Martin QUINSON
University of California, Santa Barbara (UCSB)

Automatic discovery of the characteristics and capacities of a distributed computational platform

10 mai 2004

Introduction to the Grid

Metacomputing: aggregating distributed computers and storage units

the resulting platform is usually called the Grid

- Very high potential (in power and ease of use)
- The Grid hardware is already there

  Share of local resources between several organizations $\Rightarrow$ WAN constellation of LAN
- The Grid software infrastructure only emerging.

  Difficulties come from (amongst others):
  - Heterogeneity
  - Resource sharing ($\Rightarrow$ availability variations)
  - Multiple organizations (trust issue)

Which information for which scheduling?

Random scheduling:

- Tasks list; existing hosts list

Simple scheduling:

- About tasks: theoretical complexity (like $O(n)$)
- About hosts: peak performance or on a given benchmark
- About links: maximal capacities

Current Grid scheduling:

- About hosts: up/down, CPU and memory load
- About links: current capacities matrix

Which information for which scheduling?

Random scheduling:

- Tasks list; existing hosts list

Simple scheduling:

- About tasks: theoretical complexity (like $O(n)$)
- About hosts: peak performance or on a given benchmark
- About links: maximal capacities

Current Grid scheduling:

- About hosts: up/down, CPU and memory load
- About links: current capacities matrix

Information quality is crucial to scheduling quality

Overview of this work

Our goal: provide the information needed by the scheduler.

I. Quantitative knowledge of **needs** (tasks) and **availabilities** (servers and network)

NWS + FAST

II. Qualitative knowledge of network **topology**

ENV$\longrightarrow$ ALNeM

Overview of this work

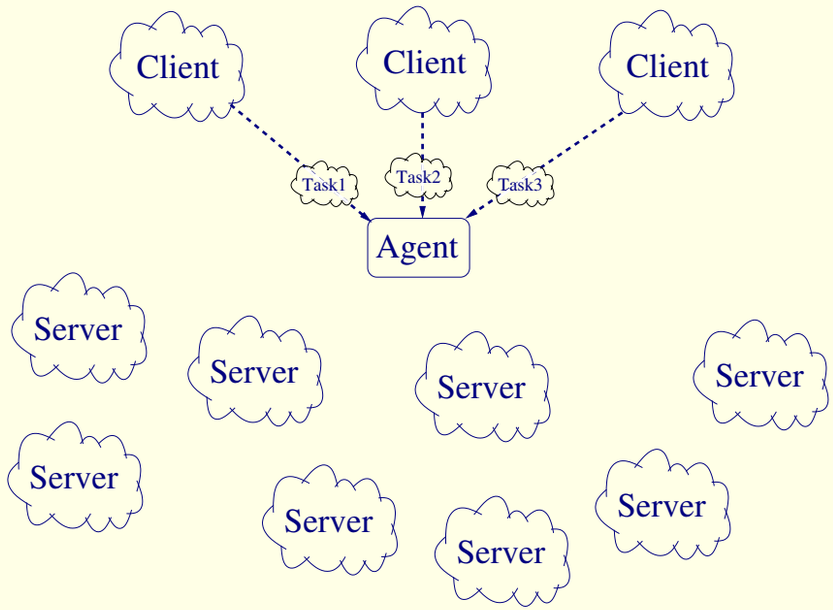
Our goal: provide the information needed by the scheduler.

I. Quantitative knowledge of **needs** (tasks) and **availabilities** (servers and network)

NWS + FAST

II. Qualitative knowledge of network **topology**

ENV$\longrightarrow$ ALNeM



Overview of this work

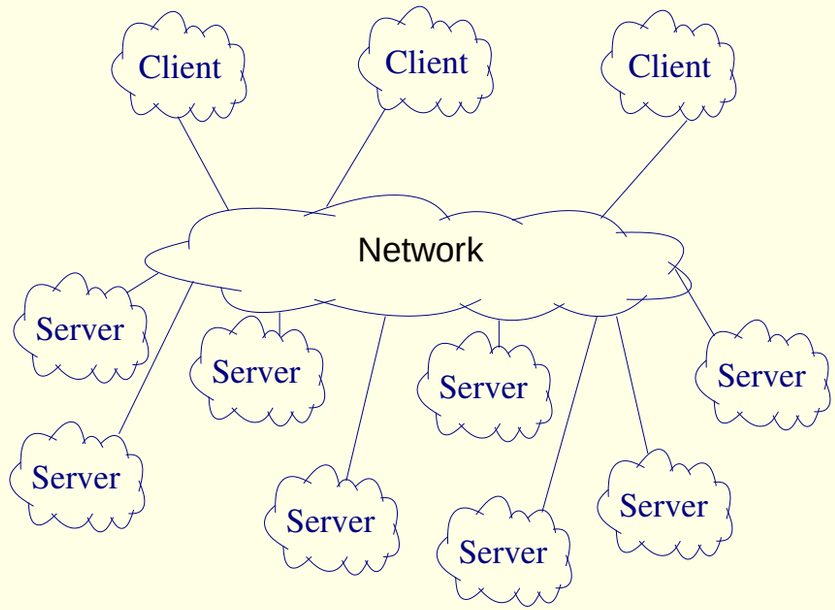
Our goal: provide the information needed by the scheduler.

I. Quantitative knowledge of **needs** (tasks) and **availabilities** (servers and network)

NWS + FAST

II. Qualitative knowledge of network **topology**

ENV$\longrightarrow$ ALNeM



Overview of this work

Our goal: provide the information needed by the scheduler.

I. Quantitative knowledge of **needs** (tasks) and **availabilities** (servers and network)

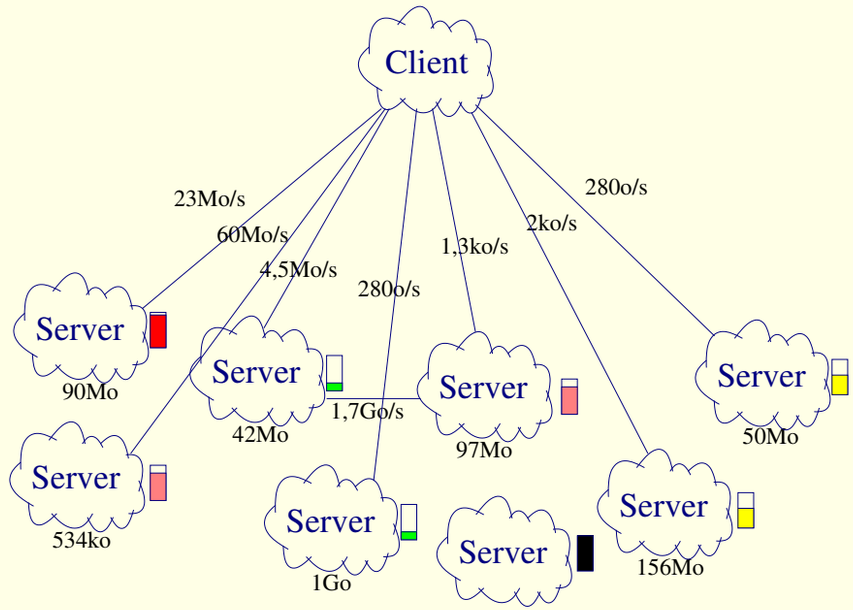
NWS + FAST

II. Qualitative knowledge of network **topology**

ENV$\longrightarrow$ ALNeM

NWS [RSH99] forecasts:

- bandwidth, latency, memory, disk space, . . .
- host load as percentage



Overview of this work

Our goal: provide the information needed by the scheduler.

I. Quantitative knowledge of **needs** (tasks) and **availabilities** (servers and network)

NWS + FAST

II. Qualitative knowledge of network **topology**

ENV$\longrightarrow$ ALNeM

NWS [RSH99] forecasts:

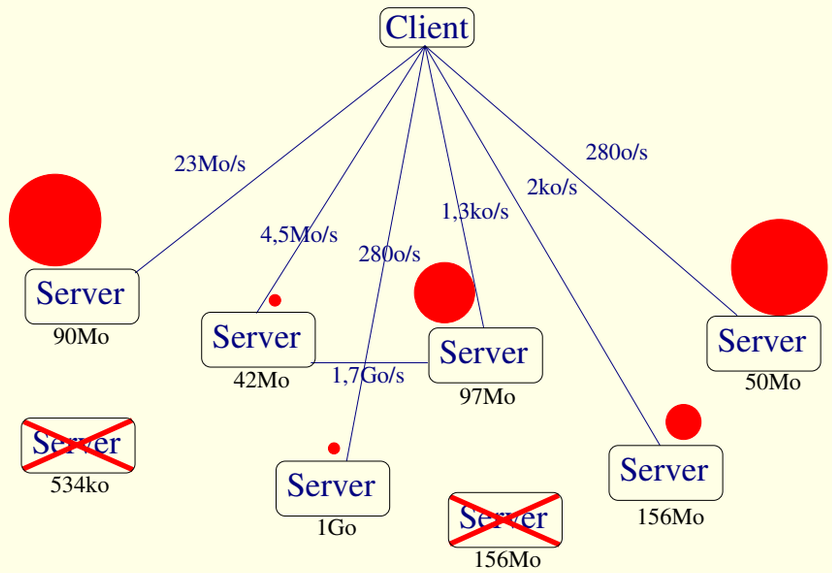
- bandwidth, latency, memory, disk space, . . .
- host load as percentage

FAST [Qui02b] provides:

- Task needs benchmarking

  time and memory size (fitting to the host)

$\Rightarrow$ Duration of the task on each server



Overview of this work

Our goal: provide the information needed by the scheduler.

I. Quantitative knowledge of **needs** (tasks) and **availabilities** (servers and network)

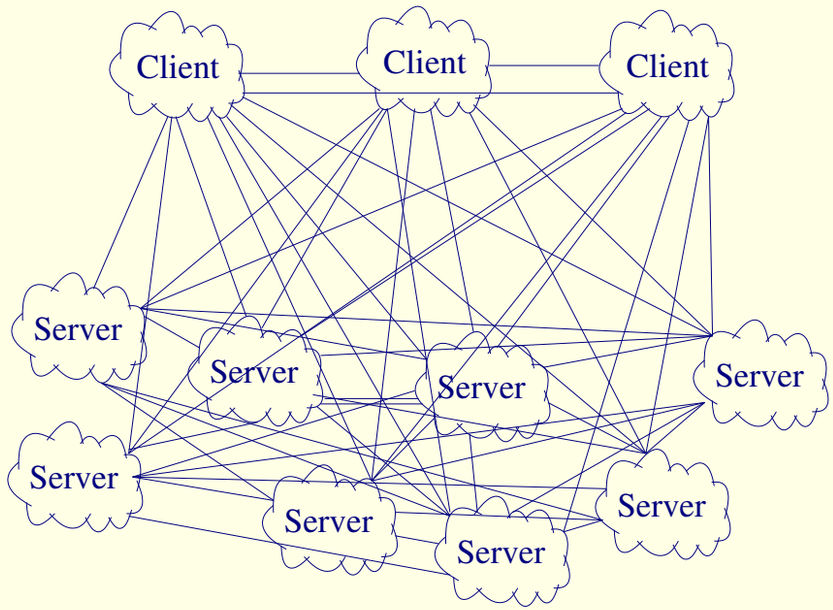
NWS + FAST

II. Qualitative knowledge of network **topology**

ENV→ ALNeM

Motivating example: how to configure NWS?

- Simplest: measure everything



Overview of this work

Our goal: provide the information needed by the scheduler.

I. Quantitative knowledge of **needs** (tasks) and **availabilities** (servers and network)

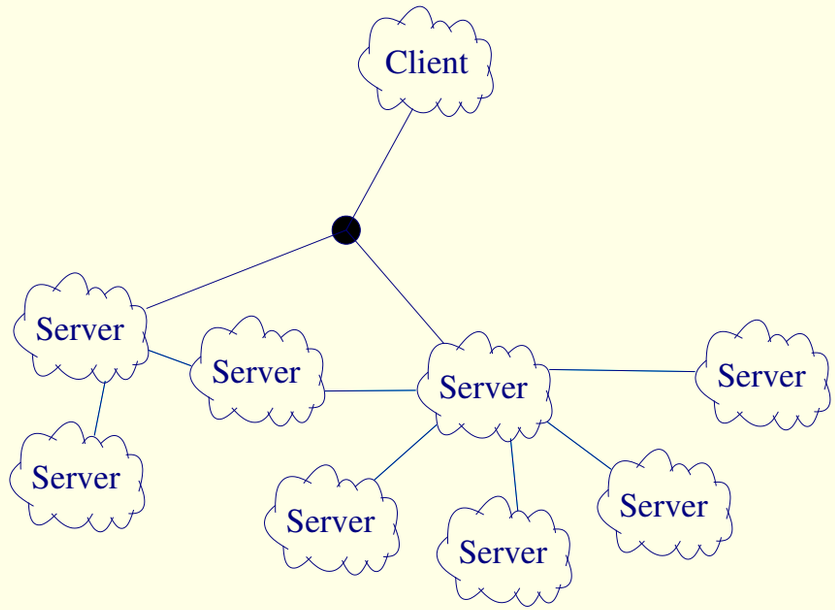
NWS + FAST

II. Qualitative knowledge of network **topology**

ENV$\longrightarrow$ ALNeM

Motivating example: how to configure NWS?

- Simplest: measure everything
- Better: hierarchical



Overview of this work

Our goal: provide the information needed by the scheduler.

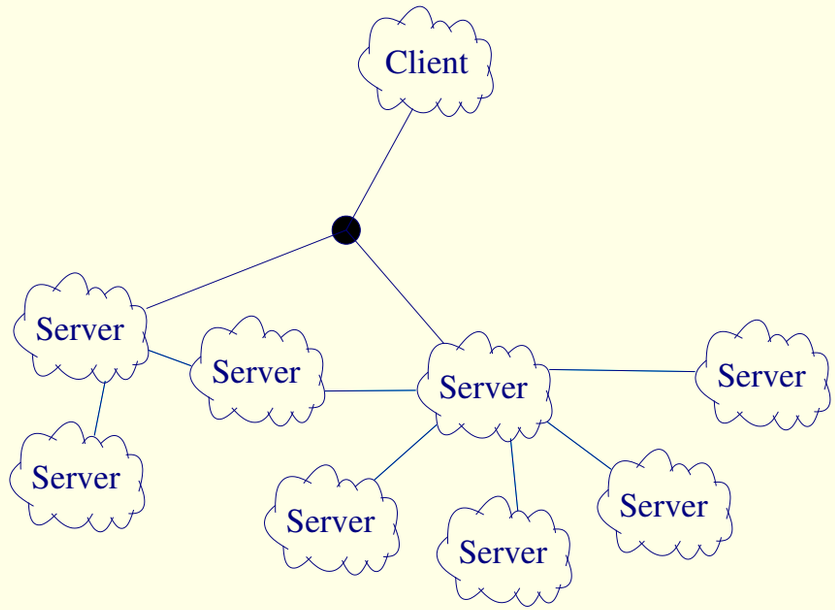
I. Quantitative knowledge of **needs** (tasks) and **availabilities** (servers and network)

NWS + FAST

II. Qualitative knowledge of network **topology**

ENV$\longrightarrow$ ALNeM

Motivating example: how to configure NWS?



- Simplest: measure everything
- Better: hierarchical

Target:

- logical topology (end-host)
- interferences

Overview of this work

Our goal: provide the information needed by the scheduler.

I. Quantitative knowledge of **needs** (tasks) and **availabilities** (servers and network)

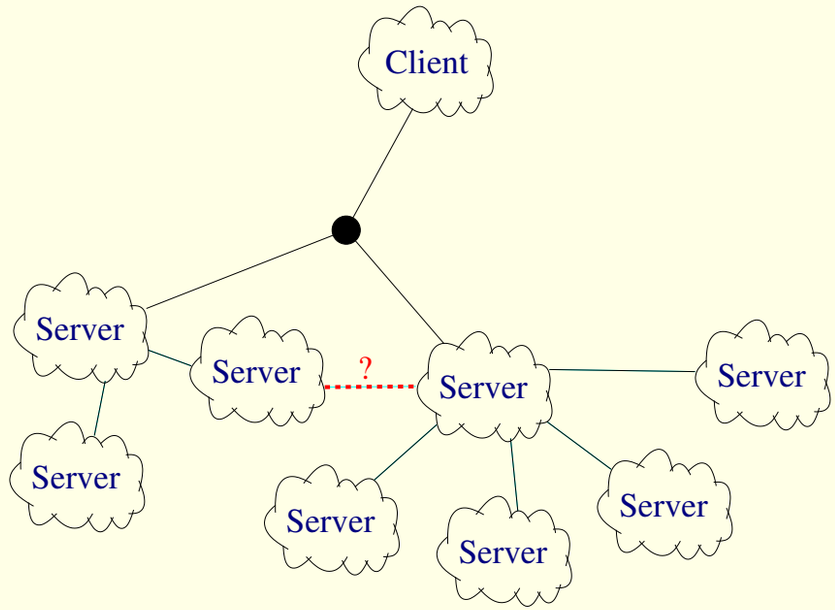
NWS + FAST

II. Qualitative knowledge of network **topology**

ENV$\longrightarrow$ **ALNeM**

ENV [SBW99]:

- ☺ maps the network without root access
- ☹ only hierarchical (tree)



Overview of this work

Our goal: provide the information needed by the scheduler.

I. Quantitative knowledge of **needs** (tasks) and **availabilities** (servers and network)

NWS + FAST

II. Qualitative knowledge of network **topology**

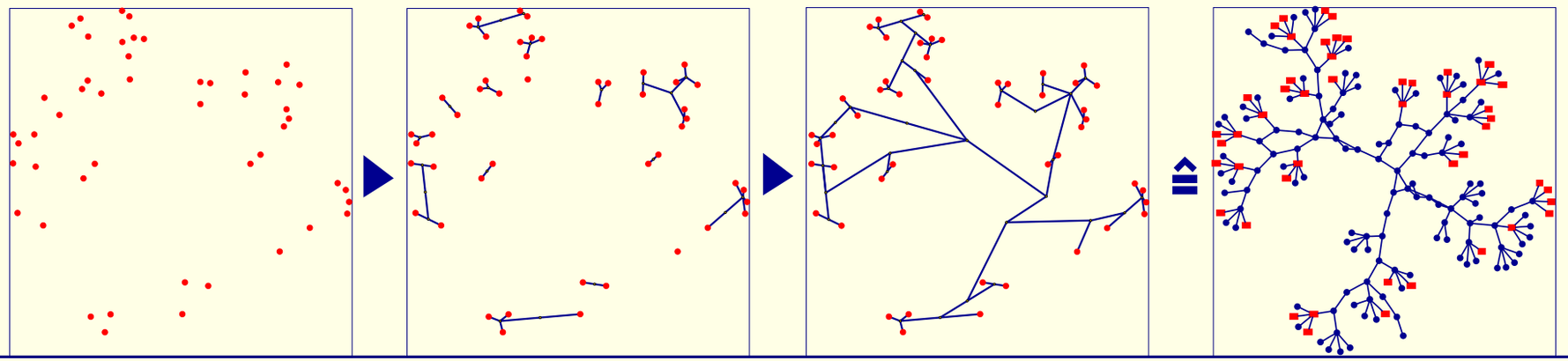
ENV $\longrightarrow$ **ALNeM**

ENV [SBW99]:

- ☺ maps the network without root access
- ☹ only hierarchical (tree)

ALNeM [LQ04]

- Same approach than ENV, generalized
- Stronger theoretical basements



Overview

- Introduction
- NWS: Network Weather Service
- FAST: Fast's Agent System Timer
- ALNeM: Application-Level Network Mapper
- Conclusion

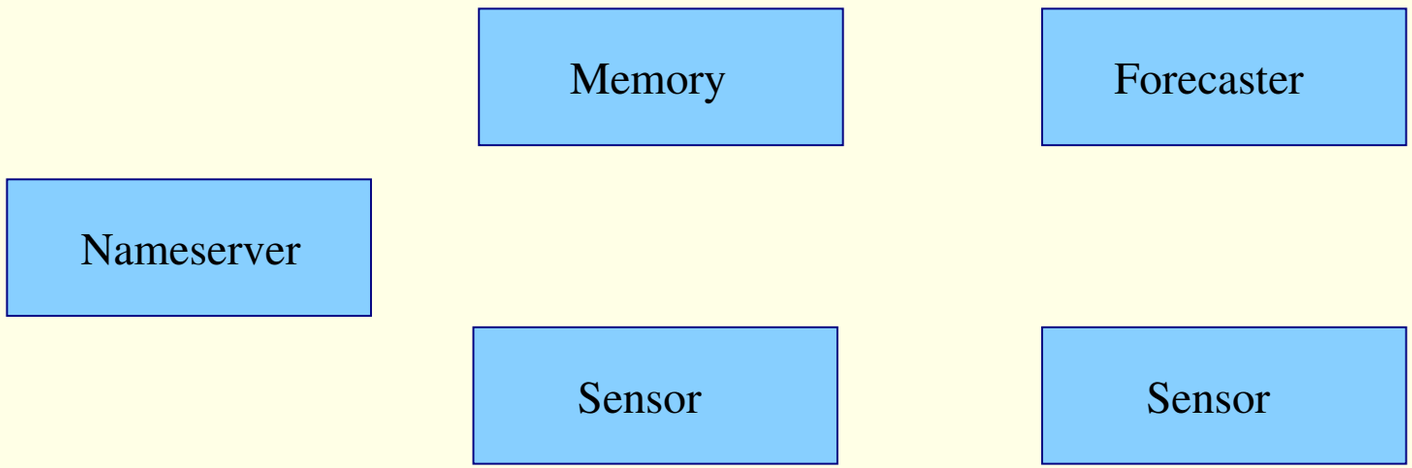
The Network Weather Service: presentation

Goal: (Grid) system availabilities measurement and forecasting

Leaded by Prof. Wolski (UCSB), used by AppLeS, Globus, NetSolve, Ninf, DIET, . . .

Architecture: Distributed system

Sensor: conducts the measurements
Memory: stores the results
Forecaster: forecasts statistically the tendencies
Name server: directory service like LDAP



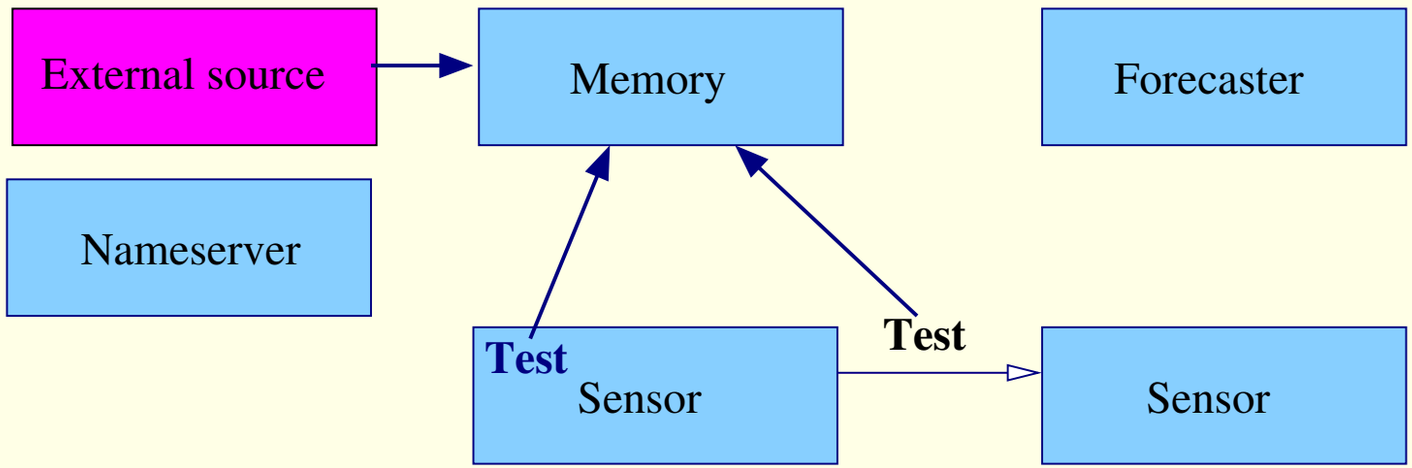
The Network Weather Service: presentation

Goal: (Grid) system availabilities measurement and forecasting

Leaded by Prof. Wolski (UCSB), used by AppLeS, Globus, NetSolve, Ninf, DIET, . . .

