On Dimensioning Optical Grids and the Impact of Scheduling

C. Develder[1], B. Dhoedt[1], B. Mukherjee[2], P. Demeester[1]

1: Dept. of Information Technology (INTEC), Ghent University – IBBT
G. Crommenlaan 8 bus 201, BE-9050 Ghent, Belgium
email: {chris.develder, bart.dhoedt, piet.demeester}@intec.ugent.be

2: Dept. of Computer Science, University of California
1 Shields Av., 2063 Kemper Hall, Davis CA 95616, CA, USA
email: mukherje@cs.ucdavis.edu

***Abstract*—When deploying Grid infrastructure, the problem of dimensioning arises: how many servers to provide, where to place them, and which network to install for interconnecting server sites and users generating Grid jobs? In contrast to classical optical network design problems, it is typical of optical Grids that the destination of traffic (jobs) is not known beforehand. This leads to so-called anycast routing of jobs. For network dimensioning, this implies the absence of a clearly defined (source,destination)-based traffic matrix, since only the origin of Grid jobs (and their data) is known, but not their destination. The latter depends not only on the state of Grid resources, including network, storage, and computational resources, but also the Grid scheduling algorithm used. We present a phased solution approach to dimension all these resources, and use it to evaluate various scheduling algorithms in two European network case studies. Results show that the Grid scheduling algorithm has a substantial impact on the required network capacity. This capacity can be minimized by appropriately choosing a (reasonably small) number of server site locations: an optimal balance can be found, in between the single server site case requiring a lot of network traffic to this single location, and an overly fragmented distribution of server capacity over too many sites without much statistical multiplexing opportunities, and hence a relatively large probability of not finding free servers at nearby sites.**

***Index Terms*—Optical networks, Grids, anycast, dimensioning, ILP, simulation**

1. Introduction

GRIDS originated from the eScience community dealing with large experimental data sets (particle physics, astrophysics etc.): to meet computational and storage demands, cluster centers were interconnected via networks to achieve a huge common resource pool to process the tasks (jobs). Yet, also business/consumer oriented applications can benefit from Grid infrastructure. Consider high-definition (HD) video editing: applying effects, requiring one or multiple operations per pixel, or en/decoding these high resolution and high frame rate image streams quickly leads to a non-negligible amount of processing which already is challenging on today's PCs. In such cases, also considering evolutions to more thin client based consumer solutions, Grids offering off-site computational and storage capacity also make sense for business/consumer solutions. Both high data rates typical of eScience applications and low latency requirements of consumer/business applications (with their typically high degree of interactivity) can effectively be addressed by optical network technology interconnecting users and resources. Thus, Grids based on optical network infrastructure promise to offer cost and resource efficient delivery of network services with possibly high data rate, processing and storage demands, for a geographically widely dispersed user base. To fulfill that promise, fundamental questions need to be addressed, including (re)designing the architecture of a flexible optical layer, dimensioning and routing/scheduling algorithms. Fundamental differences with traditional network design arise from for instance traffic volume being dependent on dimensions and locations of computational/storage resources, as well as the job scheduling algorithm, and the fact that Grid users generally do not care where their jobs are processed (that is, destinations are unknown, hence there is no clearly defined traffic matrix). These Grid specific aspects give rise to multiple challenging research questions [1, 2].

For the optical network architecture, whether to adopt Optical Circuit Switching (OCS) or Optical Packet/Burst Switching (OPS/OBS) is debatable. Depending on the ratio signaling time/job transmission time, OCS can be acceptable [3]. For small jobs, complex grooming/aggregation at the OCS edges will be required. As job data size reduces and/or latency-sensitivity increases, OBS will be more efficient [4]. Another advantage of OBS is its ease in dealing with highly dynamic traffic patterns (both in space and time). The methodology proposed in this paper addresses both OBS and OCS alternatives.

Given the optical network architecture of choice, the so-called anycast routing principle has a major impact on the scheduling and routing decision: how to decide where to execute a job submitted to the Grid system, and how to get the job there? In contrast to routing and wavelength assignment problems in more traditional optical networks, an extra degree of freedom arises since not only the route, but also the destination itself can be chosen. This typically leads to multi-cost routing problems [6], incorporating the state of both network and computational/storage Grid resources. Given this paper's focus on dimensioning, we will assume relatively simple routing and scheduling algorithms.

