Implementation of a prototype and deployement on a platform of the Overlay Network Arigatoni

Luigi Liquori, Projet Mascotte

STAGE ENSL

1 Localisation

- **Projet(s) de recherche :** Mascotte
- **Responsable du dossier :** Luigi.Liquori@sophia.inria.fr
- **Correspondant scientifique du dossier :** Luigi.Liquori@sophia.inria.fr
- **Intitulé du dossier :** Expérimentation sur échelle de l'Overlay Network Arigatoni

2 Abstract

This work represents the first light-weight overlay network called Arigatoni[1] that is suitable to deploy, via the Internet the *Global Computing Communication Paradigm*, *i.e.*, computation via a seamless, geographically distributed, open-ended network of bounded resources owned by agents (called *Global Computers*) acting with partial knowledge and no central coordination. The paradigm provides uniform services with variable guarantees. Aggregating many Global Computers sharing similar or different resources leads to a *Virtual Organization*, sometimes called *Overlay Computer*. Finally, organizing many Overlay Computers, using, *e.g.* tree- or graph-based topology leads to an *Overlay Network*, *i.e.* the possibility of programming a *collaborative Global Internet* over the *plain Internet*.

The main challenge in this research field is how single resources, offered by the Global/Overlay Computers are discovered. The process is often called *Resource Discovery*: it requires an *up-to-date* information about widely-distributed resources. This is a challenging problem for large distributed systems when taking into account the continuously changing state of resources offered by Global/Overlay Computers and the possibility of tolerating intermittent participation and dynamically changing status/availability of the latter.

Entities in Arigatoni are organized in *Colonies*. A colony is a simple virtual organization composed by exactly one *Leader*, offering some broker-like services, and some set of *Individuals*. Individuals are Global Computers (think it as an *Amoeba*), or subcolonies (think it as a *Protozoa*). Global Computers communicate by first registering to the colony and then by mutually asking and offering services. The leader, called *Global Broker* analyzes service requests/responses, coming from its own colony or arriving from a surrounding colony, and routes requests/responses to other individuals. After this discovery phase, individuals get in touch with each other without any further intervention from the system, in a P2P fashion.

Symmetrically, the leader of a colony can arbitrarily unregister an individual from its colony, *e.g.*, because of its bad performance when dealing with some requests, or because of its high number of "embarrassing" requests for the colony. This mechanism/strategy reminiscent of the Roman *"do ut des"*, is nowadays called, in Game Theory, *"tit-for-tat"* [6]. This strategy is commonly used in economics, social sciences, and it has been implemented by a computer program as a winning

[1]The Arigatoni model, protocol and middleware, is copyrighted by Luigi Liquori (INRIA) under the CECIL License.

strategy in a chess-play challenge against humans (see also the well known *prisoner dilemma*). In computer science, the tit-for-tat strategy is the main principle of Bittorrent P2P protocol [2]. Once a Global Computer has issued a request for some services, the system finds some Global Computers (or, recursively, some subcolonies) that can offer the resources needed, and communicates their identities to the (client) Global Computer as soon as they are found.

The model also offers some mechanisms to dynamically adapt to *dynamic topology changes* of the Overlay Network, by allowing an individual (Global Computer or subcolony) to log/delog in/from a colony. This essentially means that the process of routing request/responses may lead to failure, because some individuals delogged or because they are temporarily unavailable (recall that Individuals are not *slaves*) [5]. This may also lead to temporarily *denials of service* or, more drastically, to the complete delogging of an individual from a given colony in the case where the former does not provide enough services to the latter. We have designed an operational semantics (Intermittence semantics) via a labeled transition system, that describes the main operations necessary in the Arigatoni model to perform leader negotiation, joining/leaving a colony, linking two colonies and moving one GC from one colony to another. Our formalization results to be adequate w.r.t. the algorithm performing peer logging/delogging and colony aggregation.

Indeed, dealing only with Resource Discovery has one important advantage: the complete generality and independence of any given requested resource. Arigatoni can fit with various scenarios in the Global Computing arena, from classical P2P applications, like file- or band-sharing, to more sophisticated Grid applications, like remote and distributed big (and small) computations, until possible, futuristic *migration computations*, *i.e.* transfer of a non completed local run in another GC, the latter scenario being useful in case of catastrophic scenarios, like fire, terrorist attack, earthquake, etc., in the vein of Global Programming Languages *à la* Obliq [3] or Telescript [8].

