Journal of Computer and System Sciences 77 (2011) 262–281

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Network-aware heuristics for inter-domain meta-scheduling in Grids

Agustín Caminero [a], Omer Rana [b,*], Blanca Caminero [a], Carmen Carrión [a]

[a] The University of Castilla La Mancha, Campus Universitario s/n, 02071, Albacete, Spain
[b] Cardiff School of Computer Science, 5 The Parade, Cardiff, CF24 3AA, UK

ARTICLE INFO

Article history:
Received 8 April 2009
Received in revised form 3 September 2009
Available online 29 January 2010

Keywords:
Grid computing
Network-aware
Inter-domain
Peer-to-peer
Meta-scheduling

ABSTRACT

Grid computing generally involves the aggregation of geographically distributed resources in the context of a particular application. As such resources can exist within different administrative domains, requirements on the communication network must also be taken into account when performing meta-scheduling, migration or monitoring of jobs. Similarly, coordinating efficient interaction between different domains should also be considered when performing such meta-scheduling of jobs. A strategy to perform peer-to-peer-inspired meta-scheduling in Grids is presented. This strategy has three main goals: (1) it takes the network characteristics into account when performing meta-scheduling; (2) communication and query referral between domains is considered, so that efficient meta-scheduling can be performed; and (3) the strategy demonstrates scalability, making it suitable for many scientific applications that require resources on a large scale. Simulation results are presented that demonstrate the usefulness of this approach, and it is compared with other proposals from literature.

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*), and keeping their independence and autonomy [1]. 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 [2]. A well-known example of such applications is the Grid-based worldwide data-processing infrastructure deployed for the Large Hadron Collider (LHC) experiments at CERN [3].

The variability of Grids makes *Quality of Service* (*QoS*) highly desirable, though often very difficult to achieve in practice [4]. 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 [5]), 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. In a VO, entities communicate with each other using an interconnection network – resulting in the network playing an essential role in Grid systems [4].

As a VO is made of different organizations (or domains), the interactions between different domains become important when executing jobs, since jobs belonging to a user with particular QoS requirements may need to be executed in a computing resource from a different administrative domain. This can be graphically seen in Fig. 1, which depicts such sharing of resources across organizational boundaries.

* Corresponding author.
E-mail addresses: agustin@dsi.uclm.es (A. Caminero), o.f.rana@cs.cardiff.ac.uk (O. Rana), blanca@dsi.uclm.es (B. Caminero), carmen@dsi.uclm.es (C. Carrión).

0022-0000/$ – see front matter
doi:10.1016/j.jcss.2010.01.006

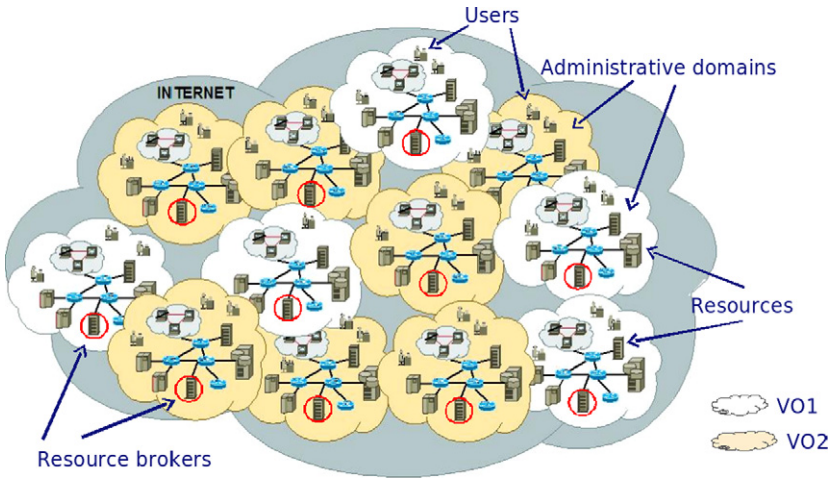


Fig. 1. A Grid, made of several administrative domains.

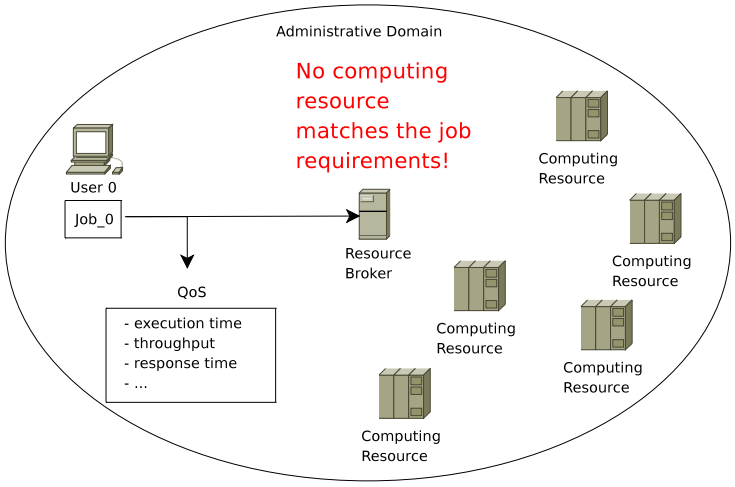


Fig. 2. Match-making between job requirements and computing resources.

A resource broker may be utilized to discover computing resources fulfilling the particular job properties required by a user, as shown in Fig. 2. A user wishing to execute a job with particular QoS requirements, such as execution time or response time, must contact a resource broker in order to get a computing resource fulfilling those requirements. It now becomes necessary to consider an alternative administrative domain, if local resources cannot be found to fulfill these QoS requirements.

The main contribution of this paper is a heuristic intended to manage QoS in a Grid system, concerned with the interactions between administrative domains when performing the meta-scheduling of jobs to computing resources. It is implemented in an entity called *Grid Network Broker* (GNB), first presented in [6], and which has been extended in this paper to perform inter-domain meta-scheduling. The heuristic utilizes Peer-2-Peer (P2P) ideas centered on query routing, for identifying suitable neighbouring domains which may contain the required resources. P2P and Grid systems share many properties – for instance, they involve resource sharing across different administrative domains to support particular application behaviours. Traditionally, in Grid computing, such resource sharing has been undertaken through the use of specialist security infrastructure, limiting access by users to particular types of resources. In P2P systems however, resource sharing has been less constrained, enabling more ad hoc modes of interaction between resource users and providers. The query referral mechanism used in P2P content sharing networks is of most interest to us in this context. Primarily, this is undertaken by enabling requirements for particular types of resources (or data) to be forwarded to the appropriate peer (or in the context of Grid computing – administrative domain). In P2P systems, a variety of mechanisms are available to achieve this, ranging from network flooding, more constrained gossiping protocols, to the use of structured overlays based on distributed hash tables. In this work, our focus is similar, i.e. to utilize properties of domains to determine where a request for resources should be forwarded, akin to the approach adopted in P2P systems. To achieve this, we make use of a heuristic, the main goals of which are: (1) to take the network into account to perform meta-scheduling; (2) to focus on communication

between domains, so that efficient meta-scheduling can be performed; and (3) do this in a scalable way, making it suitable for realistic deployments.

The paper is structured as follows: Section 2 reviews current efforts on network QoS in Grids, and the lack of attention paid to inter-domain relationships. Existing proposals for inter-domain meta-scheduling are also reviewed, along with work that combines P2P and Grid computing. Section 3 explains our approach for inter-domain meta-scheduling, and Section 4 provides an evaluation, demonstrating the usefulness of this work. Finally, Section 5 provides conclusions and Section 6 provides directions for future work.

2. Related work

The provision of QoS in Grids has been addressed by several research projects, but only *General-purpose Architecture for Reservation and Allocation* (*GARA*) and *Network Resource Scheduling Entity* (*NRSE*) deal explicitly with inter-domain relations. The *General-purpose Architecture for Reservation and Allocation* (*GARA*) [4] system can perform reservations on a variety of resources, such as computing, storage or networks. It is however difficult to scale, since users (or a broker acting on his behalf) needs to authenticate with all domains. Besides, GARA does not consider the network when performing the match-making between jobs and computing resources. On the other hand, *Network Resource Scheduling Entity* (*NRSE*) [7] is similar to GARA but limited to network reservations. It is able to automatically negotiate a multi-domain reservation by communicating with its counterpart on the remote network, on behalf of its client. Reservations across multiple domains are made using two NRSEs, one at each end (improving on GARA's limitation), but it relies on the assumption that the core network is over-provisioned. Besides, NRSE is only aimed at performing network reservations, not meta-scheduling of jobs to computing resources. The proposal presented in this paper is aimed at the provision of QoS in Grids by means of efficient and network-aware inter-domain meta-scheduling, it does not perform reservations yet, but this is considered as a part of the future work.

Interactions between different administrative domains have been studied, among others, in [8–11], but they are mainly concerned with security issues, not scheduling. Furthermore, the combination of Grid computing with P2P has been studied, among others, in [12–15]. Talia et al. [12] propose a P2P protocol for efficient invocation of Grid Services, and an architecture for resource discovery that adopts a P2P approach to extend the model of the GT3 information service. They propose a modified Gnutella discovery protocol – *Gridnut* – which makes it suitable for Grids following the *Open Grid Services Architecture* (*OGSA*). In particular, Gridnut uses appropriate message buffering and merging techniques to support interaction between Grid Services in a P2P fashion.

Xion et al. [13] develop an algorithm for finding services in a P2P Grid. In this algorithm, Grid resources are at first aggregated into a *GridPeer*. Subsequently, when a Grid resource is needed, a genetic algorithm is used to find the *closest* GridPeer. The authors also discuss the use of an ant-based optimization algorithm for improving the choice of resources that may be made available within a GridPeer. Similarly, Xu et al. [14] presented a framework for the QoS-aware discovery of services, where QoS is based on feedback from users. Gu et al. [15] proposed a scalable aggregation model for P2P systems to automatically aggregate services to support distributed application delivery, which satisfy user specified QoS guarantees.

Given the scenario where no suitable computing resource is available in the local administrative domain, a major issue is choosing the neighbor domain to which the query should be re-submitted. The e-Protein Project [16] makes use of a *Job Yield Distribution Environment* (*JYDE*) [17] to achieve this. One of its components is the *Grid Distribution Manager* (*GriDM*), a P2P system that performs inter-domain meta-scheduling and load balancing, extending the capability of intra-cluster schedulers like SGE and Condor. 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 meta-scheduling decisions [17]. Thus, network QoS provision cannot be considered as one of the aims of this proposal.

Gnutella [18] 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 first.

DIANA [19] 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 unsuitable for a realistic Grid testbed, such as the LHC Computing Grid [20], which has around 200 sites and tens of thousands of CPU (for a map showing real time information, see [21]).