In this paper we address the Grid dimensioning problem. The input is a given network topology comprising the locations of the sites where jobs originate (or aggregation points, for example points-of-presence (PoP) nodes of Grid service providers) and the (backbone) network interconnecting them, and the amount of jobs generated. We want to answer the question how to decide where to provide server capacity, and how to figure out the network dimensions required to process the submitted jobs. The major difference with classical (optical) network dimensioning again arises from the aforementioned anycast principle: we are not given the complete so-called traffic matrix, since only the source of the jobs is given, not the destination (that can be freely chosen by some job scheduling algorithm).

The remainder of this paper discusses our phased approach to Grid dimensioning. A concise problem statement, and an overview of related work on Grid dimensioning is presented in Section 2. Our solution method is described in Section 3, followed by a case study and its discussion in Section 4. The paper is concluded in Section 5.

2. The Grid dimensioning problem

A classical network design problem is dimensioning: figuring out how much capacity is needed for the network to be able to transport a given amount of traffic. Typically, this traffic is specified in a traffic matrix: for each source site i and destination site j, the amount of traffic flowing from site i to j is given by as a number T_{ij} (say in Mbit/s). A broad range of dimensioning algorithms is available, either based on heuristics or exact solution methods using for example Integer Linear Programming (ILP). The algorithms vary depending on the network technologies and topologies (for instance single- or multi-layer scenarios involving one or more network layers such as when jointly dimensioning IP routers and WDM cross-connects [8], with or without grooming [9] where for instance IP flows between different end points can share the same WDM circuits over multiple hops bypassing some of the IP routers; for ring [10] or mesh networks), design criteria (such as survivability [11], availability), single or multi-period planning [12] (where the network evolves over time, in response to a changing traffic demand over a longer time interval spanning multiple years), single domain or hierarchical networks [13] (providing algorithms for deciding how to partition the network in access and backbone nodes, as well as designing the backbone topology), etc. Yet, if we want to apply any of these approaches for dimensioning grids, the problem arises of accurately estimating the traffic matrix. Indeed, given the anycast principle typical of Grids, the destination of the traffic (Grid jobs) is not given a priori.

Another aspect in Grid dimensioning is that not only the network resources, but also the computational and/or storage resources need to be dimensioned: how many servers need to be installed, and at which sites? Note that the latter will have an impact on where jobs will end up being executed, thus the eventual traffic matrix, hence the network dimensions. It is clear that jointly determining both server and network dimensions is a very hard problem (note that even single-period dimensioning, where a single traffic matrix is given specifying the average demand between every node pair, may already be NP-hard [14]). Therefore, we will propose a phased approach, dimensioning first the servers and then the network (see Section 3).

Related work on dimensioning Grids is scarce. In [15] analytical ILP and heuristic approximations are used to cater for excess load: it is assumed that each of the grid sites (dimensioned for the locally generated jobs) may suffer from overload, and network dimensions (number of wavelengths and fibers used) are determined by a finding a global optimum over all single-site overload problems.

One way to deal with the unknown destination for Grid jobs is to assume that the fraction of jobs (originating at a particular site) going to a given computational Grid site is known, thus fixing a priori the arrival rates of jobs at each job execution site. This approach is taken in [16], where an analytical methodology known as reduced load fixed-point approximation [17] is used to dimension both network and computational resources. In this paper however, we focus on a ‘clean slate’ or greenfield Grid dimensioning problem finding the complete Grid capacity required to meet a given Grid job arrival pattern. Also, we assume fully flexible scheduling strategies without any knowledge of probabilities for selecting a given destination site.

This presents a viable dimensioning methodology, and assesses the impact of the scheduling algorithm on Grid network dimensions. Yet, since development of scheduling algorithms as such is not this paper’s primary concern, we will assume fairly straightforward scheduling strategies, based on a single all-knowing scheduler, finding a free server for every arriving job based solely on the job’s arrival time and duration, and server processing speed and occupation. For examples of more advanced scheduling algorithms, including QoS support and advance reservation concepts and, we refer to [5] and [7] respectively. We believe that adding QoS support or advance reservations is unlikely to affect the qualitative comparison of the different scheduling strategies discussed further.

3. A PHASED SOLUTION OF THE GRID DIMENSIONING PROBLEM

We will take an iterative dimensioning approach, starting with an algorithm for choosing appropriate server site locations: not every grid site will necessarily be a server site. Next we will calculate the amount of servers needed (and distribute them amongst the chosen server site locations). Subsequently the inter-site job rates are determined, and hence required bandwidth. In the work presented, we focus on computational

Grids, where jobs consist of a single unit of work submitted to the Grid, characterized by a data size, and a computational requirement (say, expressed in number of floating point operations, FLOP). An accurate problem statement is the following:

- Given:
  - Graph representing the network topology (nodes representing Grid sites and switches, links the optical fibers interconnecting them),
  - Arrival process of jobs originating at each site,
  - Job processing capacity of a single server CPU (an average of μ jobs/s), and
  - Target maximum job loss rate,
- Find:
  - Locations of the server sites,
  - Amount of Grid server CPUs at each site, and
  - Amount of link bandwidth to install,
  - While meeting the maximum job loss rate criterion and minimizing network capacity.

