Improving Grid Inter-Domain Scheduling with P2P Techniques: A Performance Evaluation

Agustín Caminero 1, Omer Rana 2, Blanca Caminero 1, Carmen Carrión 1

1The University of Castilla La Mancha.
Campus Universitario s/n. 02071. Albacete. Spain.
{agustin, blanca, carmen}@dsi.uclm.es

2Cardiff School of Computer Science.
5 The Parade, Cardiff. CF24 3AA. UK.
o.f.rana@cs.cardiff.ac.uk

Abstract

The aggregation of geographically distributed resources in the context of a particular application has been made possible thanks to the deployment of Grid technologies. As Grids are extremely distributed systems, requirements on the communication network should also be taken into account when performing usual 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 of tasks. In a previous contribution, we enhanced an existing framework that provides scheduling of jobs to computing resources to allow multi-domain scheduling based on peer-to-peer techniques. This paper provides a performance evaluation from the users’ point of view, carried out by means of simulation, in which our framework is compared with existing approaches for inter-domain task scheduling. Results show that our proposal provides the best results in terms of scalability and job completion time.

1 Introduction

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) [16]. During the last decade, Grid computing has emerged as an enabling technology for many data and/or computing intensive applications. Through the use of Grid technologies it is possible to aggregate dispersed heterogeneous resources for solving various kinds of large-scale parallel applications in science, engineering and commerce [15]. A well-known example of such applications is the Grid-based worldwide data-processing infrastructure deployed for the LHC experiments at CERN [10].

Grid’s variability makes Quality of Service (QoS) highly desirable, though often very difficult to achieve in practice [30]. 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 [23]), 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 [28]. In a VO, entities communicate with each other using an interconnection network – resulting in the network playing an essential role in Grid systems [30].

Interactions between domains within a VO remains an important research topic [36] – as users in one domain may need to access resources in another. Also, jobs belonging to a user with particular QoS requirements may need to be executed in a computing resource from a different administrative domain. A resource broker may be utilized to discover computing resources fulfilling the particular job properties required by a user.

Our work focuses on the provision of network QoS, by integrating network characteristics when choosing resources to execute jobs. As mentioned above, if no suitable resource can be found locally, a resource from a different domain may be chosen. Therefore, the connections between domains should be considered when performing the scheduling of tasks – we assume this connectivity to follow
The provision of network QoS in a Grid system has been explored by a number of research projects, namely GARA [30], NRSE [5], G-QoSM [2], GNRB [1] and VIO-OLA [33] [34]. They are briefly reviewed below.

General-purpose Architecture for Reservation and Allocation (GARA) [30] 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. The most important limitation regarding network QoS is that 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 multi-domain reservations, GARA must exist in all the traversed domains, and the user (or a broker acting on his behalf) has to be authenticated in all the domains. This makes GARA difficult to scale.

The Network Resource Scheduling Entity (NRSE) [5] 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, a limitation found in GARA [5]. 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.

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 and G-QoS, and the schedulers used are DSRT [11] and PBS [25] in GARA, whilst G-QoS uses DSRT. 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 [33] [34] provides a meta-scheduling 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 reserve a common time slot to execute various components of an application. A key feature in VIOLA is the network reservation capability, that allows the network to be treated as a resource within a meta-scheduling application. In this context, VIOLA is somewhat similar to our approach – in that it also considers network as a key part in the job allocation process. However, the key difference is the focus in VIOLA on co-allocation and reservation – which is not always possible if the network is under ownership of a different administrator.

Also, when no computing resource can be found in the local administrative domain, a major issue is deciding which neighbor domain to chose. DIANA [3] performs global meta-scheduling in a local environment, typically in a LAN. In DIANA, a set of meta-schedulers are used that work in a P2P manner. Each site has a meta-scheduler that communicates with all other meta-schedulers on other sites. DIANA has been developed to make decisions based on global information. This makes DIANA not suitable for a realistic Grid testbed, such as the LHC Computing Grid [22], which has around 200 sites and tens of thousands of CPU (for a map showing real time information, see [18]).

