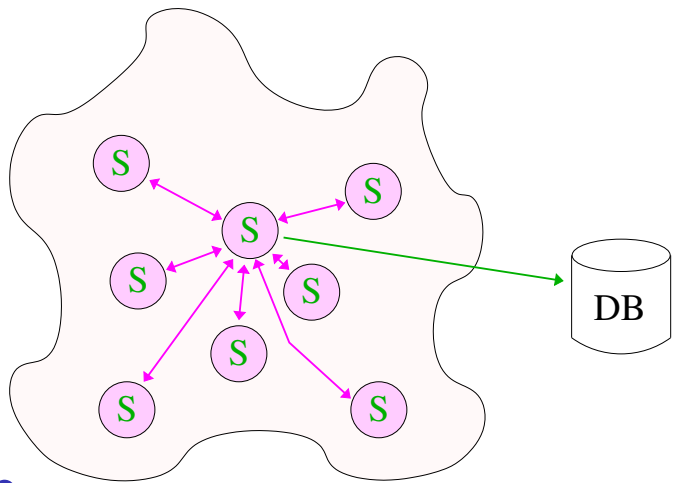
Architecture

- Lightweight distributed measurement infrastructure (collection of sensors)
- MySQL measurement database

ALNeM (Application-Level Network Mapper)

Presentation

- Long-term goal: be a tool providing topology to network-aware applications
- Short-term goal: allow the study of network mapping algorithms



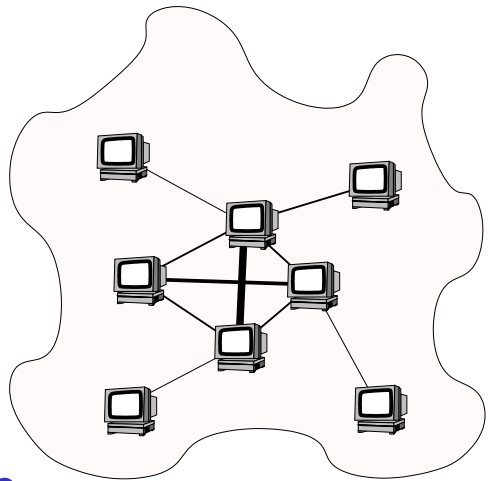
Architecture

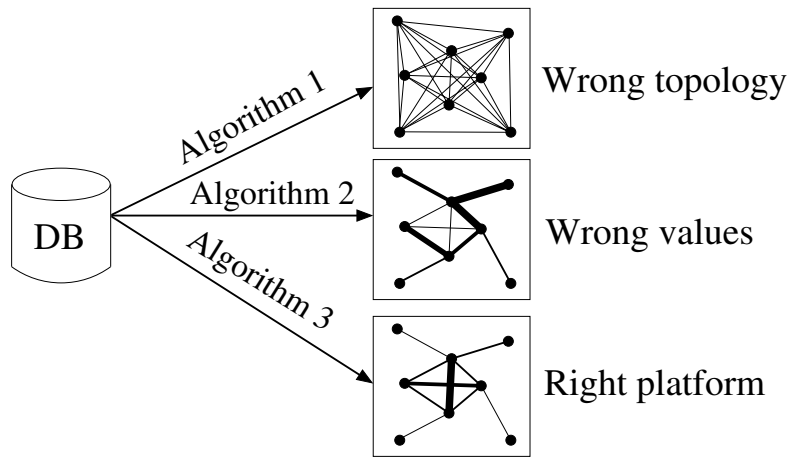
- Lightweight distributed measurement infrastructure (collection of sensors)
- MySQL measurement database

ALNeM (Application-Level Network Mapper)

Presentation

- Long-term goal: be a tool providing topology to network-aware applications
- Short-term goal: allow the study of network mapping algorithms





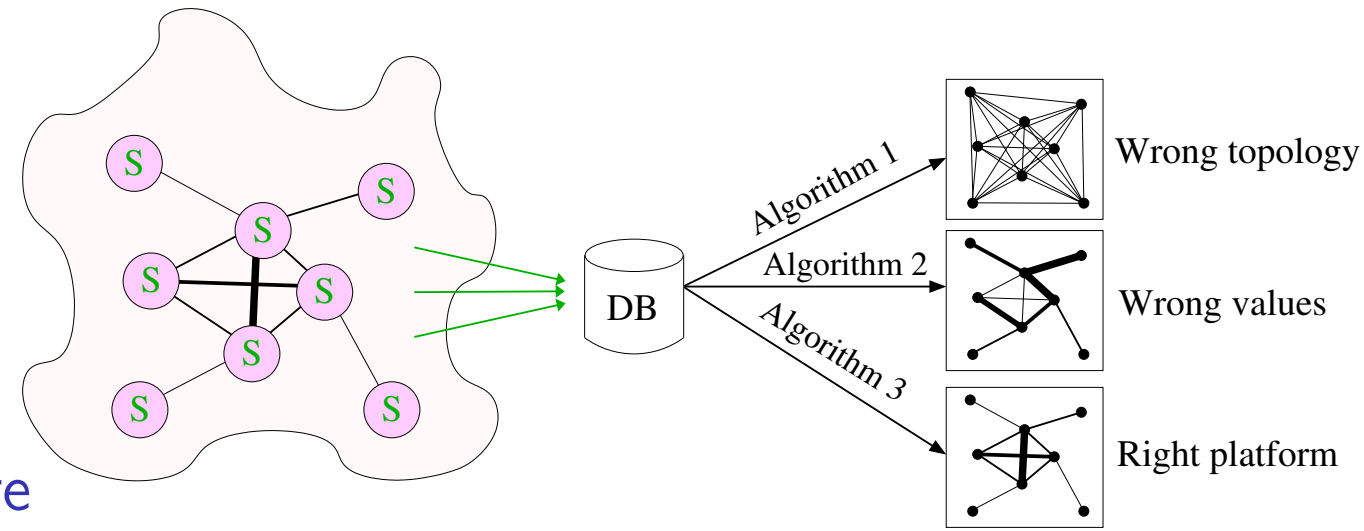
Architecture

- Lightweight distributed measurement infrastructure (collection of sensors)
- MySQL measurement database
- Topology builder, with several reconstruction algorithms

ALNeM (Application-Level Network Mapper)

Presentation

- Long-term goal: be a tool providing topology to network-aware applications
- Short-term goal: allow the study of network mapping algorithms



Architecture

- Lightweight distributed measurement infrastructure (collection of sensors)
- MySQL measurement database
- Topology builder, with several reconstruction algorithms

Development on simulator, use in real life

- Implemented using GRAS (same code running in both contexts)

Evaluation methodology

Goal: Quantify similarity between initial and reconstructed platforms. Not so easy!

Evaluation methodology

Goal: Quantify similarity between initial and reconstructed platforms. Not so easy!

4 evaluation approaches

- Visual evaluation (structural comparison)
- Compare end-to-end measurements (communication-level)
- Compare interference amount:

$$Interf\left((a,b),(c,d)\right) = 1 \text{ iff } \frac{BW(a \to b)}{BW(a \to b \parallel c \to d)} \approx 2$$

- Compare application running times (application-level)

	Comm. schema	// comm	# steps
Token-ring	Ring	No	1
Broadcast	Tree	No	1
All2All	Clique	Yes	1
Parallel Matrix Multiplication	2D	Yes	$\sqrt{procs}$

Evaluation methodology

Goal: Quantify similarity between initial and reconstructed platforms. Not so easy!