Given the complexity of the problem (such as the dependence of the network capacities on the choices of server locations and capacities), we opt for a phased solution approach comprising subsequent steps. The first step will be to find K server locations (out of the N grid sites), while a second step finds the server capacities at each of the K chosen sites. The third step will determine the amount of jobs exchanged between the grid sites and the server locations. The final fourth step will be to calculate the actual network dimensions, that is link bandwidth. Each of these steps is now discussed in detail.

3.1. Finding the K best server locations

The aim of the first step in solving our Grid dimensioning problem is to figure out which locations are best suited for placing the servers. The cost criterion to measure by will be the total expected link bandwidth. The major difficulty in evaluating that cost for a given choice of K locations, is that the required bandwidth depends also on the amount of server capacity installed at each of the server sites and possibly the Grid scheduling and routing algorithm. Therefore, we make some simplifying assumptions: (i) each Grid site i will send all its jobs to a single destination D_i, and (ii) shortest path routing is used. Hence, given a choice of K locations, a site i will send its jobs to server site j if the routing distance H_{ij} is the minimum over all H_{ik} values for $k = 1..K$.

Finding the optimal choice of K sites hence is a k-means clustering problem (or rather a k-medoid problem, since cluster centers are actual data points): we are looking for K cluster centers, the centers representing the server sites, and the cluster members the Grid sites sending their jobs to that center (server). A well-known heuristic k-means clustering algorithm [18] solving the problem is rephrased in Fig. 1 as a k-medoids algorithm for the Grid site location problem. Repeating this algorithm for various randomly chosen initial *K* server locations leads to solutions close to the exact solution. However, given the simplifying assumptions, a fairly compact ILP formulation of the problem can be devised, as outlined in Fig. 2. Given the relatively small number of (binary) variables and equations (*O*(N^2) with *N* the number of sites), the time to solve it for the case studies we considered (comprising a few tens of nodes) was most acceptable (a few seconds at most). The results we present in this paper therefore are obtained using the ILP solution method. (Note that for other, prohibitively large problem instances, the k-medoid heuristic can provide acceptable solutions quite fast.)

```
Given constants:   $H_{ij}$ = routing distance (for instance hop count) from site i to site j  (i, j = 1..N)
                   $\lambda_i$ = job arrival rate at site i (i = 1..N)
                   K = number of clusters and centers (hence server sites) to choose
(1)  Choose K initial medoids $m_k$ (k = 1..K).
(2)  Form clusters: assign each object (Grid site) to closest centroid:
         For each i = 1..N, assign node i to the cluster set $C_k$ with centroid $m_k$ if
         $H_{i,m_k} = \min\left(H_{i,m_l}, l = 1..K\right)$
(3)  Recalculate the positions of the K medoids within their cluster:
         For each k=1..K, let $m_k$ be that node m in set $C_k$ minimizing $\sum_{i \in C_k} \lambda_i \cdot H_{im}$
(4)  Repeat steps 2-3 until the medoids $m_k$ no longer change; these cluster centers are the
     server locations.
```

Fig. 1 – K-medoids clustering algorithm for choosing K server locations. (Note: the term 'k-medoids' is used, instead of 'k-means', since the cluster centers are actual data points—in casu site locations—and not freely chosen means.)

Decision variables: T_j = 1 if and only if site j is chosen as a server site location, else 0

S_{ij} = 1 if and only if site j is the target server for traffic from site i, else 0

Given constants: H_{ij} = routing distance (for instance hop count) from site i to site j (i, j = 1..N)

λ_i = job arrival rate at site i (i = 1..N)

K = the number of server sites to choose

$$\min \sum_i \sum_j \lambda_i \cdot H_{ij} \cdot S_{ij} \quad \text{with} \quad \begin{cases} \sum_j T_j = K & \text{(only K server locations)} \\ \sum_j S_{ij} = 1 \quad \forall i & \text{(simplifying assumption: for each site i, only send jobs to a single server site j; the objective will ensure it is the closest one)} \\ S_{ij} \le T_j \quad \forall i, j & \text{(only send traffic to server sites)} \end{cases}$$

Fig. 2 – ILP for choosing K server locations.