Architecture: Distributed system

Sensor: conducts the measurements
Memory: stores the results
Forecaster: forecasts statistically the tendencies
Name server: directory service like LDAP



Steady state: regular tests

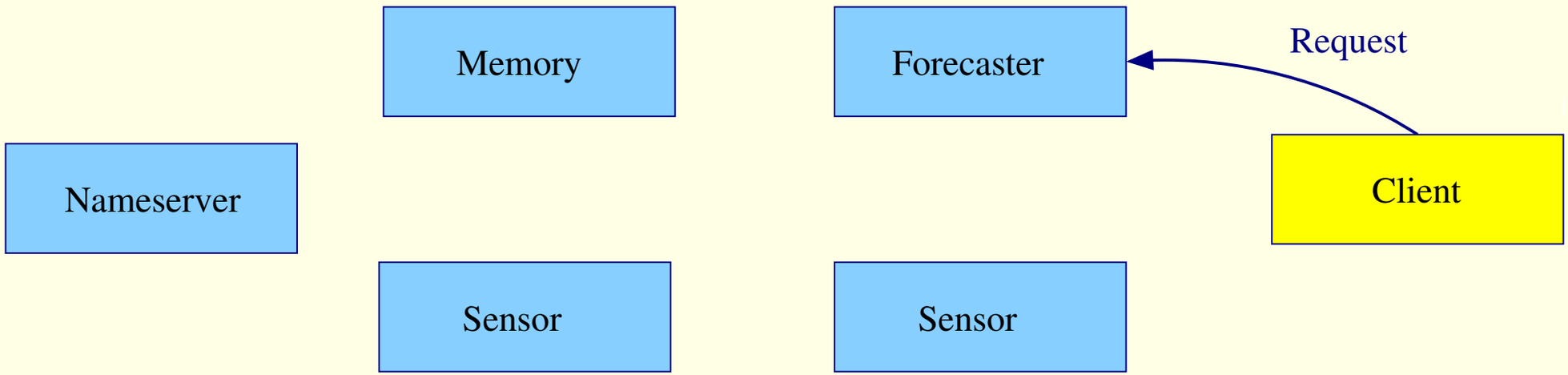
The Network Weather Service: presentation

Goal: (Grid) system availabilities measurement and forecasting

Leaded by Prof. Wolski (UCSB), used by AppLeS, Globus, NetSolve, Ninf, DIET, . . .

Architecture: Distributed system

Sensor: conducts the measurements
Memory: stores the results
Forecaster: forecasts statistically the tendencies
Name server: directory service like LDAP



Handling of a request

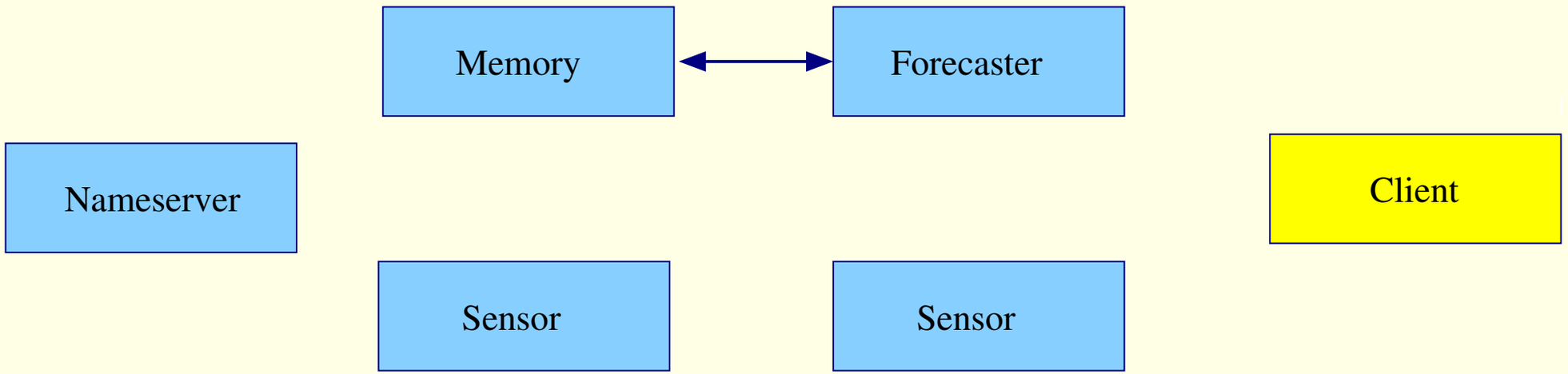
The Network Weather Service: presentation

Goal: (Grid) system availabilities measurement and forecasting

Leaded by Prof. Wolski (UCSB), used by AppLeS, Globus, NetSolve, Ninf, DIET, . . .

Architecture: Distributed system

Sensor: conducts the measurements
Memory: stores the results
Forecaster: forecasts statistically the tendencies
Name server: directory service like LDAP



Handling of a request

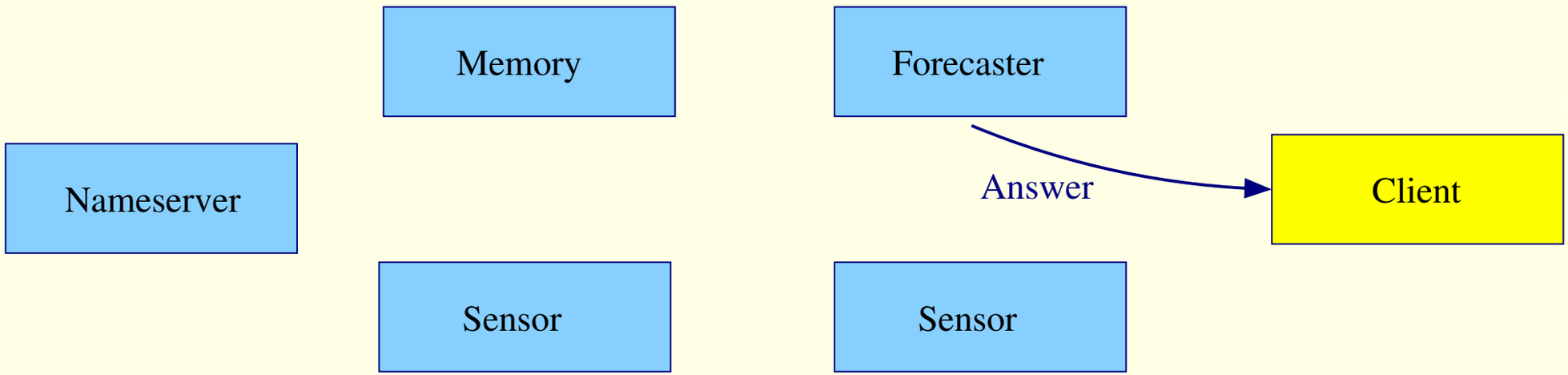
The Network Weather Service: presentation

Goal: (Grid) system availabilities measurement and forecasting

Leaded by Prof. Wolski (UCSB), used by AppLeS, Globus, NetSolve, Ninf, DIET, . . .

Architecture: Distributed system

Sensor: conducts the measurements
Memory: stores the results
Forecaster: forecasts statistically the tendencies
Name server: directory service like LDAP



Handling of a request

Measurements and Forecasting

- Provided metrics:

  availableCpu (for an incoming process), currentCpu (for existing processes), bandwidthTcp, latencyTcp (Default: 64Kb in 16Kb messages; buffer=32Kb), connectTimeTcp, freeDisk, freeMemory, ...

- Forecasting using statistics

  Data = serie: $D_1, D_2, \ldots, D_{n-1}, D_n$. We want D_{n+1}.
  Methods are applied on $D_1, D_2, \ldots, D_{n-1}$. each one predict D_n.
  Selection of the best on D_n to predict D_{n+1}.

  ## Used statistical methods

  mean: running, (adapting) sliding window ;
  median: idem ;
  gradian: $GRAD(t, g) = (1-g) \times GRAD(t-1, g) + g \times value(t)$;
  last value.

Conclusion about NWS

- ☺ Complete environment
- ☺ Designed for scheduling
- ☺ Statistical forecasting
- ☺ Widely used
- ☹ Uneasy to extend
- ☹ Sometimes difficult to deploy
- ☹ TCP only (myrinet-based?)

Related work

NetPerf: HP project to sort network components, no interactivity

GloPerf: Globus moves to NWS

PingER: Regular pings between 600 hosts in 72 countries

Iperf: Finds out the bandwidth by saturating the link for 30 seconds

RPS: Forecasting limited to the CPU load

Performance Co-Pilot (SGI):

- Same kind of architecture
- Low level data (/proc) $\Rightarrow$ not easily usable by a scheduler
- No forecasting

Overview

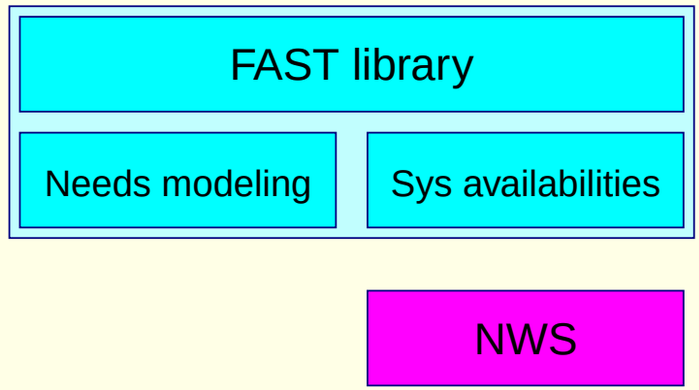
- Introduction
- NWS: Network Weather Service
- FAST: Fast's Agent System Timer
- ALNeM: Application-Level Network Mapper
- Conclusion

Fast Agent's System Timer: presentation

Goals:

- gather routine's performance on a given host at a given time
- interactivity, ease of use

Architecture:

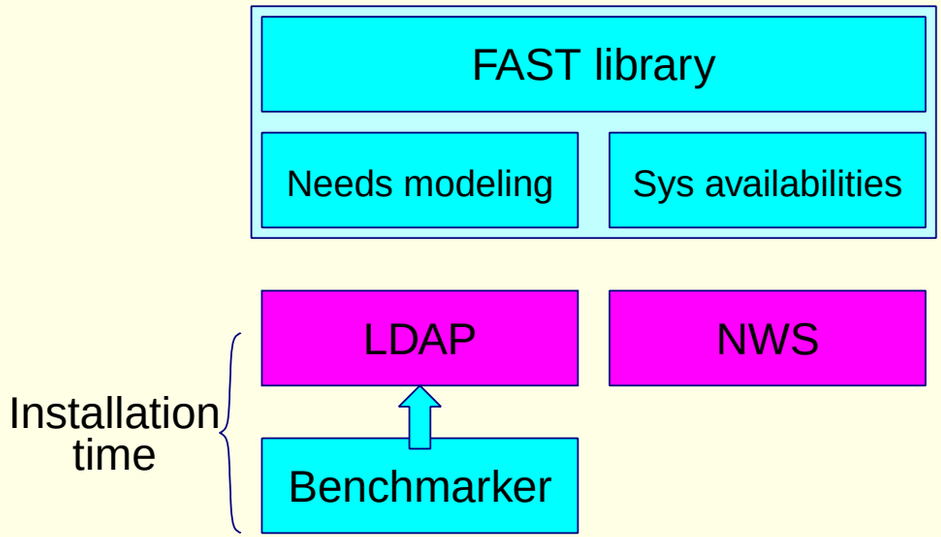


Fast Agent's System Timer: presentation

Goals:

- gather routine's performance on a given host at a given time
- interactivity, ease of use

Architecture:

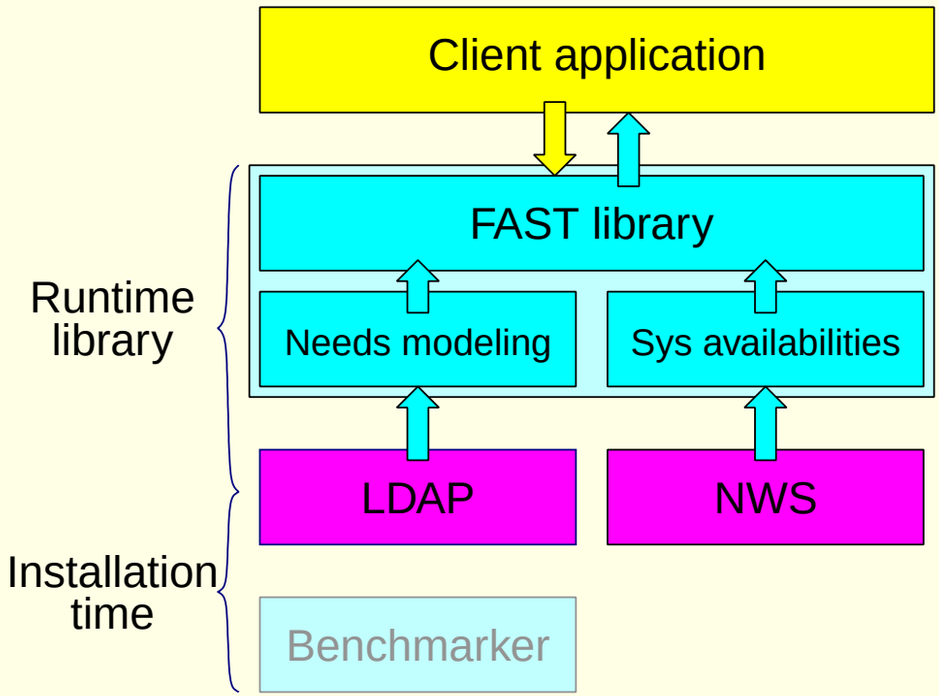


Fast Agent's System Timer: presentation

Goals:

- gather routine's performance on a given host at a given time
- interactivity, ease of use

Architecture:



Routines needs modeling

Related Work

- Elementary operation count: the myth of the constant Mflop/s
- Analytical model, micro-benchmarking: complex $\not\Rightarrow$ interactive, task description?
- Probability, Markov: how to instanciate it at a given time?

Routines needs modeling

Related Work

- Elementary operation count: the myth of the constant Mflop/s
- Analytical model, micro-benchmarking: complex $\not\Rightarrow$ interactive, task description?
- Probability, Markov: how to instanciate it at a given time?

FAST's approach

- Simple (sequential) routines like BLAS

  macro-benchmarking: benchmark {task; host} as a whole at installation
  - Getting the time: utime + stime to avoid backgroung load
  - Getting the space: step by step execution (like gdb) to track changes and search peak

    $\Rightarrow$ rather long, but only once

Routines needs modeling

Related Work

- Elementary operation count: the myth of the constant Mflop/s
- Analytical model, micro-benchmarking: complex $\not\Rightarrow$ interactive, task description?
- Probability, Markov: how to instanciate it at a given time?

FAST's approach

- Simple (sequential) routines like BLAS

  macro-benchmarking: benchmark {task; host} as a whole at installation
  - Getting the time: utime + stime to avoid backgroung load
  - Getting the space: step by step execution (like gdb) to track changes and search peak

    $\Rightarrow$ rather long, but only once
- Complex routines (ScaLAPACK)

  Structural decomposition by source analysis

Routines needs modeling

Related Work

- Elementary operation count: the myth of the constant Mflop/s
- Analytical model, micro-benchmarking: complex $\not\Rightarrow$ interactive, task description?
- Probability, Markov: how to instanciate it at a given time?

FAST's approach

- Simple (sequential) routines like BLAS

  macro-benchmarking: benchmark {task; host} as a whole at installation
  - Getting the time: utime + stime to avoid backgroung load
  - Getting the space: step by step execution (like gdb) to track changes and search peak

    $\Rightarrow$ rather long, but only once
- Complex routines (ScaLAPACK) **Freddy [CDQF03],**

  Structural decomposition by source analysis **integration underway**

Routines needs modeling

Related Work

- Elementary operation count: the myth of the constant Mflop/s
- Analytical model, micro-benchmarking: complex $\not\Rightarrow$ interactive, task description?
- Probability, Markov: how to instanciate it at a given time?

FAST's approach

- Simple (sequential) routines like BLAS

  macro-benchmarking: benchmark {task; host} as a whole at installation
  - Getting the time: utime + stime to avoid backgroung load
  - Getting the space: step by step execution (like gdb) to track changes and search peak

    $\Rightarrow$ rather long, but only once
- Complex routines (ScaLAPACK)

  Structural decomposition by source analysis

  Freddy [CDQF03], integration underway
- Irregular routines (sparse algebra)

  No forecasting $\Rightarrow$ selection of the fastest host

  Decomposition to extract simple parts

  Input of estimators from the application

Quality of the modeling

Time modeling

	dgeadd		dgemm		dtrsm	
	icluster	paraski	icluster	paraski	icluster	paraski
Maximal	0.02s	0.02s	0.21s	5.8s	0.13s	0.31s
error	6%	35%	0.3%	4%	10%	16%
Average	0.006s	0.007s	0.025s	0.03s	0.02s	0.08s
error	4%	6.5%	0.1%	0.1%	5%	7%

dgeadd: Matrix addition

dgemm: Matrix multiplication

dtrsm: Triangular resolution

icluster: bi-Pentium II, 256Mb, Linux, IMAG (Grenoble).

paraski: Pentium III, 256Mb, Linux, IRISA (Rennes).

network: Intra: LAN, 100Mb/s; Inter: VTHD network, 2.5Gb/s.

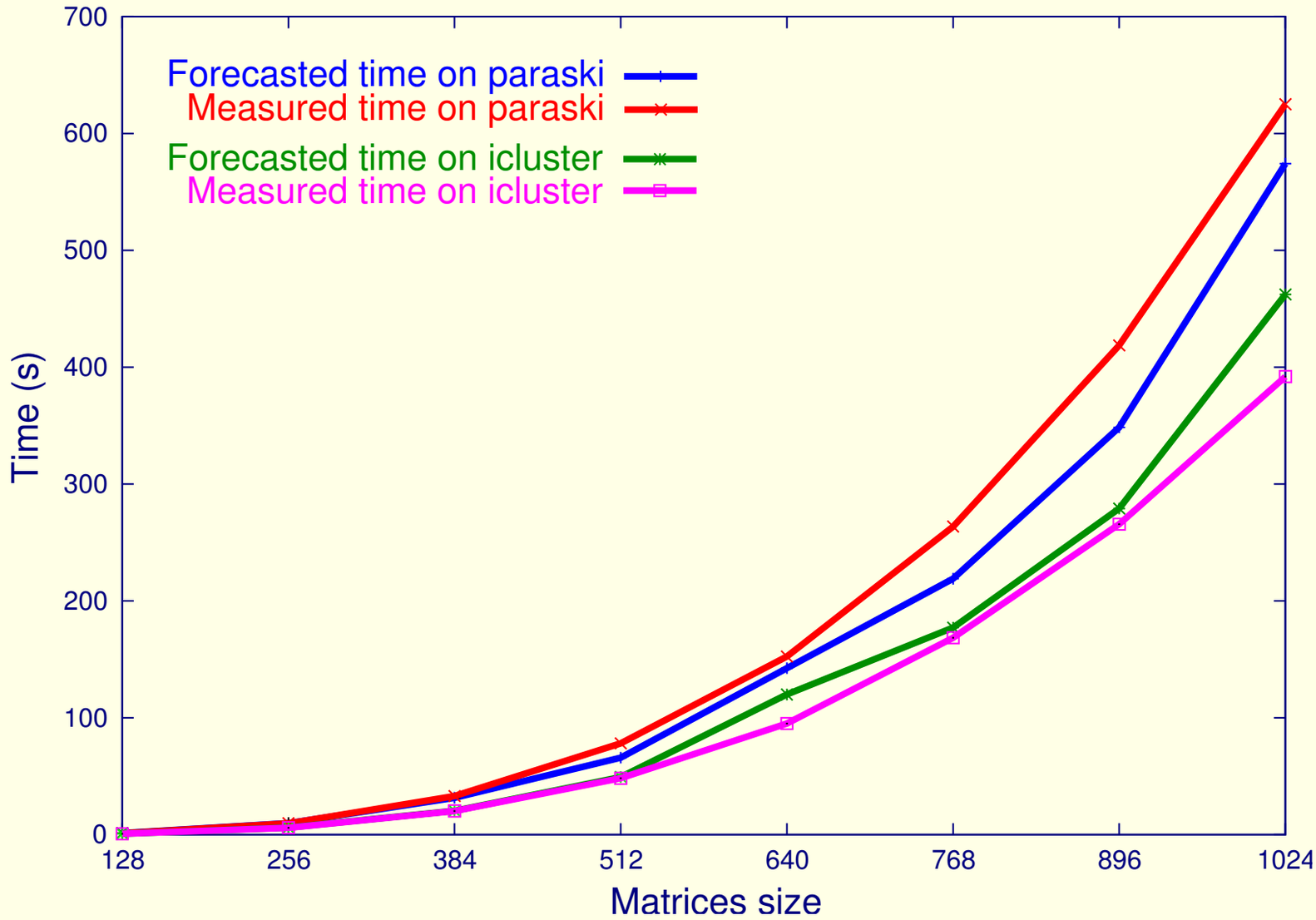
Space modeling

Almost perfect: Maximal error < 1% ; Average error $\approx$ 0.1%

Code size (constant) + Matrix size (polynomial)

Forecasting with background load

dgemm with background load (CPU-intensive process in background).



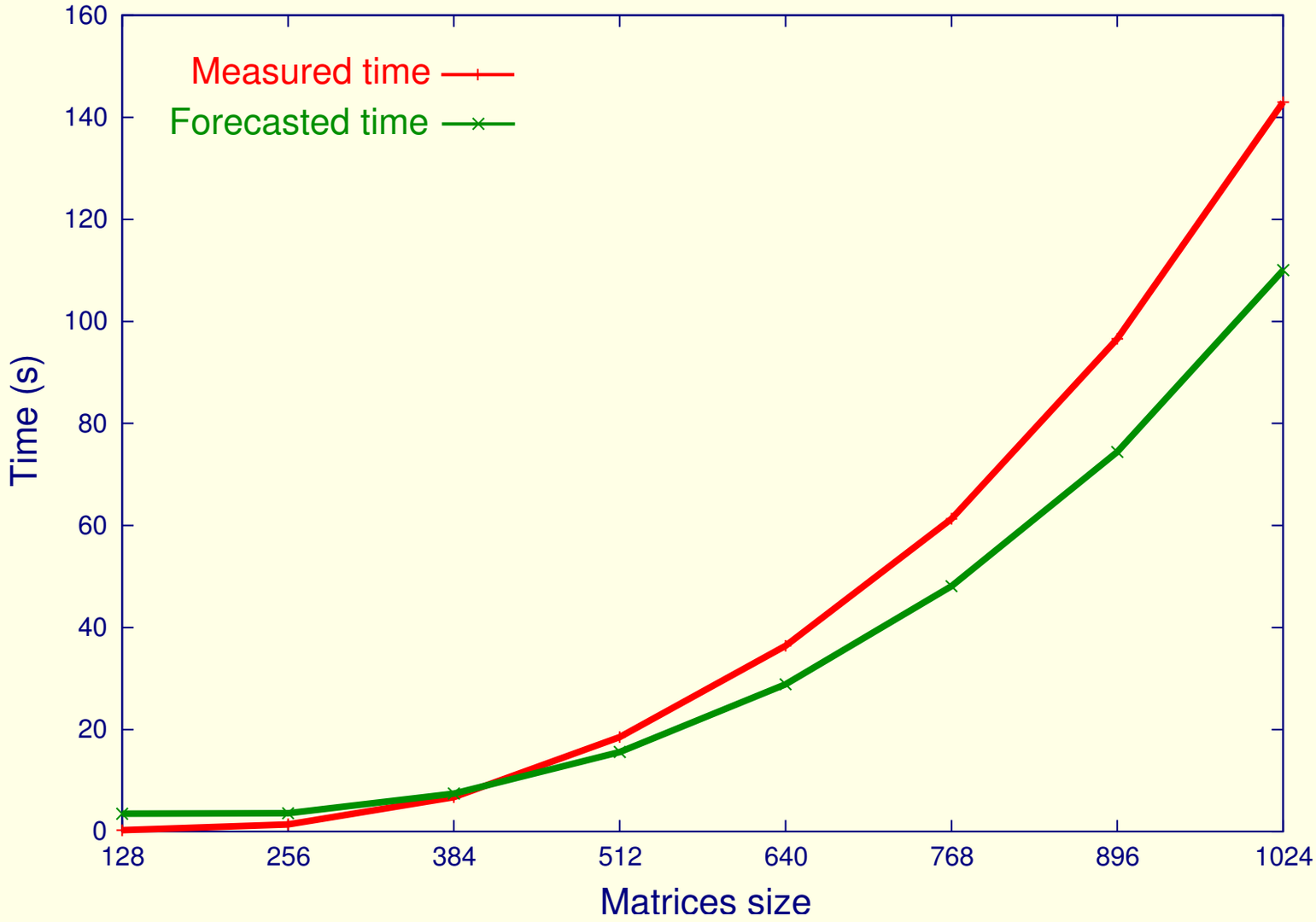
Maximal error: 22%

Average error$<$10%

Forecasting of sequence with background load

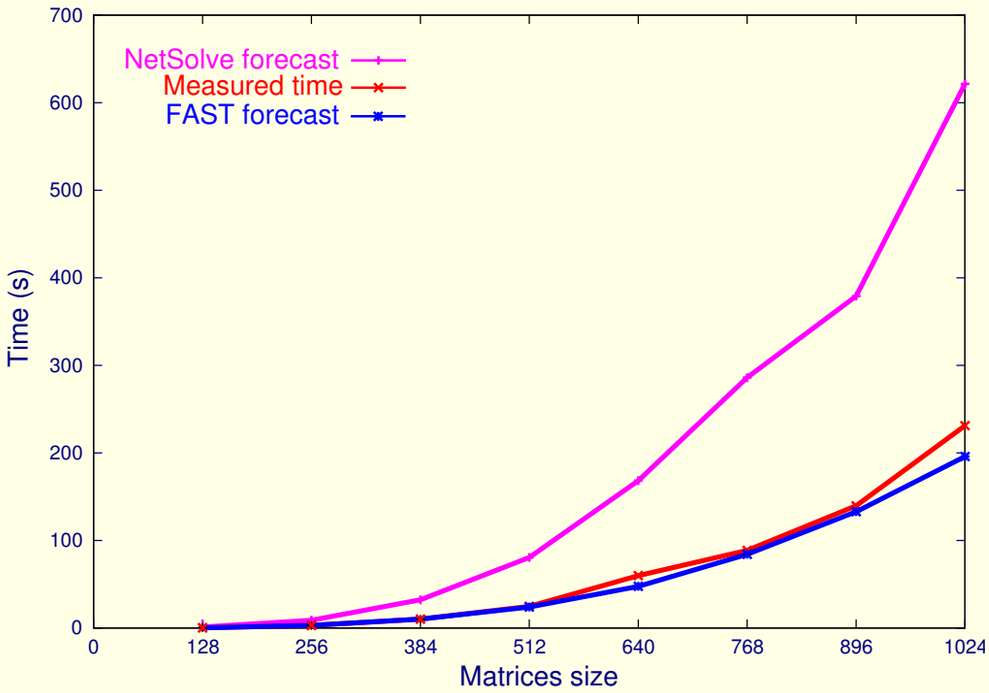
$$C = \begin{cases} C_r = A_r \times B_r - A_i \times B_i \\ C_i = A_r \times B_i + A_i \times B_r \end{cases}$$