The main ingredients of Arigatoni are one protocol, the *Global Internet Protocol,* GIP, and three main units:

• A *Global Computer Unit,* GC, *i.e.* the basic peer of the Global Computer paradigm; typically it is a small device, like a PDA, a laptop or a PC, connected via IP.

• A *Global Broker Unit,* GB, is the basic unit devoted to register and unregister GCs, to receive service queries from client GCs, to contact potential servants GCs, to negotiate with the latter the given services, to trust clients and servers, and to send all the informations useful to allow the client GC, and the servants GCs to be able to communicate. Every GC can register to *only one* GB, so that every GB controls a *colony* of collaborating Global Computers. Hence, communication intra-colony is initiated via only one GB, while communication inter-colonies is initiated through a chain of GB-2-GB message exchanges. In both cases, when a client GC receives an acknowledgment for a request service (with related trust certificate) from the leader GB, then the client enjoys the service directly from the servant(s) GC, *i.e.* without a further mediation of the GB itself.

• A *Global Router Unit,* GR, is a simple basic unit that is devoted to send and receive packets of the Global Internet Protocol GIP and to forward the payload to the units which is connected with this router. Every GC and every GB have one personal GR. The connection between router and peer is ensured via suitable API.

Figure 1 shows the Arigatoni Overlay Network.

Effective use of computational grids via Overlay Networks requires *up-to-date* information about widely-distributed resources. This is a challenging problem for very large distributed systems particularly taking into account the continuously changing state of the resources. Discovering dynamic resources must be scalable in number of resources and users and hence, as much as possible, fully decentralized. It should tolerate intermittent participation and dynamically changing status/availability.

The Arigatoni overlay network is, by construction, independent from any given resource request. We could envisage at least the following scenarios to be completely full-fitted in our model (list not exhaustive)

• Ask for computational power (*i.e.* the Grid).
• Ask for memory space.
• Ask for bandwidth (*i.e.* VoIP).
• Ask for file retrieving (*i.e.* P2P).

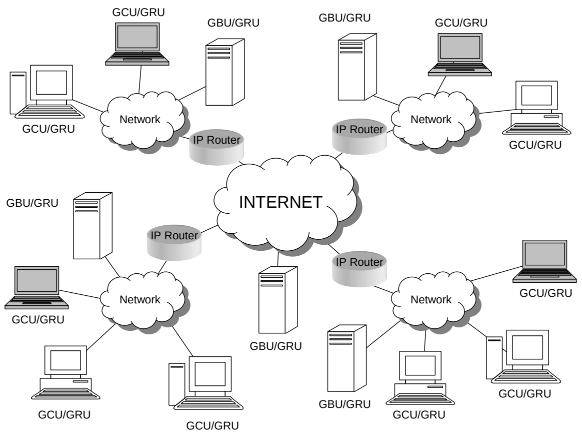


Figure 1: ArigatoNet

• Ask for web service (*i.e.* Google).
• Ask for a computation migration (*i.e.* transfer one partial run in another GC saving the partial results, as in a truly mobile ubiquitous computations).
• Ask for a *Human Computer Interaction* ...

Our paper tries to fill some of the objectives fixed in the seminal paper of [7], where the requirements and the resource management for future generation Grids are discussed. More generally, Arigatoni is *parametric* in a given application, or *universal* in the sense of Universal Turing Machine, or *generic* as the Von Neumann Computer Model. Summarizing, the original contributions of the paper are:
• A simple distributed communication model that is suitable to make Resource Discovery transparent.
• A Global Internet Protocol that allows Global Computers to negotiate resources.
• A *complete independence* of the classical scenarios of the arena, *i.e.* Grid, file/band sharing, web services, etc. This domain independence is a key feature of the model and of the protocol, since it allows the Overlay Network to be programmable.

We hope that Arigatoni could represent a little step toward a natural integration of different scenarios under the common paradigm of Global Computing.

To assess the effectiveness and the scalability of our Resource Discovery mechanisms, we have conducted simulations using large numbers of units and service requests.

