University of Castilla-La Mancha

A publication of the

Computing Systems Department

Network-aware Peer-to-Peer Based Grid Inter-Domain Scheduling

by

Agustín Caminero, Omer Rana, Blanca Caminero, Carmen Carrión

Technical Report **#DIAB-08-01-1** January, 2008

This work has been jointly supported by the Spanish MEC and European Commission FEDER funds under grants “Consolider Ingenio-2010 CSD2006-00046” and “TIN2006-15516-C04-02”; by JCCM under grants PBC-05-007-01, PBC-05-005-01 and José Castillejo.

DEPARTAMENTO DE SISTEMAS INFORMÁTICOS
ESCUELA POLITÉCNICA SUPERIOR
UNIVERSIDAD DE CASTILLA-LA MANCHA
Campus Universitario s/n
Albacete - 02071 - Spain
Phone +34.967.599200, Fax +34.967.599224

Network-aware Peer-to-Peer Based Grid Inter-Domain Scheduling

Agustín Caminero[1], Omer Rana[2], Blanca Caminero[1], Carmen Carrión[1]
[1]Computing Systems Department. Escuela Politécnica Superior
Universidad de Castilla-La Mancha. 02071 - Albacete, SPAIN
{agustin, blanca, carmen}@dsi.uclm.es
[2]Cardiff School of Computer Science.
5 The Parade, Cardiff. CF24 3AA. UK.
o.f.rana@cs.cardiff.ac.uk

March 19, 2008

Abstract

Grid technologies have enabled the aggregation of geographically distributed resources, in the context of a particular application. The network remains an important requirement for any Grid application, as entities involved in a Grid system (such as users, services, and data) need to communicate with each other over a network. The performance of the network must therefore be considered when carrying out tasks such as scheduling, migration or monitoring of jobs. Moreover, the interactions between different domains are a key issue in Grid computing, thus their effects should be considered when performing the scheduling task. In this paper, we enhance an existing framework that provides scheduling of jobs to computing resources to allow multi-domain scheduling based on peer-to-peer techniques.

1 Introduction

Grid computing enables the aggregation of dispersed heterogeneous resources for supporting large-scale parallel applications in science, engineering and commerce [10]. Current Grid systems are highly variable environments, made of a series of independent organizations that share their resources, creating what is known as *Virtual Organizations* (VOs) [11]. This variability makes Quality of Service (*QoS*) highly desirable, though often very difficult to achieve in practice [21]. One of the reasons for this limitation is the lack of control over the network that connects various components of a Grid system. Achieving an *end-to-end* QoS is often difficult, as without resource reservation any guarantees on QoS are often hard to satisfy. However, for applications that need a timely response (such as collaborative visualization [13]), the Grid must provide users with some kind of assurance about the use of resources – a non-trivial subject when viewed in the context of network QoS [17]. In a VO, entities communicate with each other using an interconnection network – resulting in the network playing an essential role in Grid systems [21].

As a VO is made of different organizations (or domains), the interactions between different domains are a key issue in Grid computing [27]. Users from a domain may have to interact with data sources, computing resources or information services, among others, which are located in a different administrative domain. Also, jobs belonging to a user may need to be executed in a computing resource from a different administrative domain. This situation is depicted in Figure 1, and shows a user who wants to run a job, which has a number of QoS requirements, such as execution time or response time. This user will contact a resource broker in order to get a computing resource fulfilling those requirements to run his job. If none of the computing resources in the local domain fulfill the job's requirements, another computing resource from another administrative domain should be allocated to run this job.

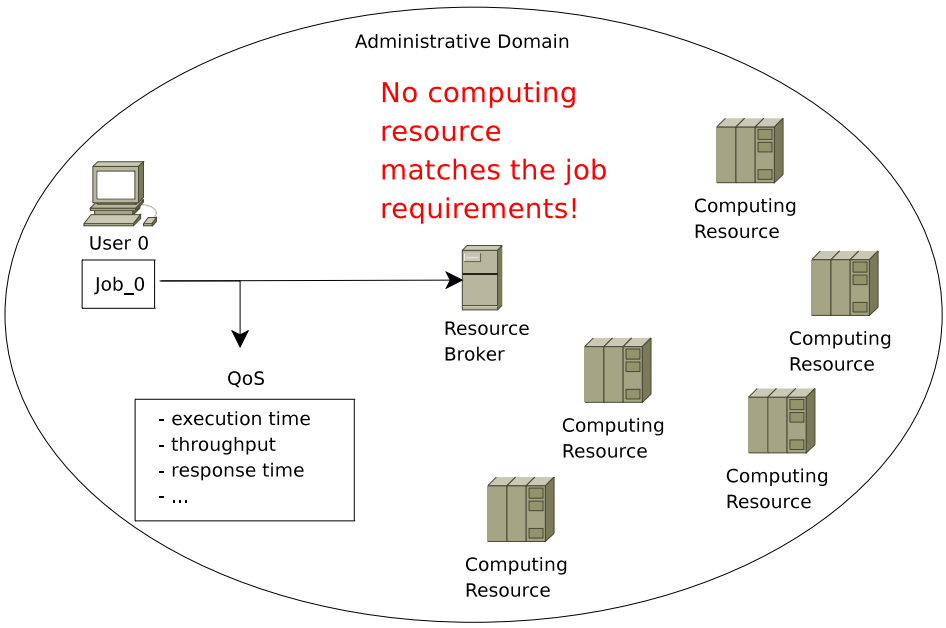


Figure 1: Match-making between job requirements and computing resources.

Our main aim is the provision of network QoS, and we try to achieve that by means of taking network into account when performing scheduling of jobs to computing resources. As mentioned above, if there is no suitable resource in the user's domain, a resource from a different domain may be chosen to run this job, so the connections between domains should be considered when performing the scheduling task (see Figure 2). The way how we will achieve this is by means of a framework based on peer-to-peer techniques.

The paper is structured as follows: Section 2 explains current proposals on network QoS in Grid, and the lack of attention they have paid to the inter-domain scheduling. Also, existing proposals for inter-domain scheduling are revised. Section 3 explains our proposal of inter-domain scheduling. Section 4 provides an evaluation, demonstrating the usefulness of our work, and Section 5 shows some guidelines for our future work.