Assunção et al. [22] provide an architecture for the inter-networking of *islands of Grids*, which identifies and proposes an architecture, mechanisms, and policies that allow the inter-connectivity of Grids, and allows Grids to grow in a similar manner to the Internet – referred to as the *InterGrid*. The proposed InterGrid architecture is composed of Gateways respon-

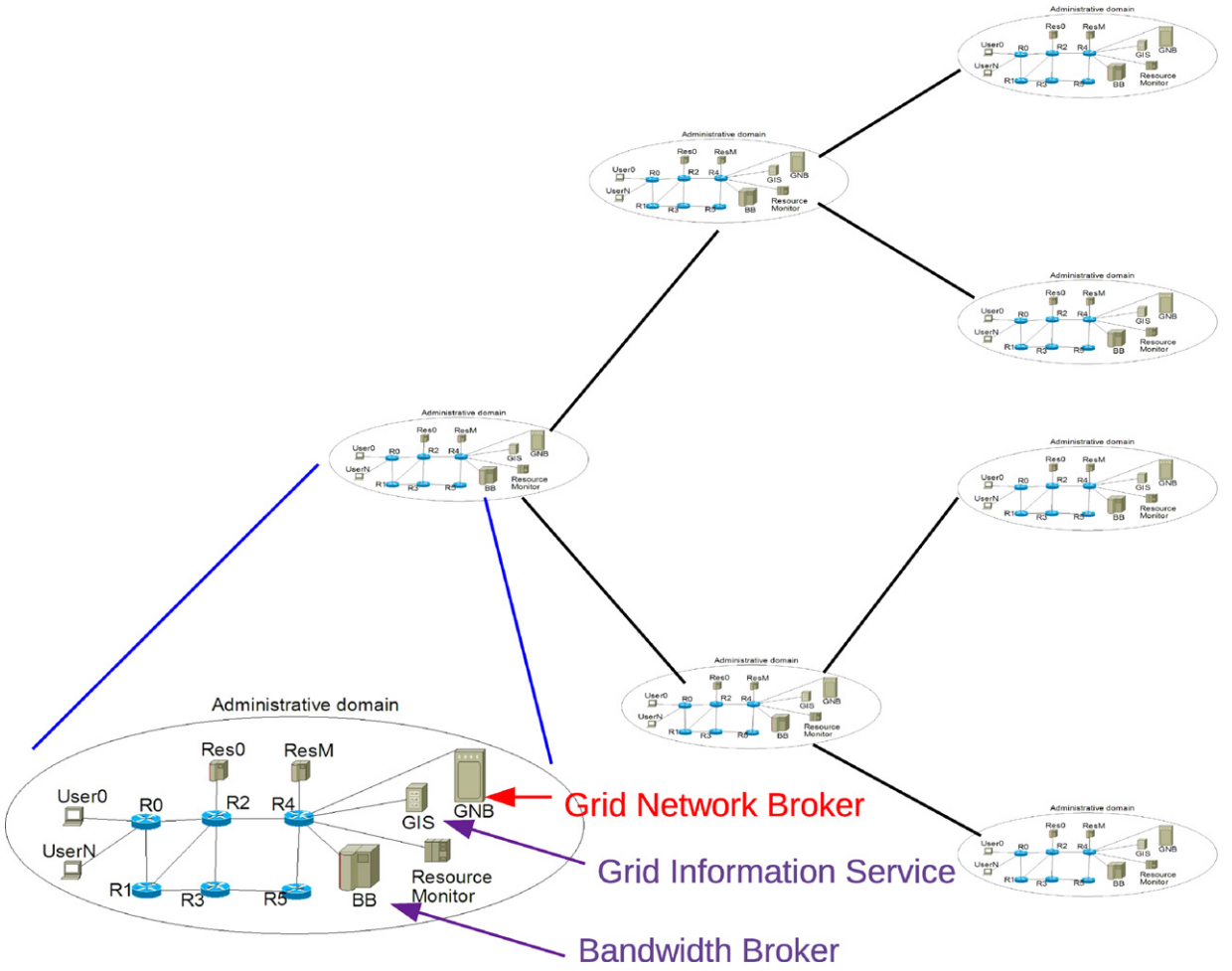


Fig. 3. Inter-domain meta-scheduling architecture.

sible for managing peering arrangements between Grids. It is useful to note that the current focus in the Grid community towards Cloud computing share many commonalities with such an architecture.

The proposal presented in this paper is aimed at the provision of QoS in Grids by means of efficient meta-scheduling. It is primarily concerned with the efficient communications and referral between administrative domains using concepts from P2P systems.

3. Inter-domain meta-scheduling

The architecture presented in this work provides meta-scheduling of jobs to computing resources in different administrative domains. When a user queries the GNB for a computing resource to run a job, the GNB will proceed with a selection procedure. If there is a suitable resource in the local domain, the job will be allocated to that resource – alternatively, a resource in another domain may be required – requiring the GNB to determine which domain should be chosen.

Fig. 3 shows the *intra-domain* meta-scheduling architecture. When the GNB of a domain receives a job to be scheduled, and no suitable computing resource exists locally, the GNB chooses one of the neighbor domains, and forwards the query to it. Apart from the GNB, each domain has other entities, such as a resource monitor (for instance, Ganglia [23]), a bandwidth broker (BB, such as [24]), and a *Grid Information Service* (GIS, such as [25]).

A number of assumptions are necessary for the effective deployment of such an architecture. The *first* assumption is that each domain must be capable of providing the resources it advertises, i.e. when a domain publishes that it has, e.g., X machines with speed Y, those machines must be available *within* the domain, and conform to the advertised specification. Hence, a domain must not contain a pointer to machines held in other domains, but should be able to offer these machines locally. A pointer to machines held elsewhere is not useful, as this does not reveal the effective bandwidth and the number of hops from the current domain to each neighbor. The *second* assumption is that the resource monitor should provide exactly the same measurements in all the domains. Otherwise, no comparison between resources available within different

Table 1
HRI for peer P_1.

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

domains can be made. When more than one network path exists from one domain to another, the *Border Gateway Protocol* (BGP) [26] will decide which is the optimal path.

Furthermore, since this architecture relies on the coordinated work of different administrative domains, issues related to authentication, authorization and accounting clearly arise. GNBs from different administrative domains must authenticate with each other prior to any information exchange. Existing techniques developed to provide security in P2P systems, such as [27], can be used. Additionally, trust and reputation must be dealt with when resources are accessed from a third party – enabling the use of techniques such as [28] may be considered. Trust issues are necessary to ensure that the capability being advertised by a provider is accurate, and reflects the true capability of the provider at the time.

The concept of *Routing Indices* (RI) [29] is used in order to forward queries to neighbors that are more likely to have the required resources. Forwarding decisions use the local RI value of neighbouring domains, rather than selecting neighbors at random or by flooding the network by forwarding the query to all neighbors.

3.1. Routing Indices

Routing Indices (RI) [29] were initially developed for document discovery in P2P systems, and have also been used to implement a Grid information service in [30]. The goal of RIs is to help users find documents with content of interest across potential P2P sources efficiently.

RI are used to make query forwarding decisions between domains in the system, and to avoid the need for flooding the entire network. The RI represents the availability of data of a specific type in the neighbor's information base. A version of RI called *Hop-Count Routing Index* (*HRI*) [29] is used, which considers the number of hops needed to reach a datum. This 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, as described in Eq. (1).

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

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 on the machine; $max\_num\_processes_i$ is the maximum number of processes that can be run on 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 is calculated by considering measurements obtained by SNMP [31], as pointed out in [6]. Predictions on the values of the current number of processes and the effective bandwidth can be used, for example, and calculated as pointed out in [6]. This formulation emphasizes that both computing and network capability are equally important, and both parameters must be considered in order to decide the *quality* of an administrative domain to run a particular job.

The $max\_num\_processes_i$ metric is used to determine the processing capability of a particular machine. It is calculated by considering the speed of the CPU and the amount of memory available at a machine. Eq. (2) shows the actual formula used.

$$max\_num\_processes = k_1 \times \frac{memory}{max(memory)} + k_2 \times \frac{cpu\_speed}{max(cpu\_speed)}. \tag{2}$$

In Eq. (2), k_1 and k_2 are two weighting constants that show the importance of each normalized parameter (memory and CPU speed) when calculating the maximum number of processes. Also, $k_1 + k_2$ represents the maximum number of processes the best of the machines should have. That is, if we consider the machine with the fastest CPU and the machine with the largest memory, $k_1 + k_2$ should represent the maximum number of processes in that machine. This is done in order to allow local administrators to set limits on the use of resources. The maximum value for memory and CPU speed must be propagated between peers, so that all the peers share the same values for them – thereby enabling a more objective comparison between resources held by different peers.

Eqs. (1) and (2) show why the two assumptions mentioned before are needed. As the effective bandwidth between domains is needed in Eq. (1), it is important that a domain correctly report its resource capabilities. Otherwise, the actual links used to transmit the job could not be accurately characteized. The second assumption requires that the domains must report the same monitoring metrics (such as CPU speed, current load and effective bandwidth), as otherwise no comparison could be made between domains.

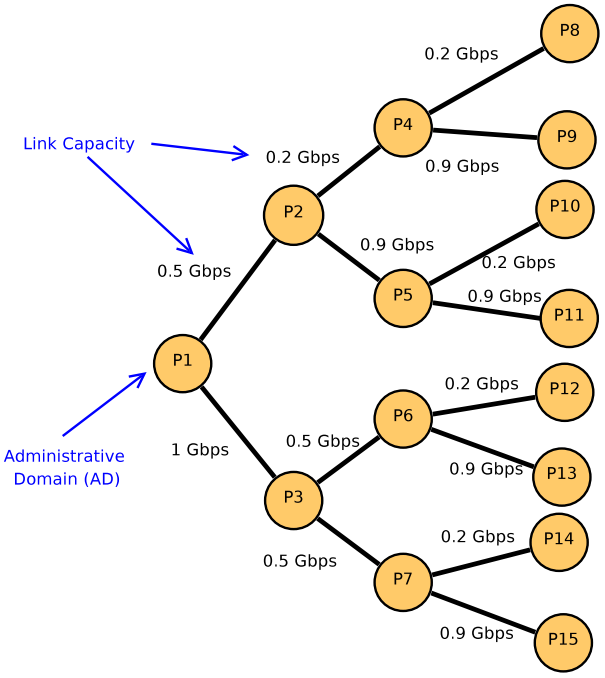


Fig. 4. Peer-to-peer relations between several administrative domains.

Table 2
Detailed HRI for peer P_1.

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

HRI have been used as described in [29]: 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 the Index: the nth position in the mth 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 are provided in Table 1 (for the topology depicted in Fig. 4), where $S_{x.y}$ is the value for peers that can be reached through peer x, and are y hops away from the local peer (in this case, P_1), and calculated as in Eq. (3). Hence, $S_{2.3}$ represents the quality of domains which can reached through peer P_2, whose distance from the local peer is 3 hops.

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

In Eq. (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 based on the distance from some 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, the information that each peer P_t (which is the neighbor of P_i) keeps about them has to be added. Hence, the HRI of peer P_1 will be calculated as shown in Table 2.