client/servers over LAN

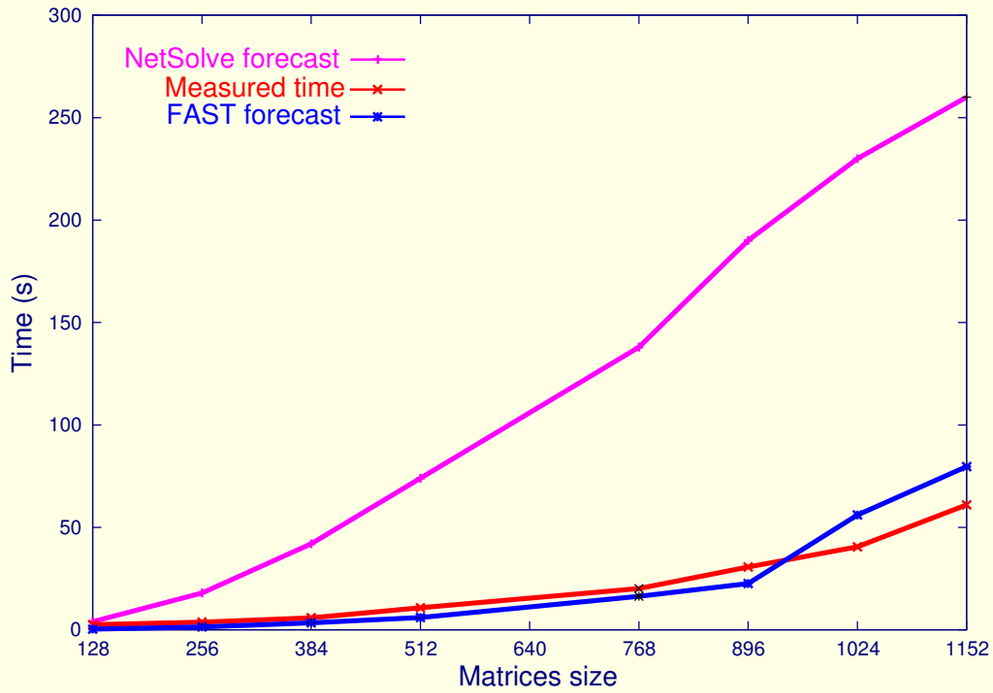


Maximal error: 25%; Average error: 13%

Comparison with NetSolve's forecaster

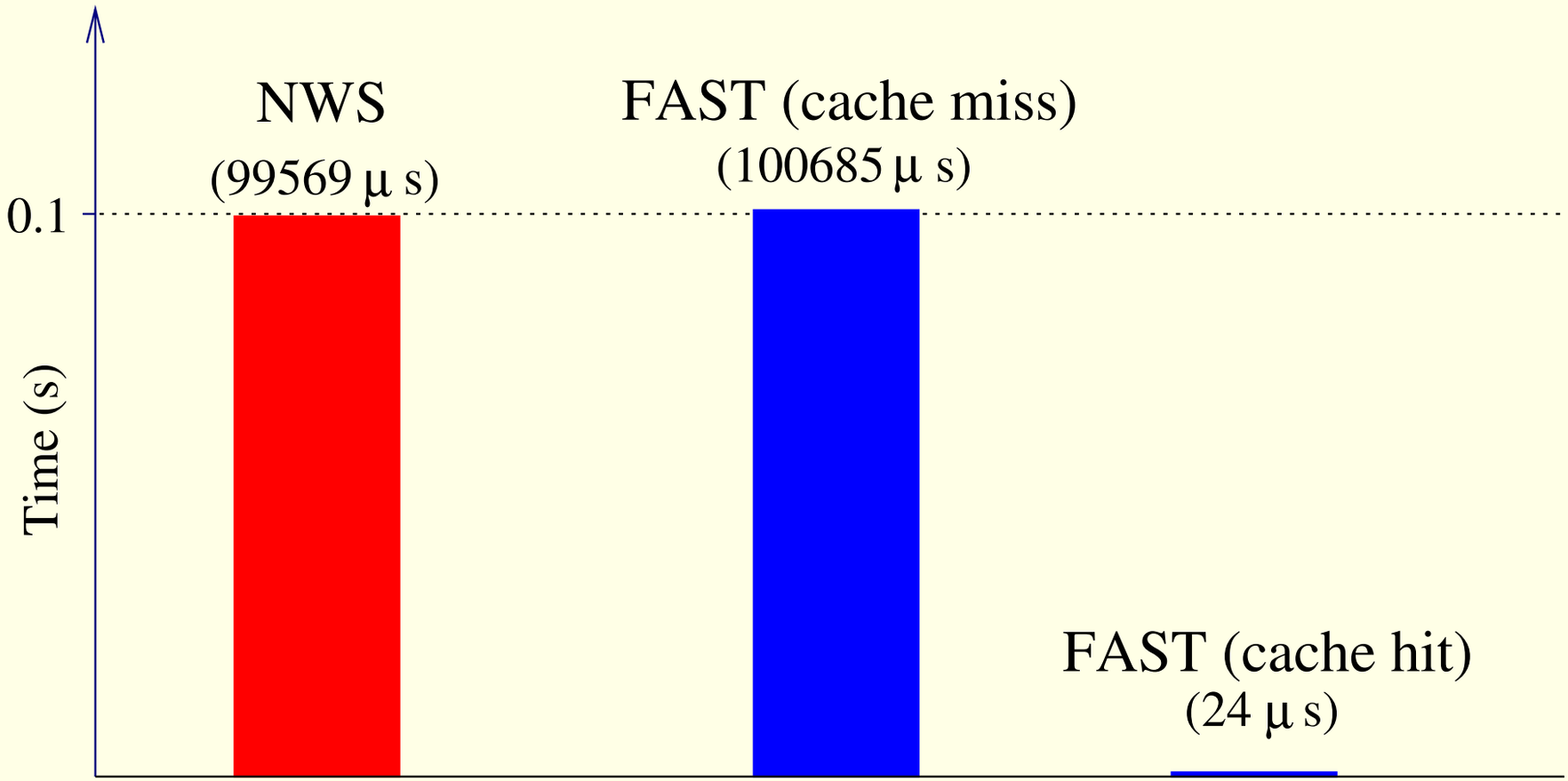


Computation time of dgemm.



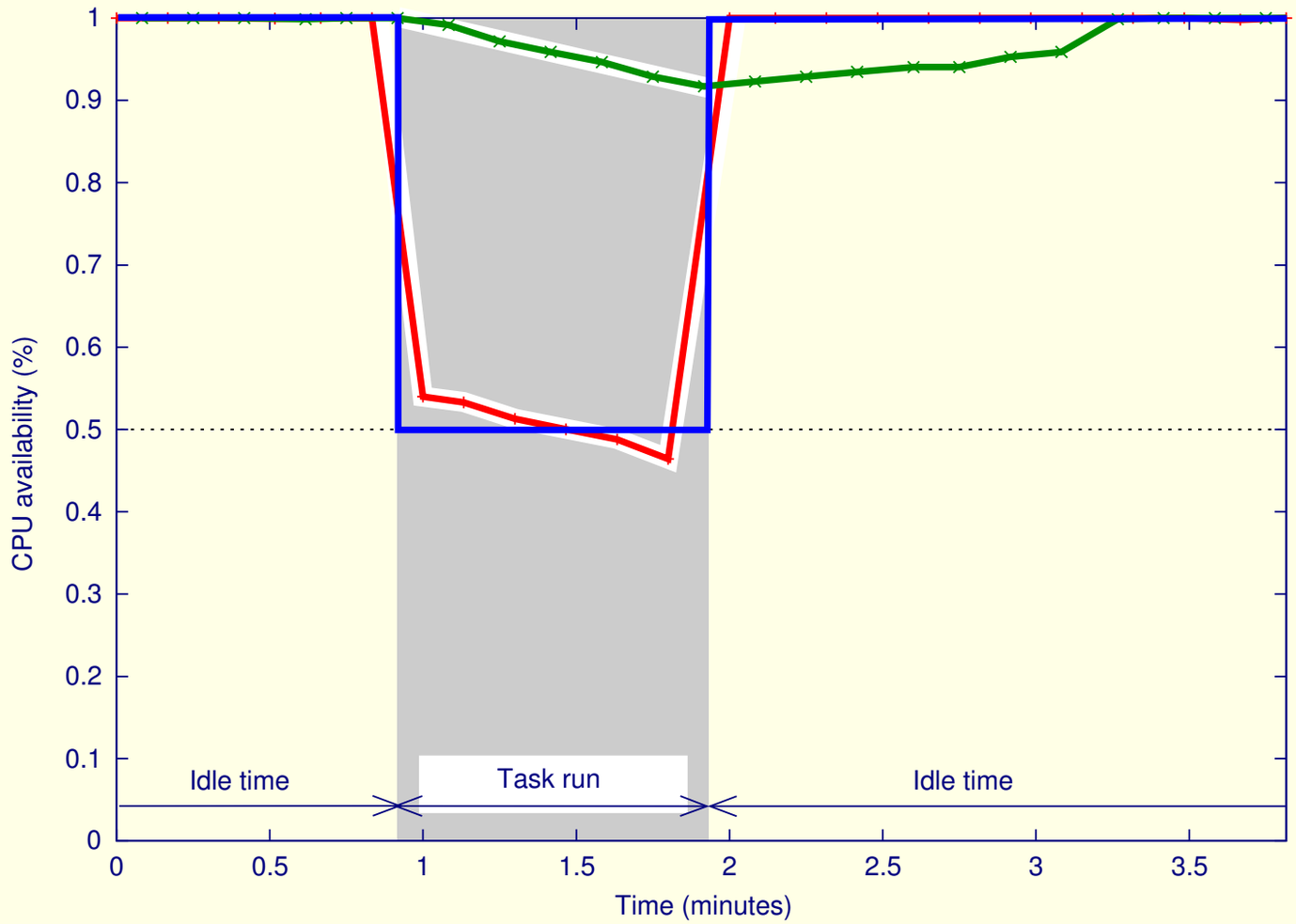
Communication time of dgemm.

Latency reduction



Responsiveness improvement

Scheduler / NWS collaboration



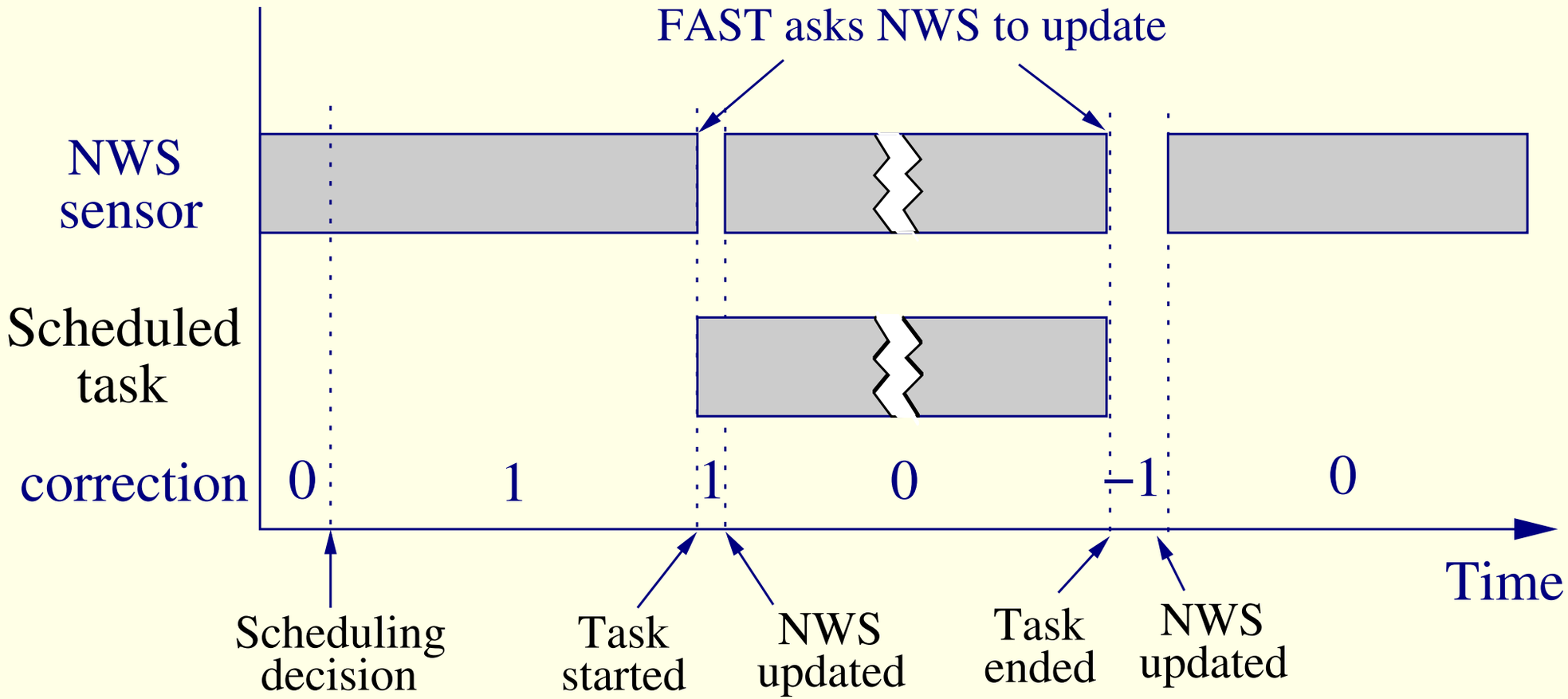
Forecasting

NWS: out of the box

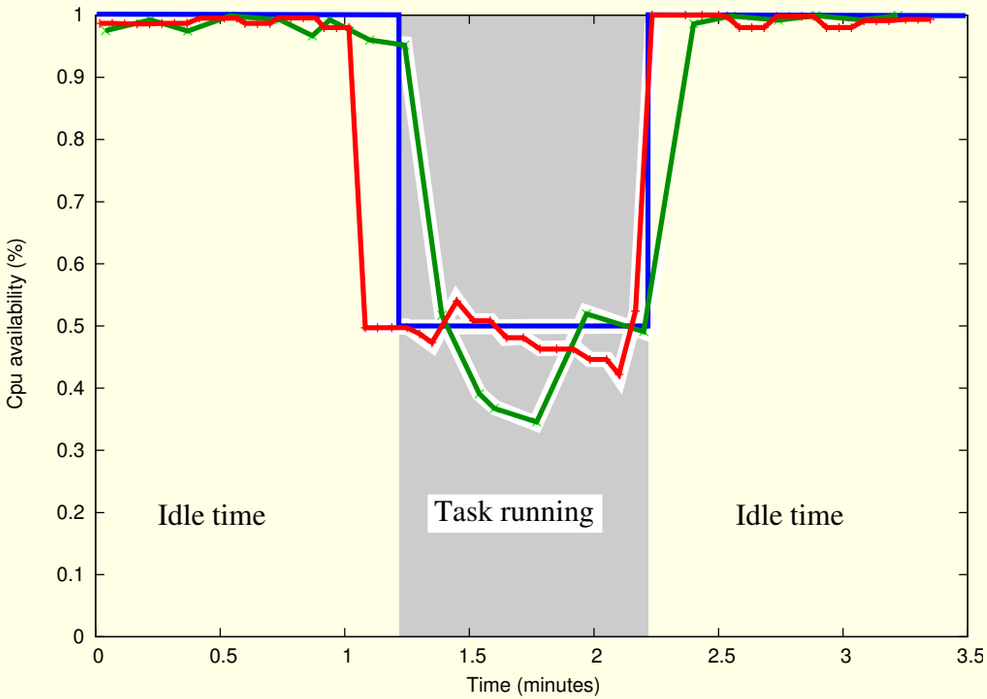
FAST: {sensors restart + forecaster reset} when the task starts or ends

Theoretical value

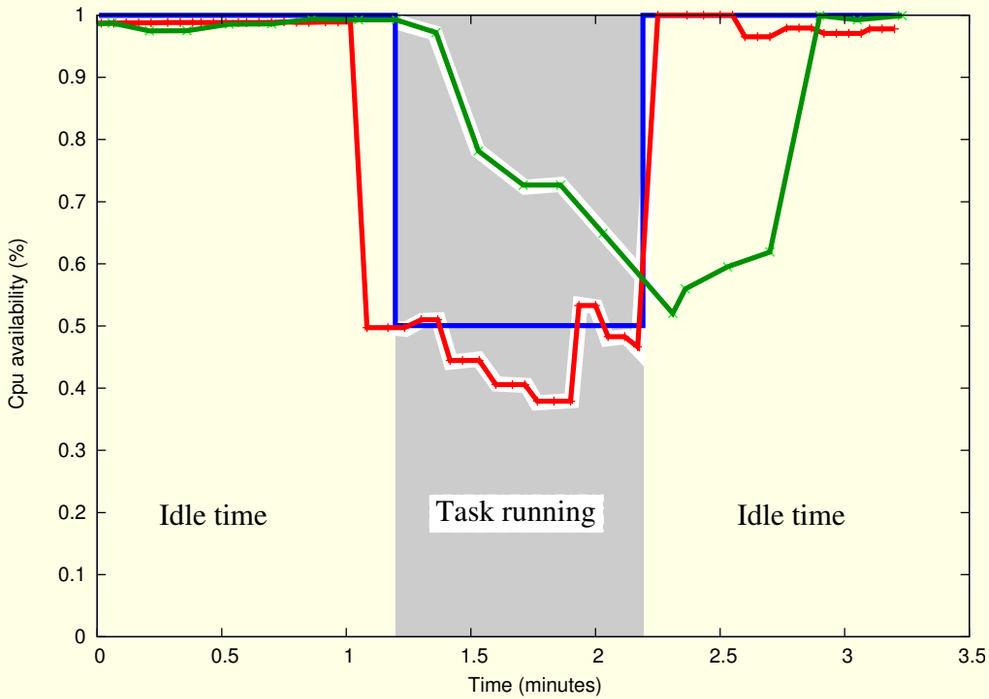
Virtual booking: How does it work?



Benefits of virtual booking



Measurements



Forecasting

NWS: ADAPT_CPU

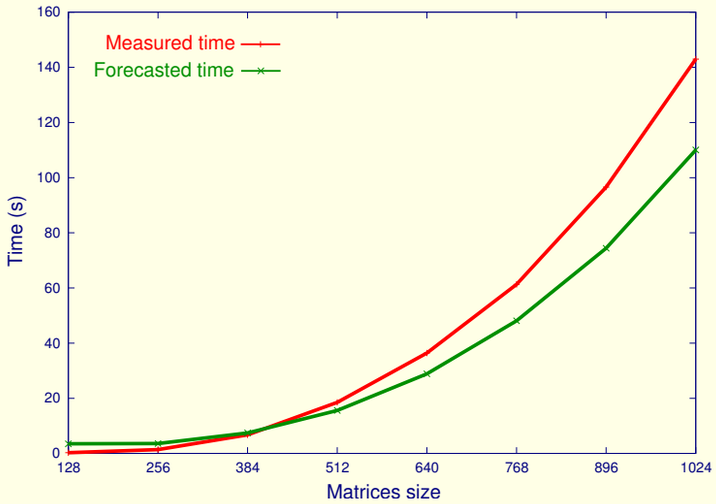
FAST: ADAPT_CPU + virtual booking + sensors restart + forecaster reset

Theoretical value

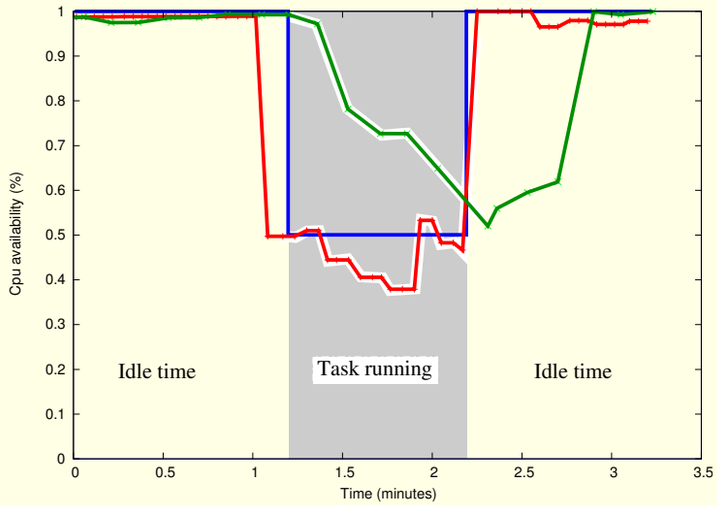
(Result of 4 different runs)

Contributions of FAST

Forecasting with load

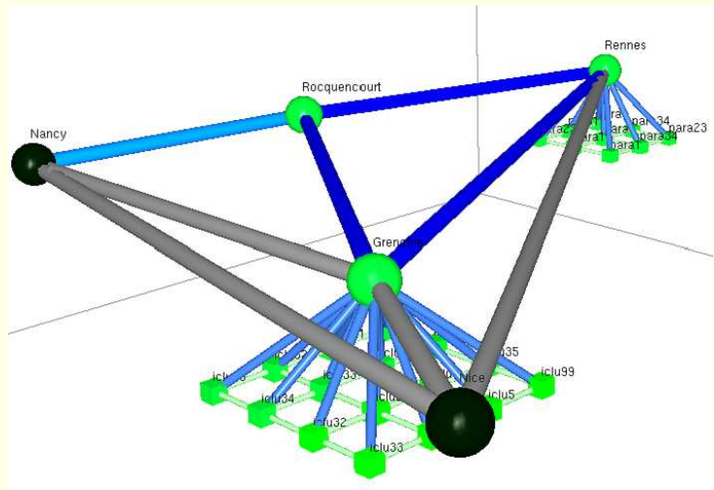


Responsiveness



Summary

- Generic benchmarking solution
- Simple interface to quantitative data
- Parallel routines handling currently integrated
- Integration: DIET, *NetSolve*, *Grid-TLSE*, cichlid
- 15 000 lines of C code, Linux, Solaris, True64
- 2 journals and 3 conferences/workshops



Overview

- Introduction
- NWS: Network Weather Service
- FAST: Fast's Agent System Timer
- ALNeM: Application-Level Network Mapper
- Conclusion

Application-Level Network Mapper

Goal: Mapping the network topology

Authors: Arnaud Legrand, Martin Quinson

Motivation: Server hosting, Simulation, Collective Communication Forecasting

Target application: NWS hosting

Problem: Network experiments must not collide (Clique concept)

Application-Level Network Mapper

Goal: Mapping the network topology

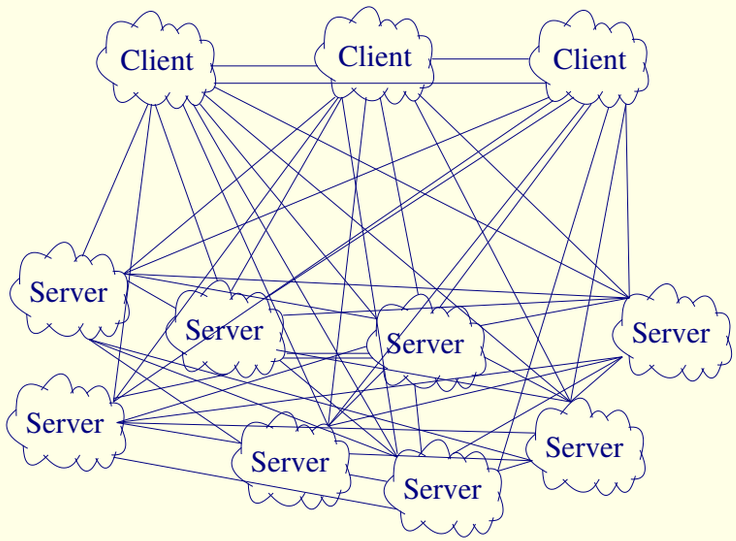
Authors: Arnaud Legrand, Martin Quinson

Motivation: Server hosting, Simulation, Collective Communication Forecasting

Target application: NWS hosting

Problem: Network experiments must not collide (Clique concept)