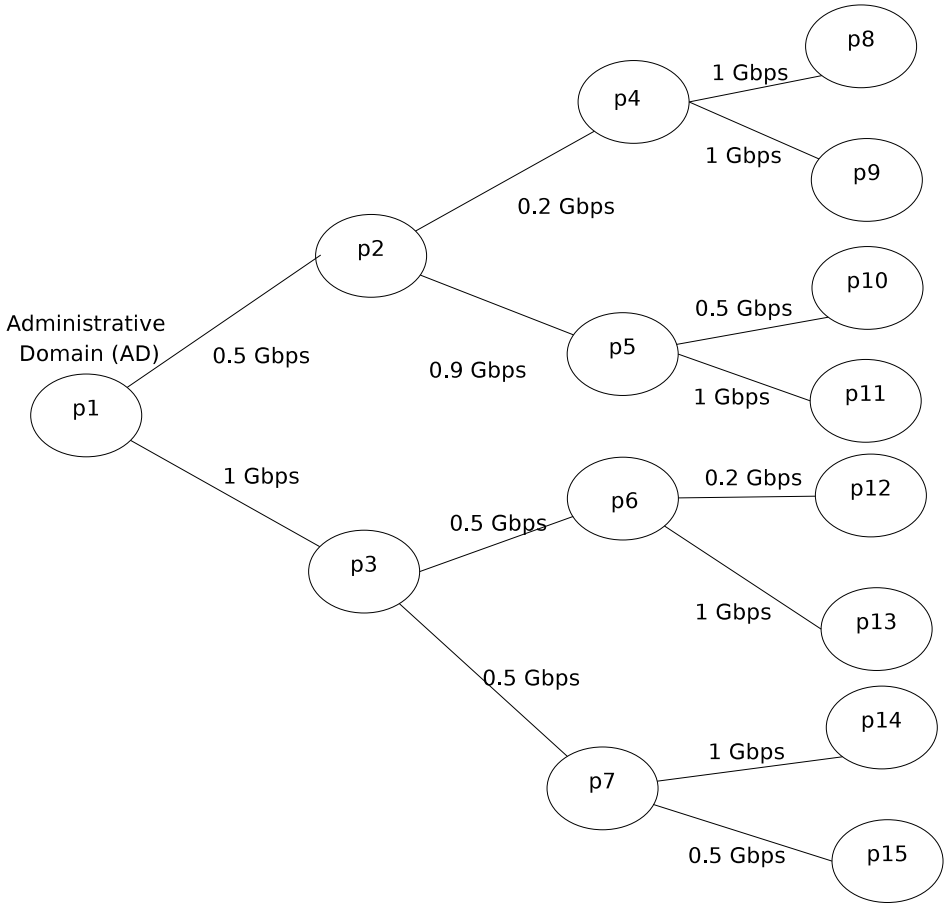


Figure 2: Several administrative domains.

2 Related work

The architecture we propose in this paper is intended to manage network QoS in a Grid system, and it is specially concerned with the interactions between administrative domains when performing the scheduling of jobs to computing resources. The way that our approach tackles the inter-domain relations is by applying peer-to-peer ideas in order to decide to which neighbor domain a query should be forwarded, in the case that there is no available resources in the current domain. Thus, we will provide an insight into existing proposals for network QoS in Grid.

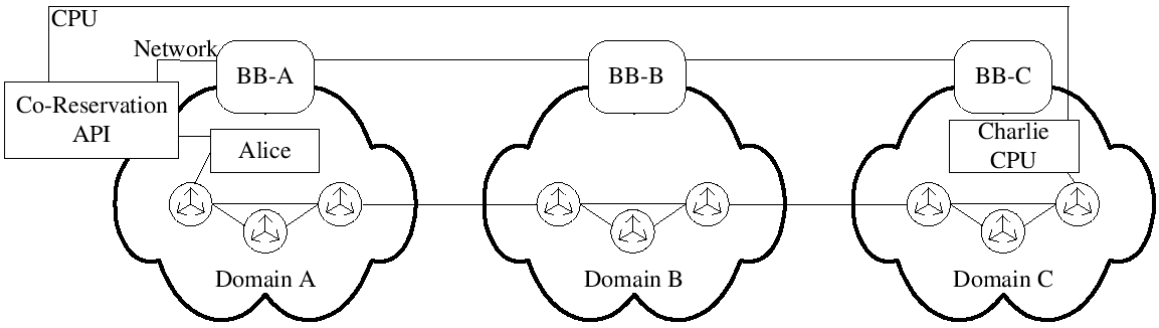


Figure 3: Problem of GARA in multi-domain scenarios (from [21]).

The provision of network QoS in a Grid system has been explored by a number of research projects, namely GARA [21], NRSE [3], G-QoSM [2], GNRB [1], GNB [4] and VIOLA [24] [25]. They are briefly reviewed below.

General-purpose Architecture for Reservation and Allocation (*GARA*) [21] provides programmers and users with convenient access to end-to-end QoS for computer applications. It provides uniform mechanisms for making QoS reservations for different types of resources, including computers, networks, and disks. These uniform mechanisms are integrated into a modular structure that permits the development of a range of high-level services. But there are also limitations. Some resources like disk space are fundamentally very different from network capacity. These resources are localized to certain end-systems and reservations can be made at the remote end-systems where such resources are located. Network capacity is a distributed resource requiring reservations at the local and remote end-systems as well as the network path between the local and remote systems. Regarding to multidomain reservations, GARA must exist in all the traversed domains, and the user (or a broker acting in his behalf) has to authenticate into all the domains (as Figure 3 depicts). This makes GARA difficult to scale.

The *Network Resource Scheduling Entity* (*NRSE*) [3] suggests that signalling and per-flow state overhead can cause end-to-end QoS reservation schemes to scale poorly to a large number of users and multi-domain operations – observed when using IntServ and RSVP, as

also with GARA [3]. This has been addressed in NRSE by storing the per-flow/per application state only at the end-sites that are involved in the communication. Although NRSE has demonstrated its effectiveness in providing DiffServ QoS, it is not clear how a Grid application developer would make use of this capability – especially as the application programming interface is not clearly defined [2].