The proposal of the e-Protein Project [29] is called Job Yield Distribution Environment (JYDE) [27]. One of its components is the Grid Distribution Manager (GrIDM, or DM), a P2P system that performs inter-domain scheduling and load balancing above the intra-cluster schedulers like SGE, Condor, etc. But network connections are not considered when deciding on where to schedule a task – only the workload at each cluster. The aim of this strategy is to keep every CPU at every site running jobs and to keep a few jobs waiting at each site at any time, but not so many that it would hinder the DM’s ability to make scheduling decisions. Thus, network QoS provision cannot be considered as one of the aims of this proposal. Similarly, Xu et al. [35] present a framework for the QoS-aware discovery of services, where QoS is based on feedback from users. Gu et al. [19] propose a scalable aggregation model for P2P systems to automatically aggregate services to create a distributed application – based on user defined QoS constraints.

Also, Assunção et al. [13] provide an architecture for the inter-networking of islands of Grids, which identifies and proposes architecture, mechanisms, and policies that allow the inter-connectivity of Grids, and allows Grids to grow in a similar manner as the Internet – referred to as the Inter-Grid. The proposed InterGrid architecture is composed of Gateways responsible for managing peering arrangements between Grids.

Our proposal is an enhancement on a previous proposal [6], which is aimed at providing scheduling of jobs to computing resources within an administrative domain. The original proposal has been extended to implement inter-domain scheduling [8], and both the original proposal and the extension will be reviewed in the next section. Subsequently, we undertake a performance evaluation of our inter-domain scheduling proposal (named Inter-Domain GNB, or ID-GNB).

3 Interdomain Scheduling: ID-GNB

We propose the use of a Grid Network Broker (GNB) [6] for providing network QoS in a single administrative domain. This is achieved by monitoring network elements in order to adapt the scheduler to the status of the system [7]. Thus, the network is a key parameter in order to choose the best computing resource to run a user’s job. The proposed inter-domain scheduling architecture, shown in Figure 2 [7], 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 [14], which keeps a list of available resources; resource monitor (for example, Ganglia [24]), which provides detailed information on the status of the resources; BB (Bandwidth Broker) such as [31], 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. This architecture works within an administrative domain, and has been extended to work in an inter-domain scenario. An in-depth description of the inter-domain architecture can be found in [9].

Our architecture is based on the following assumptions:
We use a version of RI called data of a specific type in the neighbor’s information base. The RI is a technique to choose the node to which a query should be forwarded: the RI represents the availability of resources. (1) each domain must provide the resources it announces, i.e. when a domain publishes that it has, e.g, X machines with Y speed, those machines must exist within the domain. This is needed to calculate the effective bandwidth between the user and the domain hosting the resources; (2) the resource monitor should provide exactly the same measurements in all the domains. Otherwise, no comparison among different domains can be made.

Our inter-domain scheduling approach, named Inter-Domain GNB, or ID-GNB, uses the concept of Routing Indices (RI) [12]. These enable 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. This reduces the amount of traffic generated within the network for finding suitable resources.

### 3.1 Routing Indices

The RI is used to improve the performance of our P2P 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) [12], 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.

\[
I_p^l = \left( \sum_{i=0}^{\text{num\_mach}_p} \frac{\text{max\_num\_processes}_i}{\text{current\_num\_processes}_i} \right) \times \text{eff\_bw}(l, p)
\]

where \(I_p^l\) is the information that the local domain \(l\) keeps about the neighbor domain \(p\); \(\text{num\_mach}_p\) is the number of machines domain \(p\) has; \(\text{current\_num\_processes}_i\) is the current number of processes running in machine \(i\); \(\text{max\_num\_processes}_i\) is the maximum number of processes that can be run in machine \(i\); \(\text{eff\_bw}(l, p)\) is the effective bandwidth of the network connection between the local domain \(l\) and the peer domain \(p\), and is calculated as follows. Every interval seconds, GNBs sends a query along the path to their neighbor GNBs, asking for the number of transmitted bytes, for each interface that the query goes through (the OutOctets parameter of SNMP [26]). Using two consecutive measurements (\(m_1\) 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 as follows:

\[
\text{eff\_bw}(l, p) = C - \frac{Y - X}{t_2 - t_1}
\]

In Equation (1) 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 resource load and the effective bandwidth can be used, for example, calculated as pointed out in [7]. As we can see, the network plays an important role when calculating the quality of a domain. For an in-depth description of the formulas, see [8].

