Arigatoni: A Simple, Programmable Overlay Network

Luigi Liquori,
MASCOTTE PROJECT,
INRIA/CNRS/UNSA
joint work with
Didier Benza, Raphael Chand,
Michel Cosnard, and Marc Vesin

Université Nice Sophia Antipolis
CNRS Centre National de la Recherche Scientifique
Institut National de Recherche en Informatique et en Automatique
INRIA

Talk Outline

Context:

Global Computing & Overlays Networks

The *Arigatoni* Main Units

Arigatoni's Resource Discovery Protocol (GIP)

Arigatoni's Intermittence Protocol (VIP)

Protocols Evaluation

Conclusions

This work is supported by the IST Research Project AEOLUS (Algorithmic Principles for Building Efficient Overlay Computers) of the FET Global Computing Program, by the CoreGRID Network of Excellence, and by the COLOR INRIA Sophia Antipolis.

Distributed Computer Architecture

- Conceive a Programmable **Global Computer (GC)**, a.k.a, **Overlay Computer (OC)** of Grand Scale, consisting of Internet connected, globally available, ***"Computing Individuals"*** (Laptop, PC, GSM, PDA, PCluster, SensNet, iPod, iWashing Machine, ElectroPizzas, etc)
- Able to offer, demand, and organize a rich set of resources such as: *Computational Power, Storage, Raw/Semantic Data retrieval, Bandwidth, etc.*
- Resources being available **Globally, Transparently**, **Securely**, **Efficiently**
- All GC cooperate to make the OC, ***"General Purpose"***, or ***"Programmable"***, or ***"Turing Complete"***, i.e. able to program, compute/run all **Partial Recursive Functions**, according to the well-known ***"Church's thesis"***

Distributed Computer Architecture (II)

- Conceive an **Overlay Network (ON)** of **GC**, physically connected via IP, or other *ad hoc* networks, logically organized in **Virtual Organizations (VO)**, a.k.a. ***"Colonies"*** with ***"Leaders"*** and ***"Individuals"***
- Clear rules to join/leave the colony (topology of the VO highly dynamic)
- Based on the **Pareto & Nash Equilibriums**, well known from Game Theory

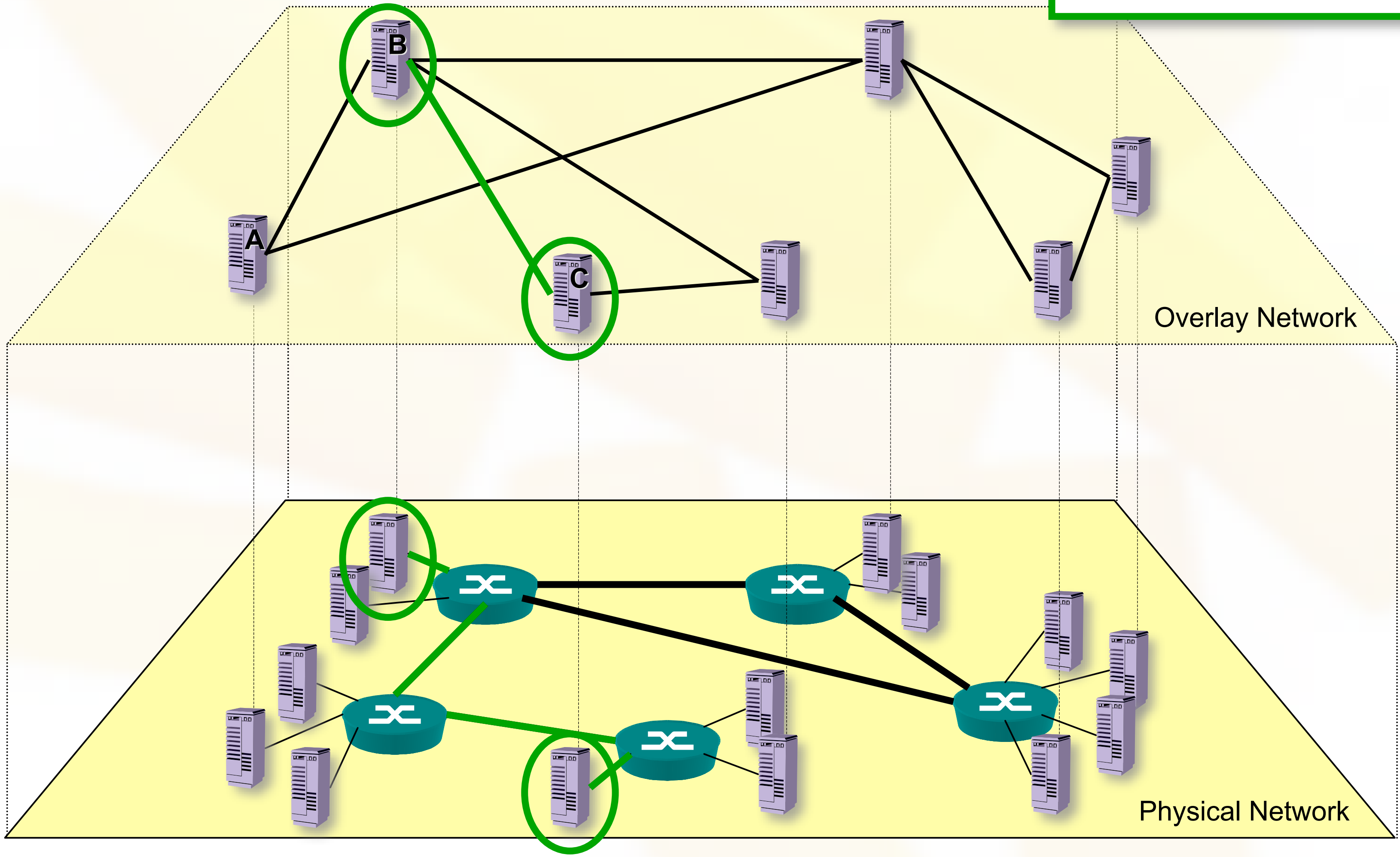
TECHNOLOGICAL ISSUES

Crossing administrative barriers, authentication, trust, routing

- Algorithms for **Routing Requests** and **Discover Resources**
- Scaling up to **Large** Overlay Computer, **Reliability** (point of failure)
- Service Availability, Load Balancing, **QOS**
- Registration policy (specialized vs. general purpose colonies), etc

Overlay Networks

Treat multiple hops through IP network as one hop in an overlay network



The Arigatoni Overlay Network (©INRIA)

Protocols

- **Resource Discovery Protocol (RDP)**
- **Virtual Intermittent Protocol (VIP)**

Resource Discovery

- How resources, offered by individuals, are discovered **transparently**
- How changed states of resources are upgraded in **routing tables**