Grid Quality of Service Management (G-QoSM) [2] is a framework to support QoS management in computational Grids in the context of the Open Grid Service Architecture (OGSA). G-QoSM is a generic modular system that, conceptually, supports various types of resource QoS, such as computation, network and disk storage. This framework aims to provide three main functions: 1) support for resource and service discovery based on QoS properties; 2) provision for QoS guarantees at application, middleware and network levels, and the establishment of Service Level Agreements (SLAs) to enforce QoS parameters; and 3) support for QoS management of allocated resources, on three QoS levels: 'guaranteed', 'controlled load' and 'best effort'. G-QoSM also supports adaptation strategies to share resource capacity between these three user categories.

The *Grid Network-aware Resource Broker* (GNRB) [1] is an entity that enhances the features of a Grid Resource Broker with the capabilities provided by a Network Resource Manager. This leads to the design and implementation of new mapping/ scheduling mechanisms to take into account both network and computational resources. The GNRB, using network status information, can reserve network resources to satisfy the QoS requirements of applications. The architecture is centralized, with one GNRB per administrative domain – potentially leading to the GNRB becoming a bottleneck within the domain. Also, GNRB is a framework, and does not enforce any particular algorithms to perform scheduling of jobs to resources.

Grid Network Broker (*GNB*) [4] is aimed at providing network QoS in a single administrative domain, and is the only proposal which considers the network when performing the

scheduling of jobs to computing resources. Thus, the network is a key parameter in order to choose the best computing resource to run a user's job. But the way this is done is not efficient, as it considers that only Grid transmissions go through the network, which is not true. Also, there is one GNB per administrative domain, which potentially may become a bottleneck within the domain.

As we said previously, we want to provide network QoS by means of an efficient scheduling. Many of the above efforts do not take network capability into account when scheduling tasks. The proposals which provide scheduling of users' jobs to computing resources are GARA, G-QoSM and GNB, and the schedulers used are DSRT [7] and PBS [15] in GARA, whilst G-QoSM uses DSRT. GNB is the only proposal which considers the network, but as mentioned above it is not done in a realistic way. These schedulers (DSRT and PBS) only pay attention to the load of the computing resource, thus a powerful unloaded computing resource with an overloaded network could be chosen to run jobs, which decreases the performance received by users, especially when the job requires a high network I/O.

Finally, VIOLA [24] [25] provides a metascheduling framework that provides co-allocation support for both computational and network resources. It is able to negotiate with the local scheduling systems to find and to reserve a common time slot to execute various components of an application. The metascheduling service in VIOLA has been implemented via the UNICORE Grid middleware for job submission, monitoring, and control. This allows a user to describe the distribution of the parallel MetaTrace application and the requested resources using the UNICORE client, while the remaining tasks like allocation and reservation of resources are executed automatically. A key feature in VIOLA is the network reservation capability, that allows the network to be treated as a resource within a metascheduling application. In this context, VIOLA is somewhat similar to our approach – in that it also considers the network as a key part in the job allocation process. However, the key difference is the fo-

cus in VIOLA on co-allocation and reservation – which is not always possible if the network is under ownership of a different administrator.

Also, given the scenario where no suitable computing resource is available in the local administrative domain, a major issue is choosing to which neighbor domain the query will be resubmitted. With this regard, some proposals have been presented. The Grid Distribution Manager (GridDM) is part of the e-Protein Project [18], a peer-to-peer system that performs inter-domain scheduling and load balancing above the intra cluster scheduling level where scheduling and load balancing are performed by standard schedulers like SGE, Condor etc. Similarly, Xu et al. [26] presented a framework for the QoS-aware discovery of services, and the QoS is based on feedback from users. Gu et al. [12] proposed a scalable aggregation model for P2P systems to automatically aggregate services into a high performance distributed application delivery with quality-of-service guarantees to fulfill the user's requirements.

Our current proposal is based on the architecture presented in [4] and extended in [5]. This architecture provides scheduling of jobs to computing resources within an administrative domain. The key element of that architecture is an entity named *Grid Network Broker* (GNB), which provides scheduling of jobs to computing resources, and considers the network as a key parameter for that. When a user queries the GNB for a computing resource to run a job, the GNB will proceed with a selection procedure. As a result, if there is a suitable resource in this domain, the job will be allocated to that resource. But, if there are no suitable computing resources in this domain, a problem will arise. This problem can be summarized with the question: *"to which neighbor domain the query should be forwarded?"*. We propose an answer to this question based on peer-to-peer systems, and this will be explained the next.

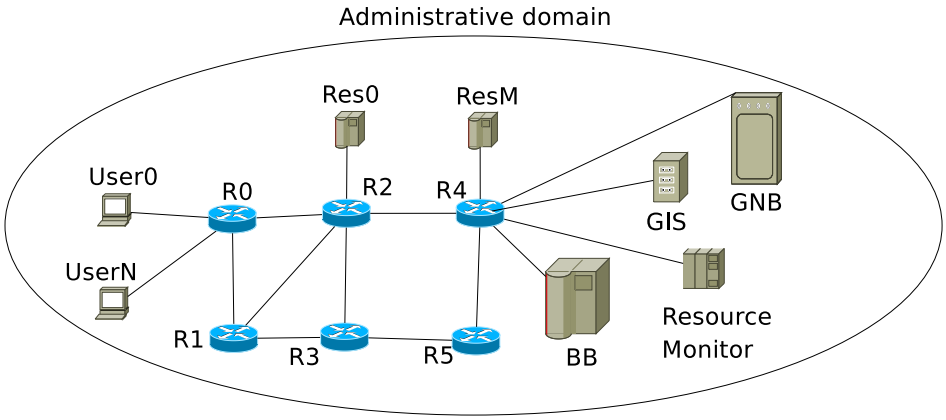


Figure 4: One single administrative domain.

3 Inter-domain scheduling

The proposed architecture is shown in Figure 4 and has the following entities: **Users**, each one with a number of jobs; **computing resources**, e.g. clusters of computers; **routers**; **GNB** (*Grid Network Broker*), a job scheduler; **GIS** (*Grid Information Service*), such as [9], which keeps a list of available resources; **resource monitor** (for example, Ganglia [14]), which provides detailed information on the status of the resources; **BB** (*Bandwidth Broker*) such as [22], which is in charge of the administrative domain, and has direct access to routers. BB can be used to support reservation of network links, and can keep track of the interconnection topology between two end points within a network. A more in-depth description of the functionality of the architecture can be found in [5].