3.2. Goodness function

In order to use RIs, a key component is the *goodness function* [29], which is needed to decide the quality of each neighbor domain. This is done by considering the quality of peers that can be reached through each neighbor, and their distance from the local peer. In other words, for each direct neighbor of the local peer, the goodness function will decide how good each of them is.

For example, consider the topology depicted in Fig. 4. If peer P_1 needs to forward a job to one of its neighbors, it will have to decide between P_2 and P_3. So, P_1 will apply the goodness function to both of them, and one of them will be chosen. When applying the goodness function to P_2, the quality of peers that can be reached through it (namely P_4, P_5, P_8, P_9, P_{10}, and P_{11}) will be considered. In the same way, the quality of P_3 depends on the quality of P_6, P_7, P_{12}, P_{13},

Table 3
Calculation of I_p^l.

Peer	P_2	P_3
1 hop	$I_{P_2}^{P_1} = (\frac{100}{40} + \frac{100}{40}) * 0.5 = 2.5$	$I_{P_3}^{P_1} = (\frac{100}{40} + \frac{100}{40}) * 1 = 5$
2 hops	$I_{P_4}^{P_2} = (\frac{100}{40} + \frac{100}{40}) * 0.2 = 1$	$I_{P_6}^{P_3} = (\frac{100}{40} + \frac{100}{40}) * 0.5 = 2.5$
	$I_{P_5}^{P_2} = (\frac{100}{40} + \frac{100}{40}) * 0.9 = 4.5$	$I_{P_7}^{P_3} = (\frac{100}{40} + \frac{100}{40}) * 0.5 = 2.5$
3 hops	$I_{P_8}^{P_4} = (\frac{100}{40} + \frac{100}{40}) * 0.2 = 1$	$I_{P_{12}}^{P_6} = (\frac{100}{40} + \frac{100}{40}) * 0.2 = 1$
	$I_{P_9}^{P_4} = (\frac{100}{40} + \frac{100}{40}) * 0.9 = 4.5$	$I_{P_{13}}^{P_6} = (\frac{100}{40} + \frac{100}{40}) * 0.9 = 4.5$
	$I_{P_{10}}^{P_5} = (\frac{100}{40} + \frac{100}{40}) * 0.2 = 1$	$I_{P_{14}}^{P_7} = (\frac{100}{40} + \frac{100}{40}) * 0.2 = 1$
	$I_{P_{11}}^{P_5} = (\frac{100}{40} + \frac{100}{40}) * 0.9 = 4.5$	$I_{P_{15}}^{P_7} = (\frac{100}{40} + \frac{100}{40}) * 0.9 = 4.5$

Table 4
Example HRI for peer P_1.

Peer	P_2	P_3
1 hop	2.5	5
2 hops	5.5	5
3 hops	11	11

Table 5
Calculating *goodness* function.

$goodness(P_2) = \frac{2.5}{3^0} + \frac{5.5}{3^1} + \frac{11}{3^2} = 5.5$
$goodness(P_3) = \frac{5}{3^0} + \frac{5}{3^1} + \frac{11}{3^2} = 7.8$

P_{14}, and P_{15}. This is done by means of the HRI, since it keeps information on the peers that can be reached through each neighbor peer. Consider that the best resources belong to peer P_6, and all the resources belonging to the other peers are overloaded. In this case, P_1 would choose to forward the job to P_3, because although P_3 does not have a suitable resource, it is closer to P_6 than P_2.

The goodness function developed can be seen in Eq. (4), where p is the peer domain to be considered; H is the horizon for the HRIs; and F is the fanout of the topology. As explained in [29], *horizon* provides an upper bound on the distance (number of hops) searched; peers whose distance from the local peer is greater than the horizon will not be considered. Meanwhile, the *fanout* (F) of the topology is the maximum number of neighbors a peer has.

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

3.3. Example

The use of HRIs is demonstrated through an example based on the topology depicted in Fig. 4. Suppose that all the peers (recall that each peer represents an administrative domain) in Fig. 4 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 [6] suggests. Finally, the horizon is 3, and fanout is 3. In order to calculate the HRI of peer P_1, each I_p^l is calculated as shown in Table 3. These I_p^l produce the HRI depicted in Table 4 by adding the results presented in each cell of Table 3.

If a computing resource is required at peer P_1, then the goodness function in Table 5 is applied:

The goodness function produces a higher value for P_3 compared to P_2. This occurs because the network connection to P_3 makes it more suitable to execute jobs than P_2. Thus, the job would be forwarded to P_3.

3.4. Search technique

Several techniques are used for searching in P2P networks, including flooding (e.g. Gnutella) or centralized index 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 at each neighbor.

Algorithm 1 shows the way that the architecture performs the scheduling of jobs to computing resources. In this system, when a user wants to run a job, a query is submitted to the GNB of the local domain. This query is stored (line 7) as it arrives for the first time to a GNB. The GNB looks for a computing resource in the local domain matching the requirements

Algorithm 1. Search algorithm.

```
1: q: new incoming query
2: LocalResource: a resource in the local domain
3: NextBestNeighbor: a neighbor domain selected by the goodness function
4: 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 at this domain, store the query}
9:       QueryStatus(q) := 1
10:      {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 a computing resource has been found}
27:      SendResponseToRequester(q)
28:   end if
29: end for
```

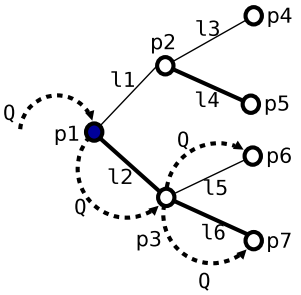


Fig. 5. A query (Q) is forwarded from $p1$ to the best neighbors ($p3$, $p6$, and $p7$).

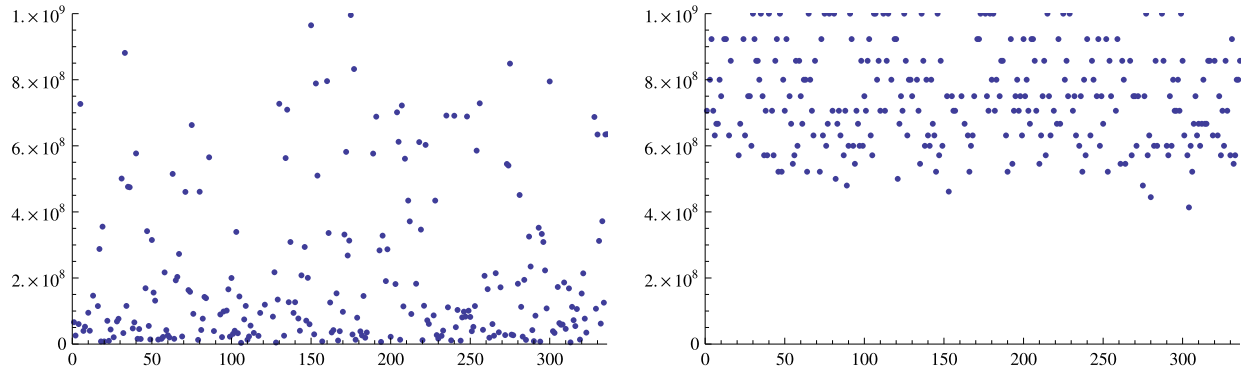
of the query (line 11). If the GNB finds a computing resource in the local domain that matches the requirements, then it tells the user to use that resource to run the job (line 27). Otherwise, the GNB will forward the query to the GNB of one of the neighbor domains. This neighbor domain will be chosen based on the *Hop-Count Routing Index, HRI,* explained before (line 16). The parameter *ToTry* is used to decide which neighbor should be contacted next, as shown in Fig. 5 (where $p3$ will contact $p6$); if the query is bounced back, then the 2nd best neighbor will be contacted ($p3$ will contact peer $p7$), and so on. Hence, a neighbor domain is only contacted if no suitable local computing resources are available.

4. Evaluation

Two types of evaluations have been undertaken to validate this approach. The first focuses on how HRIs evolve when varying system parameters – an evaluation from the point of view of the *system*. The second evaluation is carried out to evaluate the approach from the point of view of the *users*. The first evaluation is conducted in order to evaluate the general approach in a controlled environments. The aim is to validate whether the meta-scheduling approach uses network bandwidth effectively, in order to choose the most suitable neighbour peer to forward a query to. The second evaluation is more involved, and focuses on a users' perspective. Such an evaluation also enables ease of comparison with related approaches, such as GriDM and flooding.

4.1. System point of view

For the first evaluation, the topology presented in Fig. 5 is used; all the data presented here refer to peer $p1$. In the simplest case, all link bandwidths are assumed to be 1 Gbps, and all the peers have 1 resource with a single machine, with 4 Gb of memory and a CPU speed of 1 GHz.



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

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

Fig. 6. Variation of link usage.

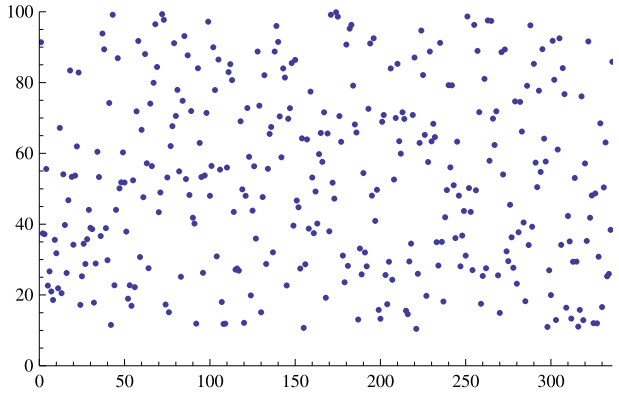


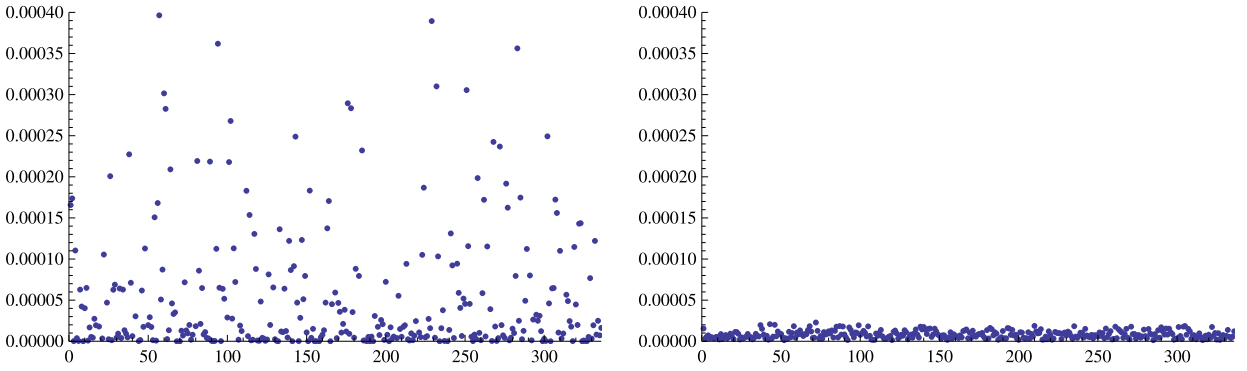
Fig. 7. Variation of the number of processes (Uniform distribution).