Simplest: One big clique



Application-Level Network Mapper

Goal: Mapping the network topology

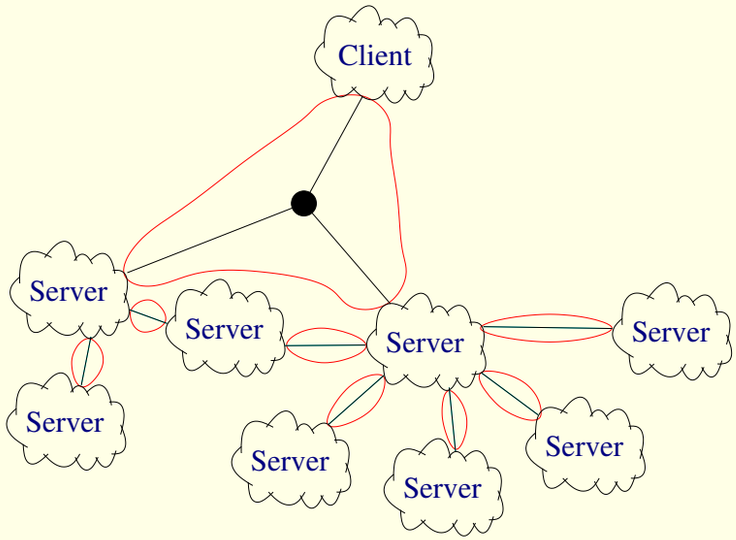
Authors: Arnaud Legrand, Martin Quinson

Motivation: Server hosting, Simulation, Collective Communication Forecasting

Target application: NWS hosting

Problem: Network experiments must not collide (Clique concept)

Simplest: One big clique ; **Better:** Hierarchical



Application-Level Network Mapper

Goal: Mapping the network topology

Authors: Arnaud Legrand, Martin Quinson

Motivation: Server hosting, Simulation, Collective Communication Forecasting

Focus: Discover interferences (limiting common links), not really packet paths

Application-Level Network Mapper

Goal: Mapping the network topology

Authors: Arnaud Legrand, Martin Quinson

Motivation: Server hosting, Simulation, Collective Communication Forecasting

Focus: Discover interferences (limiting common links), not really packet paths

Related work

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

ALNeM: Notations

Def (non-interference): $(ab) \mathbin{/\!\!/_{rl}} (cd) \iff \frac{bw_{/\!\!/ cd}(ab)}{bw(ab)} \approx 1$

Def (interference): $(ab) \mathbin{\curlyvee_{rl}} (cd) \iff \frac{bw_{/\!\!/ cd}(ab)}{bw(ab)} \approx 0.5$

Def: Interference matrix $I(V, \curlyvee_{rl})$

$$I(V, \curlyvee_{rl})(a,b,c,d) = \begin{cases} 1 & \text{if } (ab) \mathbin{\curlyvee_{rl}} (cd) \\ 0 & \text{if not} \end{cases}$$

INTERFERENCEGRAPH: Given $\mathcal{H}$ and $I(\mathcal{H}, \curlyvee_{rl})$,
Find a graph $G(V, E)$ and the associated routing satisfying:

$$\begin{cases} \mathcal{H} \subset V \\ I(\mathcal{H}, \curlyvee_G) = I(\mathcal{H}, \curlyvee_{rl}) \\ |V| \text{ is minimal.} \end{cases} .$$

ALNeM: Notations

Def (non-interference): $(ab) \mathbin{/\!\!/_{rl}} (cd) \iff \frac{bw_{/\!\!/ cd}(ab)}{bw(ab)} \approx 1$

Def (interference): $(ab) \mathbin{\curlyvee_{rl}} (cd) \iff \frac{bw_{/\!\!/ cd}(ab)}{bw(ab)} \approx 0.5$

Def: Interference matrix $I(V, \curlyvee_{rl})$

$$I(V, \curlyvee_{rl})(a,b,c,d) = \begin{cases} 1 & \text{if } (ab) \curlyvee_{rl} (cd) \\ 0 & \text{if not} \end{cases}$$

INTERFERENCEGRAPH: Given $\mathcal{H}$ and $I(\mathcal{H}, \curlyvee_{rl})$,
Find a graph $G(V,E)$ and the associated routing satisfying:

$$\begin{cases} \mathcal{H} \subset V \\ I(\mathcal{H}, \curlyvee_G) = I(\mathcal{H}, \curlyvee_{rl}) \\ |V| \text{ is minimal.} \end{cases} .$$

ALNeM: Notations

Def (non-interference): $(ab) \mathbin{/\!\!/}_{rl} (cd) \iff \frac{bw_{\mathbin{/\!\!/} cd}(ab)}{bw(ab)} \approx 1$

Def (interference): $(ab) \mathbin{\curlyvee}_{rl} (cd) \iff \frac{bw_{\mathbin{/\!\!/} cd}(ab)}{bw(ab)} \approx 0.5$

Def: Interference matrix $I(V, \curlyvee_{rl})$

$$I(V, \curlyvee_{rl})(a,b,c,d) = \begin{cases} 1 & \text{if } (ab) \mathbin{\curlyvee}_{rl} (cd) \\ 0 & \text{if not} \end{cases}$$

INTERFERENCEGRAPH: Given $\mathcal{H}$ and $I(\mathcal{H}, \curlyvee_{rl})$,

Find a graph $G(V,E)$ and the associated routing satisfying:

$$\begin{cases} \mathcal{H} \subset V \\ I(\mathcal{H}, \curlyvee_G) = I(\mathcal{H}, \curlyvee_{rl}) \\ |V| \text{ is minimal.} \end{cases} .$$

Mathematical tools

Def. (total interference): $\boldsymbol{a \perp b} \iff \forall (u,v) \in \mathcal{H},\ (au) \curlywedge_{rl} (bv)$

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

($\perp \iff \exists\ \rho$ separator)

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

Theorem (representativity): $\mathcal{C}$ equivalence class under $\perp$ (under some assumptions)

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

(you can interchange any member of the class by any other in the matrix)

Mathematical tools

Def. (total interference): $\boldsymbol{a \perp b} \iff \forall (u,v) \in \mathcal{H},\ (au) \between_{rl} (bv)$

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

($\perp \iff \exists\ \rho$ separator)

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

Theorem (representativity): $\mathcal{C}$ equivalence class under $\perp$ (under some assumptions)

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

(you can interchange any member of the class by any other in the matrix)

Mathematical tools

Def. (total interference): $\boldsymbol{a} \perp \boldsymbol{b} \iff \forall (u,v) \in \mathcal{H},\ (au) \between_{rl} (bv)$

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

($\perp \iff \exists\ \rho$ separator)

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

Theorem (representativity): $\mathcal{C}$ equivalence class under $\perp$ (under some assumptions)

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

(you can interchange any member of the class by any other in the matrix)

Mathematical tools

Def. (total interference): $\boldsymbol{a} \perp \boldsymbol{b} \iff \forall (u,v) \in \mathcal{H},\ (au) \curlywedge_{rl} (bv)$

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

($\perp \iff \exists\ \rho$ separator)

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

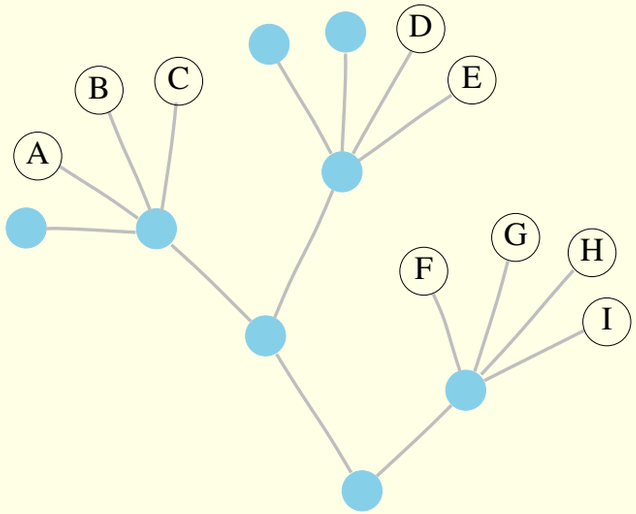
Theorem (representativity): $\mathcal{C}$ equivalence class under $\perp$ (under some assumptions)

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

(you can interchange any member of the class by any other in the matrix)

Algorithm for cliques of trees

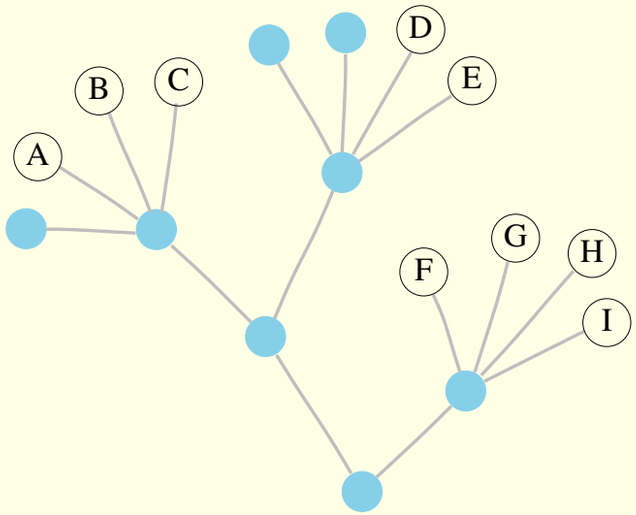
Equivalence class $\Rightarrow$ greedy algorithm *eating* the leaves

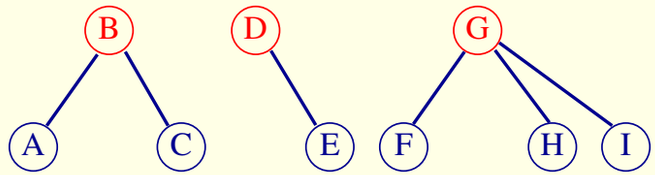


A B C D E F G H I

Algorithm for cliques of trees

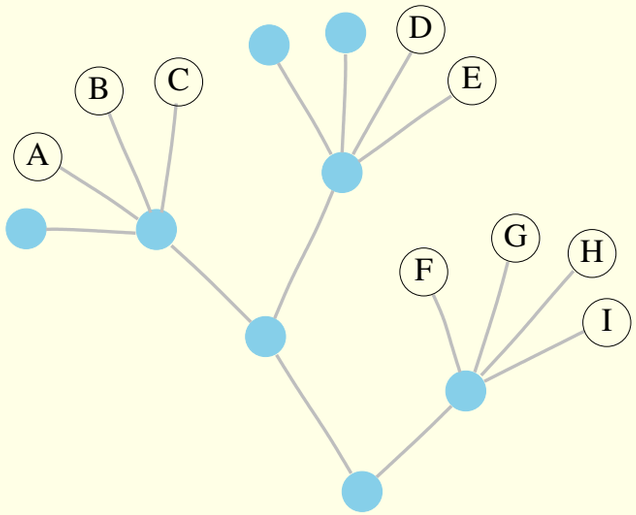
Equivalence class $\Rightarrow$ greedy algorithm *eating* the leaves

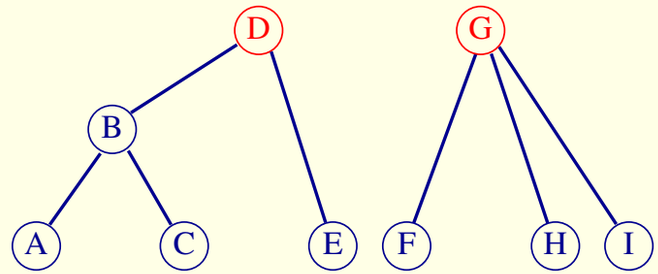




Algorithm for cliques of trees

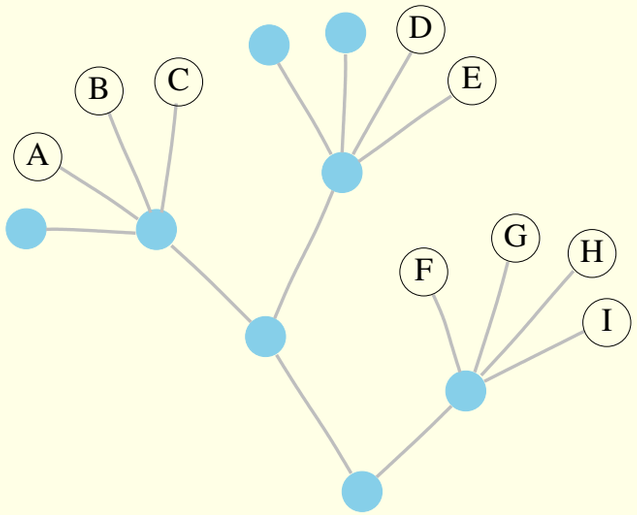
Equivalence class $\Rightarrow$ greedy algorithm *eating* the leaves

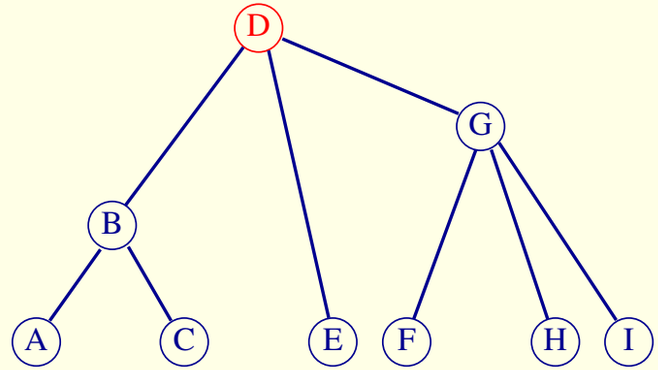




Algorithm for cliques of trees

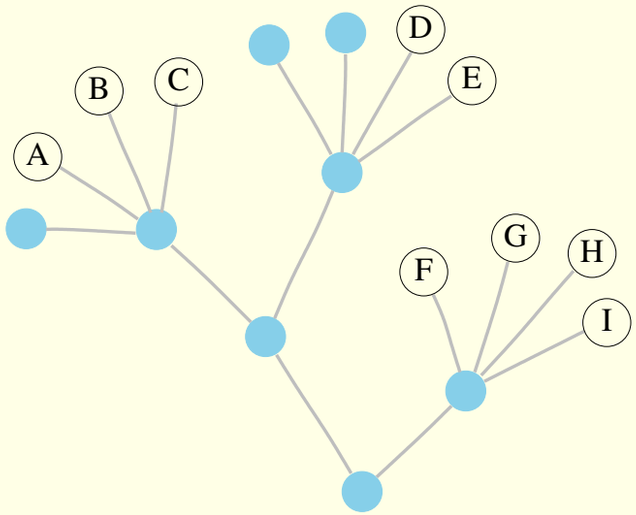
Equivalence class $\Rightarrow$ greedy algorithm *eating* the leaves

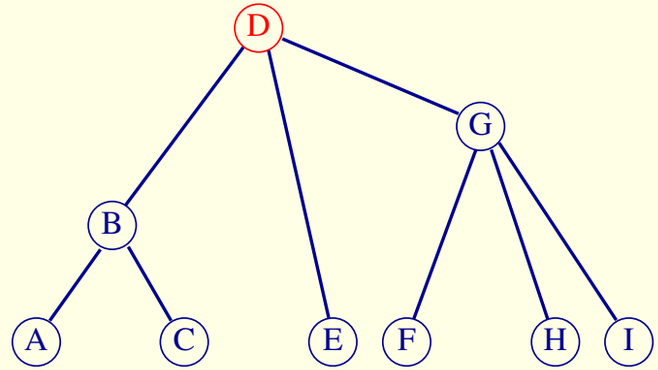




Algorithm for cliques of trees

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





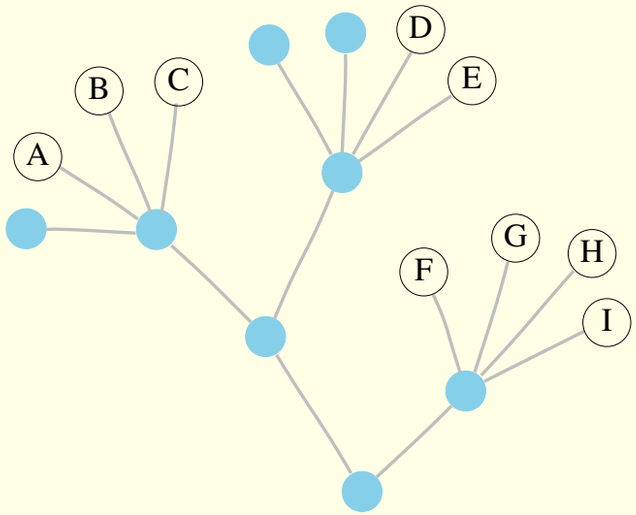
Theorem: When $|C_{\text{inf}}| = 1$, the graph built is a solution.

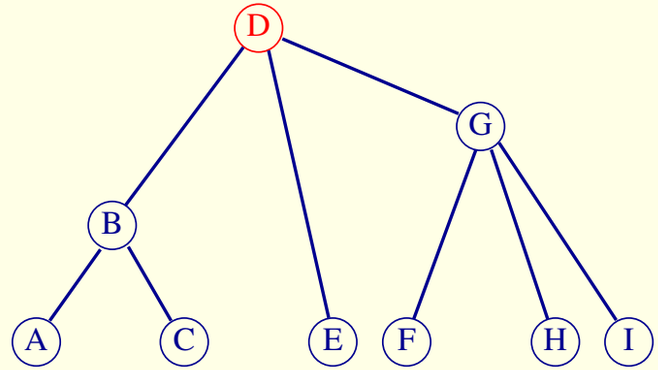
Theorem: If a tree being a solution exists, $|C_{\text{inf}}| = 1$.

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

Algorithm for cliques of trees

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





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

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

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

Theorem: When I contains no interferences, the clique of C_i is a valid solution.

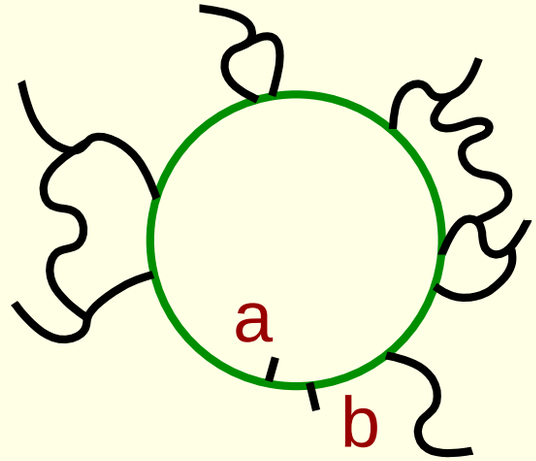
Remark: It is also optimal

Extension for cycles

Let a, b be the elements of C_i with the more interferences.

Lemma: no solution with $\exists z \in \mathcal{H}$ so that $z \in (a \rightarrow b)$

$\Rightarrow$ Cut between a and b!

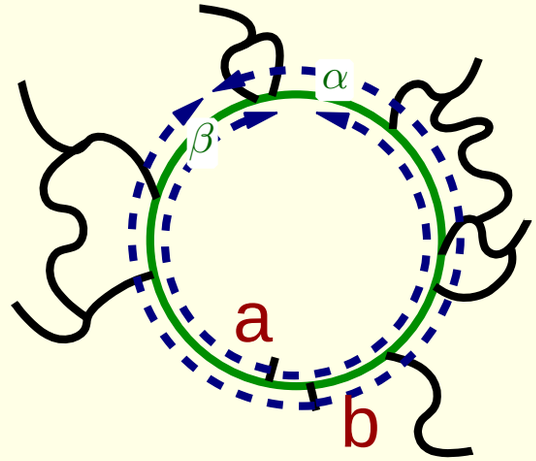


Extension for cycles

Let a, b be the elements of C_i with the more interferences.

Lemma: no solution with $\exists z \in \mathcal{H}$ so that $z \in (a \rightarrow b)$

$\Rightarrow$ Cut between a and b!



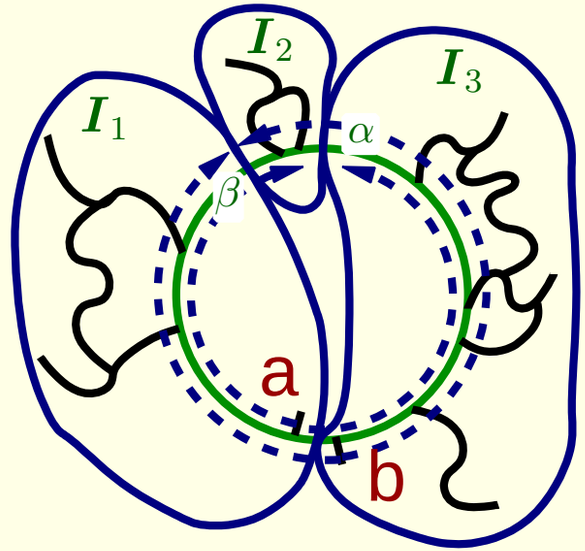
Finding out how to cut

Extension for cycles

Let a, b be the elements of C_i with the more interferences.

Lemma: no solution with $\exists z \in \mathcal{H}$ so that $z \in (a \rightarrow b)$

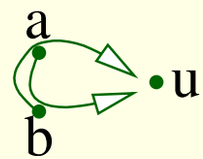
$\Rightarrow$ Cut between a and b!



Finding out how to cut

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

$$I_4 = \{a; b\}$$

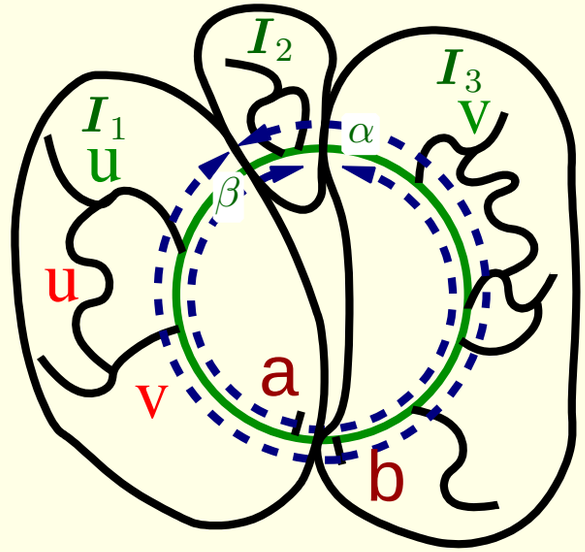
the contrary would imply 

Extension for cycles

Let a, b be the elements of C_i with the more interferences.

Lemma: no solution with $\exists z \in \mathcal{H}$ so that $z \in (a \rightarrow b)$

$\Rightarrow$ Cut between a and b!



Finding out how to cut

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

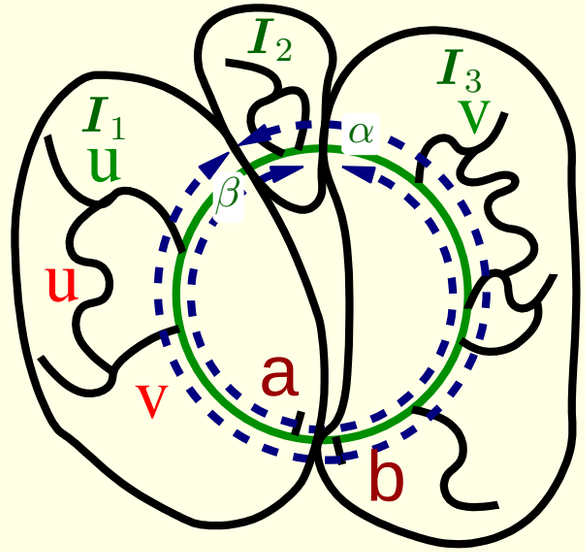
v \ u	a	α	β	b	
a	1	1	1		I_1
α	0	0\1	1		I_2
β	0	0	1		I_3
b					

Extension for cycles

Let a, b be the elements of C_i with the more interferences.

Lemma: no solution with $\exists z \in \mathcal{H}$ so that $z \in (a \rightarrow b)$

$\Rightarrow$ Cut between a and b!



Finding out how to cut

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

v\u	a	α	β	b	
a	1	1	1		I_1
α	0	0\1	1		I_2
β	0	0	1		I_3
b					

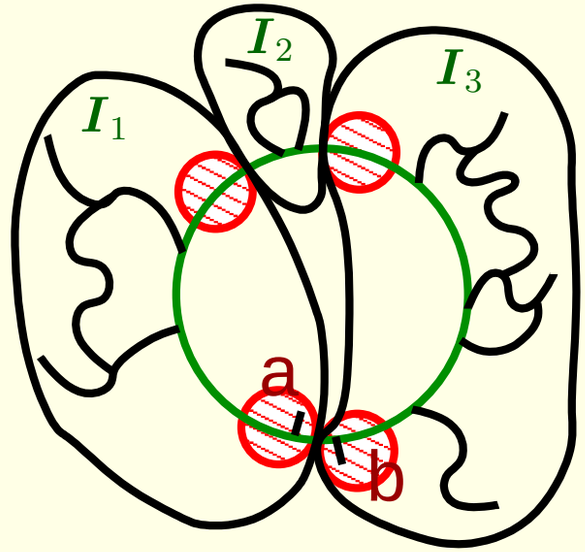
Topological sort on the graph associated to the matrix slice gives I_1, I_2, I_3

Extension for cycles

Let a, b be the elements of C_i with the more interferences.

Lemma: no solution with $\exists z \in \mathcal{H}$ so that $z \in (a \rightarrow b)$

$\Rightarrow$ Cut between a and b!