Virtual Intermittent

- How individuals organize themselves to share resources **transparently**
- How the organization evolves in time and in space **transparently**

Global Computers, Brokers, Routers

Global Computers (GC)

- Discontinuous participation in the VO (Colony)
- Partial/Zero knowledge of the ***current*** VO
- Ask and provide services with variable guarantees
- Can work in **Local Mode** or in **Global Mode**

Global Brokers (GB)

- Colony's leader, but recursively an individual in a surrounding "SuperColony"
- Register/Unregister GC in the own colony
- Send/Receive GC's queries
- Contact GCs in its population or contact its direct SuperGBs
- Trust their population at any level of the negotiation (via e.g. PKI)

Global Routers (GR)

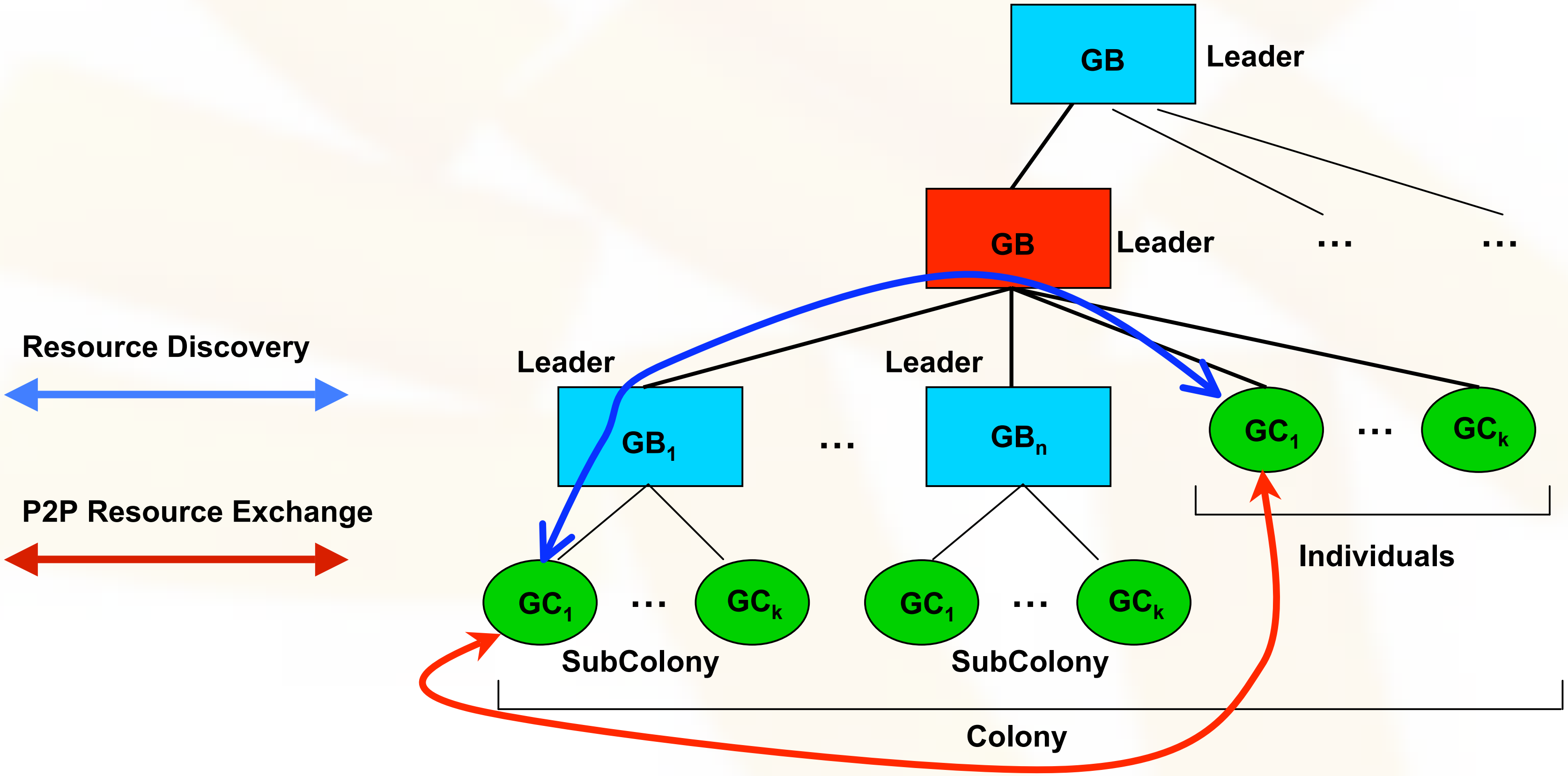
- Pack/Unpack payload in/of GIP's packets between GC and GB

Arigatoni Overlay Network Topology

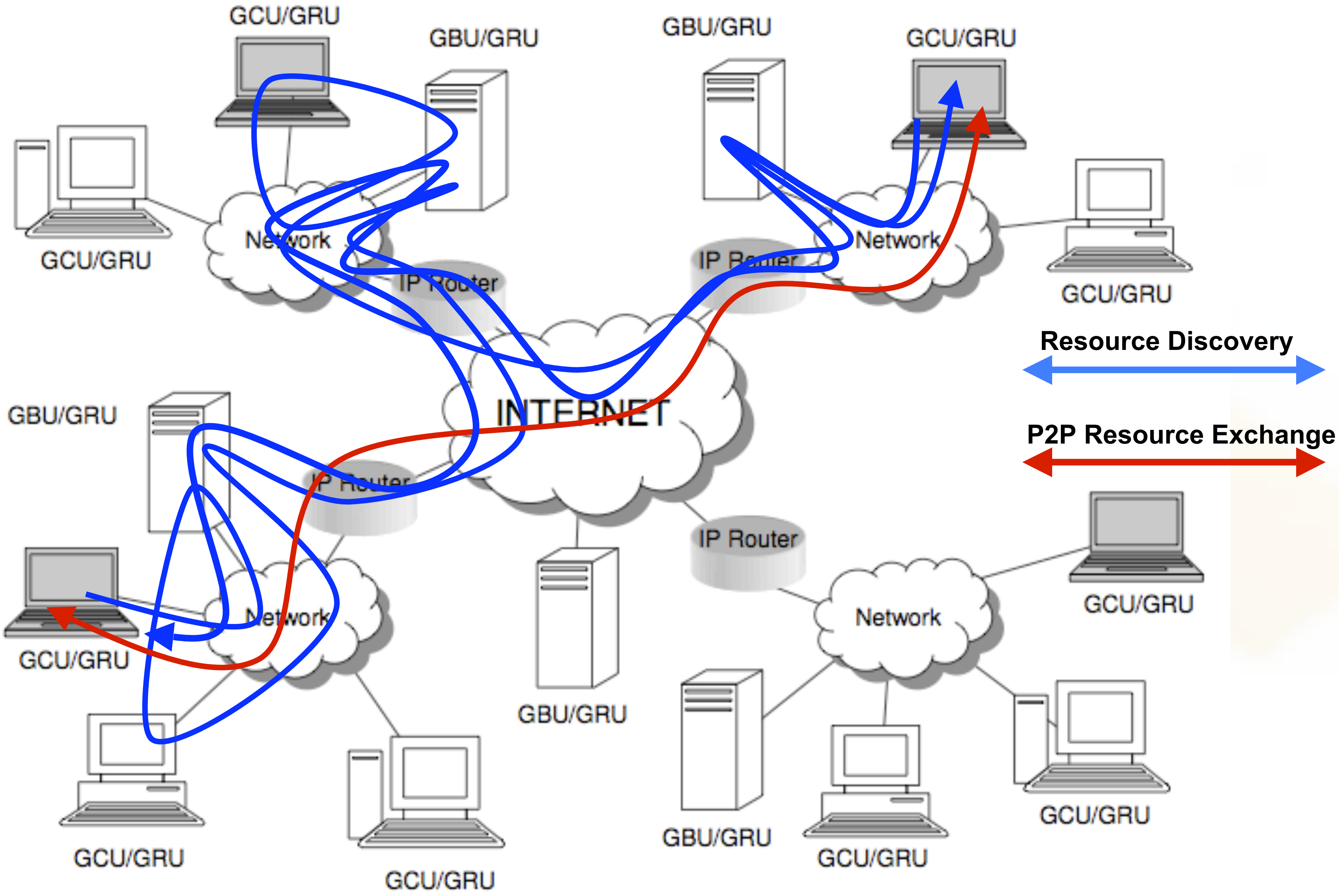
Arigatoni is composed by **Colonies and SubColonies**