For Eq. (1), the values of $current\_num\_processes_i$ have been approximated as a uniform distribution between 10 and 100, and the $max\_num\_processes_i$ as 100. Regarding the $eff\_bw(l, p)$, a Poisson distribution has been considered for those links that are heavily loaded, and a Weibull distribution for those links which are not, as [32] suggests. In [32], an study is presented which shows that an increasing number of simultaneous active connections causes a dramatic change in the statistical properties of packet traffic on an Internet link. Starting at low connection loads on an uncongested link, packet arrivals tend toward Poisson as the load increases. Besides, this study also says that the marginal distribution of the inter-arrivals is well approximated by the Weibull distribution across all values of the average number of active connections.

In Fig. 5, links with even numbered labels will be heavily used, and are depicted with a thicker line. To calculate the mean μ for the Poisson distribution, and scale β and shape α for the Weibull distribution, it has been considered that the level of use of heavily used links is 80%, whilst less heavily used links exhibit a 10% usage. Hence if a heavily used link transmits 800 Mb/s and the maximum transfer unit of the links is 1500 bytes, then the inter-arrival time for packets is 0.000015 *seconds* – corresponding to μ of the Poisson distribution. In the same way, the value for the β parameter of the Weibull distribution is calculated to be 0.00012 *seconds*.

A measurement period of 7 days has been simulated, with measurements collected every 30 minutes. These measurements are represented by the horizontal axis in all Figs. 6, 7, 8 and 9. Figs. 6 and 7 show the variation in the use of links and the number of processes, following the mathematical distributions explained before. Fig. 6 represents the level of use of links compared to the actual bandwidth (1 Gbps), per measurement. This data is used to determine along which link a query may be forwarded.

Figs. 8 and 9 present the variation of the $S_{x.y}$ for both heavily/less heavily loaded links. These figures have been calculated by means of the formulas explained in Section 3.1, and applied to the mathematical distributions mentioned above. From Tables 1 and 2, $S_{2.1} = I_{p2}^{p1}$, and $S_{3.1} = I_{p3}^{p1}$. It can be seen that the network performance affects the HRI, as was expected. A higher HRI is better, as it means that the peer is powerful and well connected. Also, when the link is not heavily loaded, S takes higher values and has a greater spread. Conversely, when the link is heavily loaded, more values are grouped



(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).

Fig. 8. Variation of $S_{x.y}$ for loaded/unloaded links.

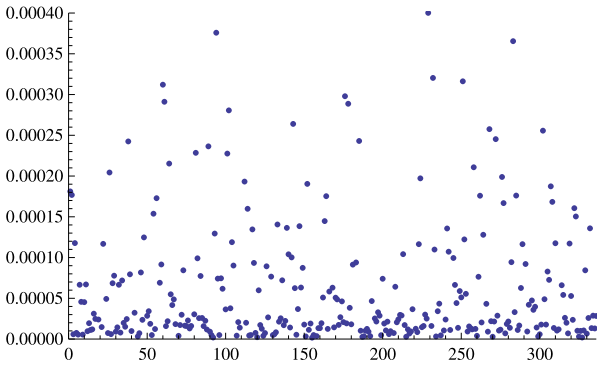
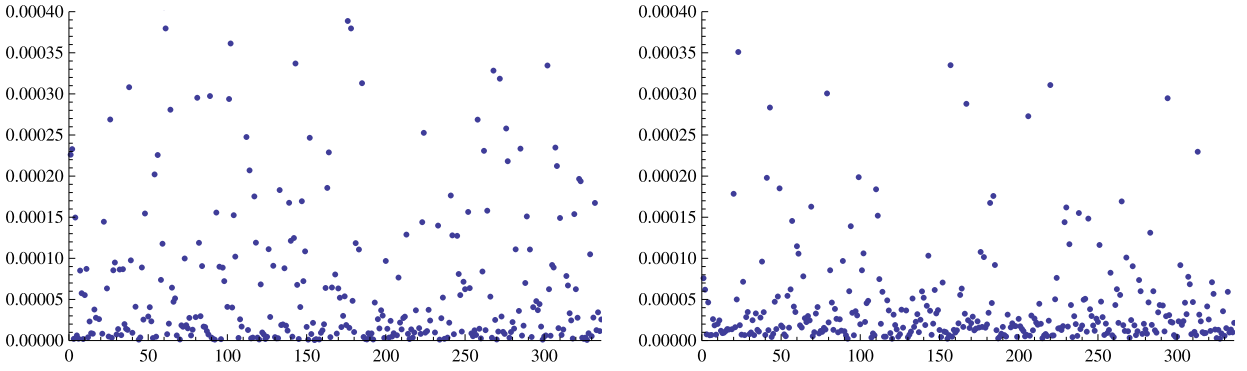


Fig. 9. $S_{2.2}$ ($S_{3.2}$ would also look like this).



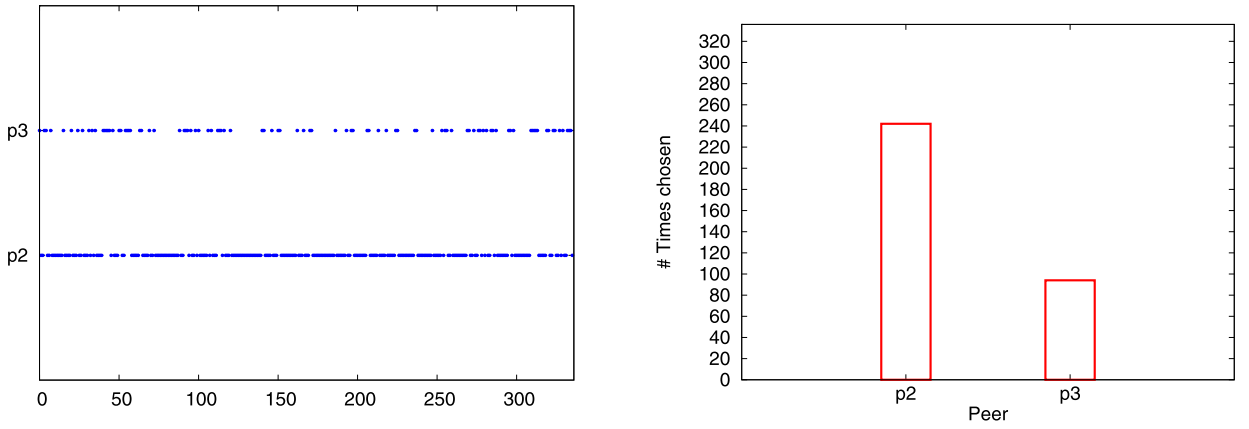
(a) Goodness function for peer $p2$ (link $p1 - p2$ unloaded).

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

Fig. 10. Variation of the goodness function.

together at the bottom of the figure. Also, for Fig. 9, $S_{2.2} = I_{P_4}^{P_2} + I_{P_5}^{P_2}$, and $S_{3.2} = I_{P_6}^{P_3} + I_{P_7}^{P_3}$, which means that to calculate $S_{2.2}$ and $S_{3.2}$, both heavily and less heavily used links are used.

Fig. 10 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. It can be seen that the goodness function for $p2$ has higher values, and for $p3$ it has more values grouped at the bottom of the figure. Thus, peer $p2$ will be chosen more often than $p3$. This is depicted in Fig. 11. Fig. 11(a) shows the peer that is chosen each time, and Fig. 11(b) shows an aggregate count of how often a particular peer was chosen. Peer $p2$ is chosen 242 times out of 336 (around 72%), and peer $p3$ is chosen 94 times (around 28%).



(a) Peer chosen. (b) Aggregate count for peer selection.

Fig. 11. Peer chosen (link $p1 - p2$ unloaded, link $p1 - p3$ loaded).

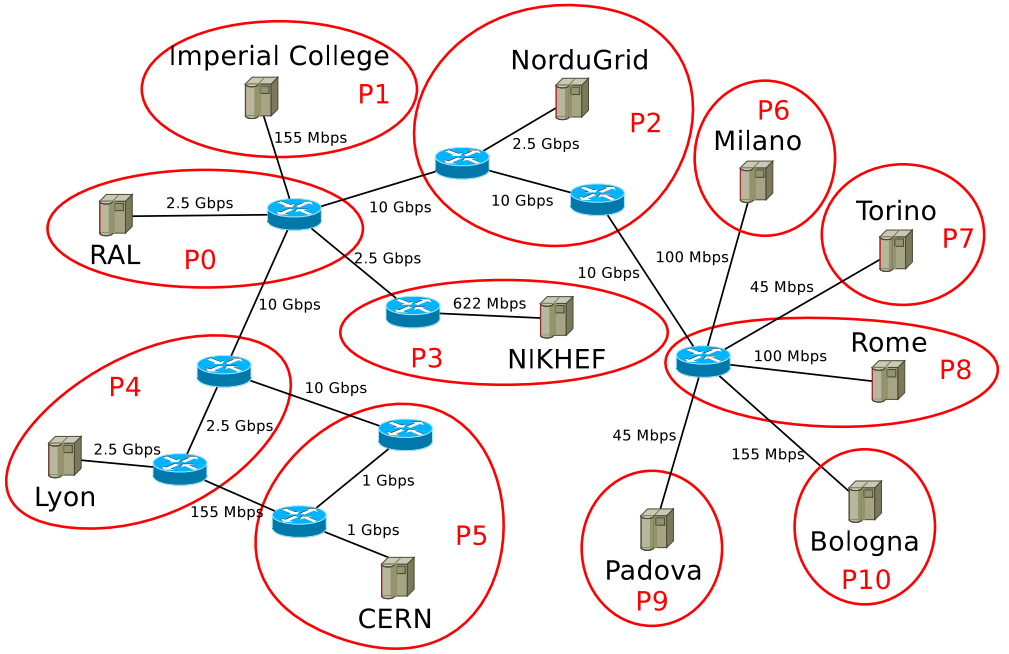


Fig. 12. EU DataGRID Testbed 1.

4.2. Users' point of view

A complementary evaluation from a users' point of view is now presented. This approach is compared with other approaches from literature, namely *GriDM* and *flooding*, which were explained in Section 2. The aim of such a comparison is to emphasise that the network is an important resource that influences the performance received by users in a Grid. Thus, approaches that do not consider the network will not perform as efficiently as possible. Besides, when a query must be forwarded, the process of finding a suitable destination must be performed in a scaleable manner, so that it can efficiently fit into such a dynamically changing environment. Furthermore, considering only the direct neighbors of an administrative domain (instead of the whole Grid system) considerably reduces the scope of the search, making the approach more scaleable.