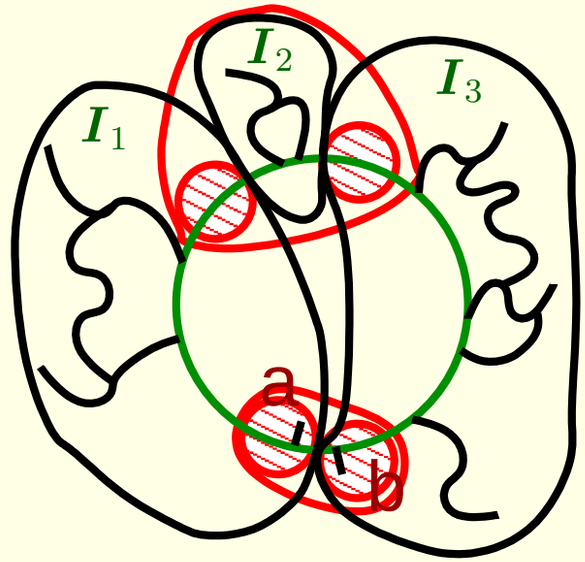
Finding out how to cut

How to connect parts afterward

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

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

Extension for cycles



Let a, b be the elements of C_i with the more interferences.

Lemma: no solution with $\exists z \in \mathcal{H}$ so that $z \in (a \rightarrow b)$

$\Rightarrow$ Cut between a and b!

Finding out how to cut

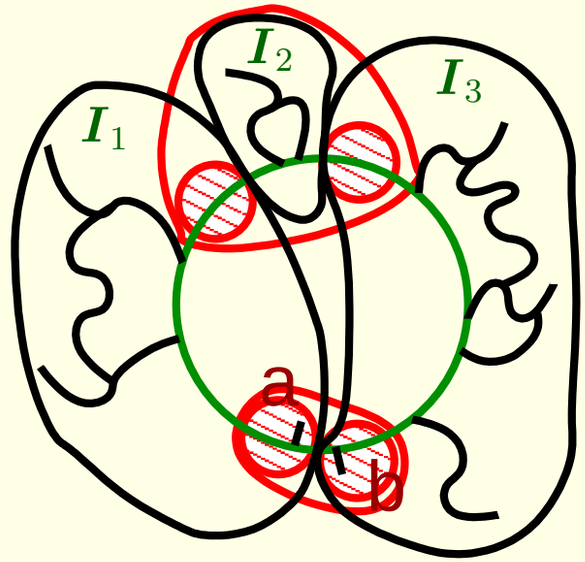
How to connect parts afterward

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

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

Reconnect I_{1_a} and I_{1_b} ; Reconnect I_{1_α} and I_{1_β}.

Extension for cycles



Let a, b be the elements of C_i with the more interferences.

Lemma: no solution with $\exists z \in \mathcal{H}$ so that $z \in (a \rightarrow b)$

$\Rightarrow$ Cut between a and b!

Finding out how to cut

How to connect parts afterward

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

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

Reconnect I_{1_a} and I_{1_b} ; Reconnect I_{1_α} and I_{1_β}.

No demonstration of this...

Data collection

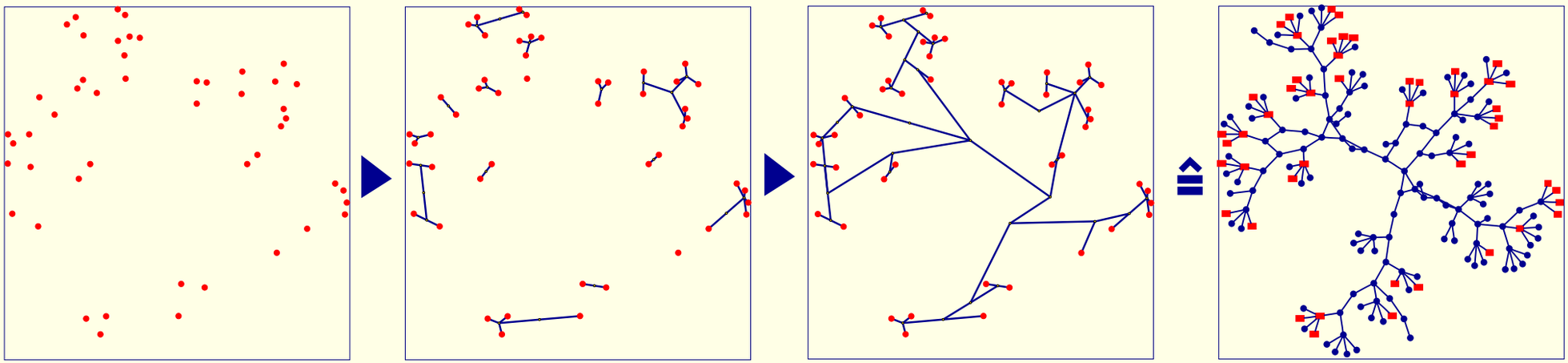
Interference measurement between each pair of hosts.

- Naïve algorithm:
  - N^4, 30s. per step $\Rightarrow$ 50 days for 20 hosts.
- Speedups thanks to traceroute or other tomography
  - Independent tests in parallel
  - Validation of information sets
- Refinement of existing graph?

Deserve more investigation

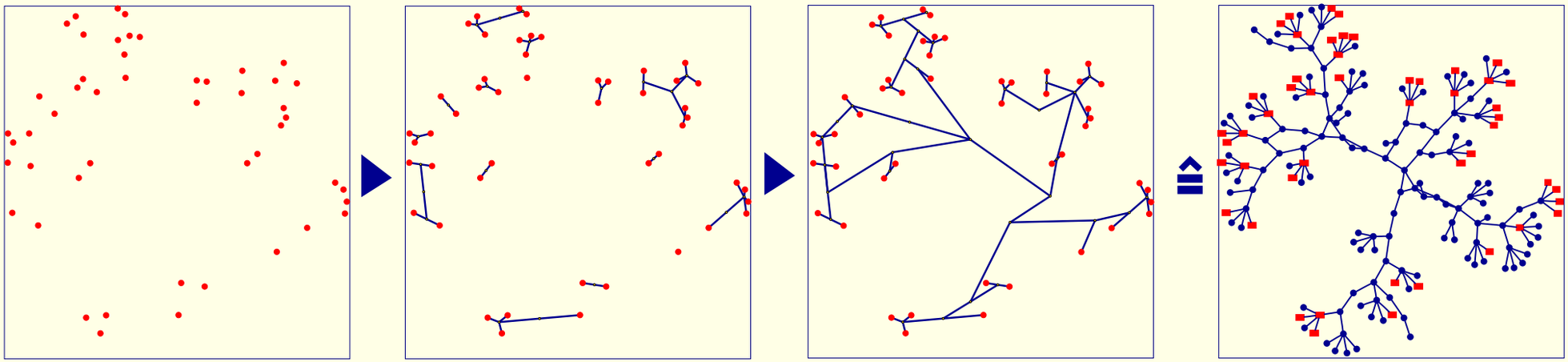
Contributions of ALNeM

- Retrieve the interference-based topology from direct measurements
- Strong mathemathical basements (optimal for cliques of trees)
- More generic than ENV (algorithm for cycles)
- 2 000 lines of C code; one research report
- Based on GRAS [Quinson03]



Contributions of ALNeM

- Retrieve the interference-based topology from direct measurements
- Strong mathemathical basements (optimal for cliques of trees)
- More generic than ENV (algorithm for cycles)
- 2 000 lines of C code; one research report
- Based on GRAS [Quinson03]
  - development on simulator (SimGrid [CLM03]) and immediate deployment
  - target: distributed event-based applications, C language
  - 10 000 lines of C code, Linux, Solaris
  - Submitted to one workshop



Overview

- Introduction
- NWS: Network Weather Service
- FAST: Fast's Agent System Timer
- ALNeM: Application-Level Network Mapper
- Conclusion

Conclusion

- Major issue on the Grid: collecting data (before scheduling)

Conclusion

- Major issue on the Grid: collecting data (before scheduling)
- Gathering quantitative data: **NWS** + **FAST**

  NWS: System availability

  FAST: Routine needs

Conclusion

- Major issue on the Grid: collecting data (before scheduling)
- Gathering quantitative data: **NWS** + **FAST**

NWS: System availability

Contributions:

- Lower latency
- Better responsiveness
- Process management

FAST: Routine needs

Future work:

- Automatic deployment

Conclusion

- Major issue on the Grid: collecting data (before scheduling)
- Gathering quantitative data: **NWS** + **FAST**

NWS: System availability

Contributions:

- Lower latency
- Better responsiveness
- Process management

Future work:

- Automatic deployment

FAST: Routine needs

Contributions:

- Generic benchmarking framework
- Unified interface to quantitative data
- Virtual booking
- Integration: DIET, NetSolve, Grid-TLSE
- 2 journals; 3 conferences/workshops

Future work:

- Integration of Freddy
- Irregular routines (sparse algebra)
- New metrics (like I/O)?
- Yet better integration within NWS

Conclusion

- Major issue on the Grid: collecting data (before scheduling)
- Gathering quantitative data: **NWS** + **FAST**
- Gathering qualitative data: **ALNeM**

ALNeM: Network topology to know about interferences

Contributions:

- Strong mathematical basements
- Optimal in size for cliques of trees
- Partial cycle handling
- GRAS: application development tool
- Submitted to one workshop

Future work:

- Proof of NP-hardness . . .
- . . . or exact algorithm
- Experimentation on real platform
- Optimization of the measurements
- Iterative algo. (modification detection)
- Integration within NWS
- Hosting of DIET

Selected publications

Book chapter: 1 national

- E. Caron, F. Desprez, E. Fleury, F. Lombard, J.-M. Nicod, M. Quinson, and F. Suter. Une approche hiérarchique des serveurs de calculs, in *Calcul réparti à grande échelle*. Hermès Science Paris, 2002. ISBN 2-7462-0472-X.

Journals: 2 internationals (+ 1 submitted), 1 national

- E. Caron, F. Desprez, M. Quinson, and F. Suter. Performance Evaluation of Linear Algebra Routines for Network Enabled Servers. *Parallel Computing, special issue on Cluters and Computational Grids for scientific computing*, 2003.
- F. Desprez, M. Quinson. Dynamic Performance Forecasting for Network-Enabled Servers in a Grid Environment. *Submitted to IEEE Transactions on Parallel and Distributed Systems*.

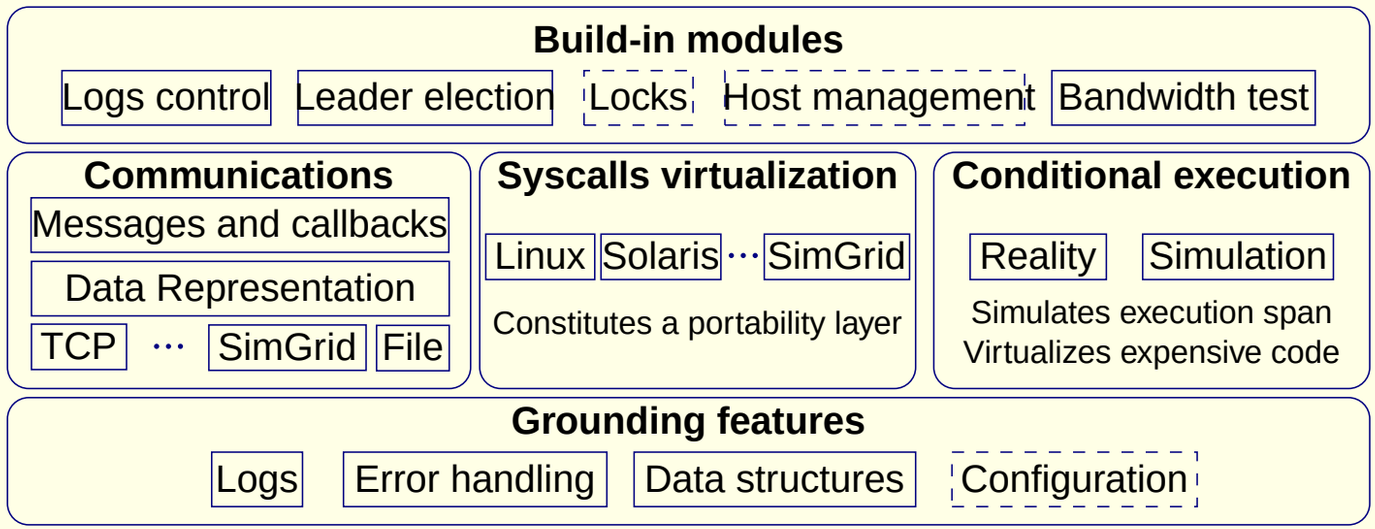
Conferences/workshops: 4 internationals (+ 2 submitted), 2 nationals.

- Ph. Combes, F. Lombard, M. Quinson, and F. Suter. A Scalable Approach to Network Enabled Servers. *Proceedings of the 7th Asian Computing Science Conference*. LNCS 2550:110–124, Springer-Verlag, Jan 2002.
- M. Quinson. Dynamic Performance Forecasting for Network-Enabled Servers in a Metacomputing Environment. *International Workshop on Performance Modeling, Evaluation, and Optimization of Parallel and Distributed Systems (PMEO-PDS'02)*, April 15-19 2002.
- A. Legrand, M. Quinson. Automatic deployment of the Network Weather Service using the Effective Network View. *Submitted to Workshop on Grid Benchmarking, associated to IPDPS'04*.
- O. Aumage, A. Legrand, M. Quinson. Reconciling the Grid Reality And Simulation. *Submitted to Parallel and Distributed Systems: Testing and Debugging, associated to IPDPS'04*.

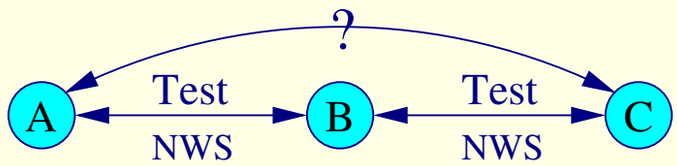
Appendix

GRAS overview

- development on simulator (SimGrid) and deployment without modification
- target: distributed event-based applications
- light virtual machine for the study and development of NWS, ALNeM, ...
- 10 000 lines of code, Linux, Solaris
- Futur: (even higher) performance and portability, interoperability



Sensor in the middle



$$bp(AC) = min\,(bp(AB); bp(BC))$$
$$lat(AC) = lat(AB) + lat(BC)$$

It's a must to reassemble measurements in hierarchical monitoring

RPC and grid computing: GridRPC

A simple idea: Implement the RPC model over the Grid

- Remote Procedure Call: run a computation remotely
- Good and simple paradigm to implement the Grid
- Some of the functionalities needed:
  - Computation scheduling, data migration
  - Security, fault-tolerance, interoperability, . . .
- 5 fundamental components:

  Client

  Server

  Agent

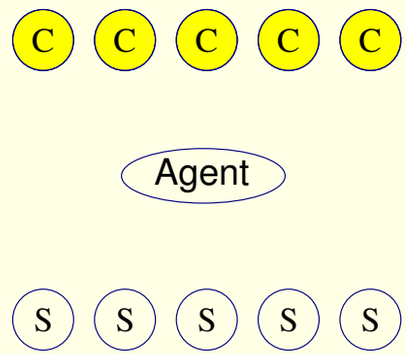
  Monitor

  Database

RPC and grid computing: GridRPC

A simple idea: Implement the RPC model over the Grid

- Remote Procedure Call: run a computation remotely
- Good and simple paradigm to implement the Grid
- Some of the functionalities needed:
  - Computation scheduling, data migration
  - Security, fault-tolerance, interoperability, . . .



- 5 fundamental components:

  Client Several user interfaces which submit the requests to servers

  Server

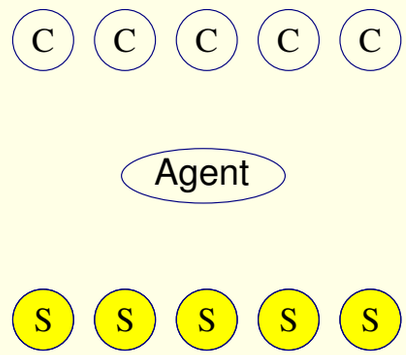
  Agent

  Monitor

  Database

RPC and grid computing: GridRPC

A simple idea: Implement the RPC model over the Grid



- Remote Procedure Call: run a computation remotely
- Good and simple paradigm to implement the Grid
- Some of the functionalities needed:
  - Computation scheduling, data migration
  - Security, fault-tolerance, interoperability, . . .
- 5 fundamental components:

  Client Several user interfaces which submit the requests to servers

  Server Runs software modules to solve client's requests

  Agent

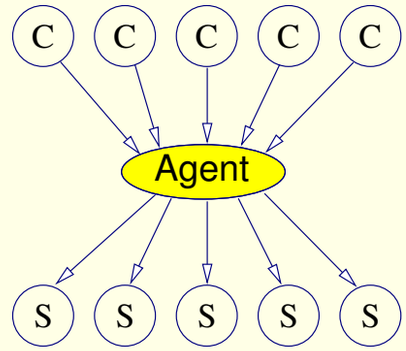
  Monitor

  Database

RPC and grid computing: GridRPC

A simple idea: Implement the RPC model over the Grid

- Remote Procedure Call: run a computation remotely
- Good and simple paradigm to implement the Grid
- Some of the functionalities needed:
  - Computation scheduling, data migration
  - Security, fault-tolerance, interoperability, . . .



- 5 fundamental components:

  Client Several user interfaces which submit the requests to servers

  Server Runs software modules to solve client's requests

  Agent Gets client's requests and schedules them onto the servers

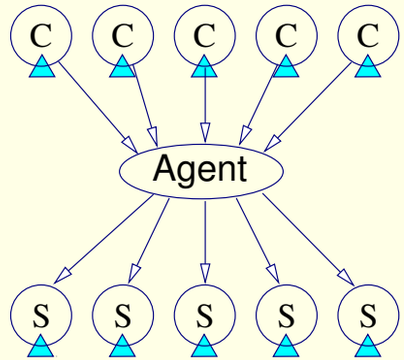
  Monitor

  Database

RPC and grid computing: GridRPC

A simple idea: Implement the RPC model over the Grid

- Remote Procedure Call: run a computation remotely
- Good and simple paradigm to implement the Grid
- Some of the functionalities needed:
  - Computation scheduling, data migration
  - Security, fault-tolerance, interoperability, . . .



- 5 fundamental components:

Client Several user interfaces which submit the requests to servers

Server Runs software modules to solve client's requests

Agent Gets client's requests and schedules them onto the servers

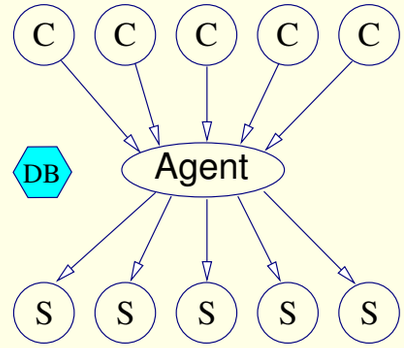
Monitor Monitors the current state of the resources

Database

RPC and grid computing: GridRPC

A simple idea: Implement the RPC model over the Grid

- Remote Procedure Call: run a computation remotely
- Good and simple paradigm to implement the Grid
- Some of the functionalities needed:
  - Computation scheduling, data migration
  - Security, fault-tolerance, interoperability, . . .



- 5 fundamental components:

Client Several user interfaces which submit the requests to servers

Server Runs software modules to solve client's requests

Agent Gets client's requests and schedules them onto the servers

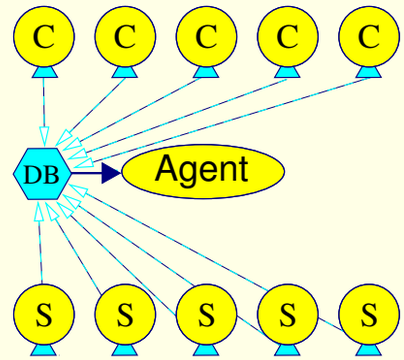
Monitor Monitors the current state of the resources

Database Contains static and dynamic knowledges about resources

RPC and grid computing: GridRPC

A simple idea: Implement the RPC model over the Grid

- Remote Procedure Call: run a computation remotely
- Good and simple paradigm to implement the Grid
- Some of the functionalities needed:
  - Computation scheduling, data migration
  - Security, fault-tolerance, interoperability, . . .



- 5 fundamental components:

Client Several user interfaces which submit the requests to servers

Server Runs software modules to solve client's requests

Agent Gets client's requests and schedules them onto the servers

Monitor Monitors the current state of the resources

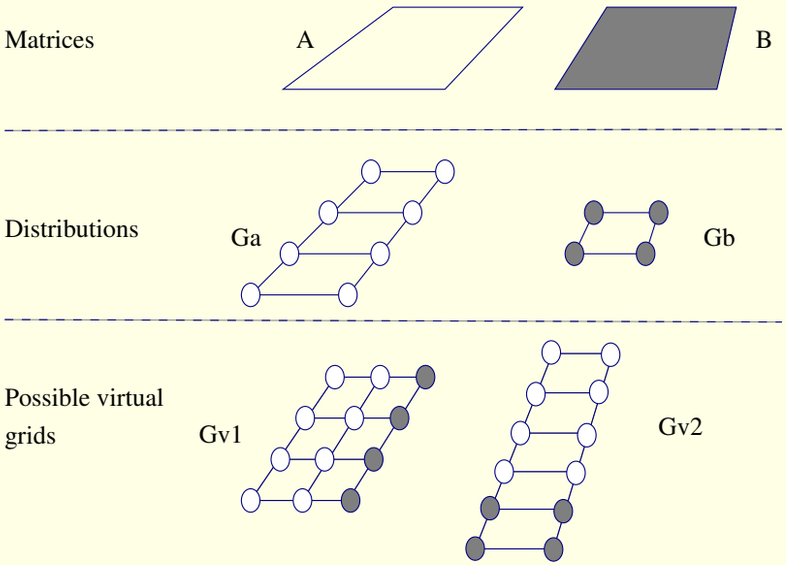
Database Contains static and dynamic knowledges about resources

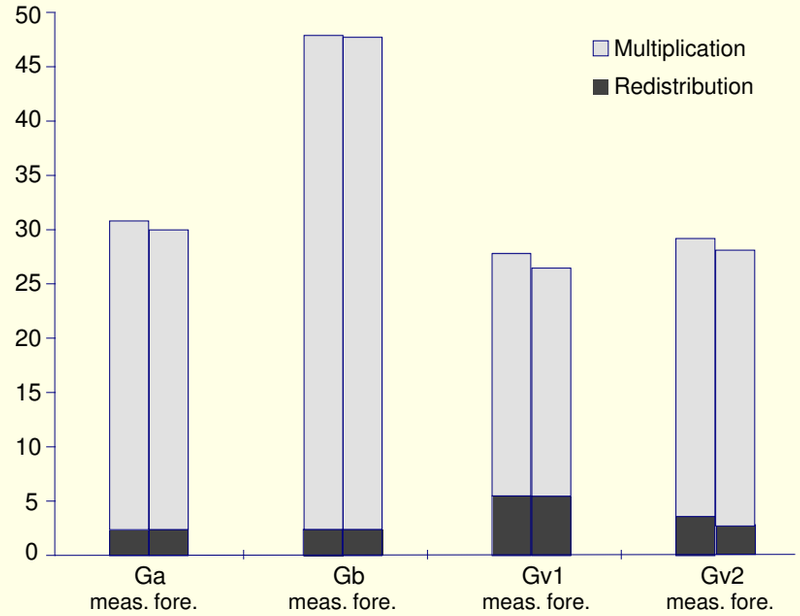
Knowing the platform is crucial for the agent

Freddy

Temps pdgemm(M, N, K) =

$$\left\lceil \frac{K}{R} \right\rceil \times \texttt{temps\_dgemm} + (M \times K)\tau_p^q + (K \times N)\tau_q^p + \left(\lambda_p^q + \lambda_q^p\right) \left\lceil \frac{K}{R} \right\rceil .$$





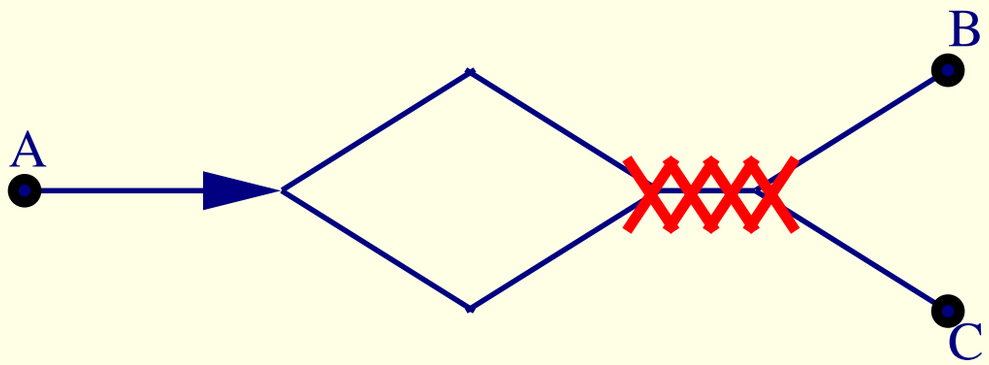
F. Suter. Parallélisme mixte et prédiction de performances sur réseaux hétérogènes de machines parallèles. *PhD thesis*, 2002.

E. Caron, F. Desprez, M. Quinson, and F. Suter. Performance Evaluation of Linear Algebra Routines for Network Enabled Servers. *Parallel Computing, special issue on Cluters and Computational Grids for scientific computing (CCGSC'02)*, 2003.