Hierarchical tree n-layer structured organization

Once the resource/s is/are negotiated, GCs communicate in P2P fashion



Arigatoni Overlay Network @ Glance



Protocols in the Arigatoni Overlay

Resource Discovery Protocol (RDP)

- Deals with Service Requests (SREQ) and Service Response (SRESP) between individuals (clients and servers)
- **Filter and route** through Colony leaders (GB)
- Completely independent on the ***"kind of service"***

Virtual Intermittent Protocol (VIP)

- Deals with intermittent participation (SREG) of individuals
- Registration and unregistration modalities of individuals in colonies
- Measure and fire ***"free riders"*** in a colony

Resource Discovery Protocol (RDP)

Routing tables maintained at each GB

- Set of services registered in each Colony

Always search in own colony first: *promote OO encapsulation*

- Local colony of GCs
- Other sub-colonies in Colony
- Other Colonies via the GB leader=recursive search, *OO-based (à la Java)*

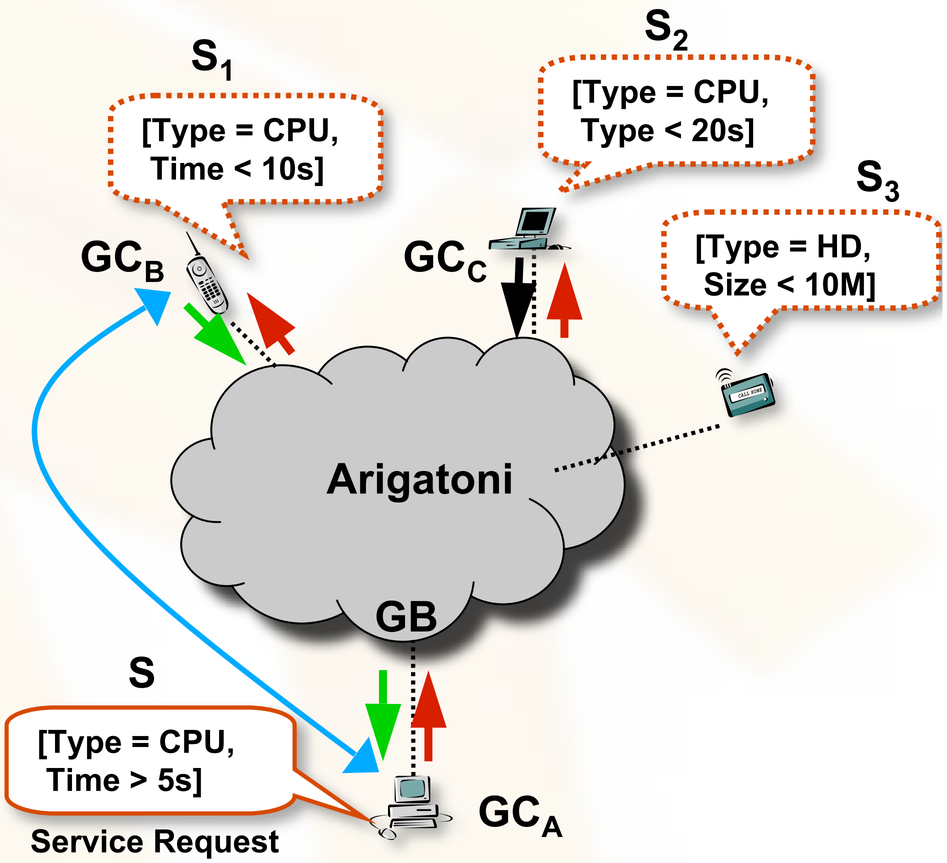
Service request for service S

- Search GCs in local colony that accepts to serve S
- Search in other sub-colonies
- If none are found, delegate to the leader GB …

Looks much like “method lookup” in OO languages …

Resource Discovery Protocol

1. GC's publish services they can offer
2. GC issues **service request S** to the system via its GB leader
3. GB finds all the resource(s) needed to satisfy the re-quested services of the GC client locally in the colony
4. If GB did not find all the resource(s) in the local colony, it will forward and delegate the request to another broker
   (Steps 3+4 can be iterated)
5. After a timeout period, or when all delegate GBs failed to satisfy the delegated request, the broker will notify to the GC client the refusal of service
6. Then, the GC client will directly talk with GC servant(s), and the latter will manage the request, as in classical P2P systems



Resource Discovery Protocol

Different intra-colony search modes

- **Selective**: Search ***one*** GC in Colony able to serve S
- **Exhaustive**: Search ***all*** GCs in Colony able to serve S

Different reply modes

- **Selective**: report ***one*** GC that accepted to serve S
- **Exhaustive**: report ***all*** GCs that accepted to serve S

Resource Discovery Protocol

S = [Type = CPU,Time > 10s]