Simulation Setup. We have generated a network topology using the transit-stub model of the Georgia Tech Internetwork Topology Models package [9], on top of which we added the Arigatoni Overlay Network. The resulting network topology, shown in Figure 2, contains 103 GBs. GB_2 (highlighted with a square in Figure 2) was chosen as the root of the topology. GCs were not directly simulated in the network topology. Instead, to simulate the population of GCs, we added a GC *agent* to each GB in the system. The GC agent of a GB represents the local Colony of GCs that are attached to that GB as their leader.

We considered a finite set of resources $R_1 \cdots R_r$ of variable size r, and represented a service by a direct mapping to a resource. In other words, a service expresses the conditional presence of a single resource. We have a set of r services $\{S_1 \cdots S_r\}$, where service S_i expresses the conditional presence of resource R_i. A GC declaring service S_i means that it can provide resource R_i. This

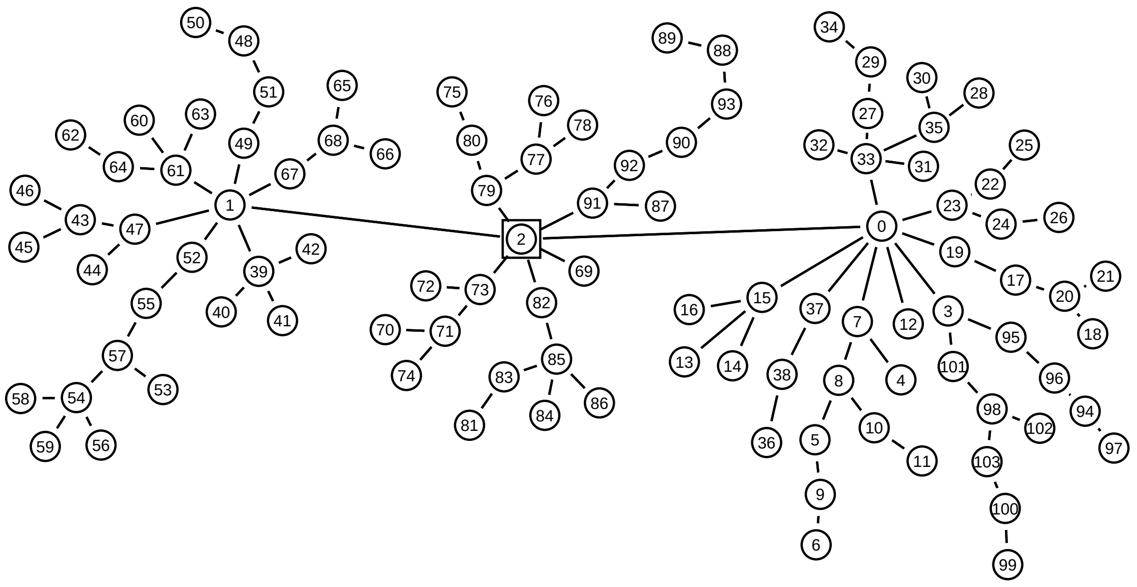
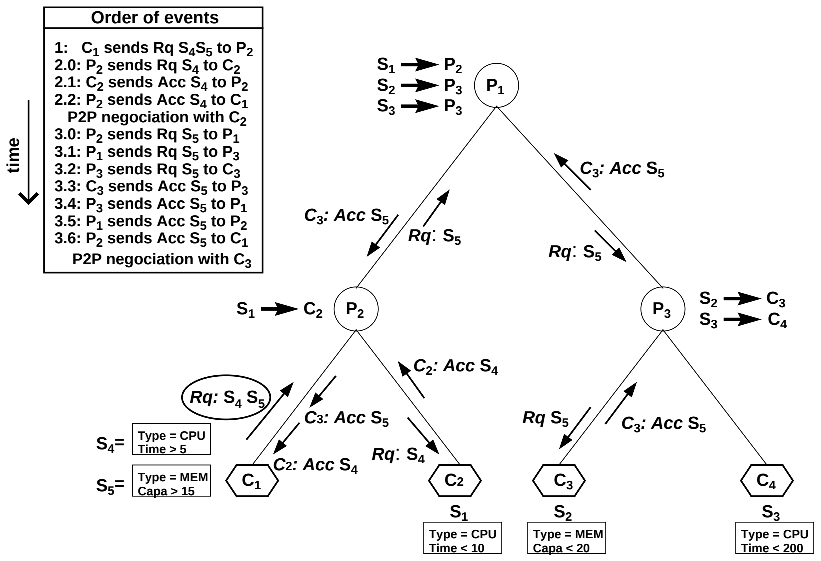