4 evaluation approaches

- Visual evaluation (structural comparison)
- Compare end-to-end measurements (communication-level)
- Compare interference amount:

$$Interf\left((a,b),(c,d)\right) = 1 \text{ iff } \frac{BW(a \to b)}{BW(a \to b \parallel c \to d)} \approx 2$$

- Compare application running times (application-level)

	Comm. schema	// comm	# steps
Token-ring	Ring	No	1
Broadcast	Tree	No	1
All2All	Clique	Yes	1
Parallel Matrix Multiplication	2D	Yes	$\sqrt{procs}$

Apply all approaches on several platforms

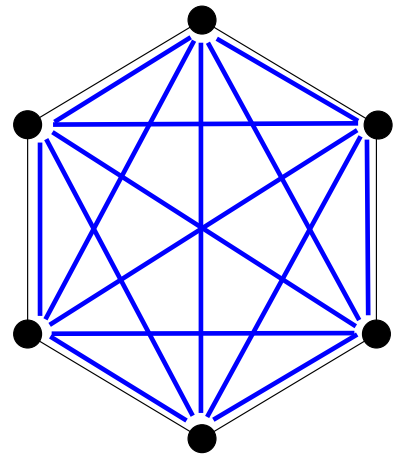
- In simulation: collect data on "real" platforms, compare reconstructed to initial
- *In situ:* most comparisons not applicable, so hard to assess quality, but still usable

Outline

Basic reconstruction algorithms

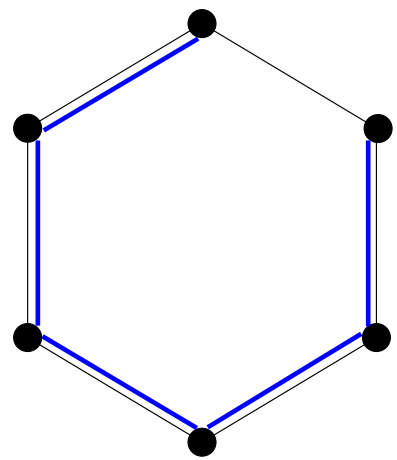
Clique

- Connect all pairs of nodes, label with measured values



Spanning trees

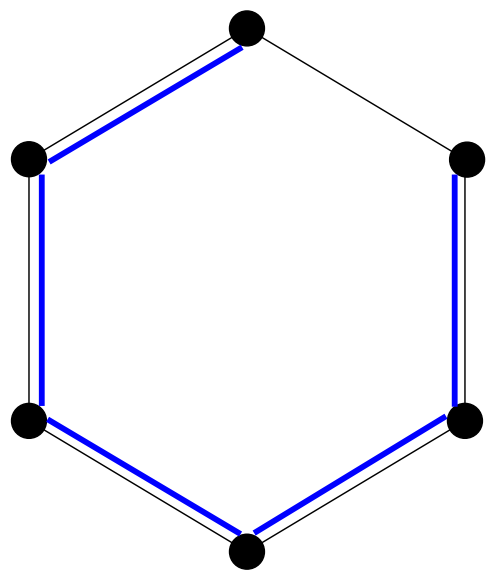
- use edges with lowest latency or highest bandwidth



New heuristic: Improved spanning tree

Algorithm

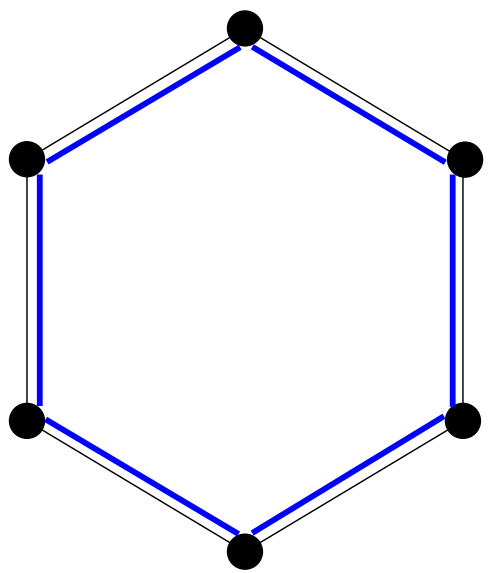
- Add links to the spanning trees, to improve predictions
- Connect the closest badly predicted nodes (latency over-estimated)



New heuristic: Improved spanning tree

Algorithm

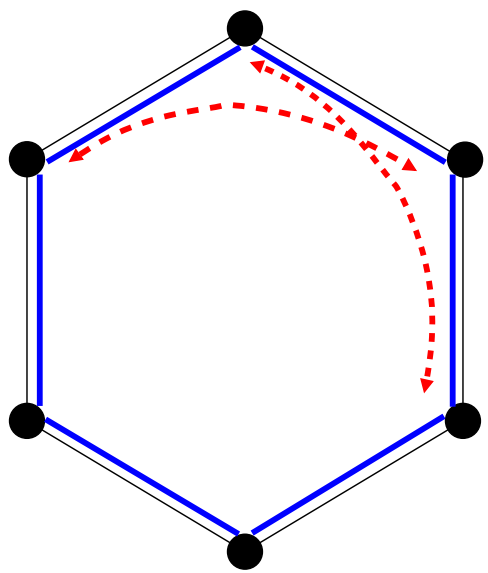
- Add links to the spanning trees, to improve predictions
- Connect the closest badly predicted nodes (latency over-estimated)



New heuristic: Improved spanning tree

Algorithm

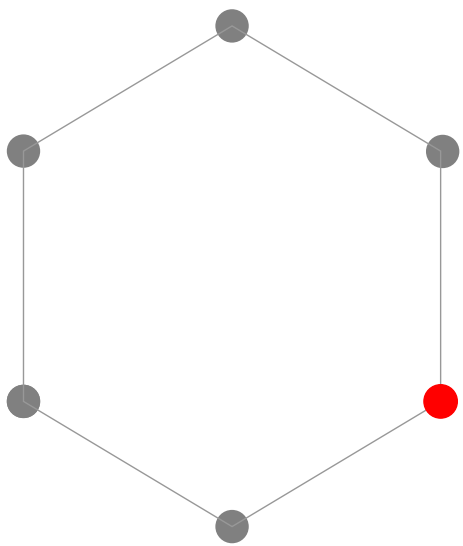
- Add links to the spanning trees, to improve predictions
- Connect the closest badly predicted nodes (latency over-estimated)
- Update the routing if it improves the predictions



New heuristic: Aggregation

Algorithm

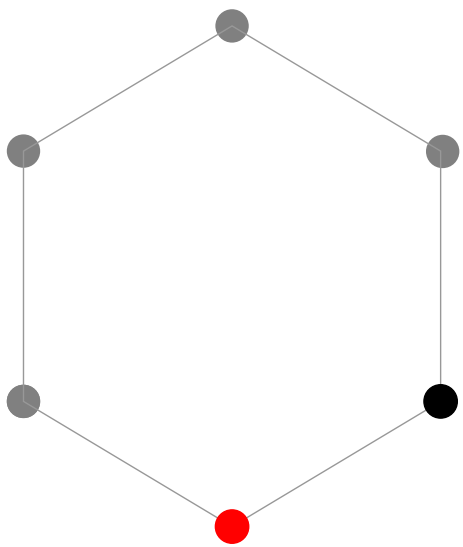
- Grow a set of *connected* nodes
- For each node:
  - Repeatedly add edges to predict better the latency
  - Until all routes from this node to the set are satisfied
- Refrain from adding "redundant" edges