Selective intra-colony search mode

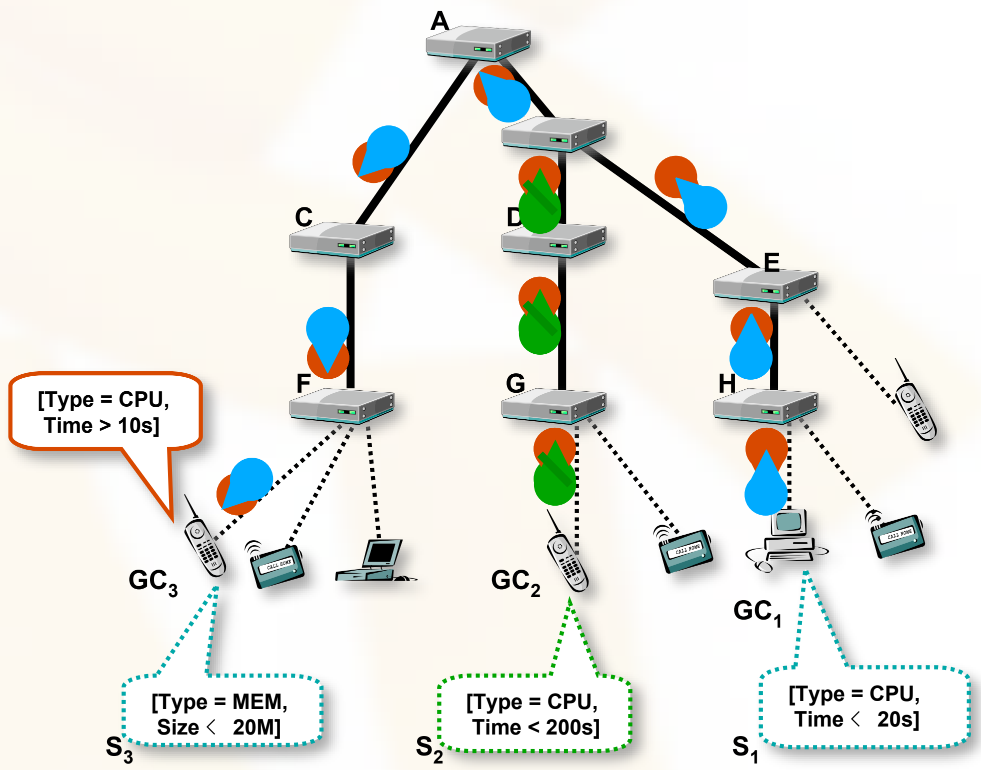
- uses less resources
- can lead to poor delay

Exhaustive search mode

- uses more resources
- can improve delay

Selective intra-colony search

Exhaustive intra-colony search



Resource Discovery Protocol V1

Algorithm 1 The Resource Discovery Routine in the Arigatoni GIP Protocol

```
case Message is
SREQ :
   ReturnPath{Message.Id} ← Message.Sender
   SendList ← SelectPeers(Message.Services, search_mode)
   for each (P, Serv(P)) ∈ SendList do
      Send ServiceRequest(Serv(P)) to P
   end for
   for each S∈Message.Services such that ∄(P, Serv(P))∈SendList, S∈Serv(P) do
      Append S to RejectList
   end for
   Send ServiceResponse({},RejectList) to ReturnPath[Id]
SRESP :
   for each S∈Message.AcceptedServices do
      if (S was not already accepted ) ∨ (EXHAUSTIVE_REPLY is set) then
         Append S to AcceptList
      end if
   end for
   SendList ← SelectPeers(Message.RejectedServices, intra_Colony_mode)
   for each (P, Serv(P))∈SendList do
      Send ServiceRequest(Serv(P)) to P
   end for
   for each S∈Message.RejectedServices such that ∄(P, S(P))∈SendList, S∈Serv(P) do
      Append S to RejectList
   end for
   Send ServiceResponse(AcceptList,RejectList) to ReturnPath[Id]
end case
```

Resource Discovery Protocol V2 and V3

When a global computer asks for a service S, it also demands for a certain number of instances of S, e.g.

SREQ : [(S,n)] SREQ : [(S,n)]

Sport V2 (ARCS 07, LNCS)

Every individual (global computer or subcolony) registers itself in the colony with a tuple of distinct (services,instances):

SREG : [(S1,n1) AND · · · AND (Sk,nk)]

Every individual arises a service request to the colony's leader:

SREQ : [(S1,n1) AND · · · AND (Sk,nk)]

The colony's leader satisfy the request by analyzing all services and instances contained in his colony and in the surrounding colonies.

Spot V3 (Future Generation Computing System, 2007)

SREQ : [· · · AND ({S1 and S2 · · · and · · · Sh},n) AND · · ·]

VIP: About the Two Registration Modalities

Registration of a GC to the GB leader of a colony belonging to the same ***current administrative domain*** of the GC

Registration via ***remote tunneling*** of an GC to another GB leader of a colony belonging to a ***different administrative domain*** of the GC

In principle, any individual can register to many brokers

This may cause typical issues of ***resource overbooking*** (flybooking)

VIP: About the One Unregistration Modality

Unregistration of a GC when there are **no pending services** demanded or requested to the leader GB of the colony it belongs. The colony accepts the unregistration only if the colony itself will not be ***corrupted***

A GB cannot unregister from its own colony (*i.e*. it **cannot discharge itself**). For fault tolerance purposes, a GB can be faulty. In that case, the GCs will unregister one after the other and the colony will "disappear"

Once a GC has been disconnected from a colony belonging to any administrative domain, it can **migrate in another colony** belonging to any other administrative domain ("emigrant-like" model)

Virtual Intermittence Protocol (VIP)

Colony syntax

$$COL ::= \{GBU\} \mid COL \cup \{GCU\} \mid COL \cup \{\,COL\,\}$$

Community (or IP Soup) syntax

$$COM ::= \emptyset \mid COM \cup \{GCU\} \mid COM \cup \{\,COL\,\}$$

VIP: Syntax Examples

$\{\mathsf{GBU}\}$ is a (small) colony √

$\{\mathsf{GBU}_1, \mathsf{GCU}_1 \ldots \mathsf{GCU}_m\}$ is a colony √

$$\{\mathsf{GBU}_1, \mathsf{GCU}_1 \ldots \mathsf{GCU}_m, \overbrace{\{\mathsf{GBU}_2, \mathsf{GCU}_{m+1} \ldots \mathsf{GCU}_{m+n}\}}^{subcolony}\}$$

is a colony (it contains a subcolony) √

$$\{\mathsf{GBU}_1, \mathsf{GCU}_1 \ldots \mathsf{GCU}_m, \mathsf{GBU}_2, \mathsf{GCU}_{m+1} \ldots \mathsf{GCU}_{m+n}\}$$

is not a colony (two GBUs) √

$$\{\mathsf{GBU}_3, \overbrace{\{\mathsf{GBU}_1, \mathsf{GCU}_1 \ldots \mathsf{GCU}_m\}}^{subcolony}, \overbrace{\{\mathsf{GBU}_2, \mathsf{GCU}_{m+1} \ldots \mathsf{GCU}_{m+n}\}}^{subcolony}\}$$

is a colony (with two subcolonies) √

$$\{\overbrace{\{\mathsf{GBU}_1, \mathsf{GCU}_1 \ldots \mathsf{GCU}_m\}}^{subcolony}, \overbrace{\{\mathsf{GBU}_2, \mathsf{GCU}_{m+1} \ldots \mathsf{GCU}_{m+n}\}}^{subcolony}\}$$

is not a colony (no leader in the toplevel colony) but it is a community √

VIP: Formalizing Protocol (DCM 06, ENTCS)