3.2. Determining the server capacities

For determining the amount of required server capacity, it is necessary to make some assumptions on the Grid job arrival process. In this paper, we focus on computational Grid jobs, hence we will dimension the servers in terms of processing capacity, expressed in number of CPUs. A job is assumed to fully occupy a single CPU for its entire duration. Furthermore, we will assume that Grid job requests are to be scheduled immediately, leading to a bufferless system model: if at job arrival no free server is found, the job is lost. Backed by real world Grid measurements [20], we will assume Poisson job arrivals (mean arrival rate λ_i at site *i*). This implies that, given the lack of buffers, we can use the *ErlangB* formula (1) to calculate the total number of server CPUs *n* required to achieve a maximal loss rate *L*.

$$L = \text{ErlangB}(n, \lambda, \mu) = \frac{(\lambda/\mu)^n / n!}{\sum_{k=0}^{n} (\lambda/\mu)^k / k!} \qquad (1)$$

To place the *n* server CPUs among the *K* (≤ *N*) server site locations chosen in step 1, we consider three strategies:

(i) *unif*: uniformly distribute the server CPUs among all *K* server sites: $n_k = n/K$, for each server location $k = 1..K$.

(ii) *prop*: distribute the server CPUs proportionally to the (cluster) arrival rate at each server site *k*: $n_k = \lambda_k^*/(K\cdot\lambda)$, with $\lambda = \Sigma\ \lambda_i$ and $\lambda_k^* = \Sigma\ \lambda_i \cdot S_{ik}$, where S_{ik} is 1 if and only if *k* is the server site closest to *i* (as defined in the ILP of Fig. 2). Note that λ_k^* equals the total job arrival rate summed over all Grid sites in cluster *k*.

(iii) *lloss*: try and achieve the same "local loss rate" at each server site, that is, use *ErlangB* to calculate n_k^* as the number of server CPUs to install locally at server site *k* to achieve loss rate *L* (solve $L = ErlangB(n_k^*, \lambda_k^*, \mu)$) and install $n_k = n \cdot n_k^*/(\Sigma\, n_i^*)$ servers.

Intuitively, we expect the *prop* and *lloss* strategies to perform better than *unif*, since more server capacity will be installed where more traffic is arriving. Case studies below will assess the penalty of using the simple *unif* strategy in terms of required network resources.

3.3. Determining the inter-site bandwidths

Once the location and the amount of CPUs are fixed, only the Grid scheduling algorithms determine the amount of jobs, and hence bandwidth, that will be exchanged between each Grid (site, server)-pair. To demonstrate that the difference may indeed be substantial, we will consider three scheduling alternatives. Note that all algorithms account for the anycast routing principle: it depends on the instantaneous availability of Grid resources, without a priori decision on where to execute a job. Since we are interested in minimizing the required amount of resources, including link bandwidth, all scheduling strategies considered will always try to use a 'local' server CPU before anything else. Jobs arriving at a site *i* belonging to cluster *k* (as derived in step 1), with m_k as cluster center (thus server site) will always be scheduled on a 'local' free server CPU at site m_k if available. The strategies only differ in choosing an alternative CPU if for a job arriving at *i*, all CPUs at its cluster center m_k are occupied:

(i) *rand*: randomly choose a free server CPU (hence, among *F* free servers, each has $1/F$ chance);

(ii) *SP*: the closest free server in terms of routing distance (H_{ij} as defined in the algorithms in Section 3.3.1, for example hop count) is chosen, thus striving to minimize network usage;

(iii) *mostfree*: choose a free CPU at server site *f*, where *f* is the server site with the highest number of free server CPUs, in an attempt to avoid overloading sites and thus limiting non-local job execution.

To calculate the job exchange rate for every Grid (site, server)-pair, we resort to simulations. In this, we make abstraction of the network capacity (since that is exactly what we want to calculate in the next step): we assume infinite link bandwidth, the only way jobs can be lost is because of lack of CPU capacity.