New heuristic: Aggregation

Algorithm

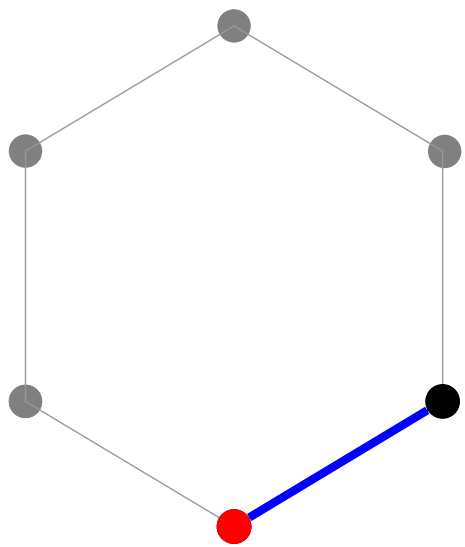
- Grow a set of *connected* nodes
- For each node:
  - Repeatedly add edges to predict better the latency
  - Until all routes from this node to the set are satisfied
- Refrain from adding "redundant" edges



New heuristic: Aggregation

Algorithm

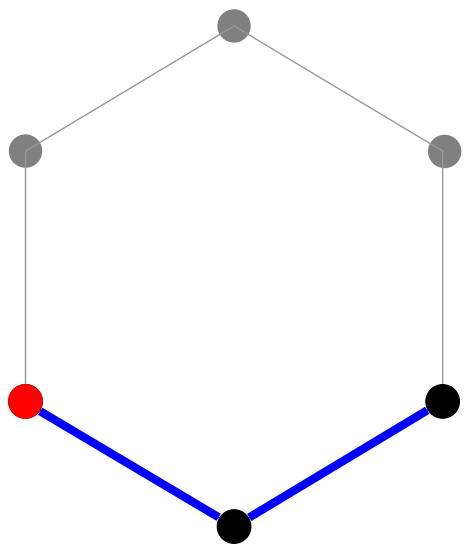
- Grow a set of *connected* nodes
- For each node:
  - Repeatedly add edges to predict better the latency
  - Until all routes from this node to the set are satisfied
- Refrain from adding "redundant" edges



New heuristic: Aggregation

Algorithm

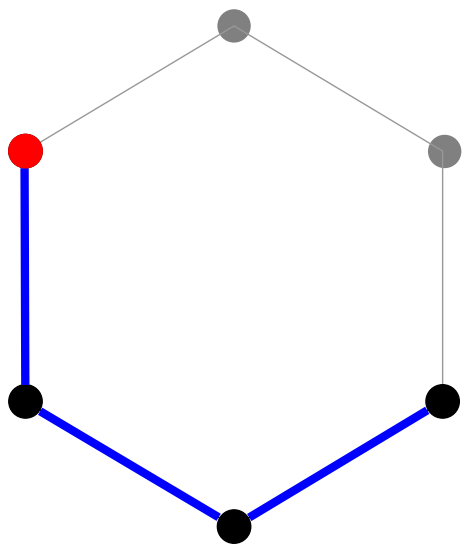
- Grow a set of *connected* nodes
- For each node:
  - Repeatedly add edges to predict better the latency
  - Until all routes from this node to the set are satisfied
- Refrain from adding "redundant" edges



New heuristic: Aggregation

Algorithm

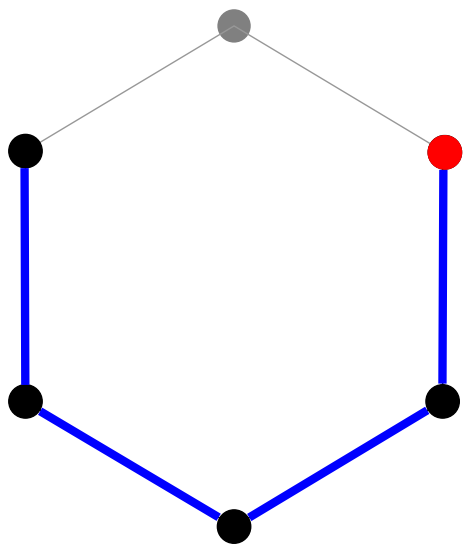
- Grow a set of *connected* nodes
- For each node:
  - Repeatedly add edges to predict better the latency
  - Until all routes from this node to the set are satisfied
- Refrain from adding "redundant" edges



New heuristic: Aggregation

Algorithm

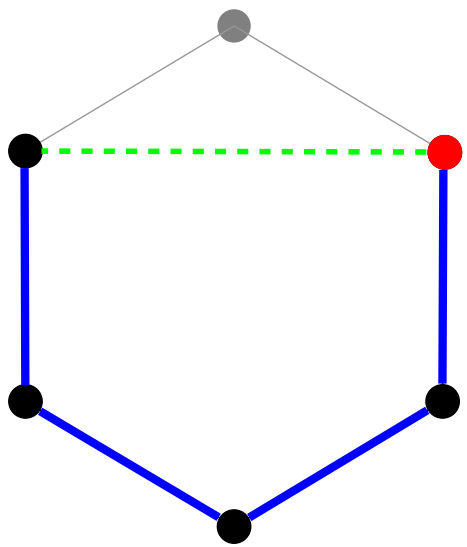
- Grow a set of *connected* nodes
- For each node:
  - Repeatedly add edges to predict better the latency
  - Until all routes from this node to the set are satisfied
- Refrain from adding "redundant" edges



New heuristic: Aggregation

Algorithm

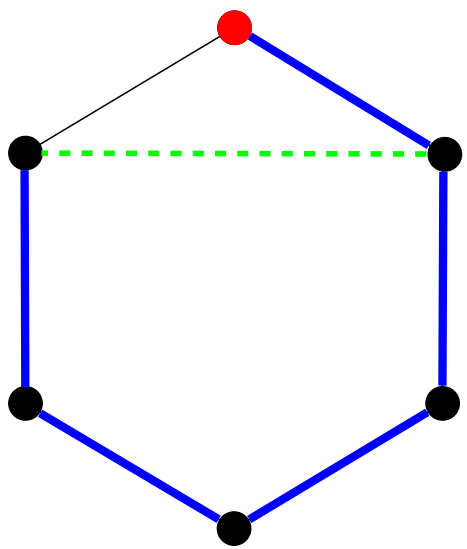
- Grow a set of *connected* nodes
- For each node:
  - Repeatedly add edges to predict better the latency
  - Until all routes from this node to the set are satisfied
- Refrain from adding "redundant" edges
  - long link ($pred > 2 \times meas$) suspected of redundancy with forthcoming links



New heuristic: Aggregation

Algorithm

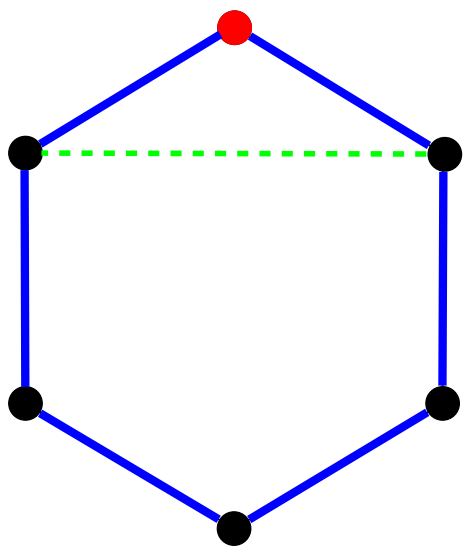
- Grow a set of *connected* nodes
- For each node:
  - Repeatedly add edges to predict better the latency
  - Until all routes from this node to the set are satisfied
- Refrain from adding "redundant" edges
  - long link ($pred > 2 \times meas$) suspected of redundancy with forthcoming links



New heuristic: Aggregation

Algorithm

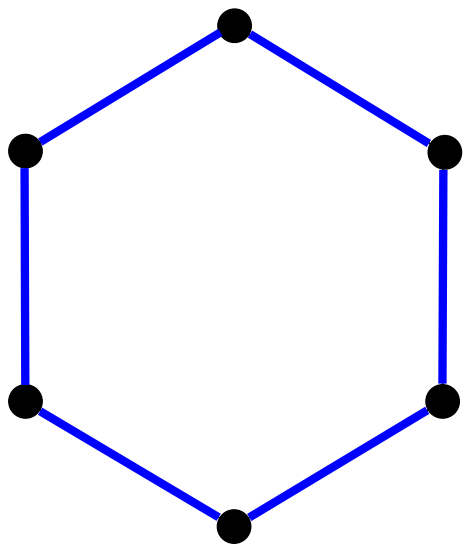
- Grow a set of *connected* nodes
- For each node:
  - Repeatedly add edges to predict better the latency
  - Until all routes from this node to the set are satisfied
- Refrain from adding "redundant" edges
  - long link ($pred > 2 \times meas$) suspected of redundancy with forthcoming links
  - Indeed, green link is redundant.