Protocol formalization using the well know Natural Semantics Mathematical tool by Gilles Kahn, Gordon Plotkin, Robin Milner

A natural set of 13 deduction rules …

$$\frac{\begin{array}{c} discover(\mathsf{GCU}) = \mathsf{GBU} \\ samedom(\mathsf{GBU},\mathsf{GCU}) = true \quad gmode(\mathsf{GCU}) = true \\ accept(\mathsf{GBU},\mathsf{GCU}) = true \;\; regmode(\mathsf{GCU}) = false \end{array}}{\{\{\mathsf{GBU},\ldots\},\mathsf{GCU}\} \rightarrow \{\{\mathsf{GBU},\mathsf{GCU},\ldots\}\}} \;(\mathsf{JoinGCU})$$

Formal mathematical/mechanical proofs not only possible but easy.

Colonies in the Same Domain

Colony Registration

$$\frac{\begin{array}{c} \mathit{discover}(\mathsf{GBU}_2) = \mathsf{GBU}_1 \\ \mathit{samedom}(\mathsf{GBU}_1, \mathsf{GBU}_2) = \mathit{true} \qquad \mathit{gmode}(\mathsf{GBU}_2) = \mathit{true} \\ \mathit{accept}(\mathsf{GBU}_1, \mathsf{GBU}_2) = \mathit{true} \quad \mathit{regmode}(\mathsf{GBU}_2) = \mathit{false} \end{array}}{\{\{\mathsf{GBU}_1, \ldots\}, \{\mathsf{GBU}_2, \ldots\}\} \rightarrow \{\{\mathsf{GBU}_1, \{\mathsf{GBU}_2, \ldots\}, \ldots\}\}} \;(\mathsf{JoinCol})$$

Colony Unregistration

$$\frac{\begin{array}{c} \mathit{pendingip}(\mathsf{GBU}_2) = \mathit{false} \\ \mathit{samedom}(\mathsf{GBU}_1, \mathsf{GBU}_2) = \mathit{true} \qquad \mathit{gmode}(\mathsf{GBU}_2) = \mathit{false} \\ \mathit{accept}(\mathsf{GBU}_1, \mathsf{GBU}_2) = \mathit{false} \quad \mathit{regmode}(\mathsf{GBU}_2) = \mathit{true} \end{array}}{\{\{\mathsf{GBU}_1, \{\mathsf{GBU}_2, \ldots\}, \ldots\}\} \rightarrow \{\{\mathsf{GBU}_1, \ldots\}, \{\mathsf{GBU}_2, \ldots\}\}} \;(\mathsf{LeaveCol})$$

Colonies not in the Same Domain

Linking two Colonies

$$\frac{\begin{array}{rr} & gmode(\mathsf{GBU}_1) = true \\ newgbu(\mathsf{GBU}_1, \mathsf{GBU}_2) = \mathsf{GBU}_3 & gmode(\mathsf{GBU}_2) = true \\ samedom(\mathsf{GBU}_1, \mathsf{GBU}_2) = false & regmode(\mathsf{GBU}_1) = false \\ agree(\mathsf{GBU}_1, \mathsf{GBU}_2) = true & regmode(\mathsf{GBU}_2) = false \end{array}}{\{\{\mathsf{GBU}_1, \ldots\}, \{\mathsf{GBU}_2, \ldots\}\} \to \{\{\mathsf{GBU}_3, \{\mathsf{GBU}_1, \ldots\}, \{\mathsf{GBU}_2, \ldots\}\}\}} \text{(LinkCol)}$$

UnLinking two Colonies

$$\frac{\begin{array}{rr} pendingip(\mathsf{GBU}_1) = false & pendingip(\mathsf{GBU}_2) = false \\ pendingip(\mathsf{GBU}_3) = false & gmode(\mathsf{GBU}_1) = false \\ newgbu(\mathsf{GBU}_1, \mathsf{GBU}_2) = \mathsf{GBU}_3 & gmode(\mathsf{GBU}_2) = false \\ samedom(\mathsf{GBU}_1, \mathsf{GBU}_2) = false & regmode(\mathsf{GBU}_1) = true \\ agree(\mathsf{GBU}_1, \mathsf{GBU}_2) = false & regmode(\mathsf{GBU}_2) = true \end{array}}{\{\{\mathsf{GBU}_3, \{\mathsf{GBU}_1, \ldots\}, \{\mathsf{GBU}_2, \ldots\}\}\} \to \{\{\mathsf{GBU}_1, \ldots\}, \{\mathsf{GBU}_2, \ldots\}\}} \text{(UnLinkCol)}$$

Free Riders in Overlay Networks

*In economics and political science, **free riders** are actors who **consume more than their fair share of a resource, or shoulder less than a fair share of the costs of its production** …*

*The free rider problem is the question of how to **prevent free riding** from taking place, or at least limit its negative effects …*

*Because the notion of "**fairness**" is a subject of controversy, free riding is usually only considered to be an **"economic problem"** when it leads to the non-production or under- production of a public good, and thus to Pareto inefficiency, or when it leads to the excessive use of a common property resource.*

[From Wikipedia].