4.2.1. Experiments and results

A network scenario based on the EU DataGRID Testbed has been created, as shown in Fig. 12 [33]. The original topology has been modified (3 links have been removed) to avoid loops when constructing Routing Indices (the issue of keeping HRI working and avoiding loops has already been described in [29], but this is not related to this Grid meta-scheduling proposal). The topology shows eleven computing resources spanning several locations in Europe. Each location is an ad-

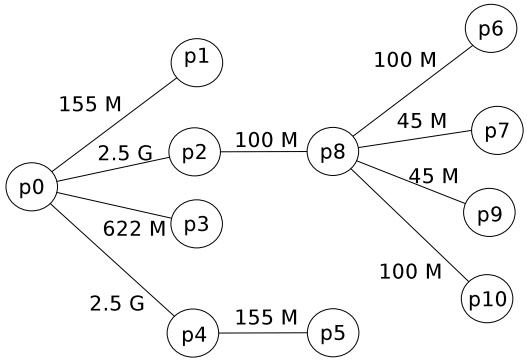


Fig. 13. Peer-to-peer topology.

Table 6
Resource specifications.

Peer ID	Res. (Location)	# Nodes	CPU Rating	# Users
0	RAL (UK)	41	49,000	12
1	Imp. College (UK)	52	62,000	16
2	NorduGrid (Norway)	17	20,000	4
3	NIKHEF (Netherlands)	18	21,000	8
4	Lyon (France)	12	14,000	12
5	CERN (Switzerland)	59	70,000	24
6	Milano (Italy)	5	70,000	4
7	Torino (Italy)	2	3000	2
8	Rome (Italy)	5	6000	4
9	Padova (Italy)	1	1000	2
10	Bologna (Italy)	67	80,000	12

ministrative domain, with the structure shown in Fig. 3. For the sake of clarity, only routers and computing resources are depicted.

Boundaries between administrative domains are shown in circles in Fig. 12, and the bandwidth of the link connecting the GNB is the same as that of the computing resource in that domain. Hence, the connectivity structure leads to the P2P topology depicted in Fig. 13, where links between peers are the bottleneck of the network paths between GNBs. From now on, link bandwidths mentioned in this section are those appearing in Fig. 13. The three proposals (ID-GNB, GriDM and flooding) have been implemented in GridSim. The following decisions have been made:

- Meta-scheduling is performed in *meta-scheduling rounds*, with an interval of 20 seconds.
- The monitoring of neighbors (ID-GNB and GriDM) is undertaken every 10 seconds. This has been chosen to allow 2 monitoring rounds to complete for every meta-scheduling round, so that more accurate information on the status of the neighbors is compiled.
- 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, the load of the computing resource provided by GridSim is used to decide which neighbor a query must be forwarded to. The least loaded computing resource is chosen each time.
- Several computing resources have full local (non-Grid) computing load, in the same way as in the intra-domain scenario. These computing resources are `Res_0`, `Res_1`, `Res_2`, `Res_3`, `Res_4`, and `Res_5`. Their local load covers around 95% of the computing power of the resources. That is, only around 5% of the computing power of each CPU at those resources is available for Grid users. For the other resources, the local load is nearly 0%. This has been decided in order to simulate a real Grid scenario, in which resources may have local load that may differ between resources.

Table 6 summarizes the characteristics of simulated resources, which were obtained from a real LCG testbed [34]. The CPU rating is defined in MIPS (*Millions of Instructions Per Second*) as per SPEC (*Standard Performance Evaluation Corporation*) benchmark. The number of nodes for each resource have been scaled down by 10, due to memory limitations – otherwise the full experiment would require more than 2 GB of memory, and would take several weeks of processing. Finally, each resource node has four CPUs. For this experiment, 100 users were created and distributed among the locations, as shown in Table 6. Each user has multiple jobs, with the processing power of each job being 1,400,000 *Million Instructions* (*MI*), which

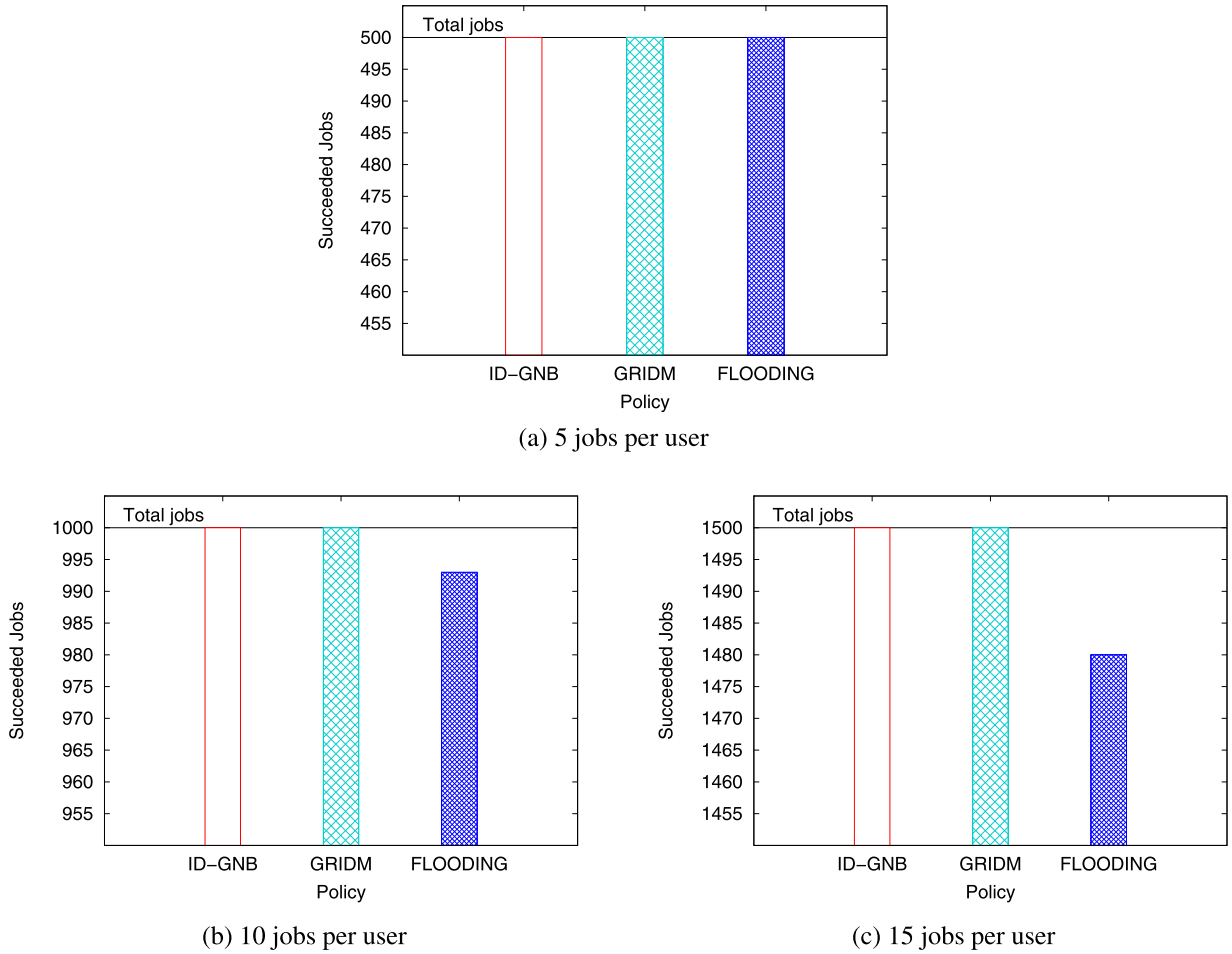


Fig. 14. Number of succeeded jobs.

means that each job takes about 2 seconds if it is run on the CERN resource. Also, I/O file sizes are 24 MB. All jobs have the same parameters that are taken from ATLAS online monitoring and calibration system [35].

This experiment is aimed at determining the behavior of the inter-domain meta-scheduling algorithm, and determining 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. Statistics related to the amount of data transferred between peers to keep HRIs up-to-date are also presented.

Fig. 14 shows the number of jobs that were successfully completed for each inter-domain meta-scheduling policy, as the number of jobs each user wants to run varies. It can be seen that there is no difference between the use of GriDM and ID-GNB approaches, since both of them can find a computing resource for all the jobs in all the experiments. On the other hand, as the number of jobs per user increases, there is an increase in the number of jobs in the flooding approach that cannot be allocated to any computing resource. Hence, these jobs remain unexecuted.

Now consider Fig. 15, which depicts the number of queries forwarded per successfully completed job. This statistic has been calculated by dividing the actual number of queries forwarded by the number of successfully completed jobs. This statistic therefore 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. Comparing with ID-GNB and GriDM, ID-GNB shows the smallest values for this statistic, and the difference gets larger as the number of jobs per user increases. For the case of 15 jobs per user, ID-GNB requires 30% less queries than GriDM, for the same amount of successful jobs.

Fig. 16 shows the amount of data forwarded through the network, and includes ping requests made from one peer to another, in order to support meta-scheduling. However, flooding does not require such information. As expected, ID-GNB requires less bytes to be forwarded, since it only requires information from the neighbors. Conversely, GriDM requires information from all the peers, thus increasing the amount of information forwarded through the network.

Fig. 17 illustrates the number of bytes transferred through the network in queries. This is calculated as the sum of the size of each query that is propagated through the system. Each query has a number of parameters, including a user