The reason for resorting to simulations is that because of the anycast principle it is hard to obtain accurate estimates for the inter-site traffic bandwidth using analytical techniques. To illustrate this, we compared our simulation results with those obtained using a fixed-point approximation methodology. We iteratively solve the equations (2)-(4), initializing the system with $\lambda_k' = \lambda_k$. Equation (2) gives the blocking probability at site *k*, being the probability that all its n_k servers are occupied. The total job arrival rate λ_k' at site *k* is calculated from equation (3) as the sum of the locally arriving jobs λ_k and the fractions f_{jk} of the jobs arriving at all other sites *j* which are blocking there with probability L_j. The fractions f_{jk} of jobs blocking at *j* and sent to *k*, are assumed to be proportional to the probability $1–L_k$ of finding a free server at site *k*, as given in equation (4). We stop the numerical iterations solving this system of equations when the difference between successive calculations of the loss rates with equation (2) is smaller than a given tolerance τ (in the results below, we set $\tau = 10^{-8}$).

$$L_k = \mathrm{ErlangB}\left(n_k, \lambda_k', \mu\right) \tag{2}$$

$$\lambda_k' = \lambda_k + \sum_{\substack{j=1 \\ j\neq k}}^{K} L_j \cdot \lambda_j \cdot f_{jk} \tag{3}$$

$$f_{jk} = \frac{1-L_k}{\sum\limits_{\substack{m=1 \\ m\neq j}}^{K} \left(1-L_m\right)} \tag{4}$$

Note that the approximation lies in fixing a priori the amount of jobs that is off-loaded to a remote site: a blocked job originally intended for site *i* is sent to a remote site *k* with probability f_{jk}, regardless of the availability of servers at site *k*. Hence, under this assumption, the blocking rate for traffic initially sent to site *i* is given by P_i as in equation (5). The total blocking probability *P*, as given by equation (6), will be larger than the ErlangB blocking of equation (1) achieved by fully sharing all available server capacity, and sending jobs using the anycast principle to any free server.

$$P_i = L_i \cdot \left(1 - \sum_{\substack{k=1 \\ k\neq i}}^{K} f_{ik} \cdot \left(1-L_k\right)\right) \tag{5}$$

$$P = \frac{\sum\limits_i \lambda_i \cdot P_i}{\sum\limits_i \lambda_{ii}} \tag{6}$$

3.4. Determining the link bandwidths

After the previous step, we know how many jobs/s are exchanged between every Grid node pair in the considered Grid network. Assuming a given data size distribution, the number of jobs/s can be translated into a bandwidth requirement (say in Mbit/s). Thus, we now have obtained a traffic matrix listing the traffic demand between each (source, destination)-pair. Hence, we can apply ‘classical’ network dimensioning algorithms to assess the required network resources (number of wavelengths on each link) for given switching technologies (such as OBS, OCS or hybrid [21]) and requirements (resilience for instance).

For the case study presented next, we will assume shortest path routing and make abstraction of the actual network technology to judge the cited scheduling and CPU dimensioning variants by. We will use the average hop count traversed by a job as a measure of network resource usage. Note that by accounting for overhead (such as packet headers or burst control packet offsets), or discrete bandwidth increments (multiples of wavelength bandwidth), the actual difference in bandwidth requirements can be larger than what the hop count based results may suggest. Yet, the qualitative conclusions will remain valid.

4. CASE STUDY

In the previous section, we have outlined a step-wise scheme to dimension both server CPU and network resources for computational Grid. We will now apply it in a realistic case study, to highlight the importance of choosing an appropriate scheduling and server CPU placement algorithm when trying to limit the network resource requirements. The scenario and input parameters are outlined and motivated in Section 4.1, whereas the actual results will be subsequently discussed in 4.2 and 4.3.

4.1. Scenario

To obtain a realistic case study, we performed measurements on a real world Grid, deployed in Europe in the frame of the Large Hadron Collider (LHC) experiments in Geneva and the Enabling Grids for E-sciencE (EGEE) project [19], referred to in short as the EGEE/LCG Grid. From Grid-wide job arrival logs, it was derived that the Poisson traffic model (with negatively exponentially distributed inter-arrival times) accurately fits the real world arrivals [20].

We considered two network topology and job demand rate cases, whose topologies are sketched in Fig. 3, and the input parameters are summarized in Table 1. The first used a fairly densely meshed European backbone network (the “Large Topology” taken from [22]), with artificially generated job arrival rates at each site (each rate λ_i was with 30% chance uniformly chosen in [1, 15] and 70% from [30, 60]).

The second case is based on measurement data from the EGEE/LCG Grid (the same data set as [20]), and for a topology based on the EGEE site locations and the Geant2 [23] network topology and its associated various national research and education networks (NRENs). The arrival rates at each site were set to the values derived from a one-month trace file. (Note that the job trace data comprised 58 Grid sites, rather than the 20 of the Geant2-inspired topology: we attributed their job arrivals to the geographically closest Geant2 site, based on the coordinates of the EGEE/LCG sites as found with IP-address-based geolocation software by MaxMind Inc.) In this EGEE/LCG case, also the average job duration was derived from the real-life trace file.