Arigatoni Rules to “Fire” Free Riders

$$\frac{\begin{array}{cc} \mathit{pendingip}(\mathsf{GCU}) = \mathit{false} & \mathit{gmode}(\mathsf{GCU}) = \mathit{true} \\ \mathit{samedom}(\mathsf{GBU}, \mathsf{GCU}) = \mathit{true} & \mathit{regmode}(\mathsf{GCU}) = \mathit{true} \\ \mathit{fairness}(\mathsf{GBU}, \mathsf{GCU}) \leq \epsilon & \mathit{notifiring}(\mathsf{GBU}, \mathsf{GCU}) \end{array}}{\{\{\mathsf{GBU}, \mathsf{GCU}, \ldots\}\} \rightarrow \{\{\mathsf{GBU}, \ldots\}, \mathsf{GCU}\}} \;(\mathsf{FireGCU})$$

$$\frac{\begin{array}{cc} \mathit{pendingip}(\mathsf{GBU}_2) = \mathit{false} & \mathit{gmode}(\mathsf{GBU}_2) = \mathit{true} \\ \mathit{samedom}(\mathsf{GBU}_1, \mathsf{GBU}_2) = \mathit{true} & \mathit{regmode}(\mathsf{GBU}_2) = \mathit{true} \\ \mathit{fairness}(\mathsf{GBU}_1, \mathsf{GBU}_2) \leq \epsilon & \mathit{notifiring}(\mathsf{GBU}_1, \mathsf{GBU}_2) \end{array}}{\{\{\mathsf{GBU}_1, \{\mathsf{GBU}_2, \ldots\}, \ldots\}\} \rightarrow \{\{\mathsf{GBU}_1, \ldots\}, \{\mathsf{GBU}_2, \ldots\}\}} \;(\mathsf{FireCol})$$

GIP: Evaluation via Simulation

Sample Topology

- GeorgiaTech-ITM (transit-stub)

103 GBs each with their local colony of 120 GC.

GCs acceptation rate = 75%

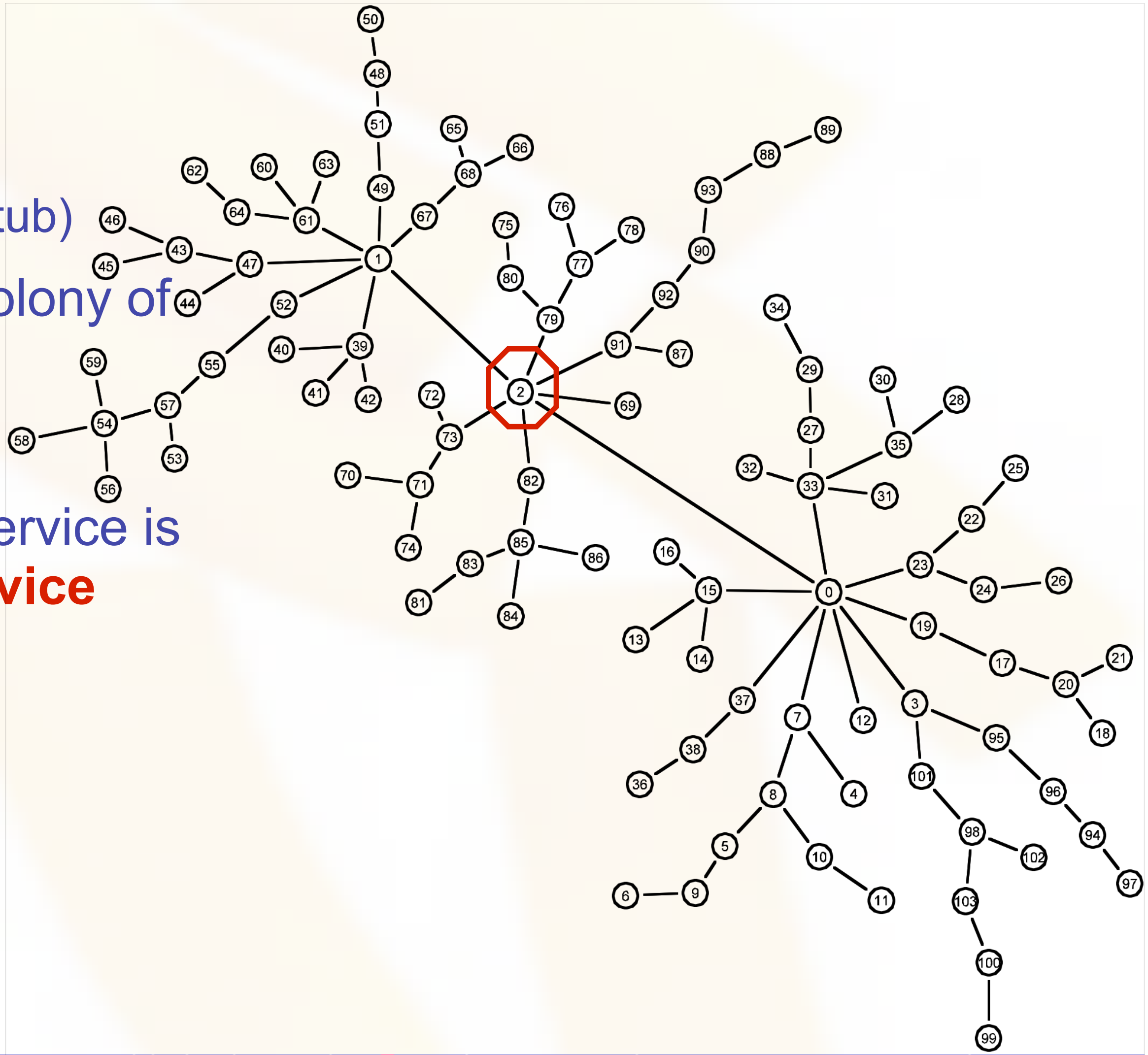
Let **ρ** be the probability that a service is provided in a local colony = **service availability**

L%= Match resources

I = Instance numbers (>1)

Evaluated

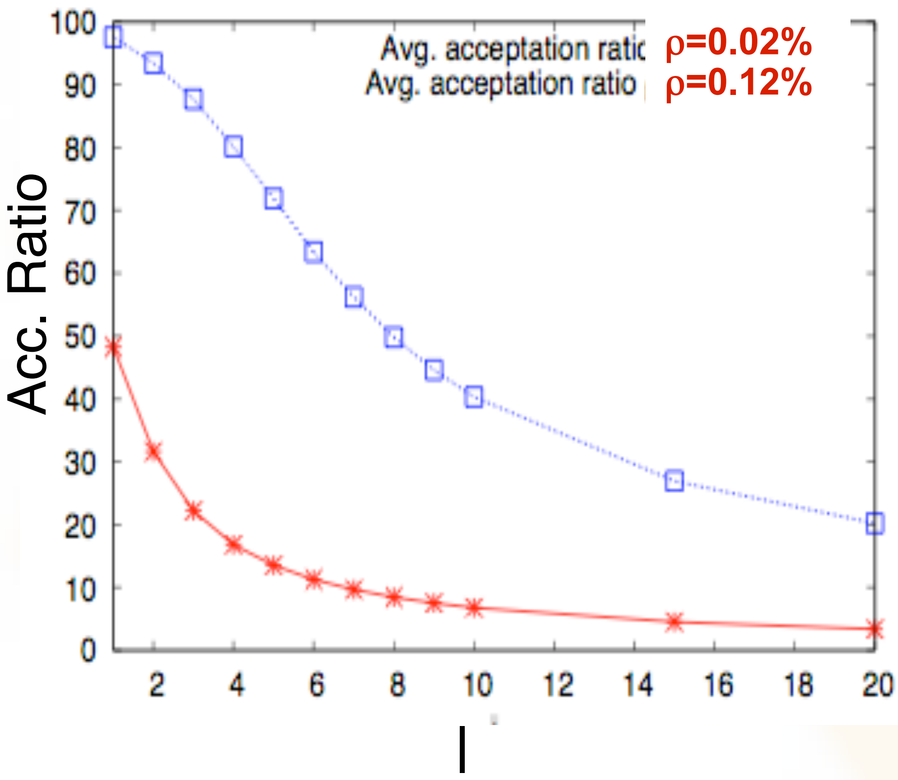
- Scalability in terms of load
- Acceptation ratio