New heuristic: Aggregation

Algorithm

- Grow a set of *connected* nodes
- For each node:
  - Repeatedly add edges to predict better the latency
  - Until all routes from this node to the set are satisfied
- Refrain from adding "redundant" edges
  - long link ($pred > 2 \times meas$) suspected of redundancy with forthcoming links
  - Indeed, green link is redundant. Thus removed.

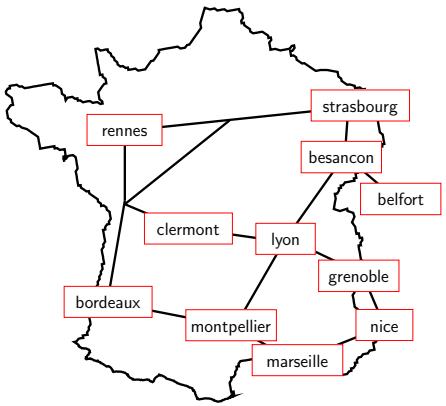


Outline

Experiments on simulator: Renater platform (1/4)

- *Real* platform built manually (real measurements + admin feedback)

Visual evaluation

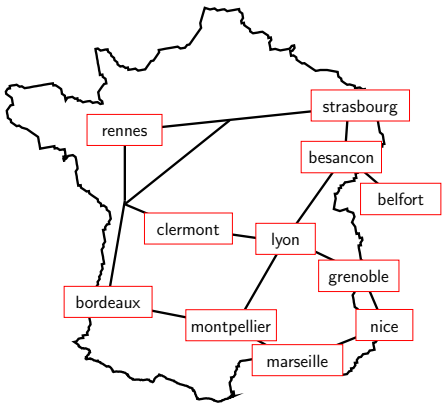


Real platform

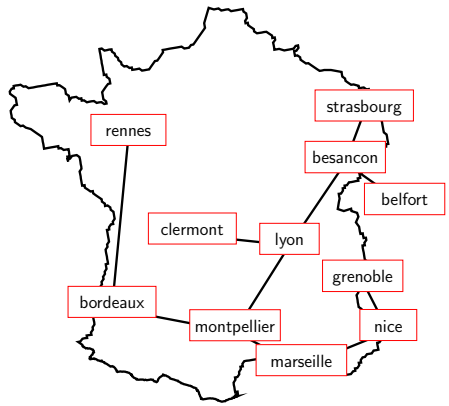
Experiments on simulator: Renater platform (1/4)

- *Real* platform built manually (real measurements + admin feedback)

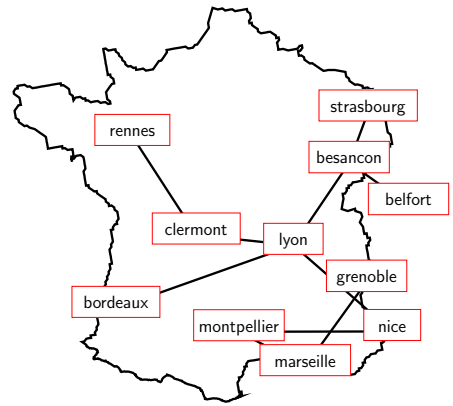
Visual evaluation



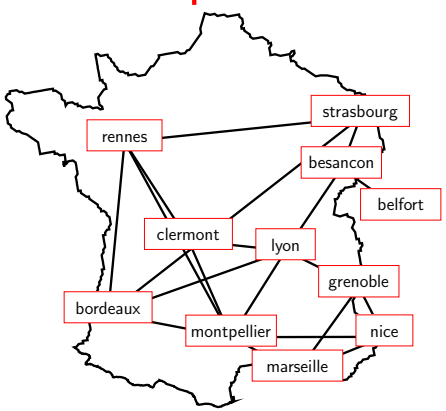
Real platform



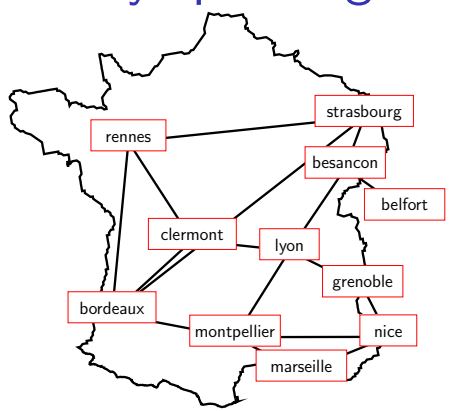
Latency spanning tree



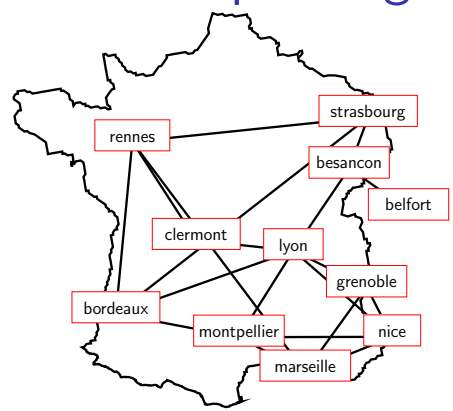
Bandwidth spanning tree



Aggregate



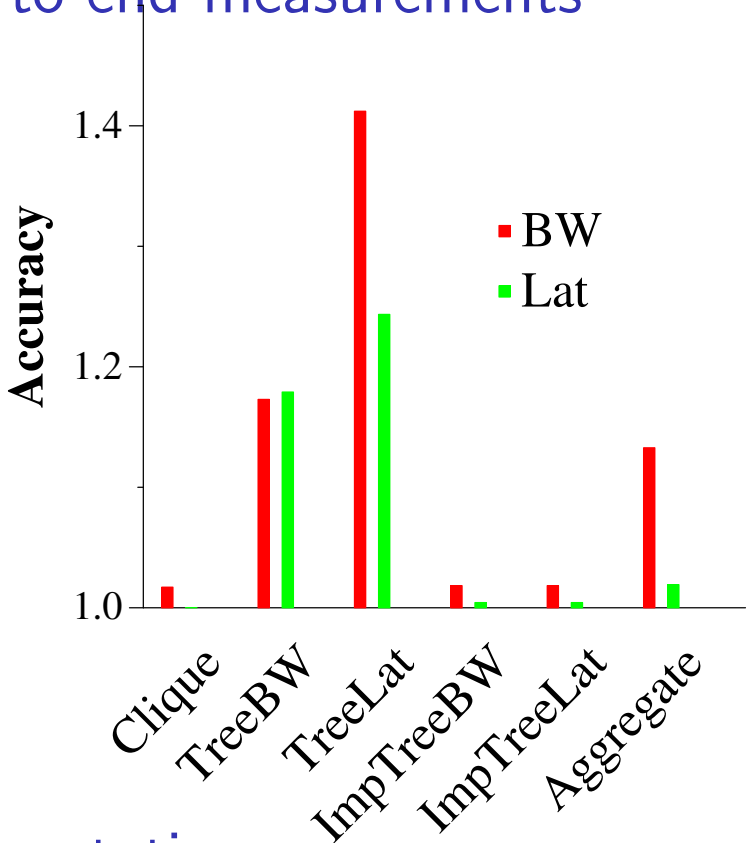
Latency improved spanning tree



Bandwidth improved spanning tree

Experiments on simulator: Renater platform (2/4)