For both cases, we applied the dimensioning strategy outlined in Sections 3.1 to 3.3. As acceptable job loss for the ErlangB calculation we chose L = 0.05.

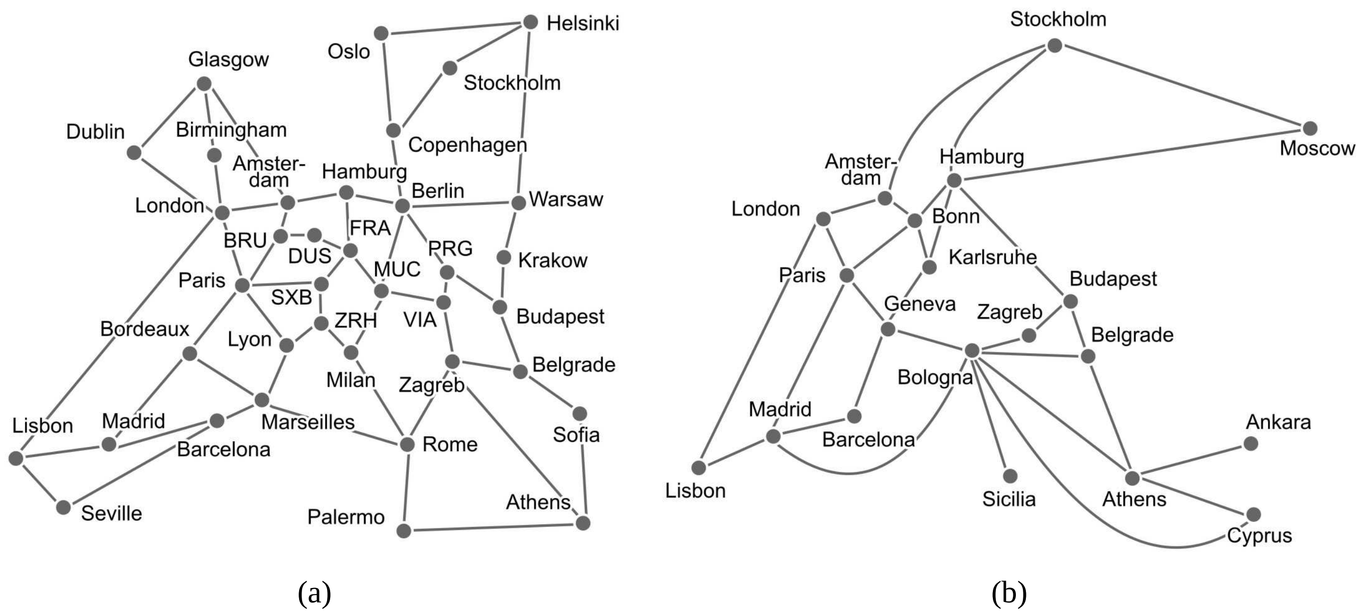


Fig. 3 – Case study topologies: (a) a European backbone network, (b) Geant2 network.

Table 1 – Case study parameters

Parameter	*Case 1 (EU)*	*Case 2 (EGEE/Geant2)*
Topology	European backbone [22]	Geant2 network
Number of nodes	37	20
Number of links	57	31
Average shortest path hop count	3.62	2.55
Job duration	100 s (per job)	854.92 s (per job)
IAT distribution	Exponential distribution	Exponential distribution
Average arrival rate over all sites	2.23E–01 jobs/s	3.56E–02 jobs/s
Stdev of arrival rate over all sites	2.02E–01 jobs/s	6.47E–02 jobs/s
Total arrival rate over all sites	8.24E+00 jobs/s	7.12E–01 jobs/s