GIP Evaluation: Load

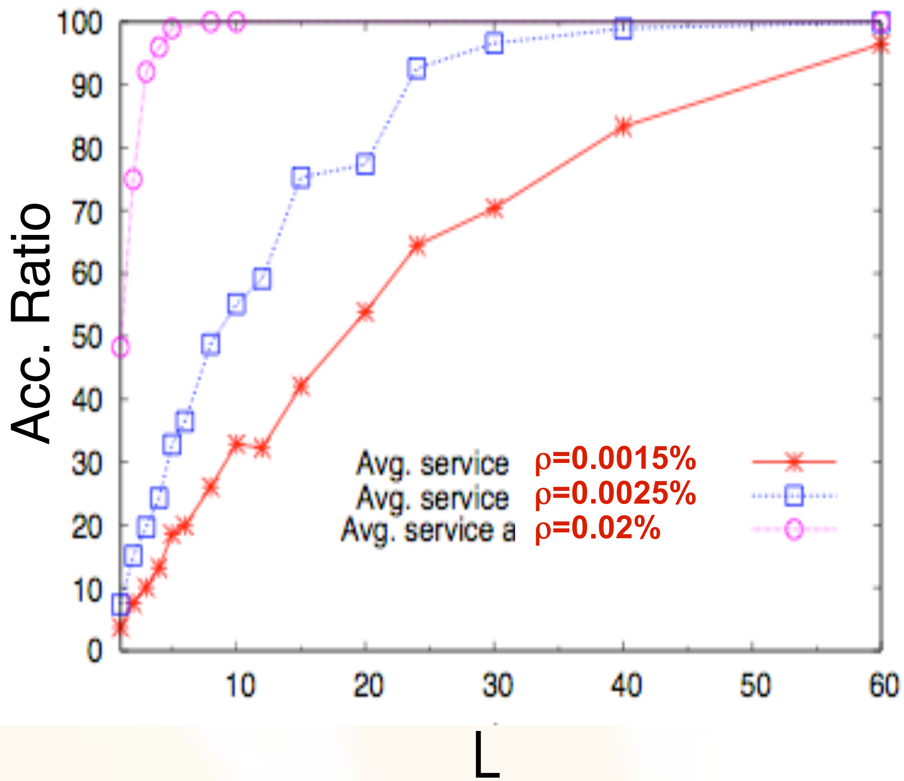
Load more evenly distributed

- **Super Brokers work less**
- **Sub Brokers work more**



Scalability: Every GBs receive ~3.5% of requests (in average)

No Super Broker overload

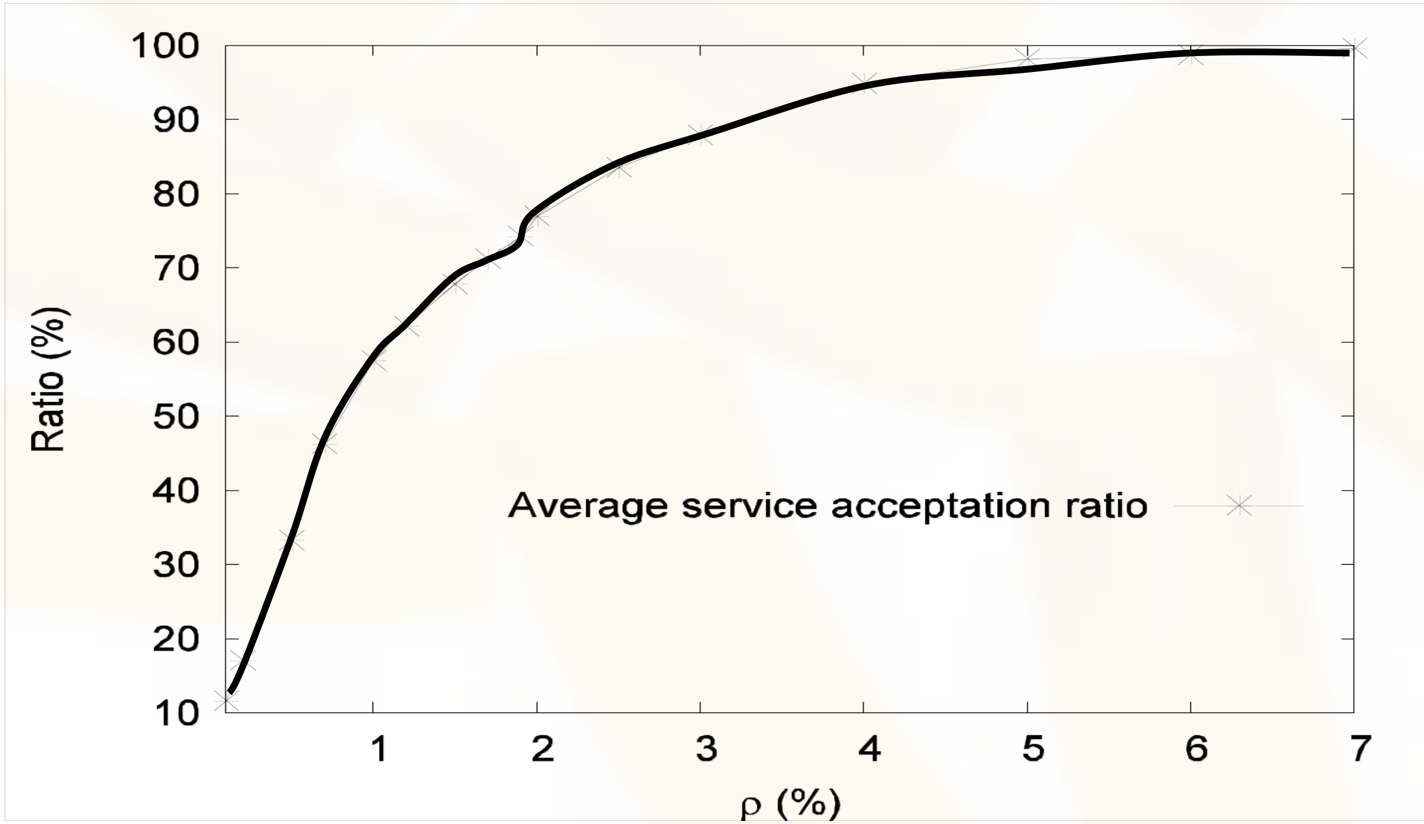


Université Nice Sophia Antipolis
CNRS Centre National de la Recherche Scientifique
INRIA