Hypothesis on the routing

Hypothesis 1: Routing consistent

- 1-to-N: no merge after branch
- N-to-1: no split after join



Hypothesis 2: Routing symmetric

Algorithm for cliques of trees

1. Initialization: $i \leftarrow 0$; $\mathcal{C}_i \leftarrow \mathcal{H}$; $E_i \leftarrow \emptyset$; $V_i \leftarrow \emptyset$
2. Classes lookup: $h_1, \ldots, h_p$: classes of $\perp$ over $\mathcal{C}_i$; $\forall i, l_i \in h_i$

   $\mathcal{C}_{i+1} \leftarrow \{l_1, \ldots, l_p\}$
3. Graph update: $V_{i+1} \leftarrow V_i$; $E_{i+1} \leftarrow E_i$

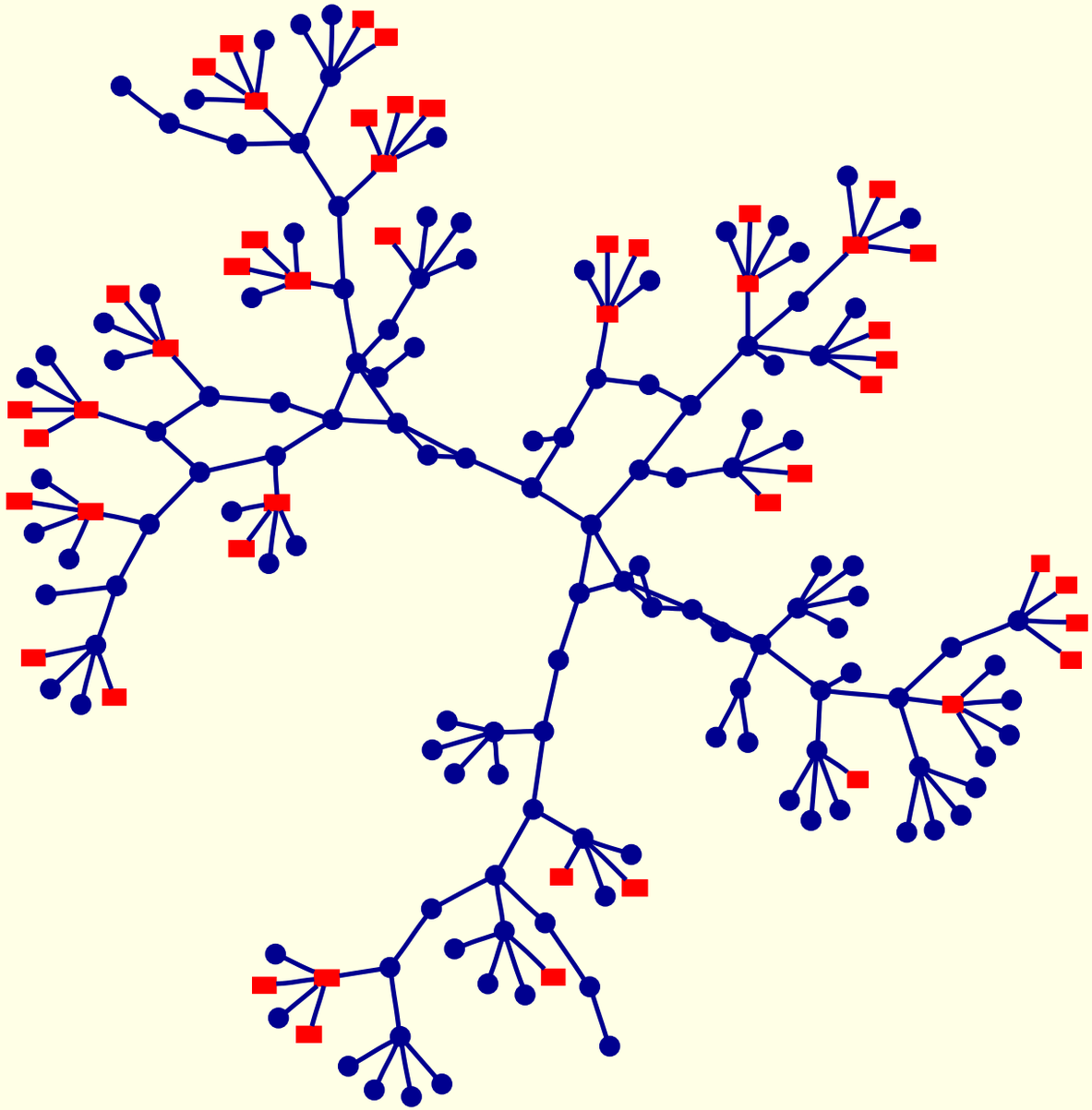
   $\forall h_j \in \mathcal{C}_i, \forall v \in h_j$, do $E_{i+1} \leftarrow E_{i+1} \cup \{(v, l_j)\}$ and $V_{i+1} \leftarrow V_{i+1} \cup \{v\}$
4. Interference matrix update

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

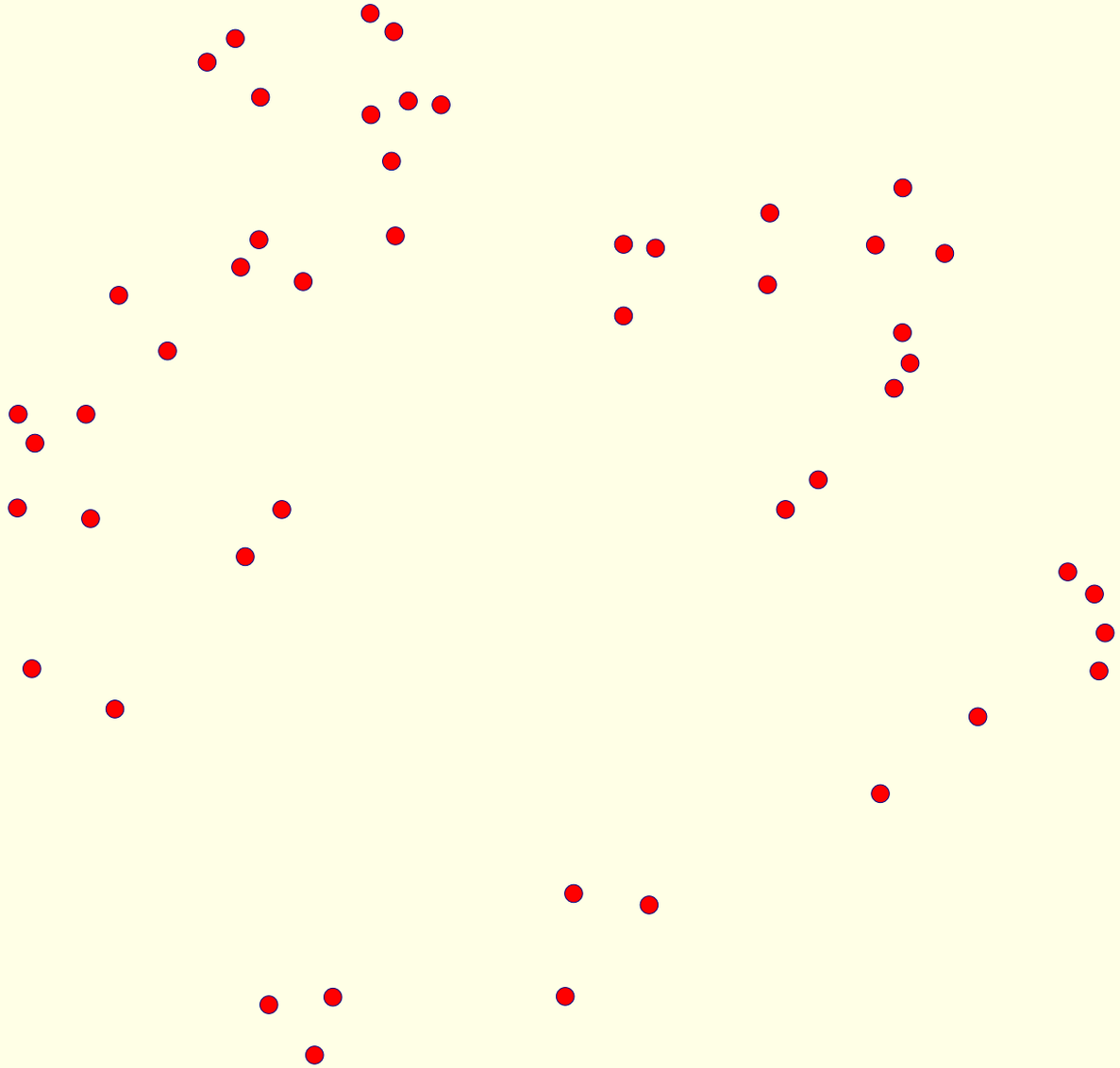
   For each $m_\alpha, m_\beta, m_\gamma, m_\delta \in \mathcal{C}_i$ so that $m_\alpha \in h_\alpha$, $m_\beta \in h_\beta$, $m_\gamma \in h_\gamma, m_\delta \in h_\delta$.

   $I(\mathcal{C}_{i+1}, \curlywedge)\left(l_\alpha, l_\beta, l_\gamma, l_\delta\right) = I\left(\mathcal{C}_i, \curlywedge\right)\left(m_\alpha, m_\beta, m_\gamma, m_\delta\right)$
5. Iterate 2–3 until $\mathcal{C}_i = \mathcal{C}_{i+1}$.