Figure 2: Simulated network topology with 103 GBs



Order of events

1: C_1 sends Rq S_4S_5 to P_2
2.0: P_2 sends Rq S_4 to C_2
2.1: C_2 sends Acc S_4 to P_2
2.2: P_2 sends Acc S_4 to C_1
P2P negociation with C_2
3.0: P_2 sends Rq S_5 to P_1
3.1: P_1 sends Rq S_5 to P_3
3.2: P_3 sends Rq S_5 to C_3
3.3: C_3 sends Acc S_5 to P_3
3.4: P_3 sends Acc S_5 to P_1
3.5: P_1 sends Acc S_5 to P_2
3.6: P_2 sends Acc S_5 to C_1
P2P negociation with C_3

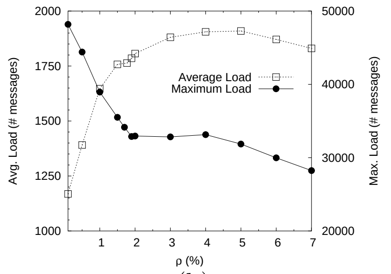
Figure 3: Resource discovery scenario.

model, while quite simple, is still generic enough, and is sufficient for the main purpose of our experiments, which is to study the scalability of Resource Discovery in our system.

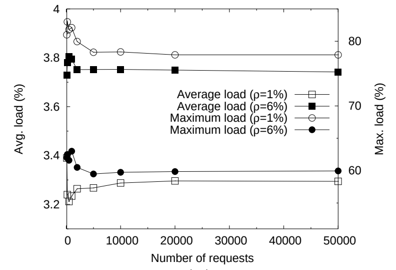
To simulate GC load, we then randomly added each service with probability ρ at each GC agent, and had it registered via the registration service of Arigatoni. The routing tables of the GBs were updated starting at the initial GB and ending at the root of the topology, GB_2. In other words, it is as if each GB has a probability ρ of having a GC which registered service S_i, for any S_i. Thus, the parameter ρ can be seen as either the global availability of services, or as the density of population of GCs (since the more the number of GCs, the more likely it is that a given service is provided).

Vars	Description	Value
K	Number of GBs	103
r	Size of services pool	128
ρ	Service availability	0.1% to 7%
α	Service accept. prob.	75%
n	Number of `SREQ` issued	100 to 50000

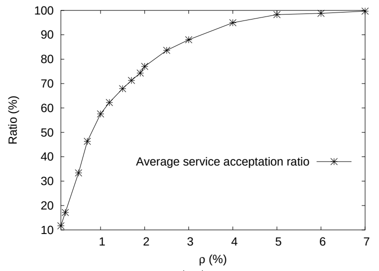
(a)



(b)



(c)



(d)

Figure 4: (a) Parameters of the experiments. (b) Average and maximum load w.r.t. service availability ρ. (c) Average and maximum load fraction w.r.t. the number of requests issued. (d) Average service acceptation ratio w.r.t. service availability ρ.

We then issued n service requests at GC agents chosen uniformly at random. Each request contained one service, also chosen uniformly at random. Each service request was then handled by the Resource Discovery mechanism of Arigatoni. We used a service acceptation probability of $\alpha = 75\%$, which corresponds to the probability that a GC that receives a service request *and* that declared itself as a potential Individual for that service (*i.e.* that registered it), accepts to serve it.

The Resource Discovery algorithm was implemented in C++ and compiled using GNU C++ version 2.95.3. Experiments were conducted on a 3.0 Ghz Intel Pentium machine with 2 GB of main memory running Linux 2.4.28. The different experimental parameters are summarized in Figure 4(a) Upon completion of the n requests, we measured for each GB its load as the number of requests (messages) that it received. We then computed the average load as the average value over the population of GBs in the system. We also computed the maximum load as the maximum value of the load over all the GB s in the system. Similarly, we computed the average and maximum load fractions as the average and max loads divided by the number of requests. The average load represents the average load of a GB due to the completion of the n requests. The average load fraction represents the fraction of requests that a GB served, in average. The maximum fraction represents the maximum fraction of the requests that a GB served. Note that since a GB receives at most one request message corresponding to a given service request, the average load fraction can be seen as the fraction of GBs in the system involved in a service request, in average.