In each peer, the HRI is represented as an \(M \times N\) 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 \(n^{th}\) row is the quality of the domains that can be reached through neighbor \(m\), within \(n\) hops. As an example, the HRI of peer \(P_1\) can be found in Table 1 (for the topology depicted in Figure 3), 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, \(P_1\)).

Here \(S_{2,2}\) is the quality of the domains which can be reached through peer \(P_2\), whose distance from the local peer is 2 hops. Each \(S_{x,y}\) at a domain \(l\) is calculated by means of the Equation 3.
In Equation 3, $d(P_x, P_i)$ is the distance (in number of hops) between peers $P_x$ and $P_i$. $S_{x.y}$ is calculated differently based on the distance from the local peer. When the distance is 1, then $S_{x.y} = I_{P_x}^y$, because the only peer that can be reached from local peer $P_x$ 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 $P_t$ (whose distance with $P_l$ is $y - 1$) keeps about them. As an example, the HRI of peer $P_1$ will be calculated as shown in Table 2.

$$S_{x.y} = \begin{cases} I_{P_x}^y, & \text{when } y = 1, \\ \sum_i I_{P_i}^y, \forall P_i, d(P_i, P_l) = y \land d(P_l, P_t) = y - 1 \land d(P_t, P_i) = 1, & \text{otherwise} \end{cases} \quad (3)$$

In Equation 3, $d(P_x, P_i)$ is the distance (in number of hops) between peers $P_x$ and $P_i$. $S_{x.y}$ is calculated differently based on the distance from the local peer. When the distance is 1, then $S_{x,y} = I_{P_x}^1$, because the only peer that can be reached from local peer $P_x$ 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 $P_t$ (whose distance with $P_l$ is $y - 1$) keeps about them. As an example, the HRI of peer $P_1$ will be calculated as shown in Table 2.

<table>
<thead>
<tr>
<th>Peer</th>
<th>1 hop</th>
<th>2 hops</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_2$</td>
<td>$S_{2,1}$</td>
<td>$S_{2,2}$</td>
</tr>
<tr>
<td>$P_3$</td>
<td>$S_{3,1}$</td>
<td>$S_{3,2}$</td>
</tr>
</tbody>
</table>

Table 1. HRI for peer $P_1$.

In order to use RIs, a key component is the goodness function [12]. 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).

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

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 [12] explains, the horizon is a threshold on the distance between peers - 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.

4 Performance Evaluation

We present a performance evaluation of our proposal, ID-GNB, from the point of view of users. Our proposal is compared with other approaches from literature, namely GriDM and flooding, and they are briefly presented next. The initial evaluation of our proposal (presented in [8]) shows an evaluation from the point of view of the system, in which measurements are varied according to mathematical distributions. Thus, it was necessary to present a complementary evaluation from a different point of view.

4.1 Other approaches: GriDM and flooding

GriDM is part of the JYDE (Job Yield Distribution Environment) [27] software package that was developed in the e-Protein project [29]. On the submission server, GriDMs form a P2P network and attempt to balance the load across them. GriDM works by constantly checking the lengths of the wait queues at each site. When a queue on a particular site falls below a threshold, new permits are issued for that site, so that more jobs can be submitted to that site. The aim of this strategy is to keep every CPU at every site running jobs, and to keep a few jobs waiting at each site at any time, but not so many that it would hinder the DM’s ability to make scheduling decisions [27]. Thus, network QoS provision cannot be considered as one of the aims of this proposal. By comparing our proposal with GriDM, we want to stress the fact that the network is an important resource that influences the performance received by users in a Grid. Thus, proposals that do not consider the network will not perform as efficiently as possible.

Gnutella [17] uses flooding, requiring each peer to forward the query to all its neighbors. Every query has a time-to-live (TTL), which is decremented each time a peer receives a query. When the TTL reaches 0, the query will be
rejected, and the user informed of the rejection. When one of the peers accepts the query, it also informs the user. Due to the fact that the number of queries increase each time they are forwarded by a peer – many different peers may accept the same query. In this case, the job will be executed in the peer whose answer reaches the user the first.