4.2. Local processing rates

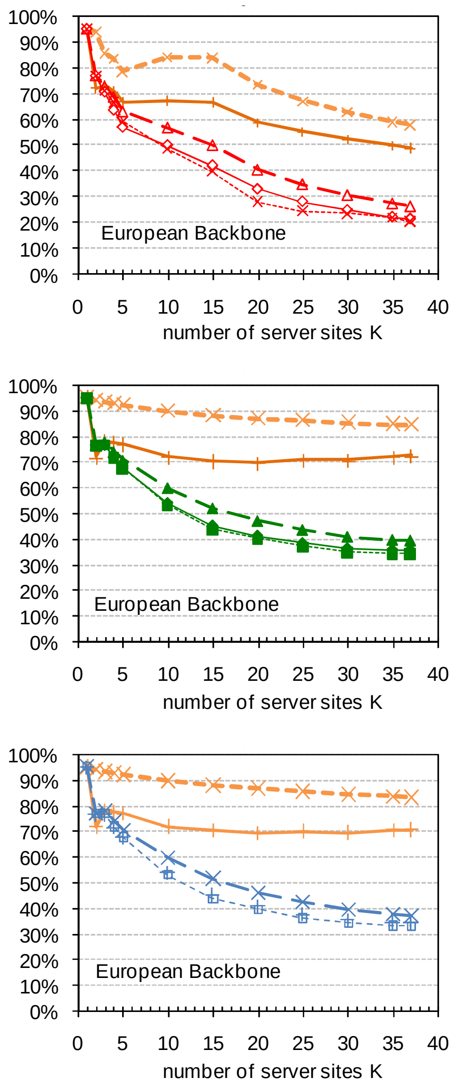
To evaluate the CPU distribution and job scheduling algorithms, a first criterion we considered was how many jobs are off-loaded to a remote site, or its complement, the so-called 'local processing rate': the fraction of jobs that is processed at the closest server. To limit network load, we strive for keeping the local processing rate high. This fraction of jobs that is executed at their respective closest server site is plotted in Fig. 4 against the number of chosen server site locations *K*. Note that the maximal value of 95% is due to the L = 0.05 target job loss rate we dimensioned the server sites for.

To assess the influence of offloading jobs to other sites, we have also calculated an upper bound for the local processing rate, using the ErlangB formula. This bound was obtained by assuming that jobs only may be executed at the closest server site (see the simplifying assumption used in the server site selection approach of Section 3.1). Given this assumption, the maximal local processing rate at server site *k* equals 1-L_k, with $L_k = L$ calculated using ErlangB formula (1) for $N = n_k$ the number of servers at site *k*, and $\lambda = \lambda_k$ the aggregate arrival rate of its closest Grid sites.

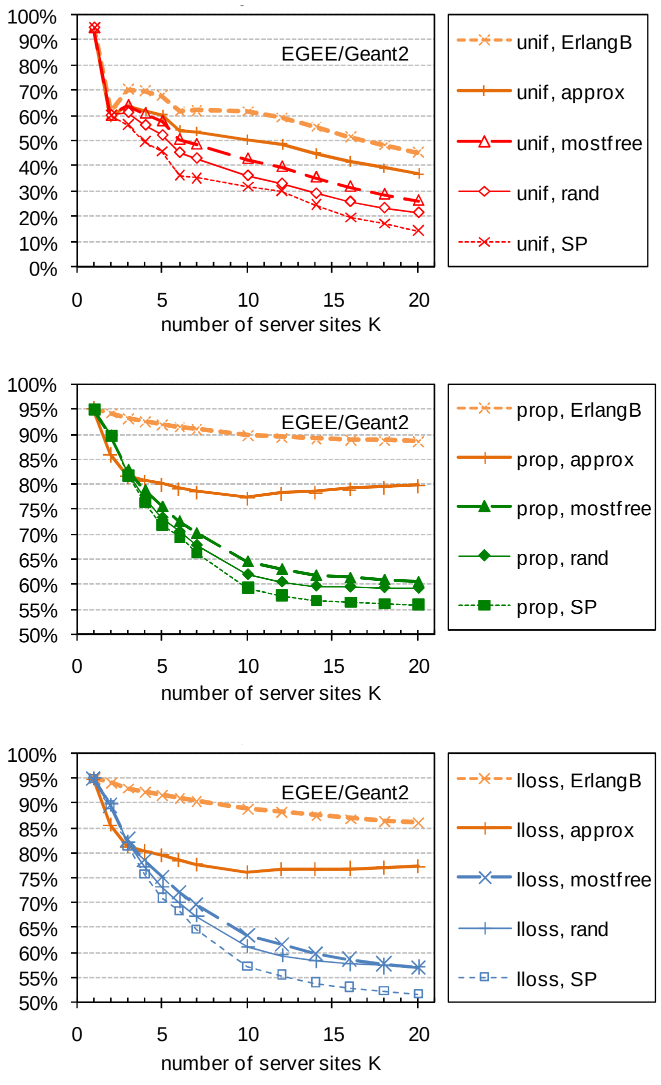
With respect to the server distribution schemes, placing more servers where more jobs originate is beneficial: the *prop* and *lloss* strategies attain higher local processing rates than *unif*. The difference between *prop* and *lloss* from this respect is minimal (relative differences less than 5%): only for the European backbone case we noted a slightly higher local rate for *lloss* for low server site counts *K*, in all other cases *prop* attains a higher fraction of jobs executed at the respective closest server site. This indicates that the more complex *lloss* dimensioning strategy doesn't seem to pay off compared to the simple *prop* strategy.