Finally, we computed the average service acceptation ratio as follows. For each GC agent, we computed the local acceptation ratio as the number of service requests that yielded a positive response (*i.e.* the system found at least one Individual), over the number of service requests issued at that GC agent. We then computed the average acceptation ratio as the average value over the number of GC agents (that issued at least one service request).

We repeated the experiments for different values of ρ and n. Results are illustrated in Figure 4. Figure 4(b) and (d) were obtained with a fixed value of n of 50000 service requests.

Results and Interpretations. Figure 4(b) shows the evolution of the average and maximum load when varying the service availability ρ. The maximum load was obtained for GB_2 or GB_0, that are both very low-depth GBs in the tree topology. It appears that the maximum load decreases with the service availability, while the average load increases. In other words, the load is more evenly distributed amongst the GBs in the system. This is due to the strategy of our Resource Discovery mechanism which consists in always searching for Individuals in its own Colony first before delegating to its leader. Indeed, as the service availability increases, GBs have a higher chance to find Individuals in their own Colony. Hence, GBs of high-depth in the topology participate more in the process of Resource Discovery, and GBs of low-depth participate less. In other words, the Resource Discovery mechanism used in Arigatoni does not overload low-depth GBs in the tree topology.

We observe for values of $2\% \leq \rho \leq 4\%$, a *plateau* in the curve of the maximum load, followed by a decreasing phase, but with a much lower slope than before ($\rho < 2\%$). This is due to the fact that for $\rho < 2\%$, GB_2 has the maximum load in the system. For $\rho > 2\%$, however, GB_0 takes over. This transition can be explained by the fact that for higher values of ρ, less messages are delegated to GB_2. At some point ($\rho \sim 2\%$), the load of GB_2 becomes less important than that of GB_0, due to the high number of colonies that the latter manages. The constantness observed in the curve around that value is probably due to the fact that a transition phase is necessary for GB_0 to be sensitive again to the increase of ρ. The following decreasing period with a lower slope corresponds to the fact that GB_0 is less sensitive to an increase of ρ (indeed, GB_0 is mostly concerned with the availability of services in its own colonies).

Finally, we observe that the average load stabilizes, which shows that the system scales to large number of GCs (since as previously mentioned, the service availability ρ can be assimilated to the number of GCs in the system).

Figure 4(c) shows the average and maximum load fractions w.r.t. the number of service requests. It appears clearly that Arigatoni scales to large numbers of requests. In fact, the average number of requests received by a GB increases linearly with the total number of requests, at a rate of $\sim 3.5\%$. In other words, in average, a GB only receives $\sim 3.5\%$ of the total number of requests. Equivalently, only 3.5% of the overall population of GBs in the system participate in the process of discovering a particular resource, in average. Figure 4(c) also shows that low level GBs in the topology are not particularly overloaded (the most overloaded GB manages 60% of the overall load for $\rho = 6\%$). Finally, it corroborates the assertion that higher values of ρ favor the maximum load over the average load, *i.e.*, load balancing gets more effective.

Figure 4(d) shows that, unsurprisingly, the average service acceptation ratio increases exponentially with the availability of services. This shows that Arigatoni is efficient in searching Individuals for requested services. Indeed, a service availability of 4% enables the system to achieve an acceptation rate of 90%. In other words, the more the number of GCs in the system, the more chances to find an Individual for a service request.

- **Description précise de l'intervention souhaitée (développement, support, encadrement, intégration, expérimentation, etc.) :**

  A solid simulator of the resource discovery algorithm has been written, in C++, by Raphael Chand (about 20K l-code). We demand to wrap this algorithm in a small Unix client that should be embedded in any GB. This would demand to completely implement the part relative to the GIP protocol and implement, via sockets or datagram the GIP packet exchange between GC and GR. The GR API and the GC client that send and receive GIP packets must be implemented. In other words, we would like to implement a real prototype of the Arigatoni Overlay Network and the subsequent deployment on any experimental platform, like the Sophia intranet or PlanetLab, or GRID5000.