4.2 Experiments and results

We create a network scenario based on the EU DataGRID Testbed 1, as shown in Figure 4 [20]. The original topology has been modified (3 links have been removed) to avoid loops when constructing Routing Indices, which has already been treated in [12]. The topology shows 11 computing resources spanning 11 locations in Europe. Each location is an administrative domain, with the structure shown in Figure 2. Limits between administrative domains are shown in circles in Figure 4. Hence, the topology creates the P2P topology depicted in Figure 5. Our experiments have been carried out by means of the GridSim [32] simulation tool.

The three proposals (ID-GNB, GriDM and flooding) have been implemented in GridSim. The following decisions have been made:

- Proposals perform the scheduling in scheduling rounds, with an interval of 20 seconds.

- Proposals which perform the monitoring of neighbors (ID-GNB and GriDM), do it every 10 seconds.

- Peers accept a job to be executed in their local resource when the resource has idle CPUs at the moment the query reaches the peer. If a query reaches the peer more than once, this is done every time the query reaches the broker.

- Job queries in both GriDM and flooding experiments have a TTL, which has been chosen to allow queries to reach all the peers in the topology. For GriDM, it is equal to 11; for flooding, the TTL is 5.

- For GriDM, we consider the load of the computing resource, provided by GridSim, as a way to decide which neighbor a query must be forwarded to. We choose the least loaded computing resource each time.

Table 3 summarizes the characteristics of simulated resources, which were obtained from a real LCG testbed [21]. The CPU rating is defined in MIPS (Million Instructions Per Second) as per SPEC (Standard Performance Evaluation Corporation) benchmark. Moreover, the number of nodes for each resource have been scaled down by 10, because the complete experiments would take several weeks of processing. Finally, each resource node has four CPUs.

For this experiment, we create 100 users and distribute them among the locations, as shown in Table 3. Each user has a number of jobs and the processing power of each job is 1400000 Million Instructions (MI), which means that each job takes about 2 seconds if it is run on the CERN resource. Also, I/O files sizes are 24 MB. All jobs have the same parameters that are taken from ATLAS online monitoring and calibration system [4]. Our experiment is aimed at determining the behavior of the inter-domain scheduling algorithm. Hence, we are trying to look at how different algorithms affect the performance received by users in terms of number of queries forwarded, rate of queries per job, and the overall job execution time.

Figure 6 presents results regarding number of succeeded jobs for each inter-domain scheduling policy, as we vary the number of jobs each user wants to run. We can see there is no difference for GriDM and ID-GNB approaches, since both of them can find a computing resource for all the jobs in all the experiments. Conversely, as we increase the num-
### Table 3. Resource specifications (TS stands for Time-Shared, while SS stands for Space-Shared).

<table>
<thead>
<tr>
<th>Peer ID</th>
<th>Res. (Location)</th>
<th># Nodes</th>
<th>CPU Rating</th>
<th>Policy</th>
<th># Users</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>RAL (UK)</td>
<td>41</td>
<td>49,000</td>
<td>SS</td>
<td>12</td>
</tr>
<tr>
<td>1</td>
<td>Imp. College (UK)</td>
<td>52</td>
<td>62,000</td>
<td>SS</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>NorduGrid (Norway)</td>
<td>17</td>
<td>20,000</td>
<td>SS</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>NIKHEF (Netherlands)</td>
<td>18</td>
<td>21,000</td>
<td>SS</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>Lyon (France)</td>
<td>12</td>
<td>14,000</td>
<td>SS</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>CERN (Switzerland)</td>
<td>59</td>
<td>70,000</td>
<td>SS</td>
<td>24</td>
</tr>
<tr>
<td>6</td>
<td>Milano (Italy)</td>
<td>5</td>
<td>70,000</td>
<td>SS</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>Torino (Italy)</td>
<td>2</td>
<td>3,000</td>
<td>TS</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>Rome (Italy)</td>
<td>5</td>
<td>6,000</td>
<td>SS</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>Padova (Italy)</td>
<td>1</td>
<td>1,000</td>
<td>TS</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>Bologna (Italy)</td>
<td>67</td>
<td>80,000</td>
<td>SS</td>
<td>12</td>
</tr>
</tbody>
</table>

The number of jobs per user, there is an increase in the number of jobs in the flooding approach that cannot be allocated to any computing resource. Hence, those jobs will not be executed.