In order to allow our inter-domain architecture to work properly, a number of assumptions should be made. The *first* assumption is that each domain must provide the resources it announces. This is, when a domain publishes that it has, e.g, X machines with Y speed, those machines must be available *within* the domain. The opposite case would be that a domain contains just a pointer to where the machines are. This last case is not correct for us,

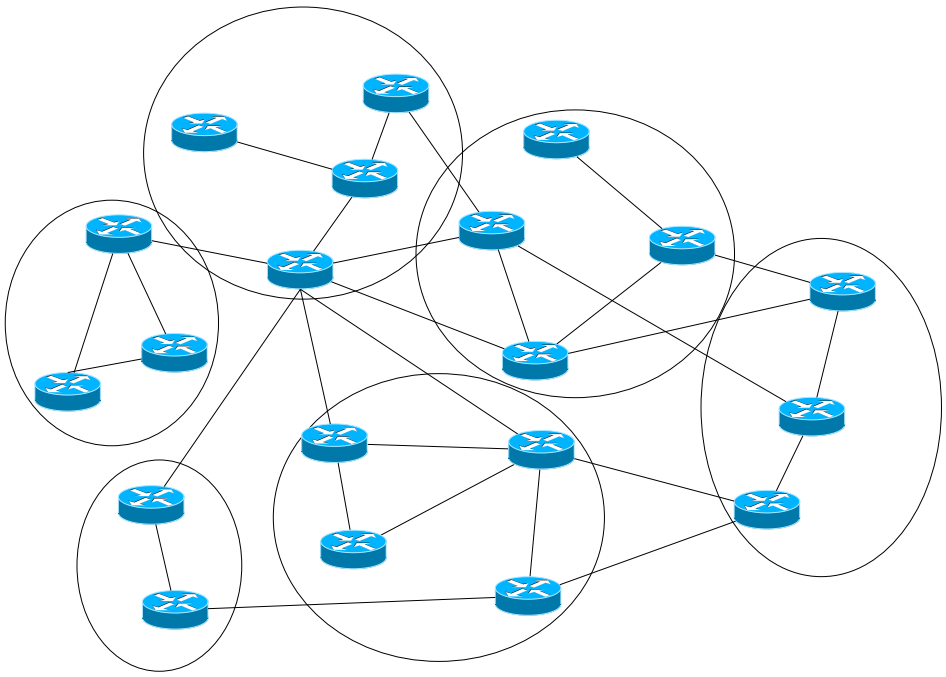


Figure 5: Several administrative domains.

because we need to use the effective bandwidth and the number of hops of the network path from the current domain to each neighbor. The reason is that this path will be used by the job during its transmission. The *second* assumption is that the resource monitor should provide exactly the same measurements in all the domains. Otherwise, no comparison among different domains can be made.

Figure 5 shows several administrative domains, and only the routers and the network links are shown for the sake of clarity. We can see that there may be one or more connections between routers from two domains. In this case, when we want to use the effective bandwidth of the link between two domains (which is an essential point of our architecture), we will rely on the *Border Gateway Protocol* (BGP) [20], which will always decide the optimal path to a destination network.

In this paper we use the concept of *Routing Indices (RI)* [8] in order to answer the question presented in the previous section. This way we allow nodes to forward queries to

neighbors that are more likely to have answers. If a node cannot find a suitable computing resource for a user's job within its domain, it forwards the query to a subset of its neighbors, based on its local RI, rather than by selecting neighbors at random or by flooding the network by forwarding the query to all neighbors. RI will be explained the next.

3.1 Routing Indices

Routing Indices (RI) [8] were initially developed for the document discovery in P2P systems, and they have also been used to implement a Grid information service in [19]. The goal of RIs is to help users find documents with content of interest across potential P2P sources efficiently.

The RI is used to improve the performance of our peer-to-peer routing, and to prevent the network from being flooded. The RI is a technique to choose the node to which a query should be forwarded: the RI represents the availability of data of a specific type in the neighbor's information base. We use a version of RI called *Hop-Count Routing Index* (*HRI*) [8], which considers the number of hops needed to reach a datum. Our implementation of HRI calculates the aggregate quality of a neighbor domain, based on the number of machines, their power, current load and the effective bandwidth of the link between the two domains. More precisely, Equation (1) is applied.

$$I_p^l = \left(\sum_{i=0}^{num\_machines_p} \frac{max\_num\_processes_i}{current\_num\_processes_i} \right) \times eff\_bw(l, p) \qquad (1)$$

where I_p^l is the information that the local domain l keeps about the neighbor domain p; $num\_machines_p$ is the number of machines domain p has; $current\_num\_processes_i$ is the current number of processes running in the machine; $max\_num\_processes_i$ is the maxi-

mum number of processes that can be run in that machine, and will be explained later on; $eff\_bw(l, p)$ is the effective bandwidth of the network connection between the local domain l and the peer domain p, and it is calculated as follows. Every *interval* seconds, GNBs forward a query along the path to their neighbor GNBs, asking for the counter of transmitted bytes of each interface the query goes through (the *OutOctets* parameter of SNMP [16]). Then, by using two consecutive measurements (let's call them m_1 and m_2, m_1 shows X bytes, and m_2 shows Y bytes), and considering the moment when they were collected (m_1 collected at time t_1 seconds and m_2 at t_2 seconds), and the capacity of the link C, we can calculate the effective bandwidth of each link as follows (the effective bandwidth of the path is the smallest effective bandwidth of links in that path.):

$$eff\_bw(l, p) = C - \frac{Y - X}{t_2 - t_1} \tag{2}$$

Regarding $max\_num\_processes_i$, this metric is the maximum number of processes that can be executed in a machine at a time, and it is used to decide how powerful a machine is. We calculate it by considering the speed of its CPU and the amount of memory it has. Equation (3) shows the actual formula used.

$$max\_num\_processes = k_1 \times \frac{memory}{max(memory)} + k_2 \times \frac{cpu\_speed}{max(cpu\_speed)} \tag{3}$$

In Equation (3), k_1 and k_2 are two constants that show how important each parameter (memory and CPU speed) is when calculating the maximum number of processes. Also, k_1+k_2 is the maximum number of processes we would like to have in the best of our machines. Moreover, each parameter is normalized by dividing it by a term, which is the maximum memory or CPU speed. These last two terms (maximum memory and CPU speed) must be propagated between peers, so that all the peers share the same values for them.