- **Si encadrement (nombre et fonctions des personnes à encadrer) :** pas de personnes à encadrer.

- **Compétences requises :** C++, Programmation par réseaux, concéption de protocoles reseaux, TCP/IP, Unix shells, Clusters, Grid, P2P.

- **Planning de déroulement (sous-tâches, durées) :**

- Heating up (0.5 Month)
- Normalization of the GIP protocol composed of the resource discovery algorithm and of the intermittence algorithm. In this part all the published papers [4, 1, 5] will be collected and *normalized* in a single "RFC"-like document. Some parts of the protocol, such as the choice of a suitable format of the packets (e.g. the ASN.1 types), still need to be worked on. The final objective of the paper is to put together all the already written parts of the GIP protocol in a document that will be suitable for publication. At this stage we do not exclude that some previous choices will be taken into account and discussed again with the new DREAM's view. (3 Months)
- By taking advantage of the previous phase, we would like to implement an operational prototype of the Arigatoni system. For that purpose, one will have to code three pieces of UNIX software: a GC client, a GR API, and a GB client/. Subsequently, we aim at deploying the Arigatoni system on a large scale, experimental platform such as the Planetlab or the GRID 5000 testbed (maybe the simple cluster at Sophia in a first place). The goal is to experiment with the conditions of the real Internet, *i.e.*, to analyze the behavior and to assess the performance of our system in realistic settings.
  - Refactoring of the existing code available in the simulator (discovery algorithm and intermittence algorithm) as an external library called by the GCU/GBU clients. (1 month)
  - Implementation of the GRU API and network part (GIP packet exchanges between peers) of GCU/GBU (1.5 Month)
  - Implementation a performance measurement API (0.5 month)
  - Implementation of a simple GCU as an interactive shell, with 2 or 3 combined resources (1.5 Month)

  (Total : 4.5 Months)
- Evaluation of the protocol; When our system has been deployed on a large scale experimental platform, we will analyze its behavior and efficiency. For that purpose, we will work out an elaborate experimental protocol so as to collect several performance measures. Those include the measures carried out using the Arigatoni simulator: *Average and maximum load w.r.t. service availability ρ, *average and maximum load fraction w.r.t. the number of requests issued, * average service acceptation ratio w.r.t. service availability, as well as several new measures that depend on the physical platform on which the Arigatoni system has been deployed: *Average and maximum service delay, *Average, maximum throughput achieved under a particular load (service requests issued at Poisson rate at GCUs). (1 Month)

  Finally, we would like to instantiate a small demo of the system with some real, meaningful, requests (using a small number of combined resources, such as CPU, Memory, Data, etc). (1 Month)

  (Total : 2 Months)

- **Dure'e de la mise a' disposition souhaite'e :** 10 Months.

References

[1] D. Benza, M. Cosnard, L. Liquori, and M. Vesin. Arigatoni: Overlaying Internet via Low Level Network Protocols. Technical Report 5805, INRIA, 2006.

[2] BitTorrent, Inc. The Bittorrent Home Page. `http://www.bittorrent.com/`.

[3] L. Cardelli. A language with distributed scope. *Computing Systems*, 8(1):27–59, 1995.

[4] R. Chand, M. Cosnard, and L. Liquori. Resource Discovery in the Arigatoni Model. Technical Report 5924, INRIA, 2006.

[5] M. Cosnard, L. Liquori, and R. Chand. Virtual Organizations in Arigatoni. *DCM: International Workshop on Developpment in Computational Models. Electr. Notes Theor. Comput. Sci.*, 2006. To appear.

[6] A. Rapoport. Mathematical models of social interaction. In *Handbook of Mathematical Psychology*, volume II, pages 493–579. John Wiley and Sons, 1963.

[7] U. Schwiegelshohn, R. Yahyapour, and P. Wieder. Resource Management for Future Generation Grids. Technical Report TR-0005, CoreGRID, 2005.

[8] J.E. White. Telescript technology: the foundation for the electronic marketplace. White Paper. General Magic, Inc., 1994.

[9] E.W. Zegura, K. Calvert, and S. Bhattacharjee. How to Model an Internetwork. In *Proc. of INFOCOM*, 1996.