Now consider Figure 7, which depicts the number of queries forwarded per succeeded job. This statistic has been calculated by dividing the actual number of queries forwarded, by the number of successfully completed jobs. Hence, this statistic includes queries forwarded for those jobs which could not be executed. As expected, flooding requires more queries per job, since each peer forwards incoming queries it cannot fulfill to all its neighbors. With regard to ID-GNB and GriDM, ID-GNB shows the smallest values for this statistic, and the difference gets bigger as we increase the number of jobs per user. For the case of 15 jobs per user, ID-GNB requires 30% less queries than GriDM, for the same amount of succeeded jobs.

The last statistic we present is the average execution latency for jobs, which is shown in Figure 8. This statistic includes the elapsed time since users submitted the job to the computing resource, until the output of the job reaches the user. It includes the transmission time, queuing time at the resource (if no CPU is idle at the moment), and the execution time of the job. This statistic is calculated for each job, for example, we calculated the average execution latency for job 0 for all the users. Figure 8 (a) shows the statistic when each user has 5 jobs, and we can see that the latencies are very similar for all the scheduling policies, although ID-GNB is slightly worse than the others. This is because the network is not loaded, thus its performance does not influence the performance received by jobs. Hence, the least loaded computing resource is the best option to run jobs. This is confirmed by statistics appearing in Figure 7 (a), since both GriDM and ID-GNB show similar numbers of queries per succeeded job.

When each user has 10 jobs (represented in Figure 8 (b)), GriDM and flooding approaches show similar results, and ID-GNB performs better. The reason is that the network is more loaded than before, thus the network performance influences the behavior of jobs more than in the previous case, and the resource workload becomes less important. This fact is supported by the number of queries per successfully completed job (presented in Figure 7 (b)), which shows that GriDM needs more queries to find a suitable resource for each job than ID-GNB.

When users have 15 jobs (shown in Figure 8 (c)), the average execution latency is higher for GriDM than for the other approaches. Since GriDM does not consider the network load, the resource chosen is not the most suitable. This is confirmed by the number of queries per successfully completed job (presented in Figure 7 (c)). Also, flooding presents similar latencies than ID-GNB, because of the nature of flooding. Recall that with flooding, each peer forwards each query to all its neighbors, thus it reaches a suitable computing resource, at the expense of a really high number of queries per succeeded job (presented in Figure 7 (c)).

The results of this evaluation show that ID-GNB outperforms both GriDM and flooding in terms of number of queries required for each job, and execution time. Inter-Domain GNB achieves better rate of successfully completed jobs and lower execution latencies, with less queries per job. This therefore demonstrates that ID-GNB is scalable, hence it is a more appropriate technique for realistic Grid environments.

### 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. In a previous contri-
bution, we presented an inter-domain Grid scheduler based on P2P techniques. This scheduler chooses the neighbor domain to which a query should be forwarded by means of Routing Indices. This paper provides a performance evaluation carried out by means of simulation, which compares the improved framework with existing proposals for inter-domain scheduling, namely flooding and GriDM.

Results presented here demonstrate the scalability of ID-GNB. We are currently also investigating the overheads caused by ID-GNB, and this constitutes future work. We are also exploring techniques for modifying neighbor connectivity. Currently we assume that the neighbors set is prefixed, and it is defined by the physical links. But we are considering a dynamic construction of the neighbors set, by

Figure 6. Number of succeeded jobs.
Figure 7. Number of queries per succeeded job.

means of dynamic overlays.

Acknowledgement

This work has been jointly supported by the Spanish MEC and European Commission FEDER funds under grants “Consolider Ingenio-2010CSD2006-00046” and “TIN2006-15516-C04-02”; jointly by JCCM and Fondo Social Europeo under grant “FSE 2007-2013”; and by JCCM under grants “PBC-05-007-01” and “PBC-05-005-01”.
Figure 8. Average execution latencies.

References