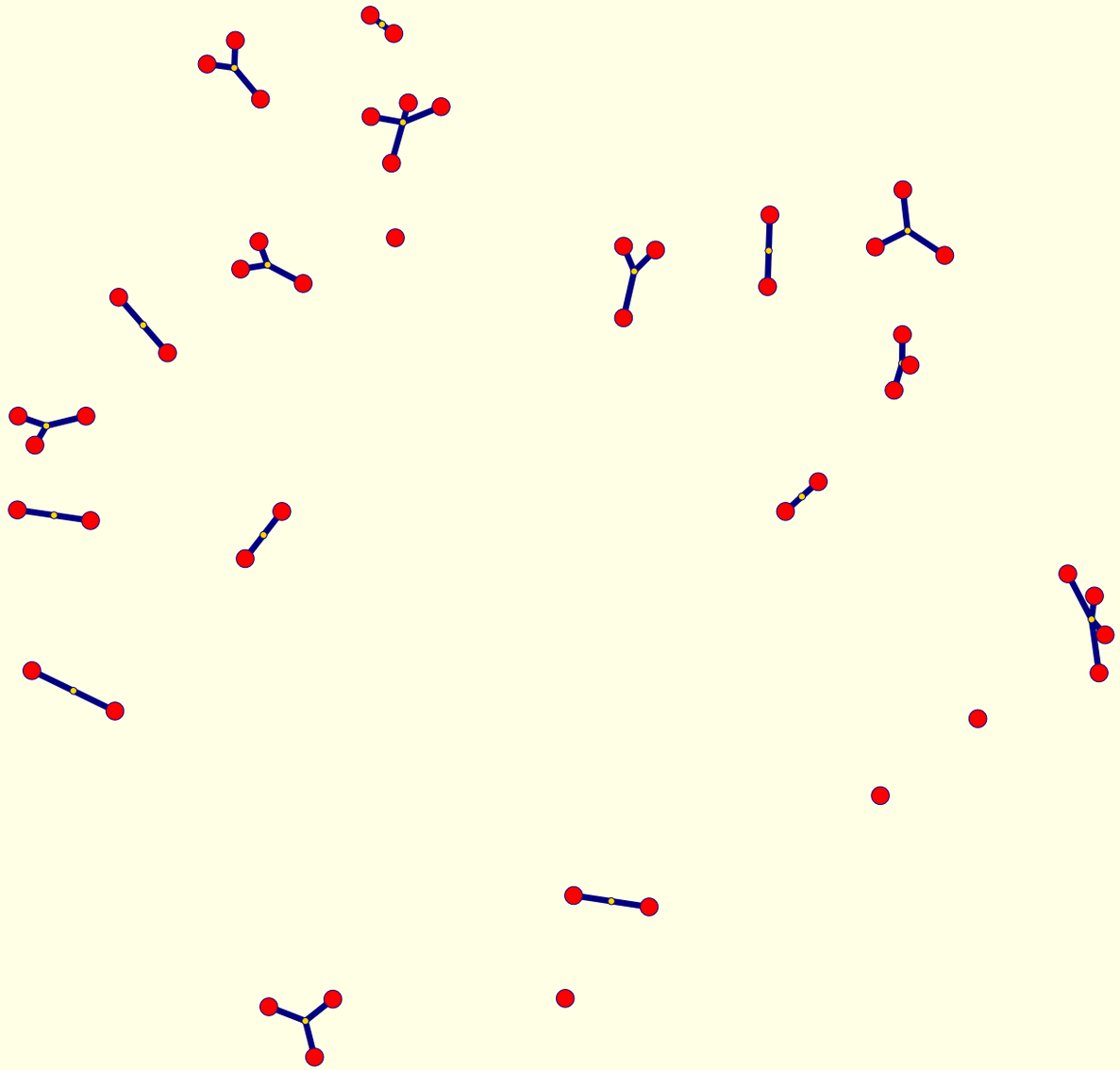
ALNeM: example of execution



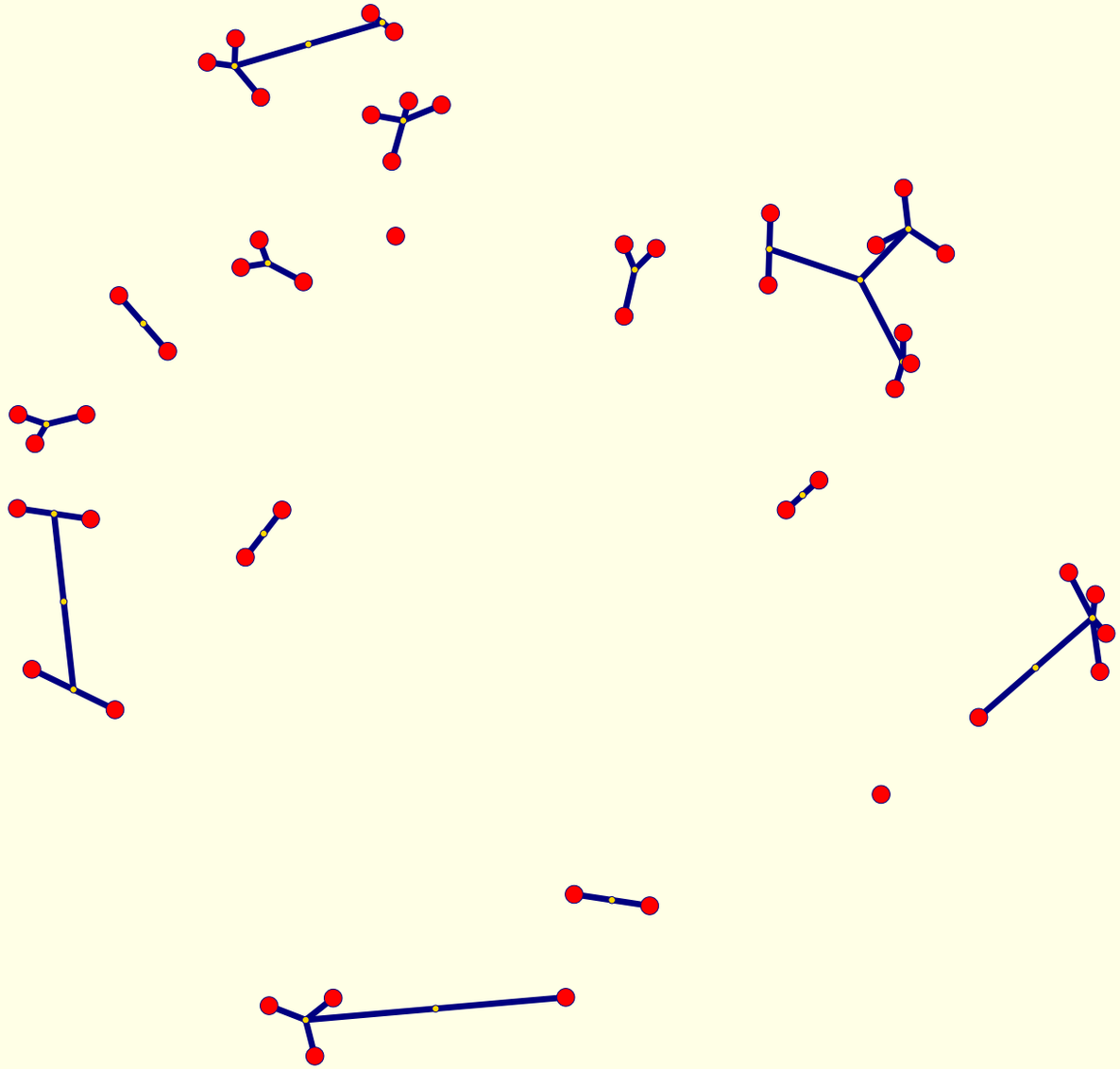
ALNeM: example of execution



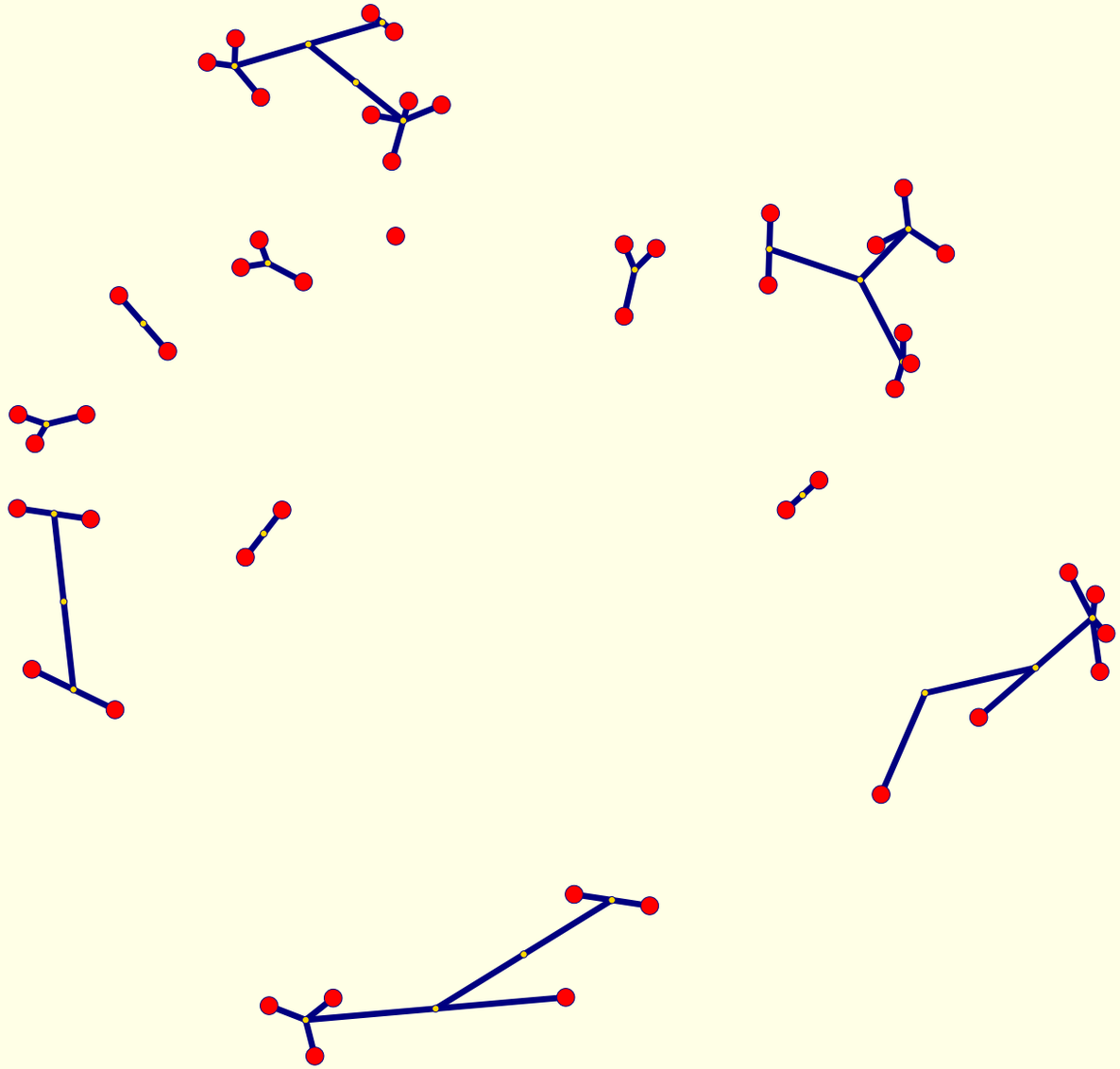
ALNeM: example of execution



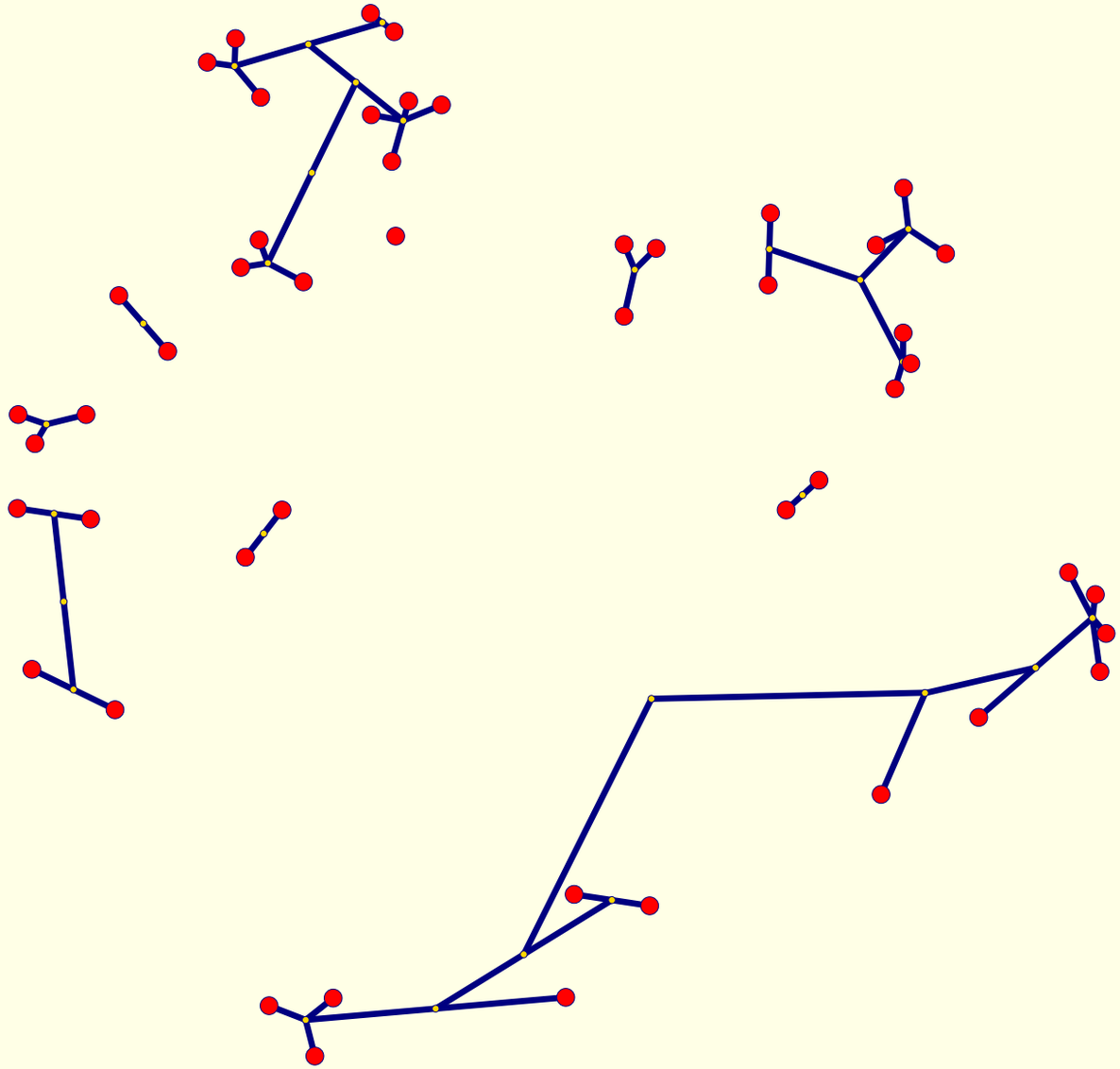
ALNeM: example of execution



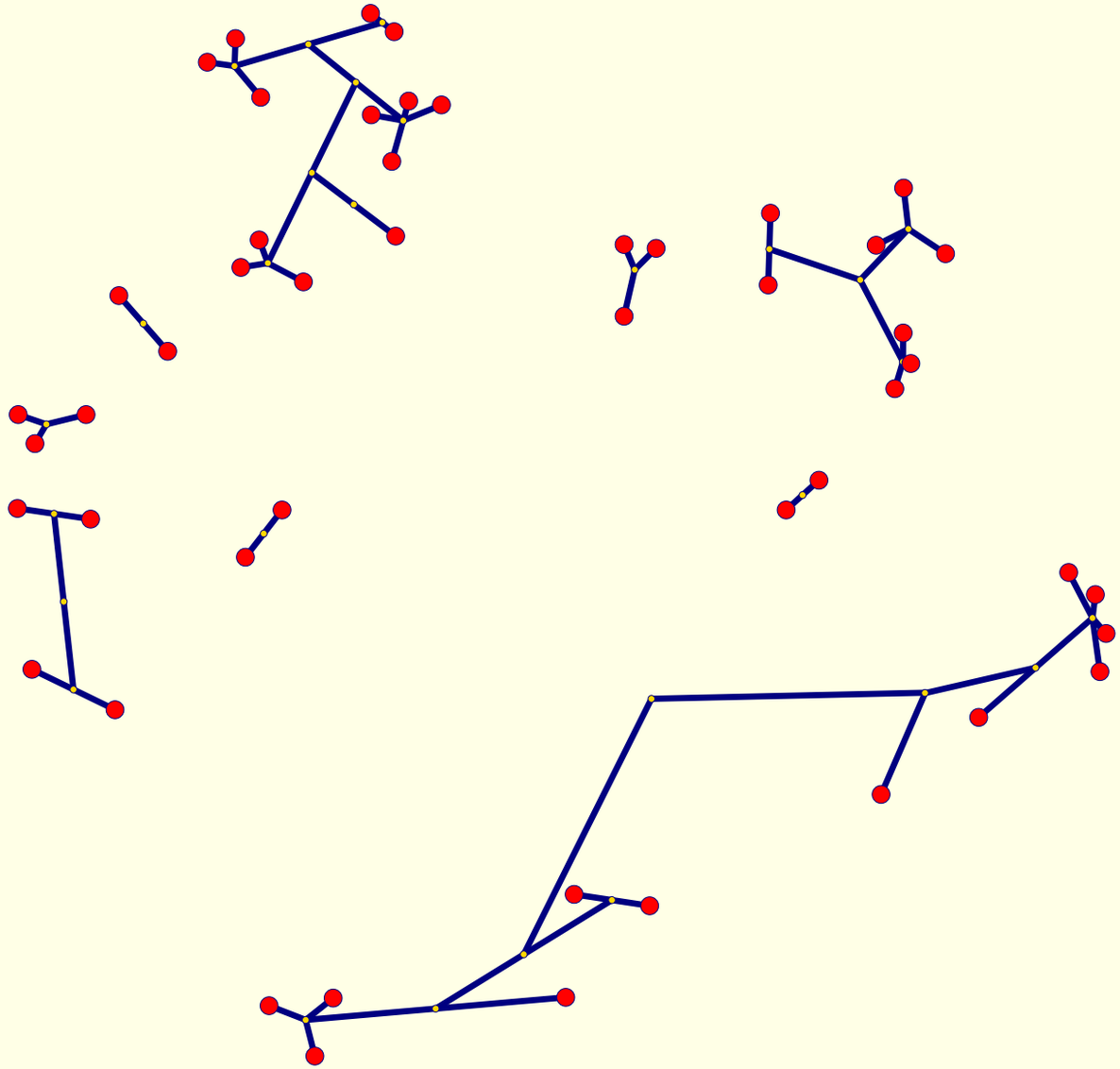
ALNeM: example of execution



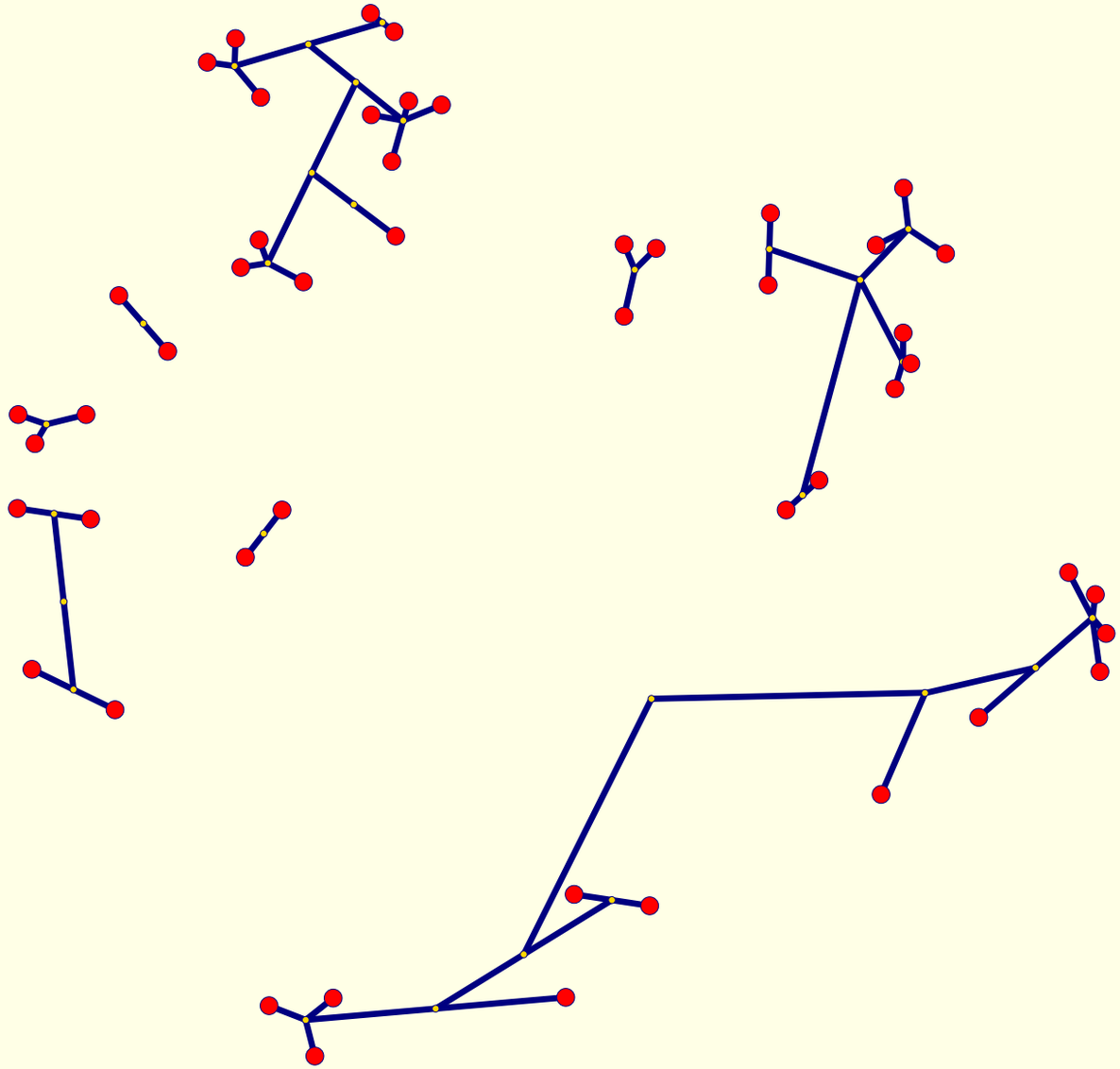
ALNeM: example of execution



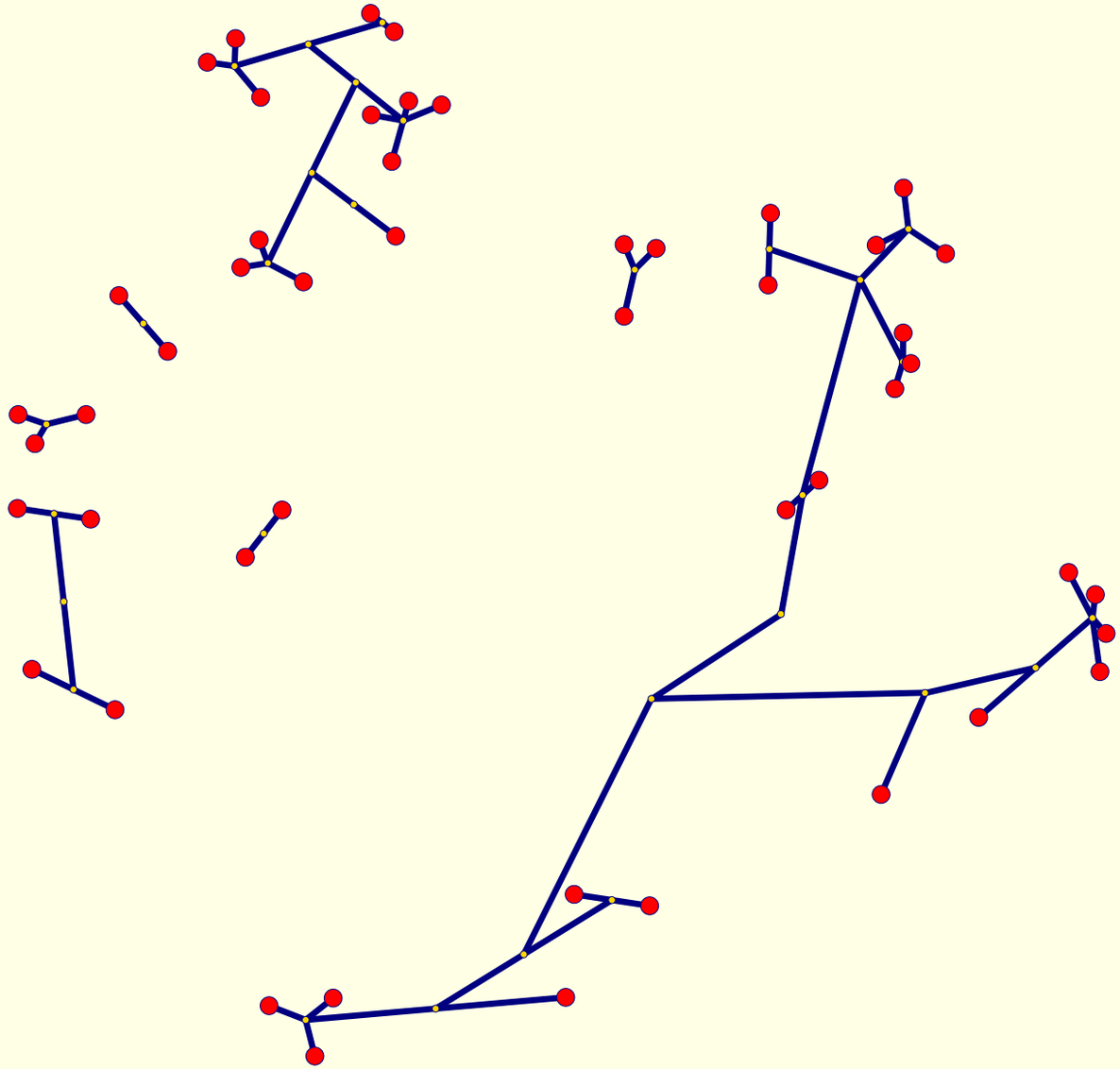
ALNeM: example of execution



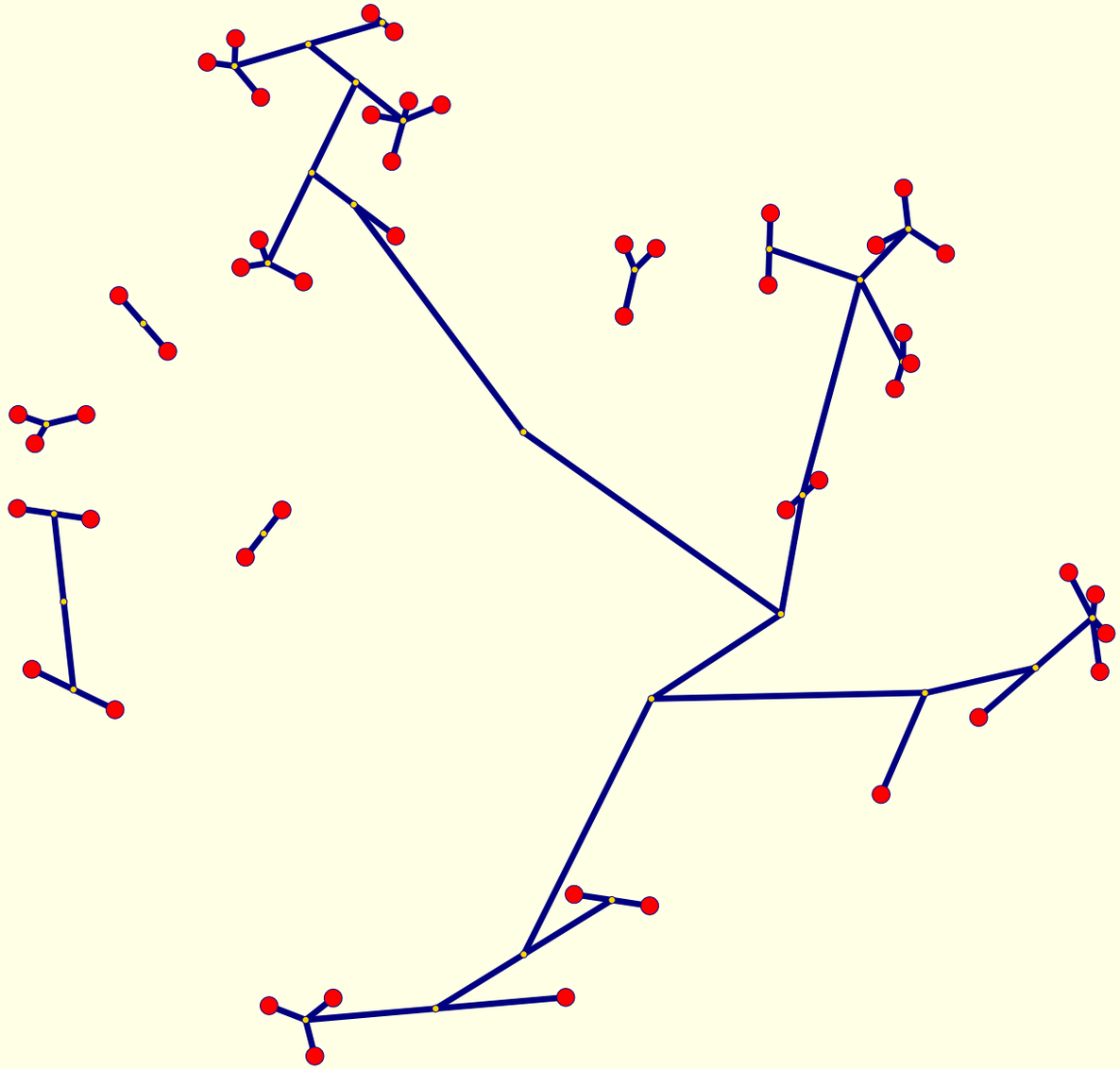
ALNeM: example of execution



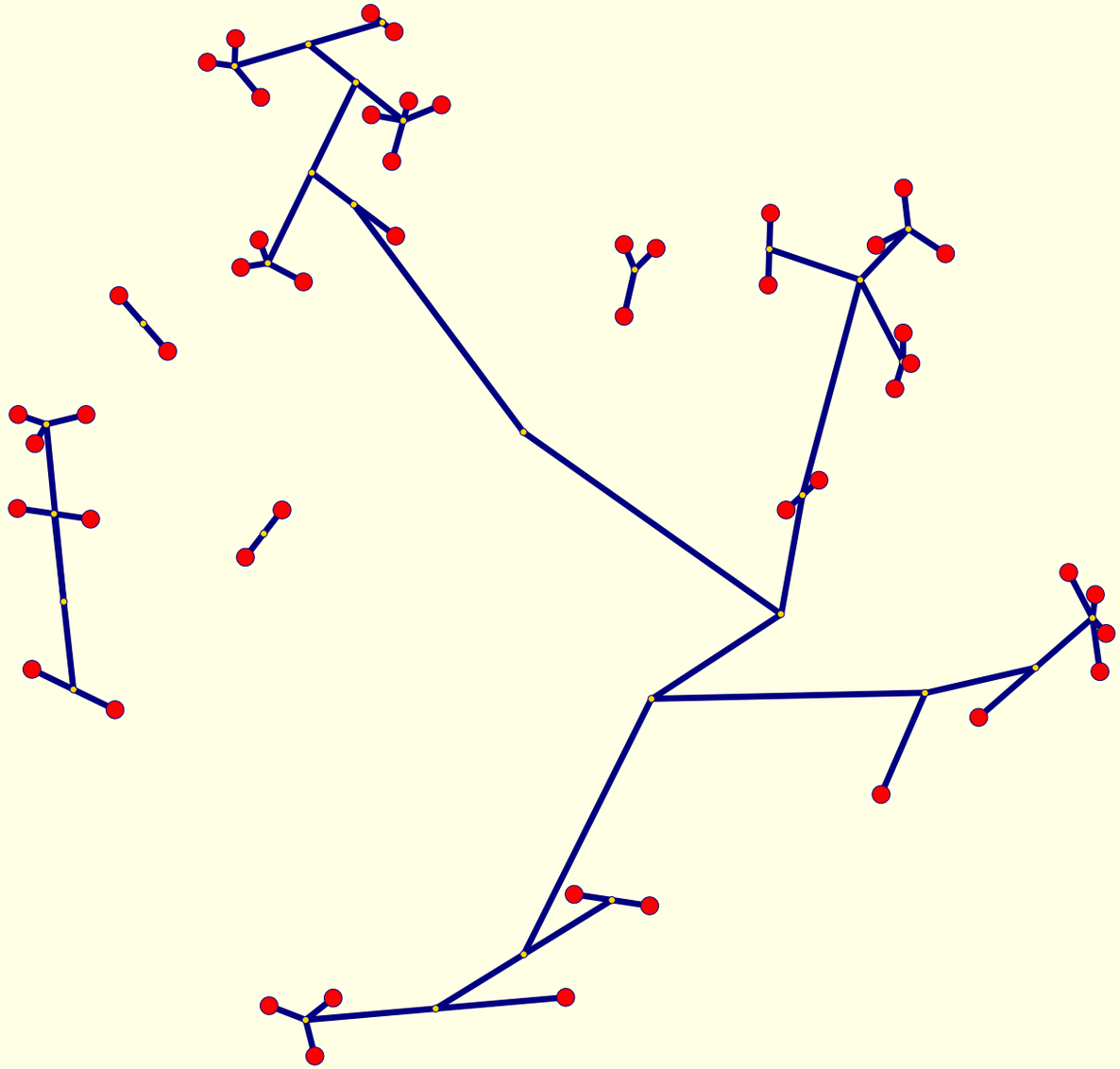
ALNeM: example of execution



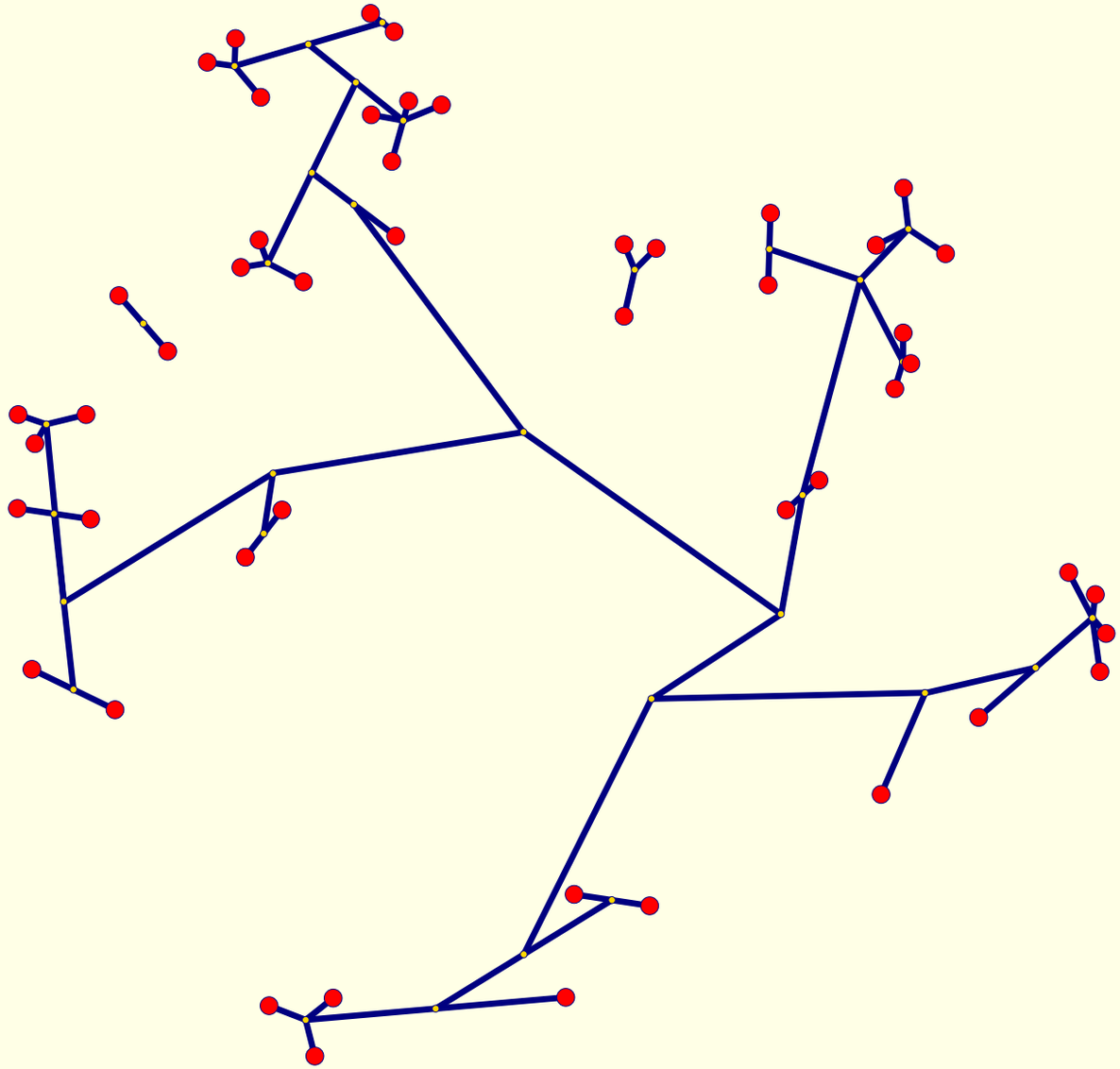
ALNeM: example of execution



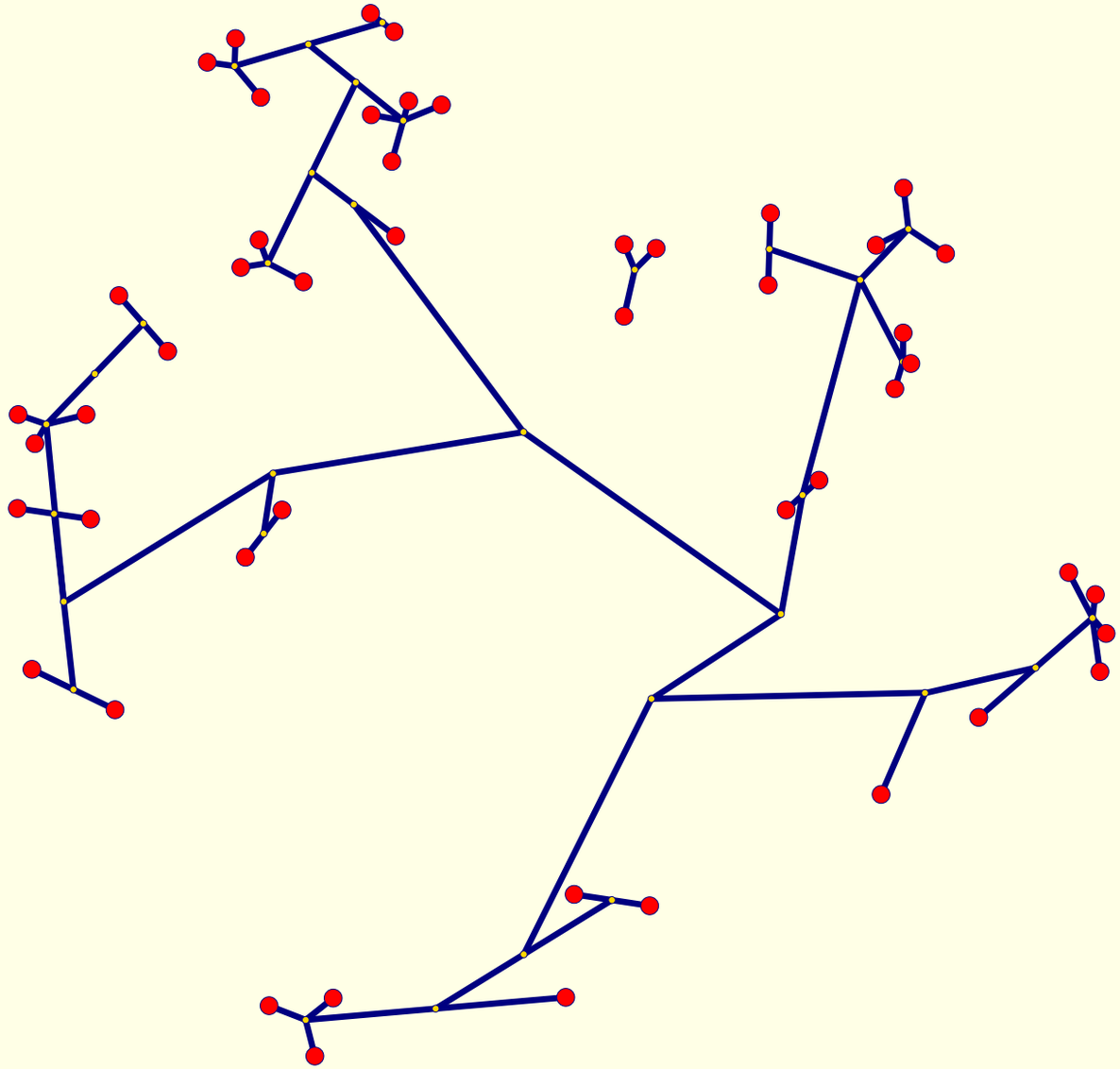
ALNeM: example of execution



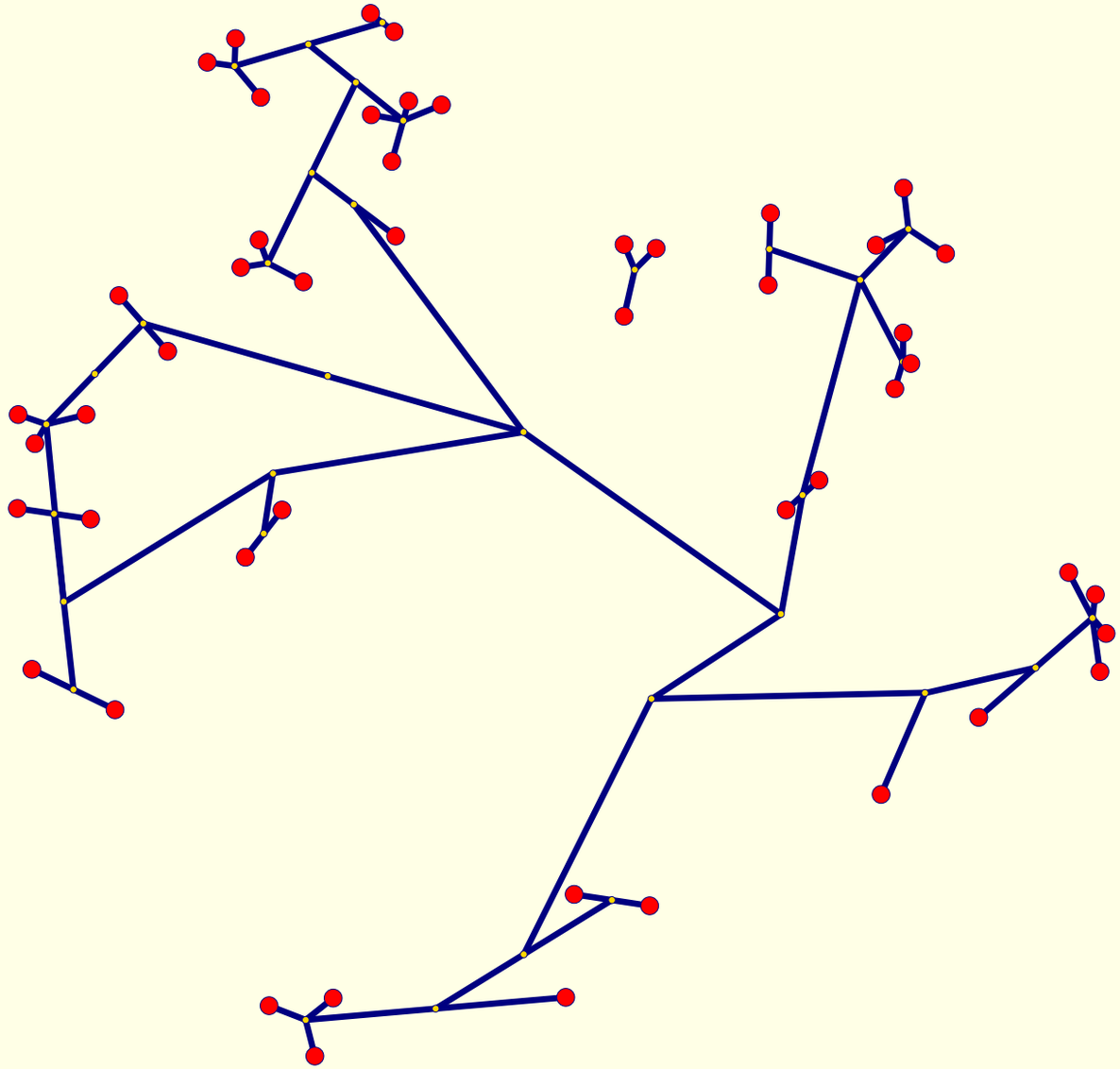
ALNeM: example of execution



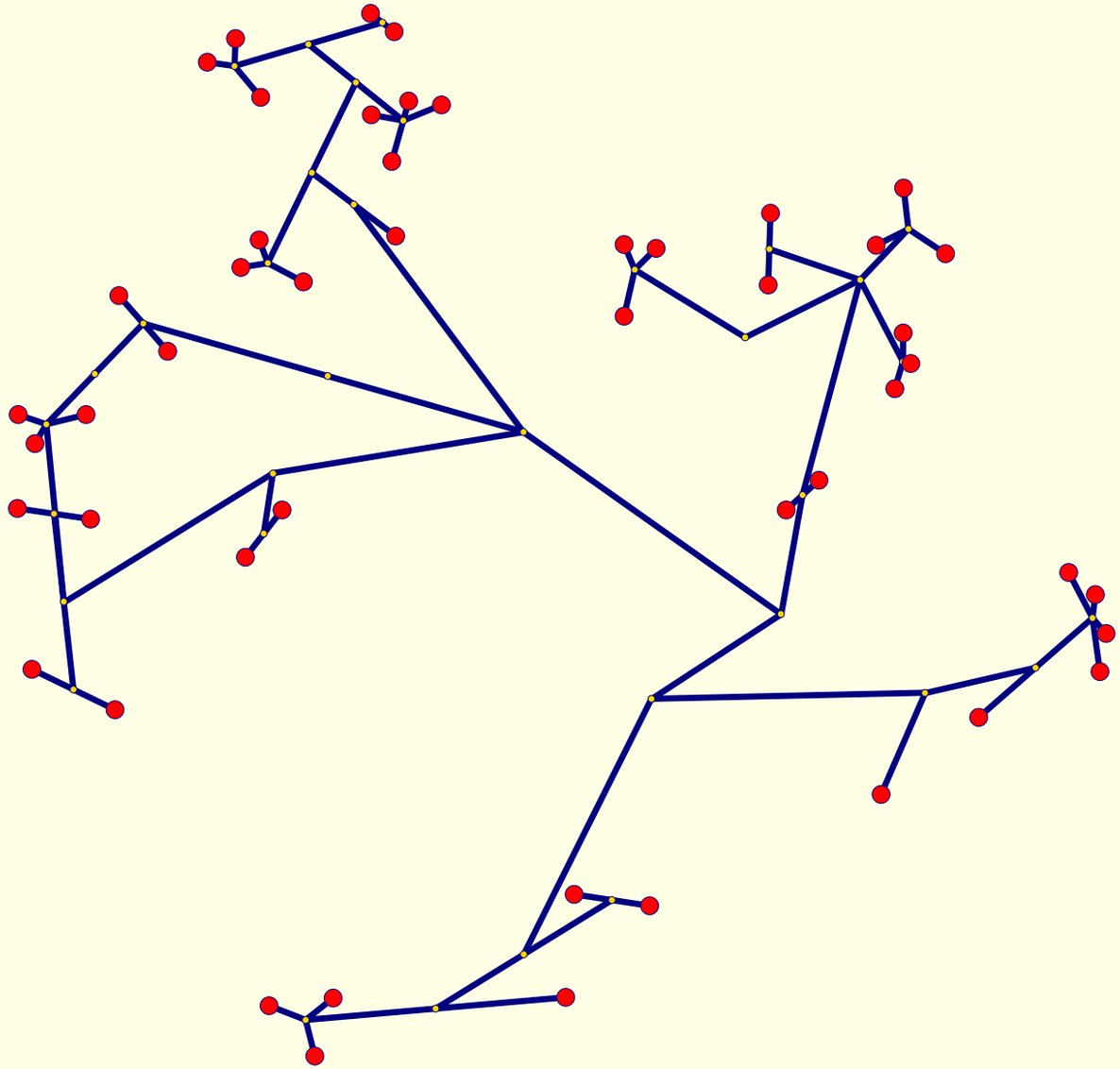
ALNeM: example of execution



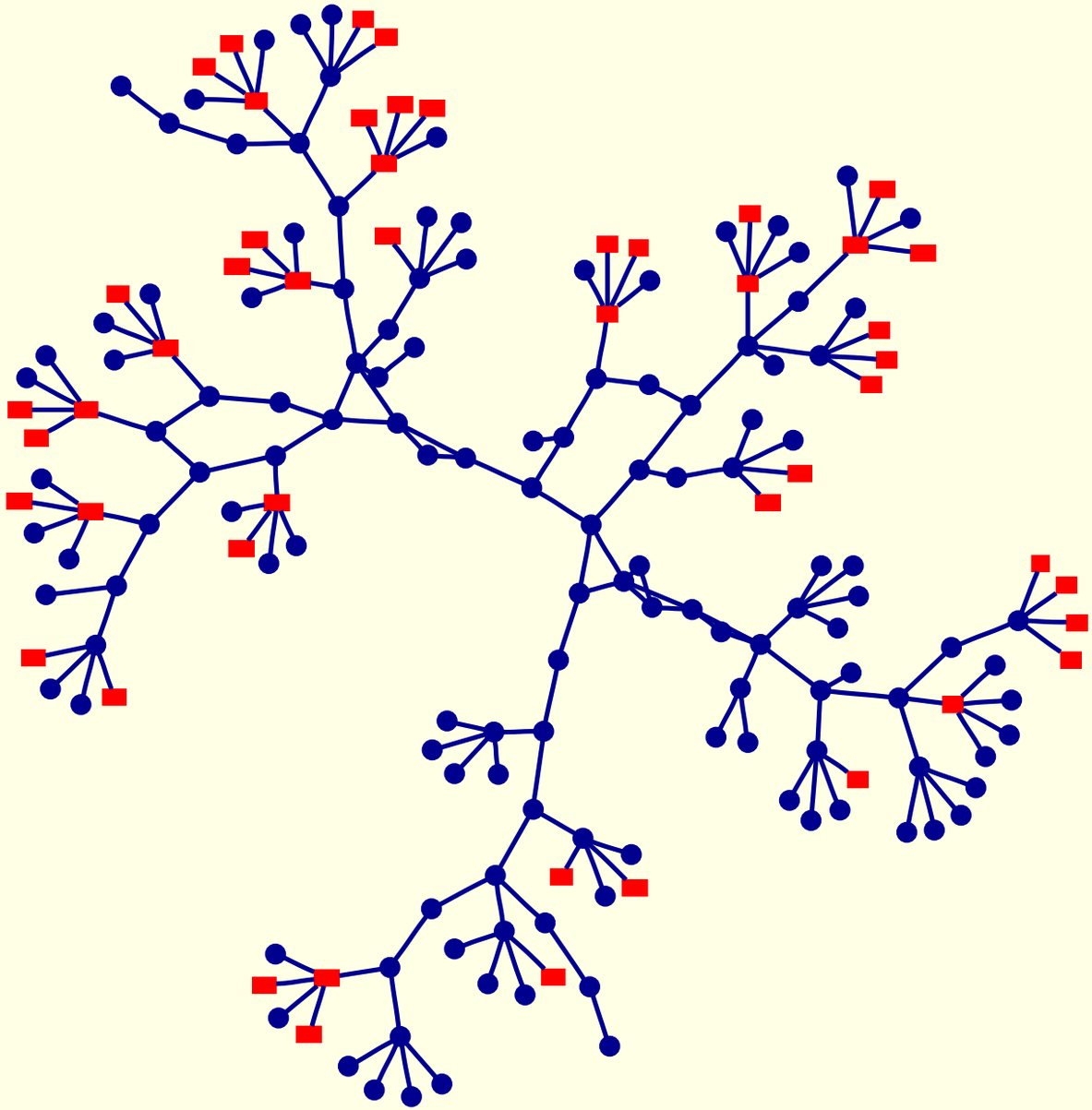
ALNeM: example of execution



ALNeM: example of execution



ALNeM: example of execution



DIET: Distributed Interactive Engineering Toolbox

Goal : Metacomputing platform (GridRPC model)

- Complete and ready to use for users
- Extensible by researchers

Main functionalities :

- Distributed and hierarchical scheduling;
- Resources localization ;
- Data persistence ;
- Platform monitoring ;

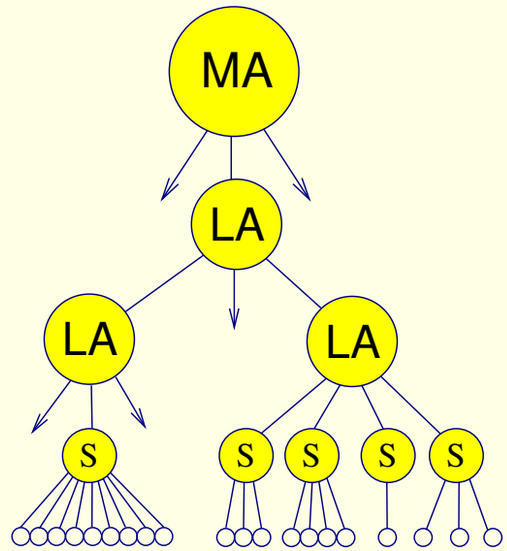
Teams : GRAAL (ENS-Lyon), U. Besançon, Insa-Lyon, Loria (Nancy), Sun.

Targeted applications : Grid-ASP

- Digital elevation model (Geology – LST ENS-Lyon) ;
- Molecular dynamics (Physique – Lyon-I *et al.*) ;
- HSEP (chemical – SRSMC Nancy) ;
- Circuit simulation (electronic – Ircom) ;
- ACI TLSE (sparse matrix expertise – Toulouse) ;

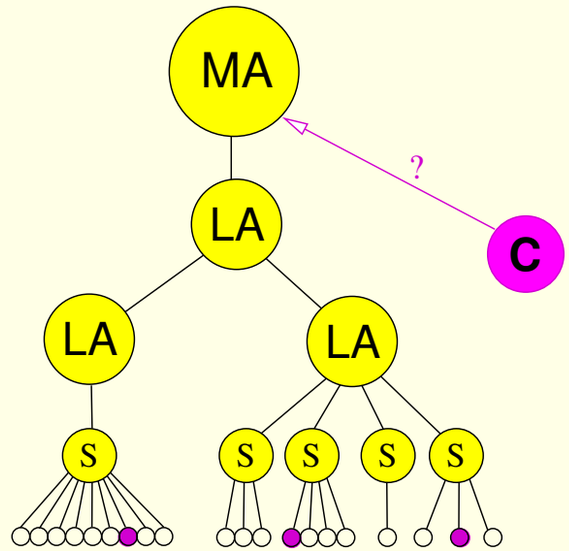
DIET : Handling of a request

1. Clients connect to the MA
2. Request transmission to servers
3. Performance evaluation : FAST (NWS)
4. Back to MA : distributed scheduling
5. (Broadcast if impossible in local tree)
6. Result sent back to the client
7. Direct client-server connection



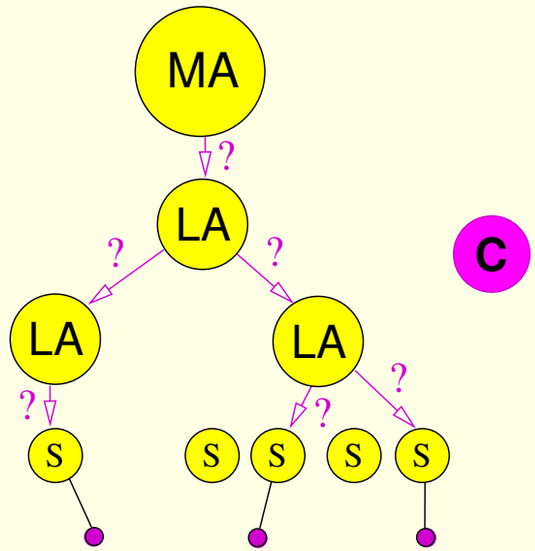
DIET : Handling of a request

1. Clients connect to the MA
2. Request transmission to servers
3. Performance evaluation : FAST (NWS)
4. Back to MA : distributed scheduling
5. (Broadcast if impossible in local tree)
6. Result sent back to the client
7. Direct client-server connection



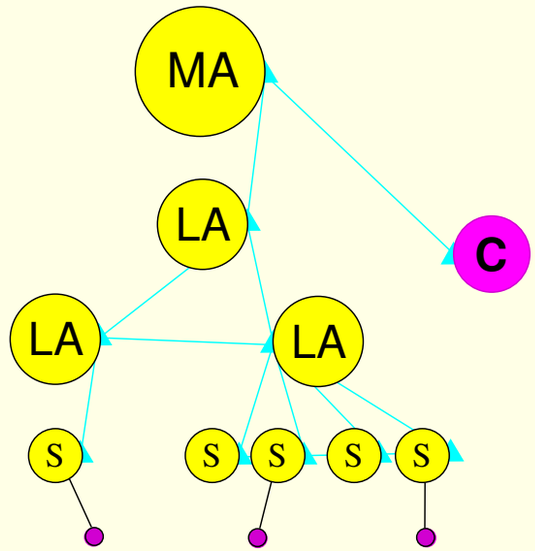
DIET : Handling of a request