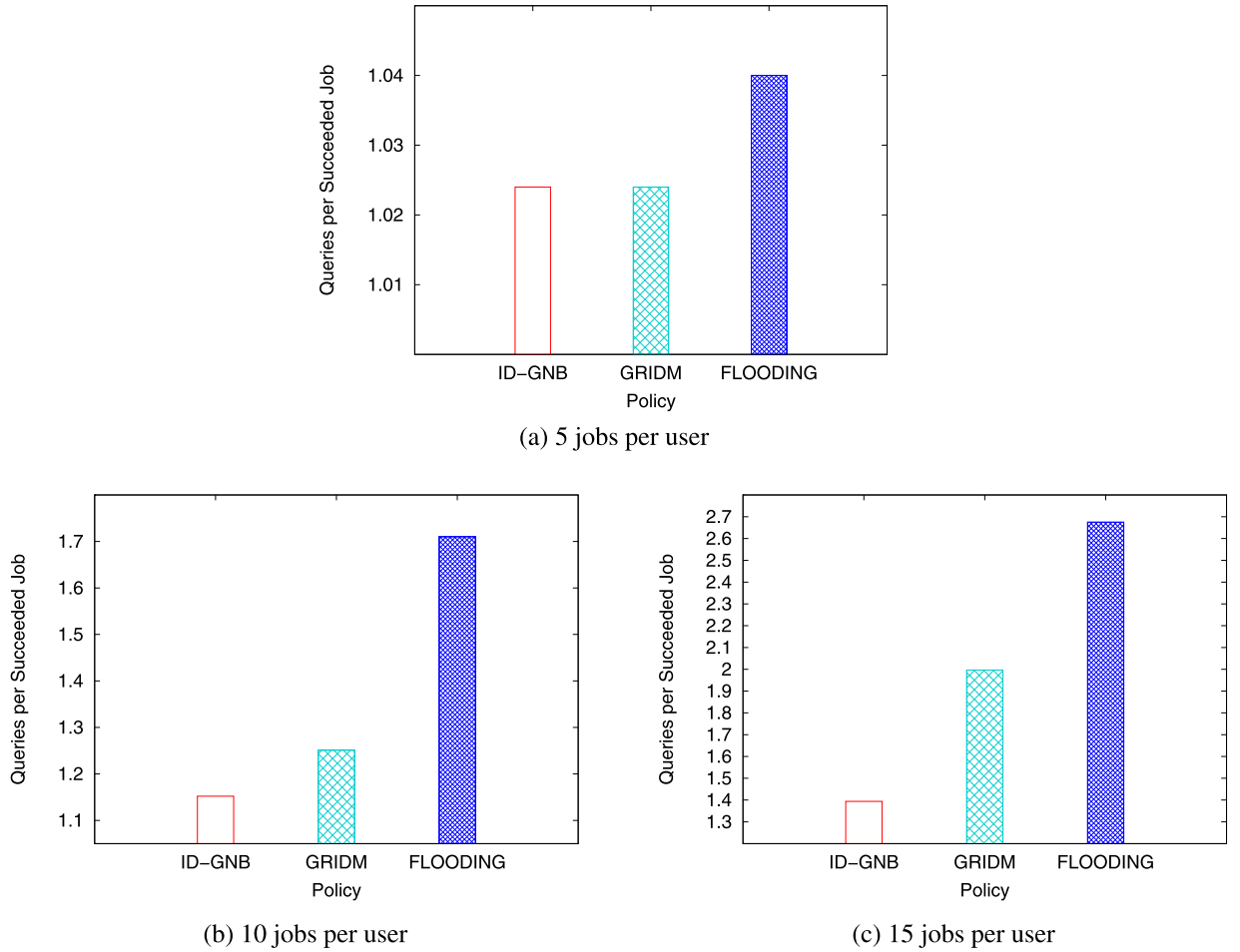


Fig. 15. Number of queries per succeeded job.

identity, identity of the computing resource chosen to run the job, TTL, identity of the GNB that forwarded the query to the current GNB, the size of the job, and sizes of input and output files. All of these parameters lead to a job request-object size of 60 bytes. The results in Fig. 17 show that when the number of jobs per user is small, there are negligible differences between strategies, but as the number of jobs per user increases, differences increase as well. As was expected, the flooding approach has the highest value for this statistic, followed by GriDM and ID-GNB respectively.

The number of jobs executed in each computing resource for 15 jobs per user is depicted in Fig. 18. When there is a smaller number of jobs per user (5 and 10 jobs per user) there are negligible differences between strategies since all the resources run the same number of jobs in each case (these results are not therefore not included here). But when each user has 15 jobs (see Fig. 18) differences clearly arise. In this last case, it can be seen that there are some computing resources that execute a high number of jobs for all the strategies (namely, `Res_0` (RAL), `Res_1` (Imperial College), `Res_5` (CERN) and `Res_10` (Bologna)). These are the most powerful computing resources, since they have more nodes than others.

Apart from this observation, when ID-GNB is being used (see Fig. 18(a)), resource `Res_4` (Lyon) runs around 175 jobs, a considerably higher number of jobs than when GriDM is used (Fig. 18(b)). This is because this resource has a higher bandwidth link of 2.5 GB (see Fig. 13) which does not get overloaded. As Routing Indices are heavily influenced by the effective bandwidth of a link, this makes `Res_4` a good candidate to execute jobs.

When ID-GNB is being used, resources `Res_6` (Milano) and `Res_7` (Torino) execute hardly any jobs, as opposed to the case when GriDM is running. This is explained by the fact that their links have low bandwidth, thus ID-GNB does not consider them as good candidates to run jobs. But they have low local load, thus GridM considers them as good candidates to run jobs.

For flooding (depicted in Fig. 18(c)), it can be seen that although computing resource `Res_8` (Rome) is less powerful, it executes more jobs than resource `Res_6` (Milano). This is because `Res_8` has 5 neighbors (as can be seen in Fig. 13). It therefore gets flooded with queries from them, and whenever its computing resource gets idle, another request arrives and is accepted for execution at that resource.

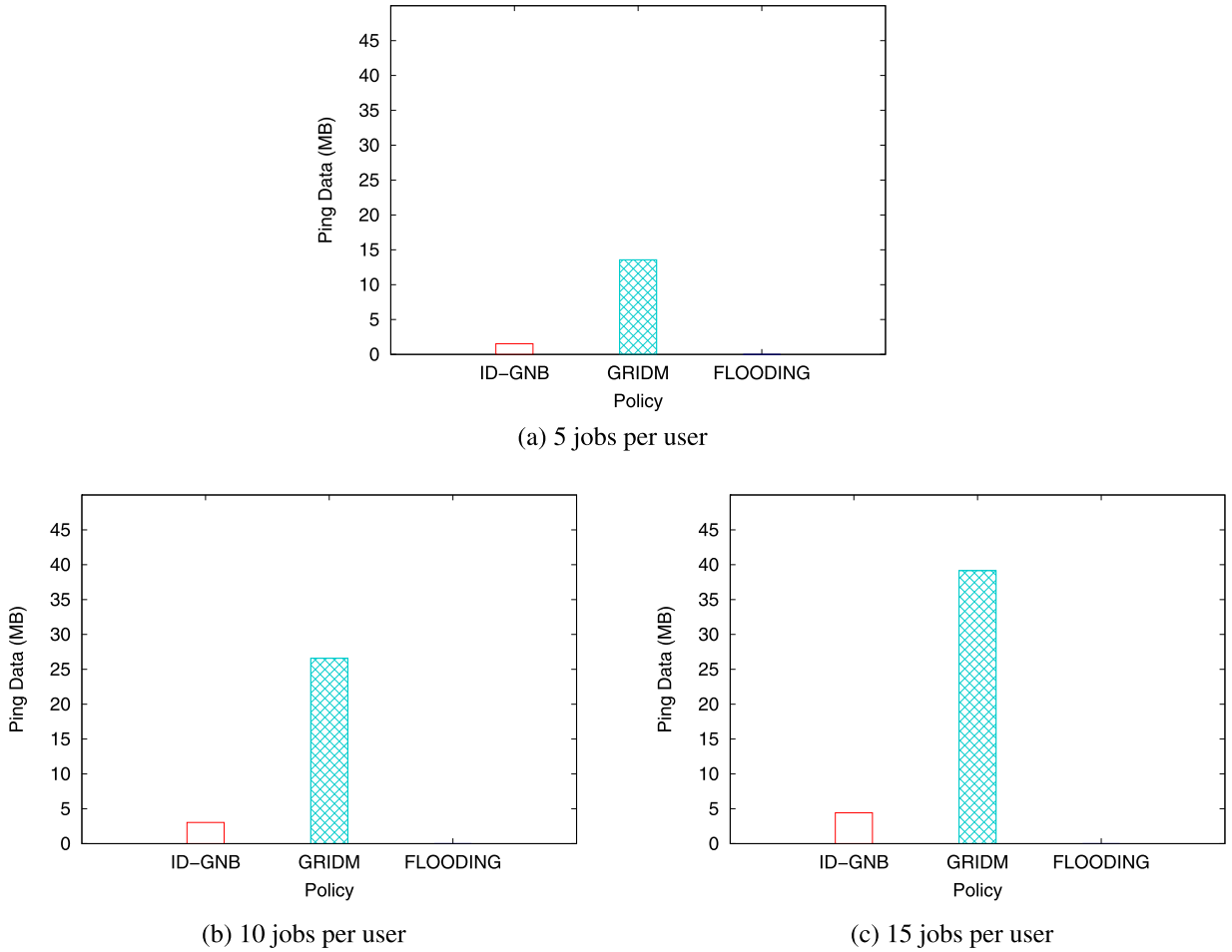


Fig. 16. Data forwarded through the network when getting information.

Fig. 19 depicts the average network latencies of jobs. This statistic is calculated for each job, for example, the average network latency is calculated for job 0 for all the users. This is undertaken to demonstrate latencies of different jobs which were submitted in the same order (jobs with the same number for all the users). As before, when each user has 5 jobs (represented in Fig. 19(a)), differences between approaches are negligible (being ID-GNB slightly worse). When each user has 10 jobs (represented in Fig. 19(b)), GriDM and flooding approaches show similar results, and ID-GNB performs better. The reason is that as data traffic increases, network performance becomes more important 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 Fig. 15(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 Fig. 19(c)), the average network 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 one. This is confirmed by the number of queries per successfully completed job (presented in Fig. 15(c)). Also, flooding presents similar latencies than ID-GNB, because of the nature of flooding. Recall that with flooding, every 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 Fig. 15(c)) and a greater amount of interchanged information in queries (presented in Fig. 17(c)).

Fig. 20 depicts the average wall clock time for jobs, and represents the total time jobs spend in computing resources, including waiting time (when no CPU is idle when the job arrives at the resource), and execution time. As before, when each user has 5 jobs (Fig. 20(a)) differences between strategies are negligible. When users have 10 jobs (Fig. 20(b)) differences start to arise, and ID-GNB performs slightly worse than the other strategies. When users have 15 jobs (Fig. 20(c)), GriDM shows the best results. This is explained by the fact that GriDM always chooses the least loaded computing resource. But differences are almost negligible (a few tens of seconds), compared with the differences in network latencies (shown in Fig. 19).