In Equations (1) and (3) we can see why the two assumptions mentioned before are needed. The first assumption assures that a domain provides the resource it announces, and it is necessary because we need to use the effective bandwidth between domains. Hence, if a domain does not contain the resources, the effective bandwidth used in Equation (1) would be useless, because the actual links used to transmit the job would not be those of the path between these two domains, but other different links. The second assumption assures that all the domains must be monitored in the same way, as otherwise the CPU speed data, current load and effective bandwidth will be useless, because no comparison could be done between domains in this case.

Also, predictions on the values of the current number of processes and the effective bandwidth can be used, for example, calculated as pointed out in [5]. As we can see, the network plays an important role when calculating the quality of a domain.

We used HRI as described in [8]: in each peer, the HRI is represented as a $M \times N$ table, where M is the number of neighbors and N is the horizon (maximum number of hops) of our Index: the n-th position in the m-th row is the quality of the domains that can be reached going through neighbor m, within n hops. As an example, the HRI of peer P_1 looks like Table 1 (for the topology depicted in Figure 2), where $S_{x,y}$ is the information for peers that can be reached through peer x, and are y hops away from the local peer (in this case, P_1).

Peer	**1 hop**	**2 hops**	**3 hops**
P_2	$S_{2.1}$	$S_{2.2}$	$S_{2.3}$
P_3	$S_{3.1}$	$S_{3.2}$	$S_{3.3}$

Table 1: HRI for peer P_1.

So, $S_{2.2}$ is the quality of the domains which can be reached through peer P_2, whose distance from the local peer is 2 hops. Each $S_{x,y}$ is calculated by means of the following formula:

$$S_{x.y} = \begin{cases} I^{P_l}_{P_x}, \text{ when } y = 1 \\ \sum_i I^{P_t}_{P_i}, \forall P_i, d(P_l, P_i) = y \wedge d(P_l, P_t) = y - 1 \wedge d(P_t, P_i) = 1, \text{ otherwise} \end{cases}$$

where $d(P_x, P_i)$ is the distance (in number of hops) between peers P_x and P_i. The formula is explained next. $S_{x.y}$ is calculated differently based on the distance from the local peer. When the distance is 1, then $S_{x.y} = I^{P_l}_{P_x}$, because the only peer that can be reached from local peer P_l through P_x within 1 hop is P_x. Otherwise, for those peers P_i whose distance from the local peer is y, we have to add the information that each peer P_t (which is the neighbor of P_i) keeps about them.

So, the HRI of peer P_1 will be calculated like this one:

Peer	**1 hop**	**2 hops**	**3 hops**
P_2	$I^{P_1}_{P_2}$	$I^{P_2}_{P_4} + I^{P_2}_{P_5}$	$I^{P_4}_{P_8} + I^{P_4}_{P_9} + I^{P_5}_{P_{10}} + I^{P_5}_{P_{11}}$
P_3	$I^{P_1}_{P_3}$	$I^{P_3}_{P_6} + I^{P_3}_{P_7}$	$I^{P_6}_{P_{12}} + I^{P_6}_{P_{13}} + I^{P_7}_{P_{14}} + I^{P_7}_{P_{15}}$

Table 2: HRI for peer P_1.

In order to use RIs, a key component is the *goodness function* [8]. The goodness function will decide how good each neighbor is by considering the HRI and the distance between neighbors. More concretely, our goodness function can be seen in Equation (4).

$$goodness(p) = \sum_{j=1..H} \frac{S_{p.j}}{F^{j-1}} \tag{4}$$

In Equation (4), p is the peer domain to be considered; H is the horizon for the HRIs; and F is the fanout of the topology. As [8] explains, the horizon is the limit distance, and those peers whose distance from the local peer is higher than the horizon will not be considered. Meanwhile, the fanout of the topology is the maximum number of neighbors a peer has.

As an example, suppose that all the peers (recall that each peer represents a whole administrative domain) in Figure 2 have 2 machines, each one with a speed of 1 GHz and 1 GB of memory; $k_1 = k_2 = 50$ and the current number of processes is 40 for all of them. Link bandwidths appearing in the figure have been calculated as [5] suggests. Table 3 shows the HRI for peer P_1. Suppose that, from peer P_1, we are looking for a computing resource. When the goodness function is applied, we get $goodness(P_2) = 3.139$ and $goodness(P_3) = 4.083$. Our goodness function gives a higher value to P_3, because within short distance (1 hop) we can reach better resources.

Peer	**1 hop**	**2 hops**	**3 hops**
P_2	1.25	2.75	8.75
P_3	2.5	2.5	6.75

Table 3: Example HRI for peer P_1.

3.2 Search technique

In literature, several techniques are used for searches in P2P networks, including flooding (e.g. Gnutella), centralized servers (e.g. Napster). More effective searches are performed by systems based on distributed indices. In these configurations, each node holds a part of the index. The index optimizes the probability of finding quickly the requested information, by keeping track of the availability of data to each neighbor.

Algorithm 1 shows the way that our architecture performs the scheduling of jobs to computing resources. In our system, when a user wants to run a job, he/she submits a query to the GNB of the local domain. This query is stored (line 7) as it arrives for the first time to a GNB. Then, the GNB looks for a computing resource in the local domain matching the requirements of the query (line 11). If the GNB finds a computing resource in the local domain that matches the requirements, then it tells the user to use that resource to run the

job (line 27). Otherwise, the GNB will forward the query to the GNB of one of the neighbor domains. This neighbor domain will be chosen based on the *Hop-Count Routing Index, HRI,* explained before (line 16). The parameter $ToTry$ is used to decide which neighbor should be contacted the next. This is, the best neighbor will be contacted first (in Figure 6, $p3$ will contact $p6$); if the query is bounced back, then the 2^{nd} best neighbor will be contacted ($p3$ will contact peer $p7$), and so on. Hence, a neighbor domain will only be contacted if there are no local computing resources available to fulfill the query (finish the job before the deadline expires, for instance).