End to end measurements



How to read:

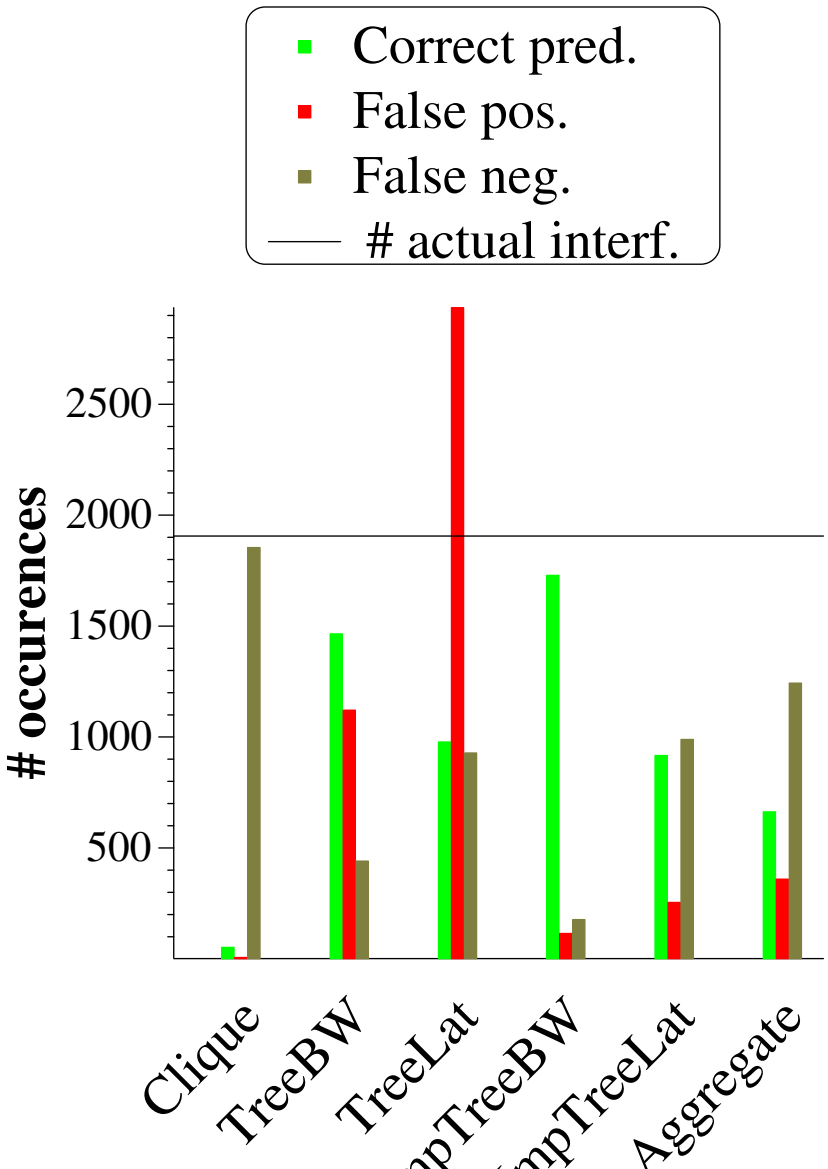
accuracy closer to 1 (lower bar)
$\rightsquigarrow$ better result

Interpretation

- Clique: very good (of course)
- Spanning Trees: missing links give bad predictions
- Improvement procedure helps
- Aggregate performs badly for bandwidth predictions

Experiments on simulator: Renater platform (3/4)

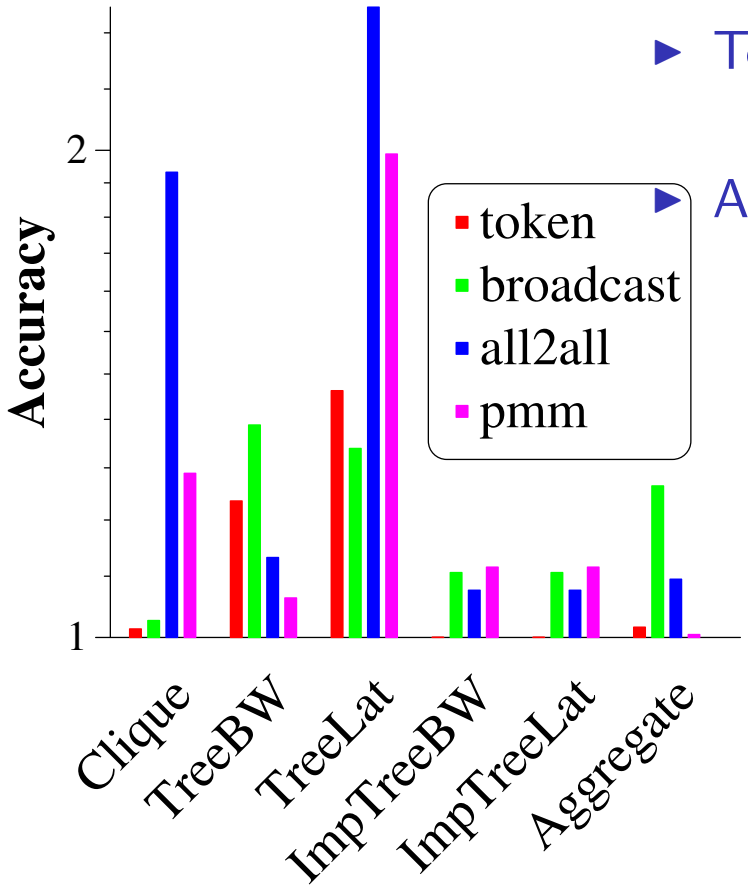
Interference measurements



- $Interf_{init}(ab, cd) \stackrel{?}{=} Interf_{recons}(ab, cd)$
- Tree*
  - Misses some links
  - Most existing interferences right
  - Lot of false positive
- Clique
  - Predicts no interference at all
  - No false positive
  - Very few actual interferences
- Improvement reduces false positives

Experiments on simulator: Renater platform (4/4)

Application-level measurements

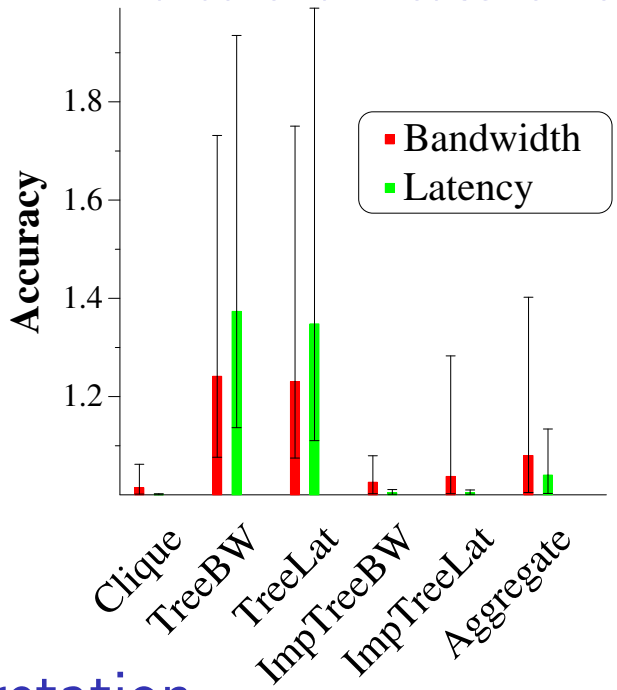


- Token and broadcast: same conclusion than end2end
  - Clique good, Trees bit worse, Improvements work
- All2all and pmm: completely new light
  - Clique dramatically underestimates times: No contention between parallel communication
  - Tree* overestimate times: Missing links (as before)
  - Improved algorithms have good predictive power