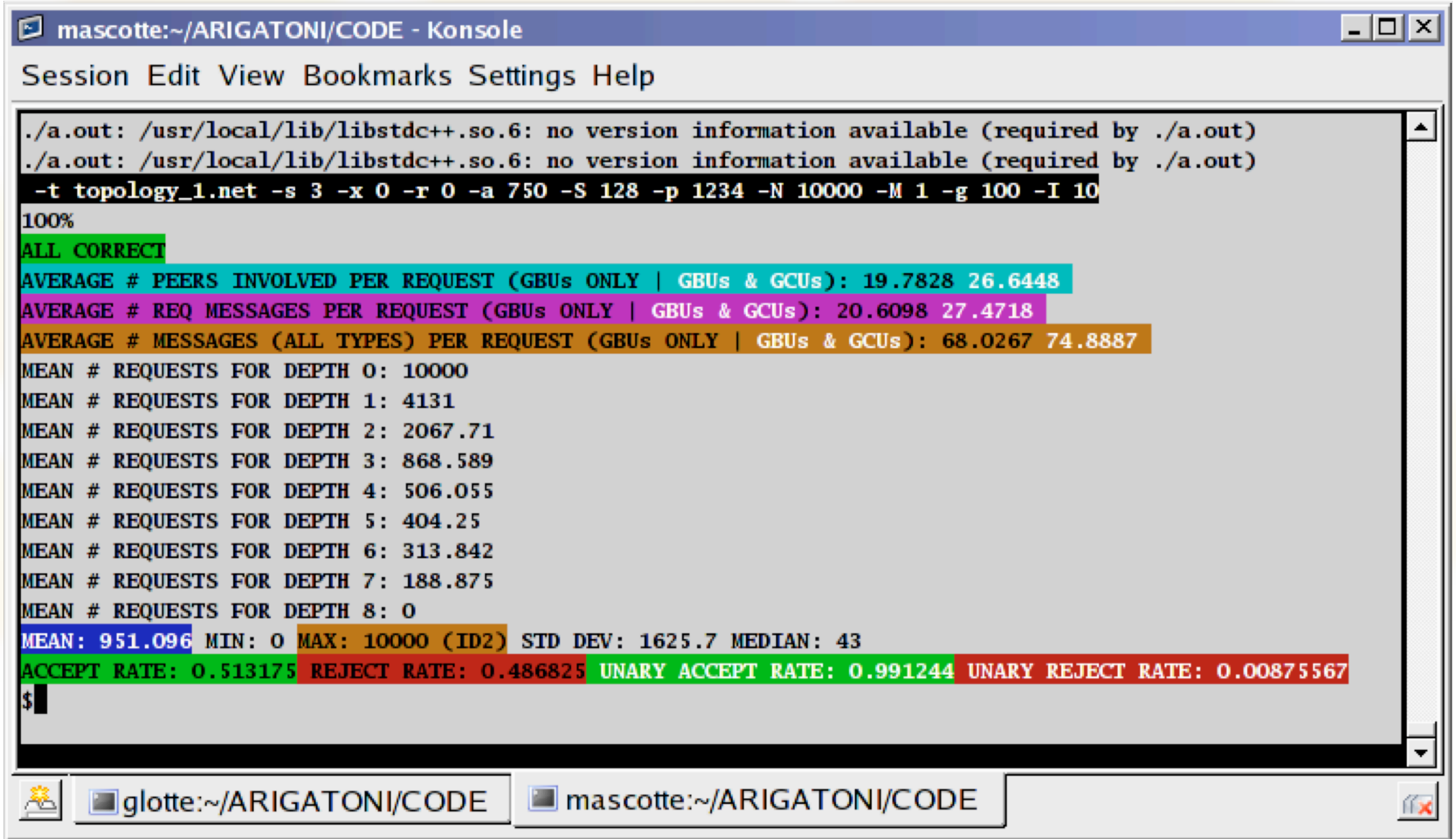
GIP Evaluation: Success Rate

Success rate depends on availability of services ρ

Efficiency: **90% for ρ = 4%**



GIP & VIP: Simulator



Overlays maybe useful for V2V and V2I?

- V2I Scenario:
  - A vehicle can play the role of a Global Computer
  - A «bus-stop» station equipped with network, routing table and computational features can play the role of Global Broker.
- V2V Scenario:
  - A « bus » or a « cab » or a « police/samu car » can play the role of Global Broker
  - Any other « car », equipped with connection and computation facilities, can play the role of a Global Computer
- Some scenarios will be proposed, simulated and possibly tested *in vitro*

In Fine …

Arigatoni: Lightweight Overlay Network for Dynamic Resource Discovery.
Main Achievements

- *Programmability, Intermittence, Generality, Asynchronicity, Scalability*

Current, future work

- More functionalities
- From n-layer tree to n-layer graph topology
- Mathematical model of the overlay (with Philippe Nain and Michel Cosnard)
- Design of an Orchestration language to deal with the P2P phase
- Proto-Implementation and Deployment

Conclusions

1. From **Physical**, Real, Proprietary, Personal, more or less expensive, programmable, **Turing Complete**, Computers …
2. … to **Ethereal**,Virtual, Public, Impersonal, Ubiquitous, not expensive, programmable, **Turing Complete**, Overlay Computers
3. Looks a bit like a **Capitalist vs. Communist Social Model** ***"querelle"***, but we can also pay to be logged inside a *"rich"* colony
4. I can conjecture, in the next 2-3 decades, at least 2 categories of Overlay Computers:
   - Not-free, **Closed Overlay Networks**, made by semantically powerful Overlay Computers offering services in change of money
   - Free, **Open Overlay Networks**, made by semantically powerful Overlay Computers offering services for free

Conclusions II

- **[John Gustafson's Paper on History of Computing]**
  - Linear Equations Automatic Solvers motivates computing innovation
  - The quest for Linear Solvers helps the development of Matrices, Negative Numbers, Loops, Binary Arithmetic, LINPACK Open-source,...
- **Research on Network Computing, Overlay Computing, Overlay Networks will be motivated by**
  - Solving Linear Equations, Bio-Challenges, Cracking RSA codes, Physical simulation, Quantum computation simulation,
  - Finding life in Mars, Solving Sudokus
  - Running other more or less serious & useful algorithms,
  - Last but not least (**sure, John Von Neumann would like!**)

Yes, Downloading the Last Madonna's tube!

Real Example in a Grid Arena