Algorithm 1 Searching algorithm.

```
 1: Let q = new incoming query
 2: Let LocalResource = a resource in the local domain
 3: Let NextBestNeighbor = a neighbor domain select by the goodness function
 4: Let ToTry = the next neighbor domain to forward the query to
 5: for all q do
 6:    LocalResource := null
 7:    if (QueryStatus(q) = not present) then
 8:       {the first time the query arrives to this domain, store the query}
 9:       QueryStatus(q) := 1
10:       {we must look for a computing resource in the local domain}
11:       LocalResource := MatchQueryLocalResource(q)
12:    end if
13:    if (LocalResource == null) then
14:       {no computing resource in the local domain, so forward the query to a neighbor
          domain}
15:       ToTry := QueryStatus(q)
16:       NextBestNeighbor := HRI(q, ToTry)
17:       if (NextBestNeighbor == null) then
18:          {the query must be bounced back}
19:          Recipient := Sender (q)
20:       else
21:          Recipient := NextBestNeighbor
22:          QueryStatus(q) += 1
23:       end if
24:       ForwardQueryToRecipient(q, Recipient)
25:    else
26:       {tell the requester we have found a computing resource}
27:       SendResponseToRequester(q)
28:    end if
29: end for
```

4 Evaluation

In order to illustrate the behavior of our design, we will present an evaluation showing how our HRIs vary when varying the measurements. For this evaluation, we use the topology presented in Figure 6, and all the data we present here are referred to the peer $p1$. For the sake of easiness and clarity, we assume that all the links' bandwidth is 1 Gbps, all the peers have 1 resource made of 1 machine, with 4 Gb of memory and CPU speed of 1 GHz.

For Equation 1, we have approximated the values of $current\_num\_processes_i$ as a uniform distribution between 10 and 100, and the $max\_num\_processes_i$ as 100. Regarding the $eff\_bw(l, p)$, we have considered a Poisson distribution for those links that are heavily loaded, and Weibull distribution for those links which are not so loaded, as [6] suggests. In Figure 6, the even links will be heavily used, and are depicted with a thicker line.

To calculate the parameters for these distributions (the mean μ for the Poisson distribution, and scale β and shape α for the Weibull distribution), we have considered that the level of use of heavily used links is 80%, whilst no heavily used links exhibit a 10% usage. This way, if a heavily used link transmits 800 Mb in 1 second, and the maximum transfer unit of the links is 1500 bytes, the inter-arrival time for packets is 0.000015 $seconds$. Thus, this is the value for the μ of the Poisson distribution. In the same way, we calculate the value for the β parameter of the Weibull distribution, and the value we get is 0.00012 $seconds$. This way, by means of these mathematical distributions, we calculate the inter-arrival time for packets, and the calculation of the effective bandwidth from this is straight-forward.

We have simulated a measurement period of 7 days, with measurements collected every 30 minutes. Figures 7 and 8 present the variation on the use of links and the number of processes, following the mathematical distributions explained before. Figure 7 represents the level of use of links compared to the actual bandwidth (1 Gbps), per measurement. Heavily

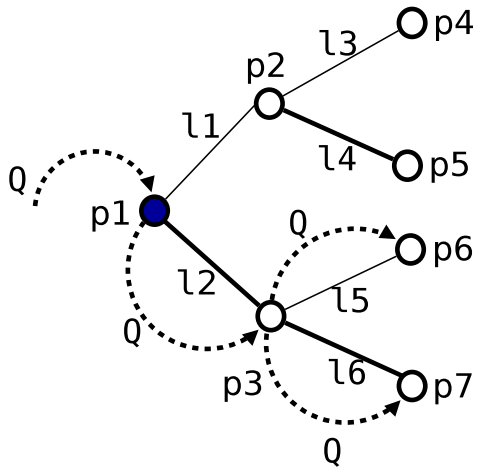
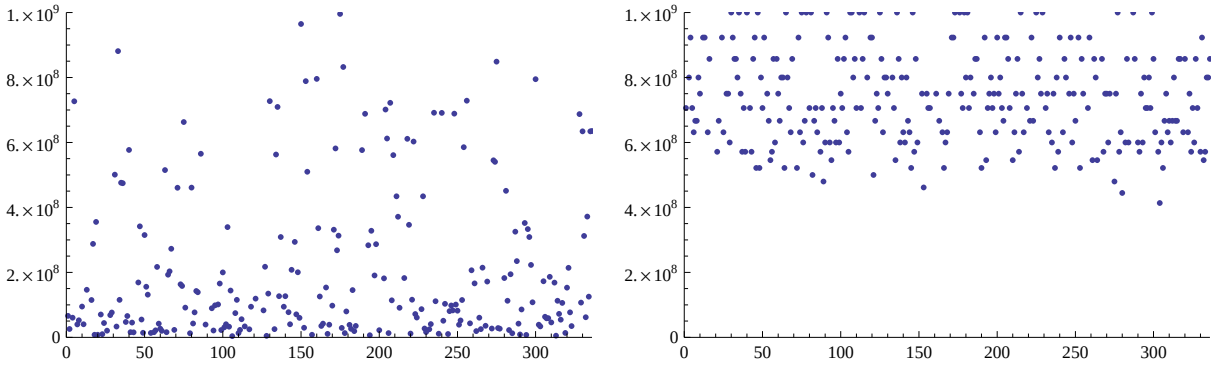


Figure 6: A query (Q) is forwarded from $p1$ to the best neighbors ($p3$, $p6$, and $p7$).

used links get a higher used bandwidth than not heavily used links. Thus, the data shown in these figures are used for our HRIs in order to decide where to forward a query.

Figure 9 and 10 present the variation of the $S_{x.y}$ for both heavily/ unheavily loaded links. These figures have been calculated by means of the formulas explained in Section 3.1, and applying them to the mathematical distributions mentioned above. As we explained in Tables 1 and 2, $S_{2.1} = I^{p1}_{p2}$, and $S_{3.1} = I^{p1}_{p3}$. We can see that the network performance affects the HRI, as was expected. We must recall that the higher the HRI is, the better because it means that the peer is powerful and well connected. Also, we see that when the link is not heavily loaded, the S has more high values, and values are more scattered across the figure. As opposed to it, when the link is heavily loaded, more values are grouped together at the bottom of the figure. Also, for Figure 10, $S_{2.2} = I^{P_2}_{P_4} + I^{P_2}_{P_5}$, and $S_{3.2} = I^{P_3}_{P_6} + I^{P_3}_{P_7}$, which means that to calculate $S_{2.2}$ and $S_{3.2}$, both heavily and not heavily used links are used.

Figure 11 shows the variation of the goodness function for the 2 neighbors of peer $p1$. Recall that the link between $p1$ and $p2$ is unloaded, and the link between $p1$ and $p3$ is loaded. We can see that the goodness function for $p2$ has higher values, and for $p3$ it has more values grouped at the bottom of the figure.