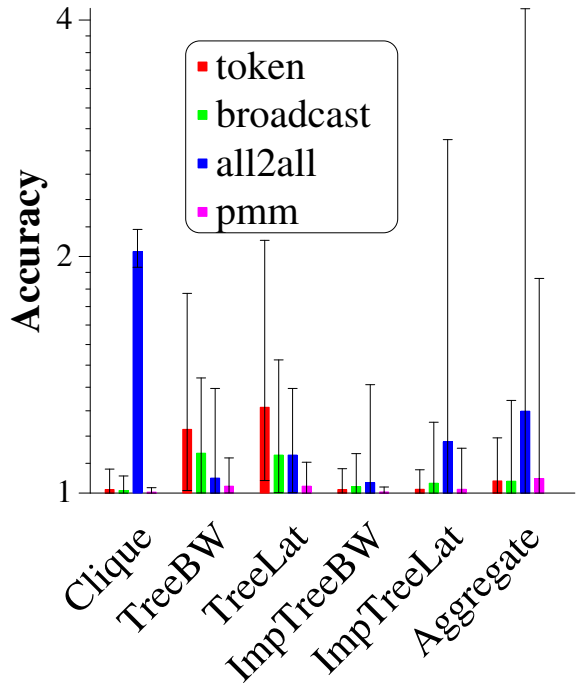
Experiments on simulator: GridG platforms

- GridG is a synthetic platform generator [Lu, Dinda – SuperComputing03]
  Generates *realistic* platforms
- Experiment: 40 platforms (60 hosts – default GridG parameters)

End to end measurements



Application-level measurements



Interpretation

- Naive algorithms get bad results
- Improved trees yield good reconstructions
  - ImpTreeBW error $\approx$ 3% for all2all (worst case)

Contributions of ALNeM

Completed a framework for reconstruction algorithm evaluation

- Several criterion of similarity between initial and reconstructed platforms visual (structural), end-to-end, interferences, application timings
- Allows comparison of reconstruction algorithms from application POV
- Runs on simulator or *in-situ* thanks to GRAS (& SimGrid)

Contributions of ALNeM

Completed a framework for reconstruction algorithm evaluation

- Several criterion of similarity between initial and reconstructed platforms visual (structural), end-to-end, interferences, application timings
- Allows comparison of reconstruction algorithms from application POV
- Runs on simulator or *in-situ* thanks to GRAS (& SimGrid)

Analyzed classical algorithms and proposed original ones

- Evaluated algorithms: Clique, Bandwidth or Latency Spanning Tree
- Evaluation conditions: Simulator, both real and synthetic platforms

Contributions of ALNeM

Completed a framework for reconstruction algorithm evaluation

- Several criterion of similarity between initial and reconstructed platforms visual (structural), end-to-end, interferences, application timings
- Allows comparison of reconstruction algorithms from application POV
- Runs on simulator or *in-situ* thanks to GRAS (& SimGrid)

Analyzed classical algorithms and proposed original ones

- Evaluated algorithms: Clique, Bandwidth or Latency Spanning Tree
- Evaluation conditions: Simulator, both real and synthetic platforms

Conclusion

- Classical algorithms are not satisfactory
  - Spanning trees: miss edges, leading to performance under-estimation
  - Clique: do not capture any existing interference
- Improving spanning trees yield much better results
  - Especially ImpTreeBW: uses both kind of measurements

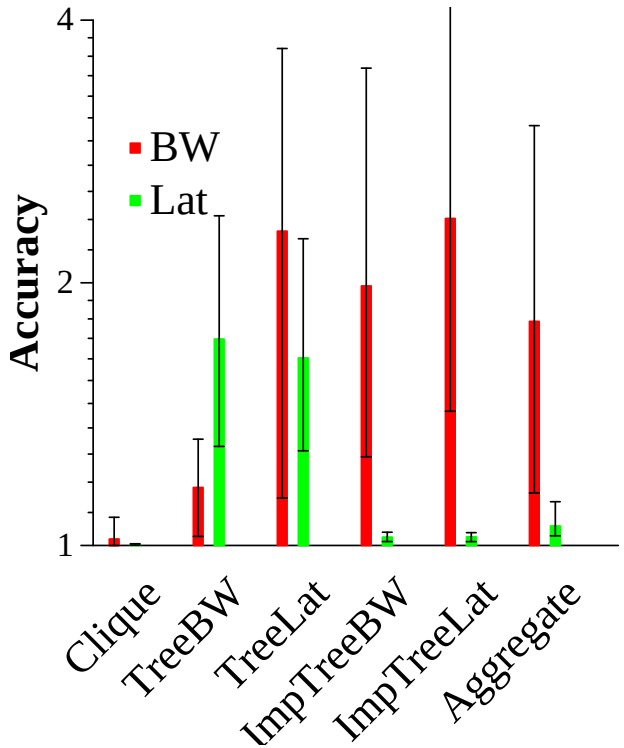
Adding routers to the picture

- New set of experiments: only *leaf* nodes run the measurement processes

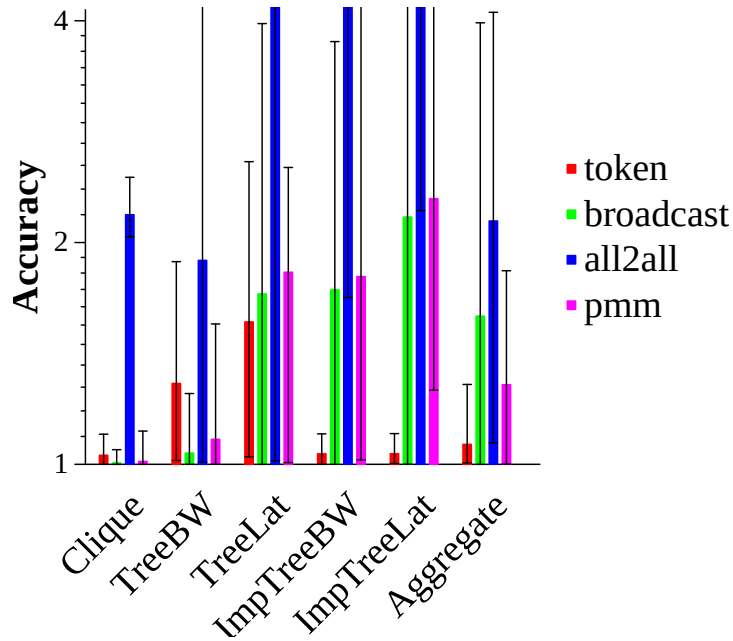
Adding routers to the picture

- New set of experiments: only *leaf* nodes run the measurement processes

End to end measurements



Application-level measurements



Interpretation

- None of the proposed heuristic is satisfactory
- Future work: improve this!

Future works on the ALNeM project

Better reconstruction algorithms

- Mix bandwidth and latency values
- Build graphs with internal (hidden) nodes

Other measurements from the sensors (new inputs to algorithms)

- Interference (but very expensive to acquire)
- Packet gap and back-to-back packets
- Packet loss, etc.