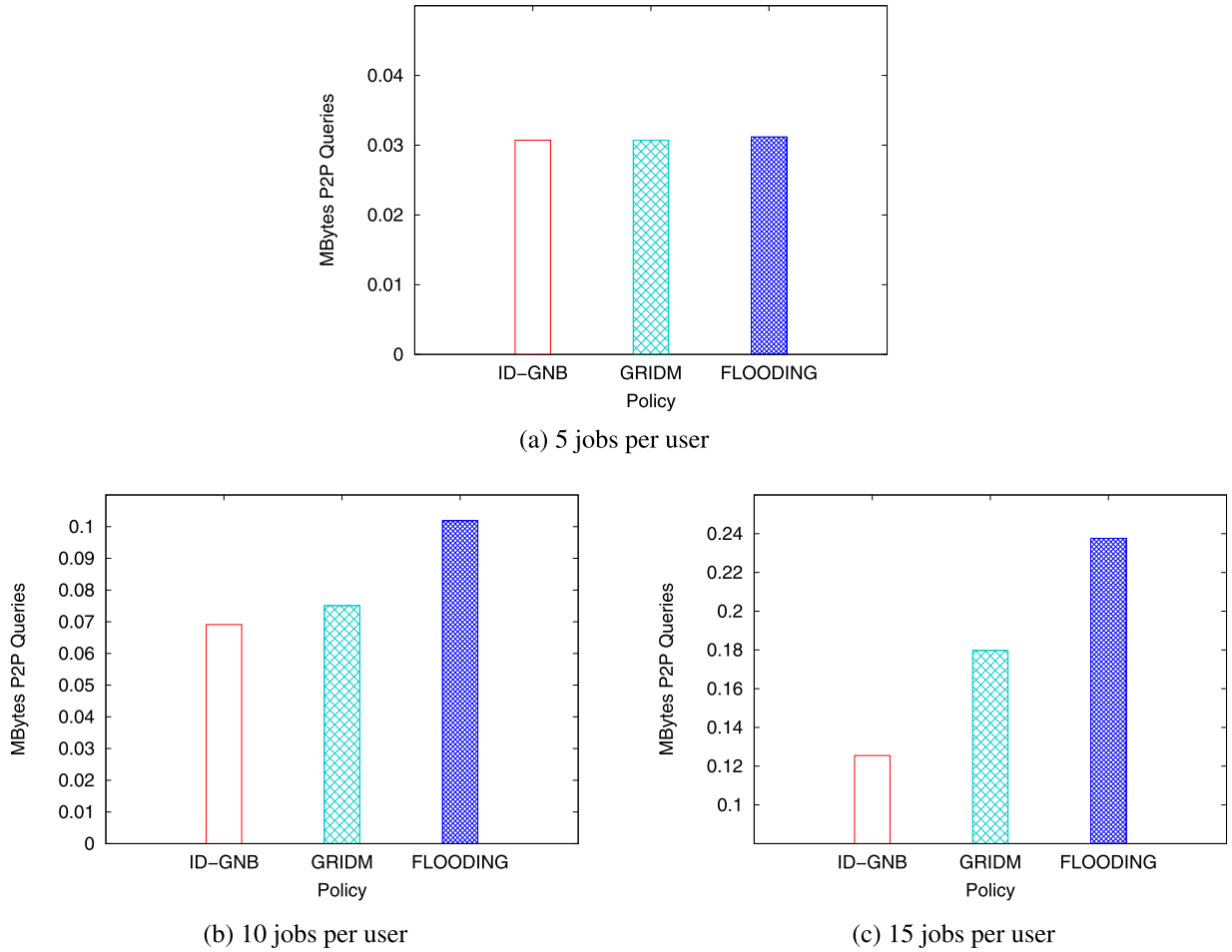


Fig. 17. Data forwarded through the network in queries.

The last statistic presented is the average total latency for jobs, which is shown in Fig. 21. 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, queueing time at the resource (if no CPU is idle at the moment), and the execution time of the job. Thus, it is the result of adding the statistics presented in Figs. 19 and 20. They show a similar trend as network latencies (presented in Fig. 19).

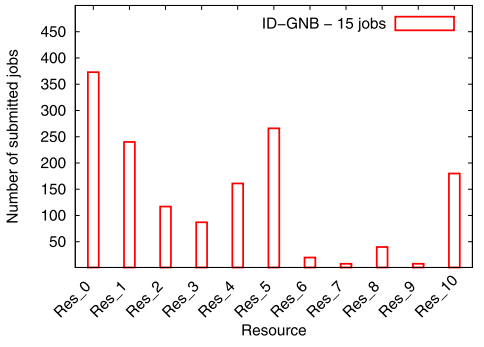
The results of this evaluation show that ID-GNB outperforms both GriDM and flooding in terms of number of queries required for each job, and network and total latency times. ID-GNB achieves a better rate of successfully completed jobs and lower latencies, with less queries per job. It is also useful to note that less information must be sent through the network to keep the architecture working. This therefore demonstrates that ID-GNB is scalable, hence it is a more appropriate technique for realistic Grid environments.

5. Conclusions

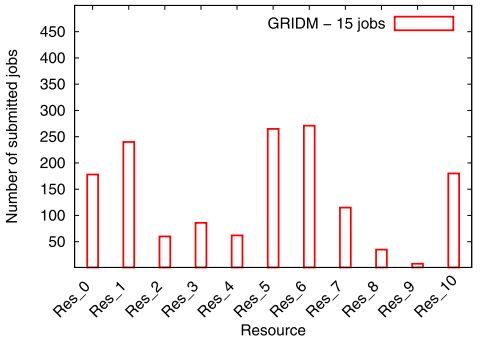
Grid computing involves use of resources in different administrative domains connected with each other. Thus, relations between domains are key, and must be considered when performing meta-scheduling. An extension to an existing meta-scheduling framework has been proposed to allow network-aware multi-domain meta-scheduling based on peer-to-peer techniques.

More precisely, the proposal is based on *Routing Indices* (RI). Using this approach, nodes are allowed to forward job queries to neighbors that are more likely to have suitable computing resources. 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 HRI, rather than by selecting neighbors at random or by flooding the network by forwarding the query to all neighbors.

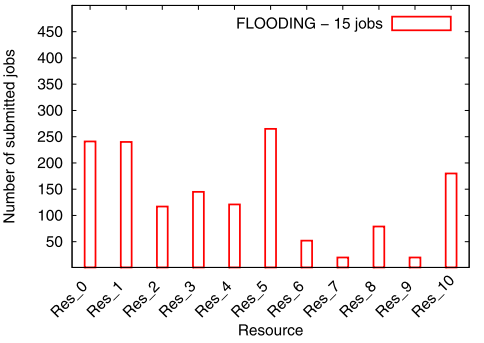
Results presented here demonstrate the better performance and the scalability of *Inter-Domain GNB*, *ID-GNB*. The results of the evaluation show that ID-GNB outperforms existing approaches in terms of number of queries required for each job,



(a) ID-GNB

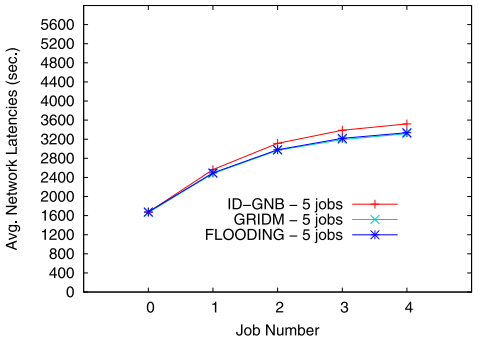


(b) GriDM

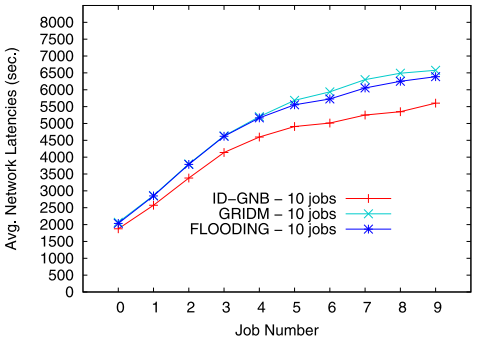


(c) Flooding

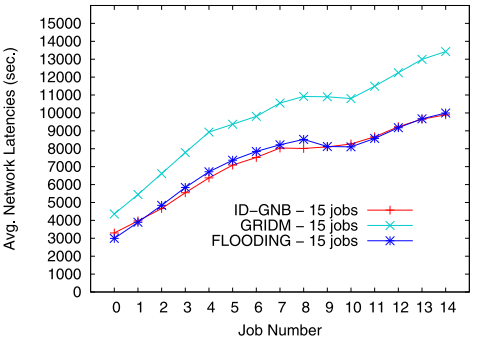
Fig. 18. Number of jobs submitted to each computing resource, for 15 jobs per user.



(a) 5 jobs per user

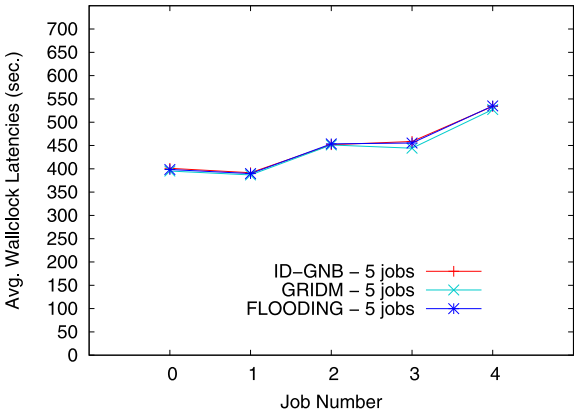


(b) 10 jobs per user

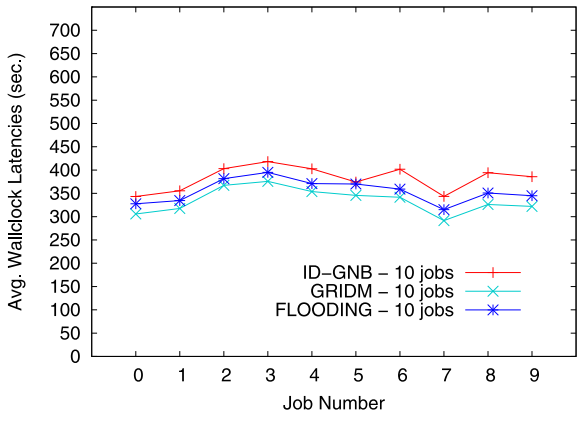


(c) 15 jobs per user

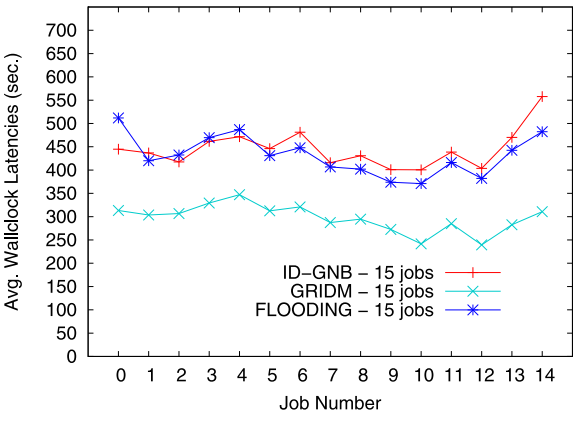
Fig. 19. Average network latencies.



(a) 5 jobs per user



(b) 10 jobs per user



(c) 15 jobs per user

Fig. 20. Average wallclock latencies.

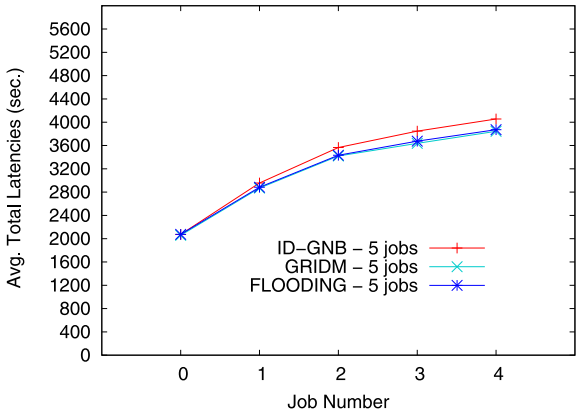
network time and total latency. ID-GNB achieves better rate of succeeded jobs and better latencies, with less queries per job. Also, less overhead is caused to keep the infrastructure working, which makes this strategy scalable.