(a) Use of a not heavily loaded link (Weibull distribution).

(b) Use of a heavily loaded link (Poisson distribution).

Figure 7: Variation of use of links.

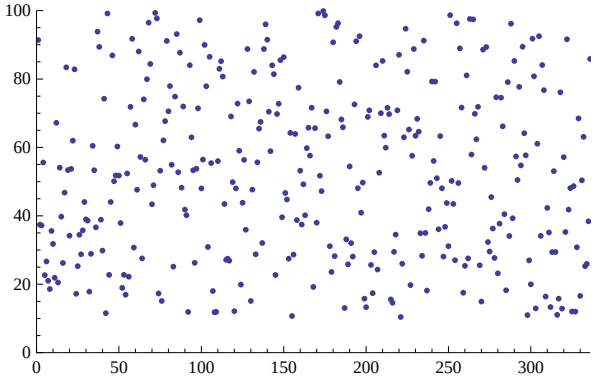
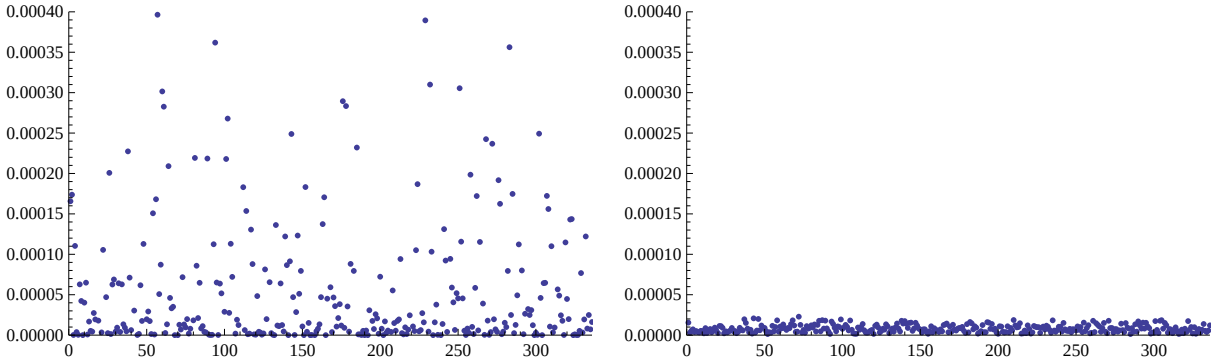


Figure 8: Variation of the number of processes (Uniform distribution).



(a) $S_{2.1} = I_{p2}^{p1}$ (link $p1 - p2$ is not heavily loaded).

(b) $S_{3.1} = I_{p3}^{p1}$ (link $p1 - p3$ is heavily loaded).

Figure 9: Variation of $S_{x.y}$ for loaded/ unloaded links.

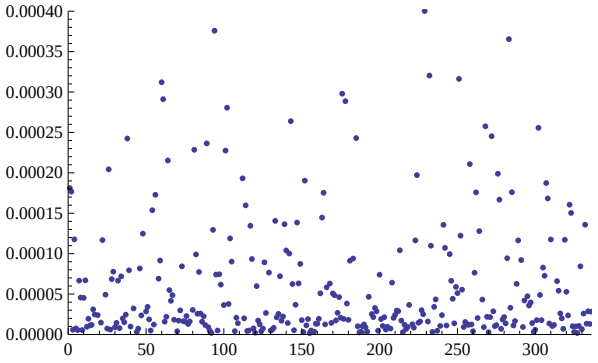
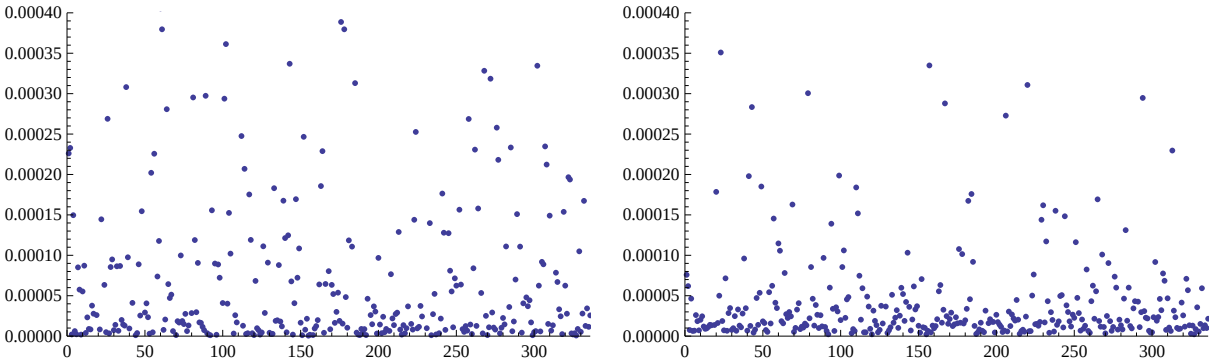


Figure 10: $S_{2.2}$ ($S_{3.2}$ would also look like this).



(a) Goodness function for peer $p2$ (link $p1 - p2$ unloaded).

(b) Goodness function for peer $p3$ (link $p1 - p3$ loaded).

Figure 11: Variation of the goodness function.

5 Conclusions and future work

The network remains an important requirement for any Grid application, as entities involved in a Grid system (such as users, services, and data) need to communicate with each other over a network. The performance of the network must therefore be considered when carrying out tasks such as scheduling, migration or monitoring of jobs. Also, inter-domain relations are key in Grid computing. We propose an extension to an existing scheduling framework to allow network-aware multi-domain scheduling based on peer-to-peer techniques.

More precisely, our proposal is based on *Routing Indices* (RI). This way we allow nodes to forward queries to neighbors that are more likely to have answers. If a node cannot find a suitable computing resource for a user's job within its domain, it forwards the query to a subset of its neighbors, based on its local RI, rather than by selecting neighbors at random or by flooding the network by forwarding the query to all neighbors.

Among the future work, we will implement the proposal using a simulation tool, GridSim [23], because a lot of work is required to set up the testbeds on many distributed sites. Even if automated tools exist to do this work, it would still be very difficult to produce performance evaluation in a *repeatable* and *controlled* manner, due to the inherent heterogeneity of the Grid. In addition, Grid testbeds are limited and creating an adequately-sized testbed is expensive and time consuming. Therefore, it is easier to use simulation as a means of studying complex scenarios.