Method based on successive refinements

1. Spanning tree as first approximation
2. Refinement by adding some missing links
3. Some (not all) interference measurements to double-check the result

Far future:

- Adapt to condition changes (bandwidth variation, node arrival/departure)
- Distribute the tool to end-users

Thank your for your attention

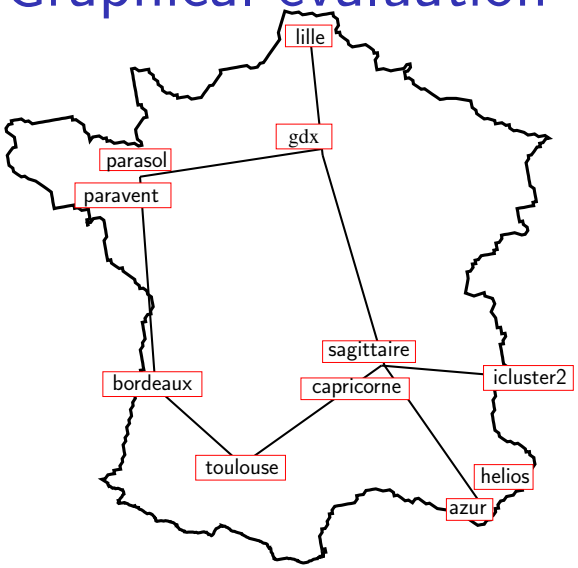
Any question?

Experiments on a real platform: Grid'5000

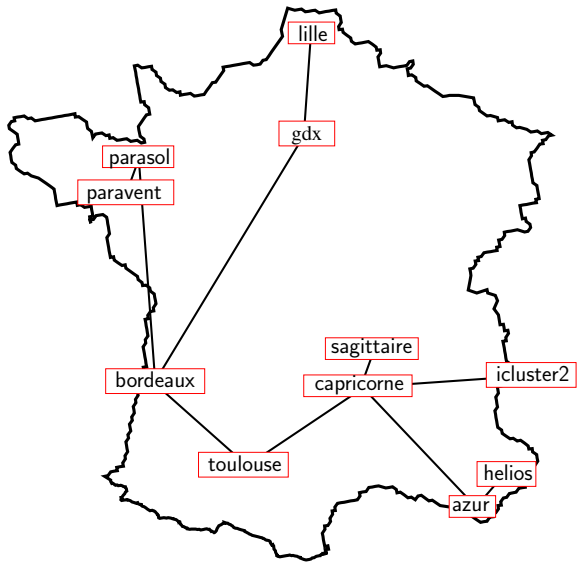
The Grid'5000 platform

- Test-bed for Grid researchers
- 9 sites in France, targets 5000 cores (2500 currently)

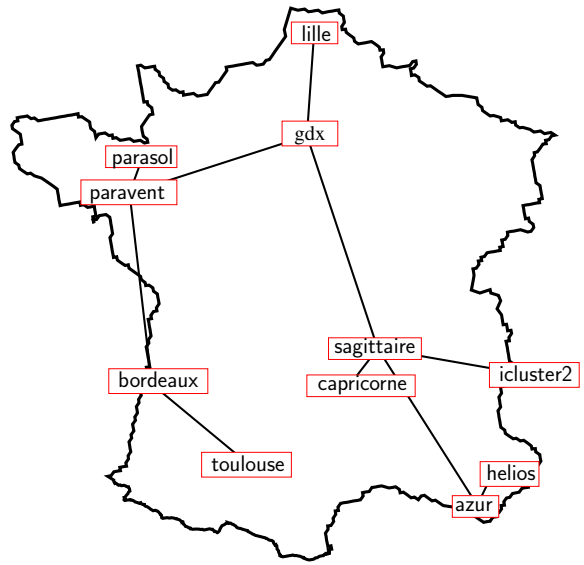
Graphical evaluation



Real platform



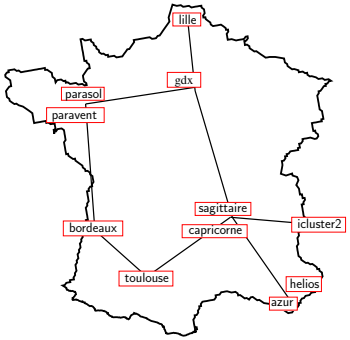
Latency spanning tree



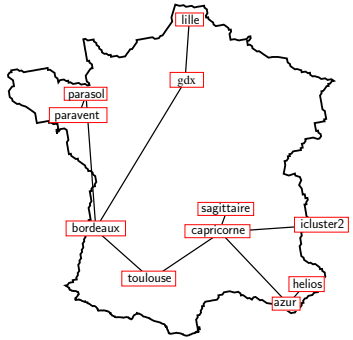
Bandwidth spanning tree

- Some links are missing, of course
- Bandwidth induced graph better, but maybe G5K more focused on bandwidth

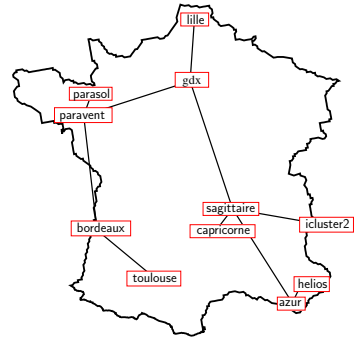
Experiments on a real platform: Grid'5000



Real platform



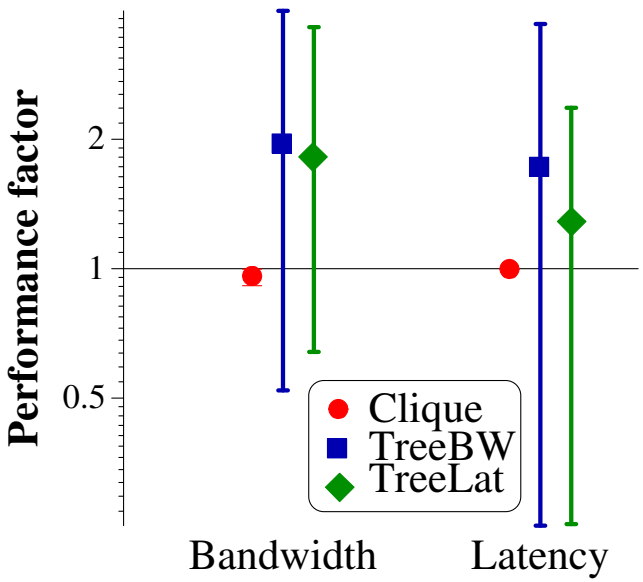
Latency spanning tree



Bandwidth spanning tree

End-to-end measurement

- Compare real measurements to the one in simulator on reconstructed platform

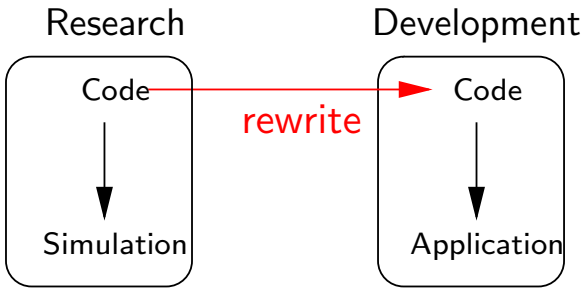


- Clique very good, but trivial result
- Latency:
  - Over-estimated when missing links ($\rightsquigarrow$ longer path)
  - Under-estimated when routing on G5K optimizes bandwidth
- Bandwidth mis-estimated:
  - Technical issue in simulator:
    assumes constant TCP window size but it varies with clusters in G5K (simulator validation issue)