6. Future work

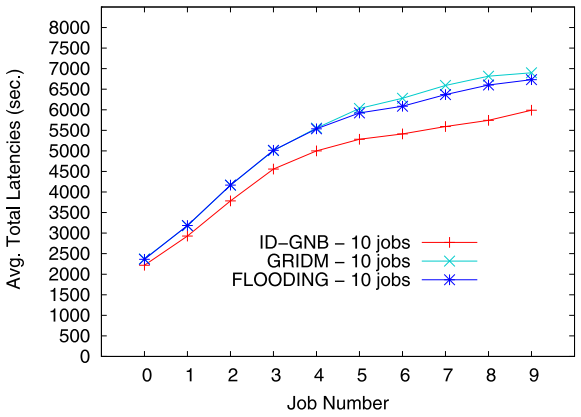
In the presented approach a combination of data on the status of the network and the status of the computing resources is used to perform inter-domain meta-scheduling, but this may not be enough. For example, a job may have the following requirements: ⟨OS = Linux, SW = Java 5, MatLab 7, HW = 200 GB available hard disk⟩. In this case, the most powerful unloaded computing resource, whose network is also powerful and unloaded cannot execute this job unless this computing resource fulfills the requirements of the job. Trying to identify how job properties precisely match resource properties has already been the subject of considerable research in matchmaking (e.g. Condor ClassAds). Without the use of such approaches, utilizing the inter-domain scenario presented here still has limitations, as GNBs must decide which information to provide to their neighbor GNBs, and this must be done in an efficient and scalable manner. The use of a summarization process may be a useful approach to summarize capabilities of multiple resource properties or job requirements, before forward requests to neighbors. Thus, further research can be conducted following this direction.

Another further extension is related to how the network topology is created. Generally peer-to-peer connectivity utilizes physical topology, which may not be the most efficient for query forwarding. Consider the case when different peers with similar characteristics are connected to opposite ends of the topology. If a query needs to be propagated and an intermediate peer is busy, the query may have to go through the whole system in order to reach the required peer. The creation of an overlay networks [36] based on node-content attributes would be useful, as peers with similar characteristics would be neighbors.

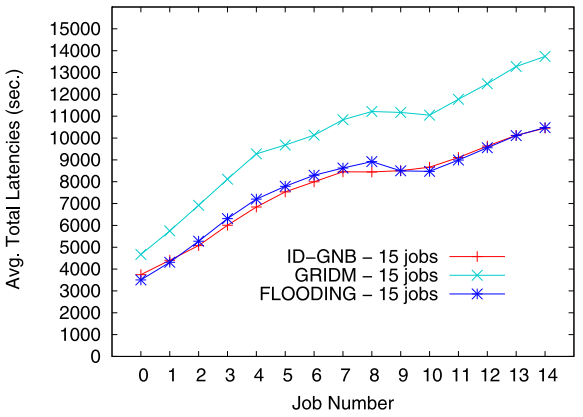
This paper demonstrates the influence of the network on the performance received by users, so it must be taken into account when providing QoS in Grid application. The only way to provide real QoS is by means of advanced reservations, including network, CPU and/or storage – and supported through *meta-scheduling* for future time slots, i.e. performing the



(a) 5 jobs per user



(b) 10 jobs per user



(c) 15 jobs per user

Fig. 21. Average total latencies.

meta-scheduling before jobs are actually submitted. Thus, the creation of a complete QoS framework supporting meta-scheduling in advance, and reservations of resources (with an special focus on the network resources) are also important tasks to support further efforts in this area.

Acknowledgments

This work has been jointly supported by the Spanish MEC and European Commission FEDER funds under grants “Consolider Ingenio-2010 CSD2006-00046” and “TIN2009-14475-C04-03”; jointly by JCCM and Fondo Social Europeo under grant “FSE 2007-2013”, and a post-doctoral contract; and by JCCM under grants “PBI08-0055-2800” and “PII1C09-0101-9476”.

References

[1] I.T. Foster, 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.
[2] I. Foster, C. Kesselman, The Grid 2: Blueprint for a New Computing Infrastructure, second ed., Morgan Kaufmann, 2003.
[3] CERN, LHC Computing, Web page at http://www.interactions.org/LHC/computing/index.html. Date of last access: 10th July, 2009.
[4] A. Roy, End-to-end quality of service for high-end applications, PhD thesis, Dept. of Computer Science, University of Chicago, 2001.
[5] F.T. Marchese, N. Brajkovska, Fostering asynchronous collaborative visualization, in: Proc. of the 11th Intl. Conference on Information Visualization, Washington DC, USA, 2007.
[6] A. Caminero, O. Rana, B. Caminero, C. Carrión, Performance evaluation of an autonomic network-aware metascheduler for Grids, Concurrency and Computation: Practice and Experience 21 (2009) 1692–1708.
[7] S. Bhatti, S. Sørensen, P. Clark, J. Crowcroft, Network QoS for Grid systems, Intl. J. High Performance Comput. Appl. 17 (3) (2003) 219–236.
[8] F. Barrère, A. Benzekri, F. Grasset, R. Laborde, B. Nasser, Automated inter-domain security policy generation, in: Proc. of the 11th Workshop of the HP OpenView University Association, Paris, 2004.
[9] B.E. Carpenter, P.A. Janson, Abstract interdomain security assertions: A basis for extra-Grid virtual organizations, IBM Systems Journal 43 (4) (2004) 689–701.

[10] E. Magaña, J. Serrat, Distributed and heuristic policy-based resource management system for large-scale Grids, in: Proc. of the First Intl. Conference on Autonomous Infrastructure, Management and Security, (AIMS), Oslo, Norway, 2007.
[11] Y. Zhao, Y. An, C. Wang, Y. Gao, A QoS-satisfied interdomain overlay multicast algorithm for live media service Grid, in: Proc. of the 4th Intl. Conference on Grid and Cooperative Computing (GCC), Beijing, China, 2005.
[12] D. Talia, P. Trunfio, A P2P Grid services-based protocol: Design and evaluation, in: Proc. of the 10th Intl. Euro-Par Conference, Pisa, Italy, 2004.
[13] Z. Xiong, Y. Yang, X. Zhang, M. Zeng, Grid resource aggregation integrated P2P mode, in: Proc. of the 4th Intl. Conference on Intelligent Computing (ICIC), Shanghai, China, 2008.
[14] D. Xu, K. Nahrstedt, D. Wichadakul, 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.
[15] X. Gu, K. Nahrstedt, 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.
[16] A. O'Brien, S. Newhouse, J. Darlington, Mapping of scientific workflow within the e-protein project to distributed resources, in: UK e-Science All-hands Meeting, Nottingham, UK, 2004.
[17] L.J. McGuffin, R.T. Smith, K. Bryson, S.A. Sorensen, D.T. Jones, High throughput profile–profile based fold recognition for the entire human proteome, BMC Bioinformatics 7 (2006) 288–299.
[18] Gnutella, Web page at http://rfc-gnutella.sourceforge.net/. Date of last access: 10th July, 2009.
[19] A. Anjum, R. McClatchey, H. Stockinger, A. Ali, I. Willers, M. Thomas, M. Sagheer, K. Hasham, O. Alvi, DIANA scheduling hierarchies for optimizing bulk job scheduling, in: Proc. of the Second Intl. Conference on e-Science and Grid Computing, Amsterdam, Netherlands, 2006.
[20] LCG (LHC Computing Grid) Project, Web page at http://lcg.web.cern.ch/LCG. Date of last access: 10th July, 2009.
[21] GridPP, real time monitor, Web page at http://gridportal.hep.ph.ic.ac.uk/rtm/. Date of last access: 10th July, 2009.
[22] M.D. de Assunção, R. Buyya, S. Venugopal, InterGrid: A case for internetworking islands of Grids, Concurrency and Computation: Practice and Experience.
[23] M.L. Massie, B.N. Chun, D.E. Culler, The Ganglia distributed monitoring system: Design, implementation, and experience, Parallel Comput. 30 (5–6) (2004) 817–840.
[24] S. Sohail, K.B. Pham, R. Nguyen, S. Jha, Bandwidth broker implementation: Circa-complete and integrable, in: Tech. Rep., School of Computer Science and Engineering, The University of New South Wales, 2003.
[25] S. Fitzgerald, I. Foster, C. Kesselman, G. von Laszewski, W. Smith, S. Tuecke, A directory service for configuring high-performance distributed computations, in: Proc. 6th Symposium on High Performance Distributed Computing (HPDC), Portland, USA, 1997.
[26] Y. Rekhter, T. Li, S. Hares, A Border Gateway Protocol 4 (BGP-4), Internet proposed standard RFC 4271 (January 2006).
[27] K. Berket, A. Essiari, A. Muratas, PKI-based security for peer-to-peer information sharing, in: 4th Intl. Conference on Peer-to-Peer Computing, P2P 2004, Zurich, Switzerland, 2004.
[28] T.G. Papaioannou, G.D. Stamoulis, Reputation-based policies that provide the right incentives in peer-to-peer environments, Computer Networks 50 (4) (2006) 563–578.
[29] A. Crespo, H. Garcia-Molina, Routing Indices for peer-to-peer systems, in: Proc. of the Intl. Conference on Distributed Computing Systems (ICDCS), Vienna, Austria, 2002.
[30] D. Puppin, S. Moncelli, R. Baraglia, N. Tonellotto, F. Silvestri, A Grid information service based on peer-to-peer, in: Proc. of the 11th Intl. Euro-Par Conference, Lisbon, Portugal, 2005.
[31] K. McCloghrie, M.T. Rose, Management information base for network management of TCP/IP-based internets: MIB-II, RFC 1213 (March 1991).
[32] J. Cao, W. Cleveland, D. Lin, D. Sun, Nonlinear Estimation and Classification, Springer Verlag, New York, USA, 2002, Ch. Internet traffic tends toward Poisson and independent as the load increases.
[33] W. Hoschek, F.J. Janez, A. Samar, H. Stockinger, K. Stockinger, Data management in an international Data Grid project, in: Proc. of the 1st Intl. Workshop on Grid Computing, Bangalore, India, 2000.
[34] LCG Computing Fabric Area, Web page at http://lcg-computing-fabric.web.cern.ch. Date of last access: 10th July, 2009.
[35] ATLAS online monitoring and calibration system, Web page at http://dissemination.interactive-grid.eu/applications/HEP. Date of last access: 10th July, 2009.
[36] D. Doval, D. O'Mahony, Overlay networks: A scalable alternative for P2P, IEEE Internet Computing 7 (4) (2003) 79–82.