Acknowledgement

This work has been jointly supported by the Spanish MEC and European Comission FEDER funds under grants “Consolider Ingenio-2010 CSD2006-00046” and “TIN2006-15516-C04-02”; by JCCM under grants PBC-05-007-01, PBC-05-005-01 and José Castillejo.

References

[1] Adami, D., et al. Design and implementation of a grid network-aware resource broker. In *Proc. of the Intl. Conference on Parallel and Distributed Computing and Networks* (Innsbruck, Austria, 2006).

[2] Al-Ali, R., et al. Network QoS Provision for Distributed Grid Applications. *Intl. Journal of Simulations Systems, Science and Technology, Special Issue on Grid Performance and Dependability 5*, 5 (December 2004).

[3] Bhatti, S., Sørensen, S., Clark, P., and Crowcroft, J. Network QoS for Grid Systems. *The Intl. Journal of High Performance Computing Applications 17*, 3 (2003).

[4] Caminero, A., Carrión, C., and Caminero, B. Designing an entity to provide network QoS in a Grid system. In *Proc. of the 1st Iberian Grid Infrastructure Conference (IberGrid)* (Santiago de Compostela, Spain, 2007).

[5] Caminero, A., Rana, O., Caminero, B., and Carrión, C. An Autonomic Network-Aware Scheduling Architecture for Grid Computing. In *Proc. of the 5th Intl. Workshop on Middleware for Grid Computing (MGC)* (Newport Beach, USA, 2007).

[6] Cao, J., Cleveland, W., Lin, D., and Sun, D. *Nonlinear Estimation and Classification*. Springer Verlag, New York, USA, 2002, ch. Internet traffic tends toward Poisson and independent as the load increases.

[7] Chu, H.-H., and Nahrstedt, K. CPU service classes for multimedia applications. In *Proc. of Intl. Conference on Multimedia Computing and Systems (ICMCS)* (Florence, Italy, 1999).

[8] Crespo, A., and Garcia-Molina, H. Routing Indices For Peer-to-Peer Systems. In *Proc. of the Intl. Conference on Distributed Computing Systems (ICDCS)* (Vienna, Austria, 2002).

[9] Fitzgerald, S., Foster, I., Kesselman, C., von Laszewski, G., Smith, W., and Tuecke, S. A directory service for configuring high-performance distributed computations. In *Proc. 6th Symposium on High Performance Distributed Computing (HPDC)* (Portland, USA, 1997).

[10] Foster, I., and Kesselman, C. *The Grid 2: Blueprint for a New Computing Infrastructure*, 2 ed. Morgan Kaufmann, 2003.

[11] Foster, I. T. The anatomy of the Grid: Enabling scalable virtual organizations. In *Proc. of the 1st Intl. Symposium on Cluster Computing and the Grid (CCGrid)* (Brisbane, Australia, 2001).

[12] Gu, X., and Nahrstedt, K. A Scalable QoS-Aware Service Aggregation Model for Peer-to-Peer Computing Grids. In *Proc. 11th Intl. Symposium on High Performance Distributed Computing (HPDC)* (Edinburgh, UK, 2002).

[13] Marchese, F. T., and Brajkovska, N. Fostering asynchronous collaborative visualization. In *Proc. of the 11th Intl. Conference on Information Visualization* (Washington DC, USA, 2007).

[14] Massie, M. L., Chun, B. N., and Culler, D. E. The Ganglia distributed monitoring system: design, implementation, and experience. *Parallel Computing 30*, 5-6 (2004), 817–840.

[15] Mateescu, G. Extending the Portable Batch System with preemptive job scheduling. In *SC2000: High Performance Networking and Computing* (Dallas, USA, 2000).

[16] McCloghrie, K., and Rose, M. T. Management information base for network management of tcp/ip-based internets:mib-ii. Internet proposed standard RFC 1213, mar 1991.

[17] Nabrzynski, J., Schopf, J., and Weglarz, J., Eds. *Grid Resource Management. State of the Art and Future Trends*. Kluwer Academic Publishers, 2004.

[18] O'Brien, A., Newhouse, S., and Darlington, J. Mapping of Scientific Workflow within the E-Protein Project to Distributed Resources. In *UK e-Science All-hands Meeting* (Nottingham, UK, 2004).

[19] Puppin, D., Moncelli, S., Baraglia, R., Tonellotto, N., and Silvestri, F. A Grid Information Service Based on Peer-to-Peer. In *Proc. of the 11th Intl. Euro-Par Conference* (Lisbon, Portugal, 2005).

[20] Rekhter, Y., Li, T., and Hares, S. A Border Gateway Protocol 4 (BGP-4). Internet proposed standard RFC 4271, January 2006.

[21] Roy, A. *End-to-End Quality of Service for High-End Applications*. PhD thesis, Dept. of Computer Science, University of Chicago, 2001.

[22] Sohail, S., Pham, K. B., Nguyen, R., and Jha, S. Bandwidth Broker Implementation: Circa-Complete and Integrable. Tech. rep., School of Computer Science and Engineering, The University of New South Wales, 2003.

[23] Sulistio, A., Poduval, G., Buyya, R., and Tham, C.-K. On incorporating differentiated levels of network service into GridSim. *Future Generation Computer Systems 23*, 4 (May 2007), 606–615.

[24] VIOLA : Vertically Integrated Optical Testbed for Large Application in DFN, 2006. Web site at http://www.viola-testbed.de/.

[25] Waldrich, O., Wieder, P., and Ziegler, W. A Meta-scheduling Service for Co-allocating Arbitrary Types of Resources. In *Proc. of the 6th Intl. Conference on Parallel Processing and Applied Mathematics (PPAM)* (Poznan, Poland, 2005).

[26] Xu, D., Nahrstedt, K., and Wichadakul, D. QoS-Aware Discovery of Wide-Area Distributed Services. In *Proc. of the First Intl. Symposium on Cluster Computing and the Grid (CCGrid)* (Brisbane, Australia, 2001).

[27] Zheng, W., Hu, M., Yang, G., Wu, Y., Chen, M., Ma, R., Liu, S., and Zhang, B. An efficient web services based approach to computing grid. In *Proc. of Intl. Conference on E-Commerce Technology for Dynamic E-Business, (CEC-East'04)* (Washington, DC, USA, 2004).