Goals of the GRAS project

Easing infrastructure development

Development of real distributed applications using a simulator

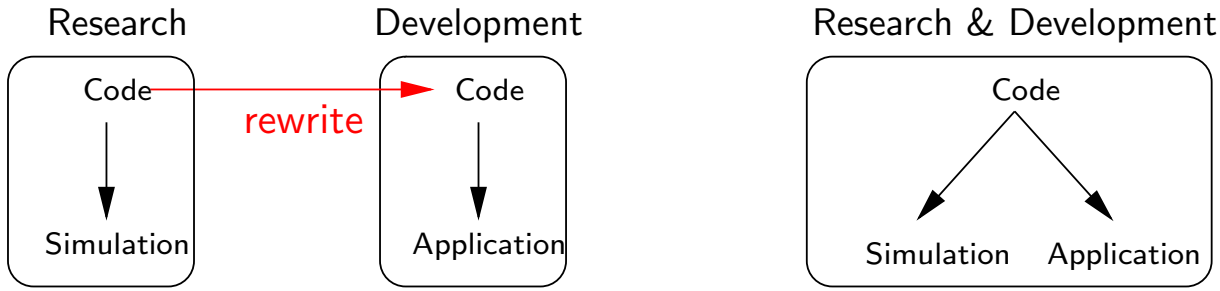


Without GRAS

Goals of the GRAS project

Easing infrastructure development

Development of real distributed applications using a simulator



Without GRAS

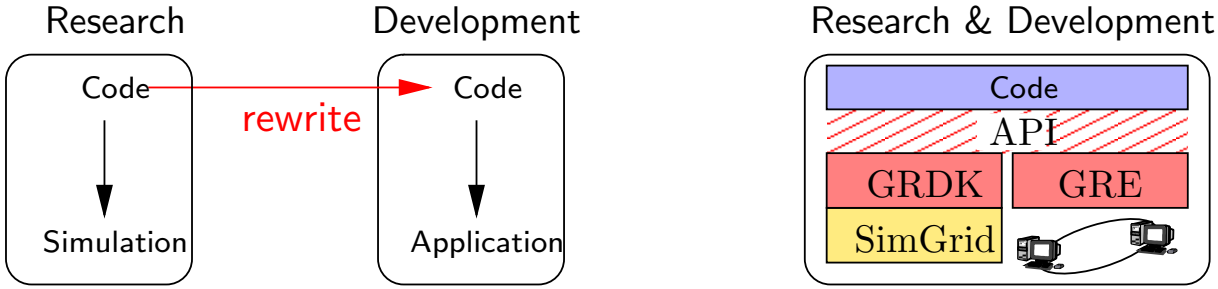
With GRAS

- Framework for Rapid Development of Distributed Infrastructure
  - Develop and tune on the simulator; Deploy *in situ* without modification

Goals of the GRAS project

Easing infrastructure development

Development of real distributed applications using a simulator



Without GRAS

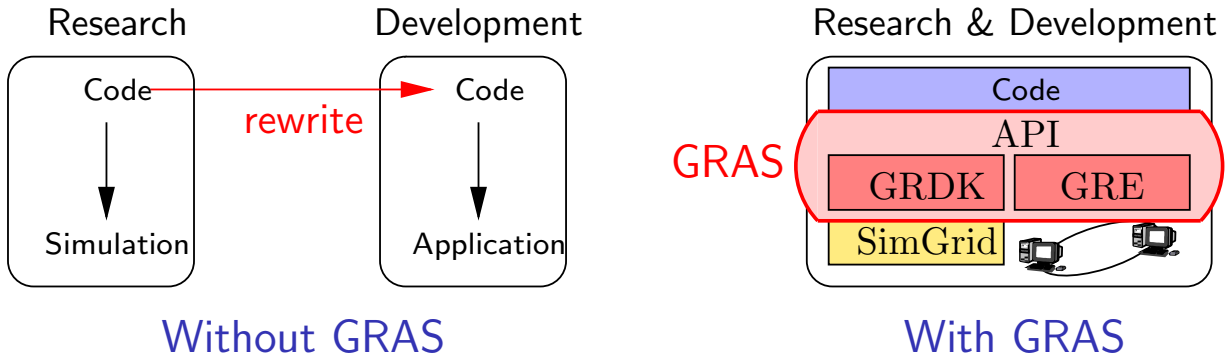
With GRAS

- Framework for Rapid Development of Distributed Infrastructure
  - Develop and tune on the simulator; Deploy *in situ* without modification
    How: One API, two implementations

Goals of the GRAS project

Easing infrastructure development

Development of real distributed applications using a simulator

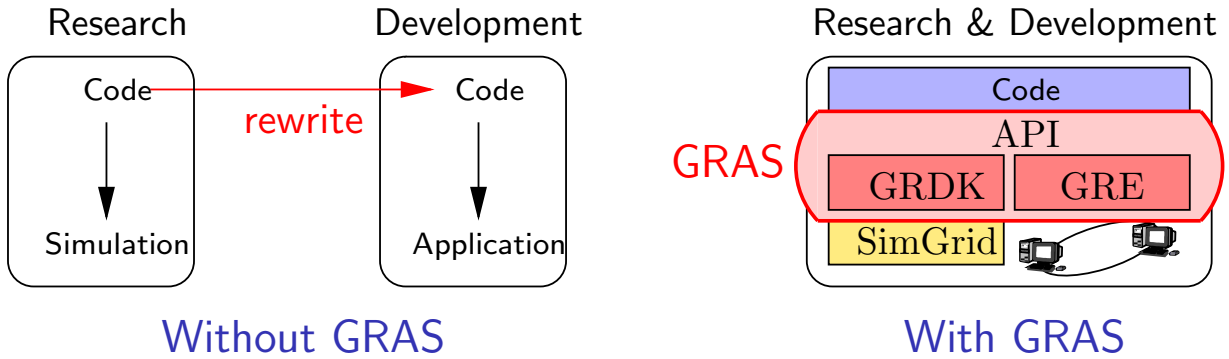


- Framework for Rapid Development of Distributed Infrastructure
  - Develop and tune on the simulator; Deploy *in situ* without modification
    How: One API, two implementations

Goals of the GRAS project

Easing infrastructure development

Development of real distributed applications using a simulator

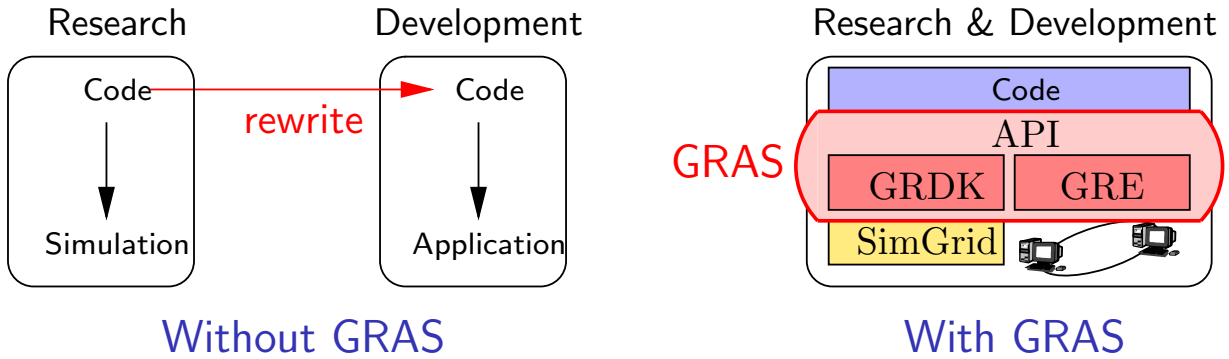


- Framework for Rapid Development of Distributed Infrastructure
  - Develop and tune on the simulator; Deploy *in situ* without modification
    How: One API, two implementations
- Efficient Grid Runtime Environment (result = application $\neq$ prototype)

Goals of the GRAS project

Easing infrastructure development

Development of real distributed applications using a simulator



- Framework for Rapid Development of Distributed Infrastructure
  - Develop and tune on the simulator; Deploy *in situ* without modification
    How: One API, two implementations
- Efficient Grid Runtime Environment (result = application $\neq$ prototype)
  - Performance concern: efficient communication of structured data
    How: Efficient wire protocol (avoid data conversion)
  - Portability concern: because of grid heterogeneity
    How: ANSI C + autoconf + no dependency