1. Clients connect to the MA
2. Request transmission to servers
3. Performance evaluation : FAST (NWS)
4. Back to MA : distributed scheduling
5. (Broadcast if impossible in local tree)
6. Result sent back to the client
7. Direct client-server connection



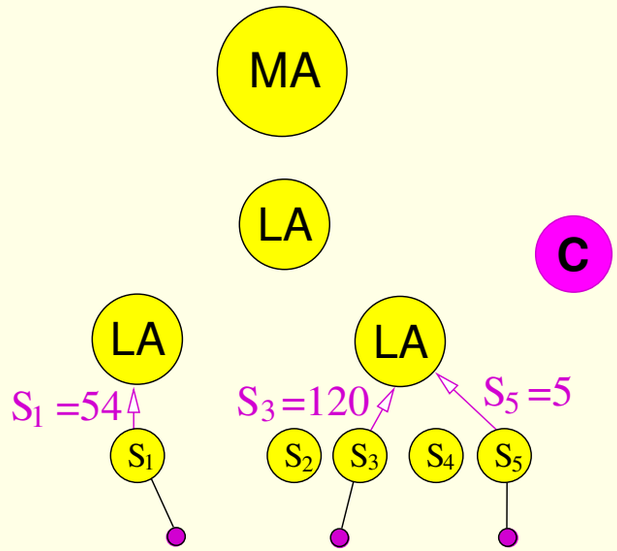
DIET : Handling of a request

1. Clients connect to the MA
2. Request transmission to servers
3. Performance evaluation : FAST (NWS)
4. Back to MA : distributed scheduling
5. (Broadcast if impossible in local tree)
6. Result sent back to the client
7. Direct client-server connection



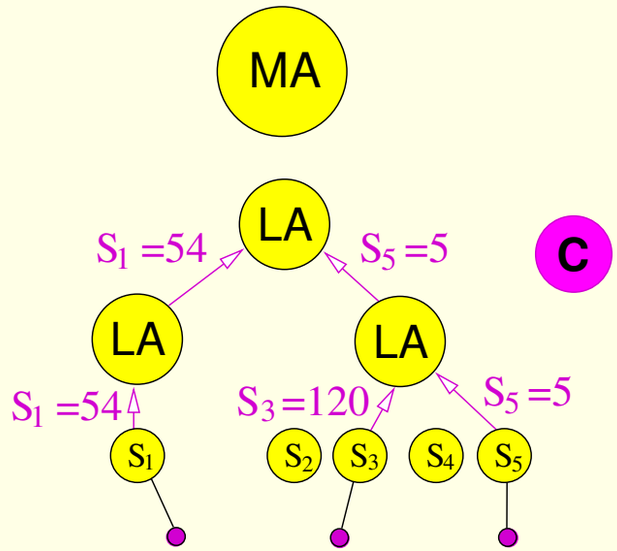
DIET : Handling of a request

1. Clients connect to the MA
2. Request transmission to servers
3. Performance evaluation : FAST (NWS)
4. Back to MA : distributed scheduling
5. (Broadcast if impossible in local tree)
6. Result sent back to the client
7. Direct client-server connection



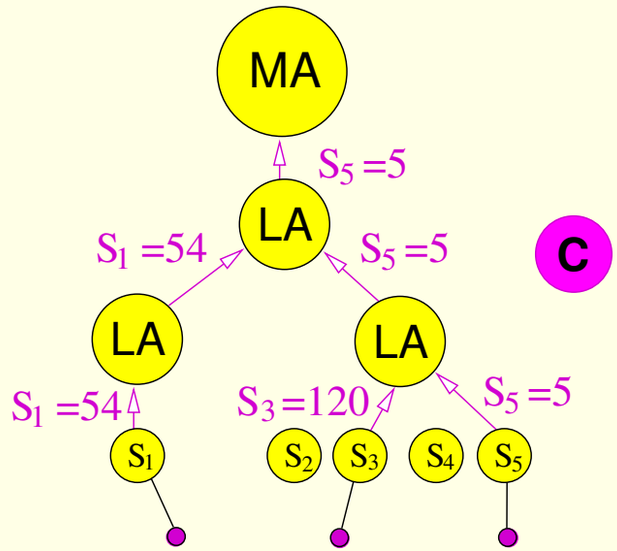
DIET : Handling of a request

1. Clients connect to the MA
2. Request transmission to servers
3. Performance evaluation : FAST (NWS)
4. Back to MA : distributed scheduling
5. (Broadcast if impossible in local tree)
6. Result sent back to the client
7. Direct client-server connection



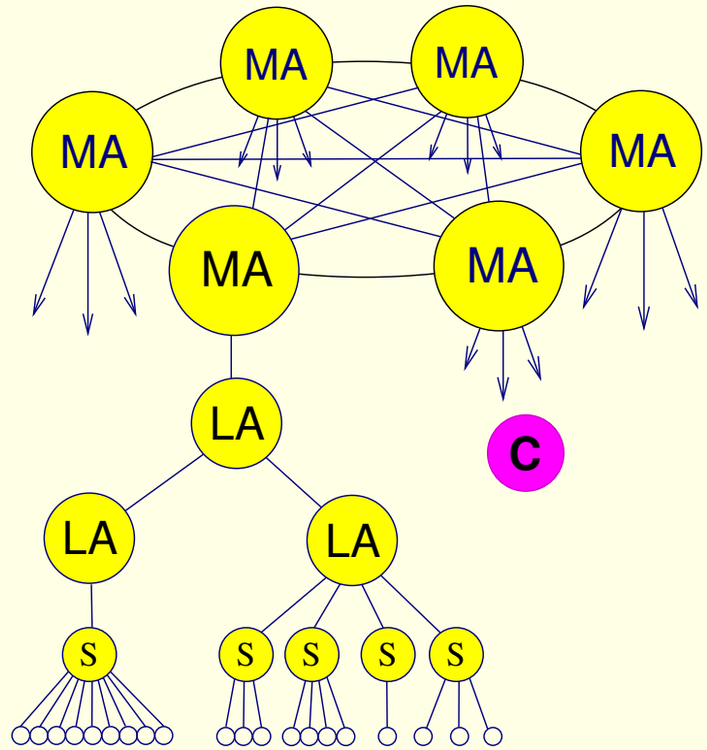
DIET : Handling of a request

1. Clients connect to the MA
2. Request transmission to servers
3. Performance evaluation : FAST (NWS)
4. Back to MA : distributed scheduling
5. (Broadcast if impossible in local tree)
6. Result sent back to the client
7. Direct client-server connection



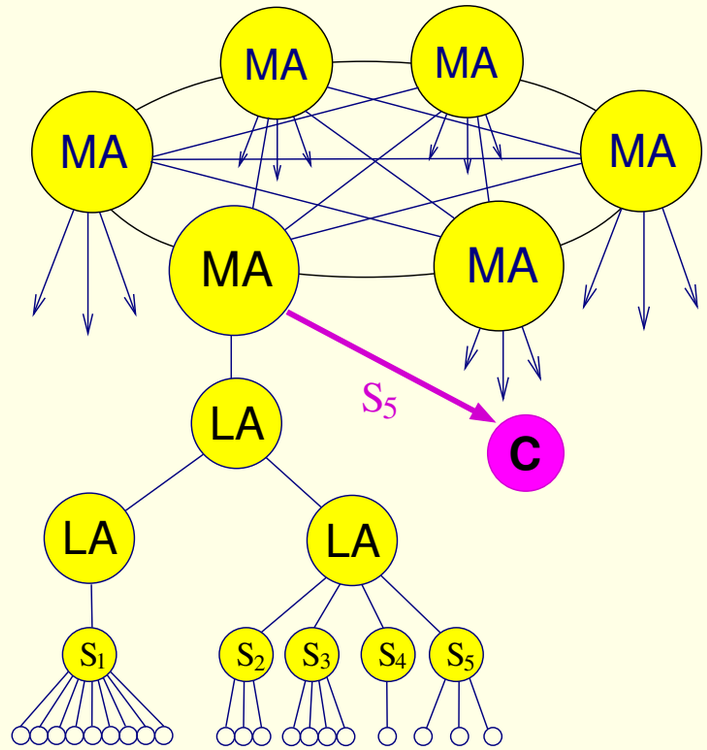
DIET : Handling of a request

1. Clients connect to the MA
2. Request transmission to servers
3. Performance evaluation : FAST (NWS)
4. Back to MA : distributed scheduling
5. (Broadcast if impossible in local tree)
6. Result sent back to the client
7. Direct client-server connection



DIET : Handling of a request

1. Clients connect to the MA
2. Request transmission to servers
3. Performance evaluation : FAST (NWS)
4. Back to MA : distributed scheduling
5. (Broadcast if impossible in local tree)
6. Result sent back to the client
7. Direct client-server connection



DIET : Handling of a request

1. Clients connect to the MA
2. Request transmission to servers
3. Performance evaluation : FAST (NWS)
4. Back to MA : distributed scheduling
5. (Broadcast if impossible in local tree)
6. Result sent back to the client
7. Direct client-server connection