From the graphs it is clear that the scheduling algorithm also impacts the local processing rate. Among the considered alternatives, *mostfree* from this perspective performs best.

With respect to the influence of the traffic and network topology, we note that in the EGEE/Geant2 case the matching the server capacities to the traffic arriving (*prop* and *lloss* cases) seems more effective: especially for larger server counts *K*, the local processing rate stabilizes around 55-65% in the EGEE/Geant2 case (compared to 30-40% for the EU case). This can be explained by the larger variance in job arrival rates in the EGEE/Geant2 case (see Table 1): for bigger discrepancies in site arrival rates, the server counts will differ more between the *unif* and *prop/lloss* dimension strategies.



(a)



(b)

Fig. 4 – The fraction of jobs executed at the respective closest server site ('local processing rate') is maximized by intelligently positioning server capacity (prop vs unif), and is well below the ErlangB upper bound: results for (a) the European backbone network, (b) the Geant2 network. Note the different Y-axis scale in the prop *and* lloss *graphs for EGEE/Geant2. The analytical fixed point approximation (approx) fails to match the more accurate simulation results for a higher amount of server sites (K > 5).*

Looking at the ErlangB upper bound, the better performance of the *prop/lloss* approaches (compared to *unif*) also is immediately apparent. It is striking that there is a rather big gap between the ErlangB upper bound, and the actually attained local processing rates. This suggests that as soon as the total amount of servers is distributed over multiple locations, there is a non-negligible amount of jobs that is sent to remote sites (instead of being dropped as in case of the ErlangB bound assumptions), which there compete for server capacity with the locally arriving jobs. Yet, by allowing this interchange of jobs, the overall success rate is improved: the target success rate attained by non-local job execution remains at 95%, whereas the ErlangB curve drops well below that (note that in case of ErlangB the ‘local processing rate’ shown in Fig. 4 is exactly the success rate since it assumes no remote execution of jobs).

Note also that the ‘approx’ curves, showing the results of the fixed point approximation solving the equations (2)-(4), do not very well match the simulation results. This approach indeed does not accurately model the site blocking probabilities’ inter-dependencies. As a result, the approximation overestimates local processing probabilities as soon as the number of Grid server sites K increases (see curves for $K > 5$).

Solely looking at local processing rates, one may be tempted to opt for a minimal number of server locations. In the following sections we will argue this is not optimal from a correct network perspective.

4.3. Used link bandwidth

From network perspective, the most important criterion is network bandwidth. To establish the network dimensions in the considered case study, we would need to choose a particular network technology (OBS versus OCS, or hybrids). Yet, these relate to the traffic matrix stating the amount of bandwidth exchanged between each node pair. A useful measure to comprehensively summarize this information in the assumed Grid context is the average hop count a job needs to traverse to reach the server it will be executed on. Hence, we will use the average job hop count as a measure to judge the network capacity requirements. We obtained this measure from the simulation approach described in Section 3.3. We summarize the average job hop count results for varying number of server sites K in Fig. 5.

These graphs also include results obtained from the analytical fixed point approximation. Given the solution of the equations (2)-(4), the bandwidth B_{ik} flowing from site i to server site k can be calculated by equation (7). From these values B_{ik} the average hop count can be calculated as from the simulation results.

$$B_{ik} = \begin{cases} \lambda_i \cdot (1 - L_k) & k = \text{closest server } c \text{ for site } i \\ \lambda_i \cdot L_c \cdot f_{ck} \cdot (1 - L_k) & k \neq \text{closest server for site } i \ (c = \text{closest server for site } i) \end{cases} \qquad (7)$$

Comparing the analytical approximation, we note a mismatch compared to simulation results for larger values of number of server sites K, as before. For the *unif* site dimensioning strategy results in case of EGEE/Geant2, we observe non-smooth fluctuations. The reason is that in the *unif* case, all server sites get an equal portion of the available server capacity, while there is quite some discrepancy in job arrival rates. Hence, adding extra server sites may result in a quite drastic change in the inter-site traffic rates (see also the less smooth ‘local processing rate curves’ in Section 3.2), stemming from a severe reduction in server capacity in certain network segments. The fact that this is far less pronounced in the European Backbone case can be explained by the smaller variation in job arrival rates. In the *prop* and *lloss* cases, the server capacities better match the arrival rates and hence the curves evolve smoothly.

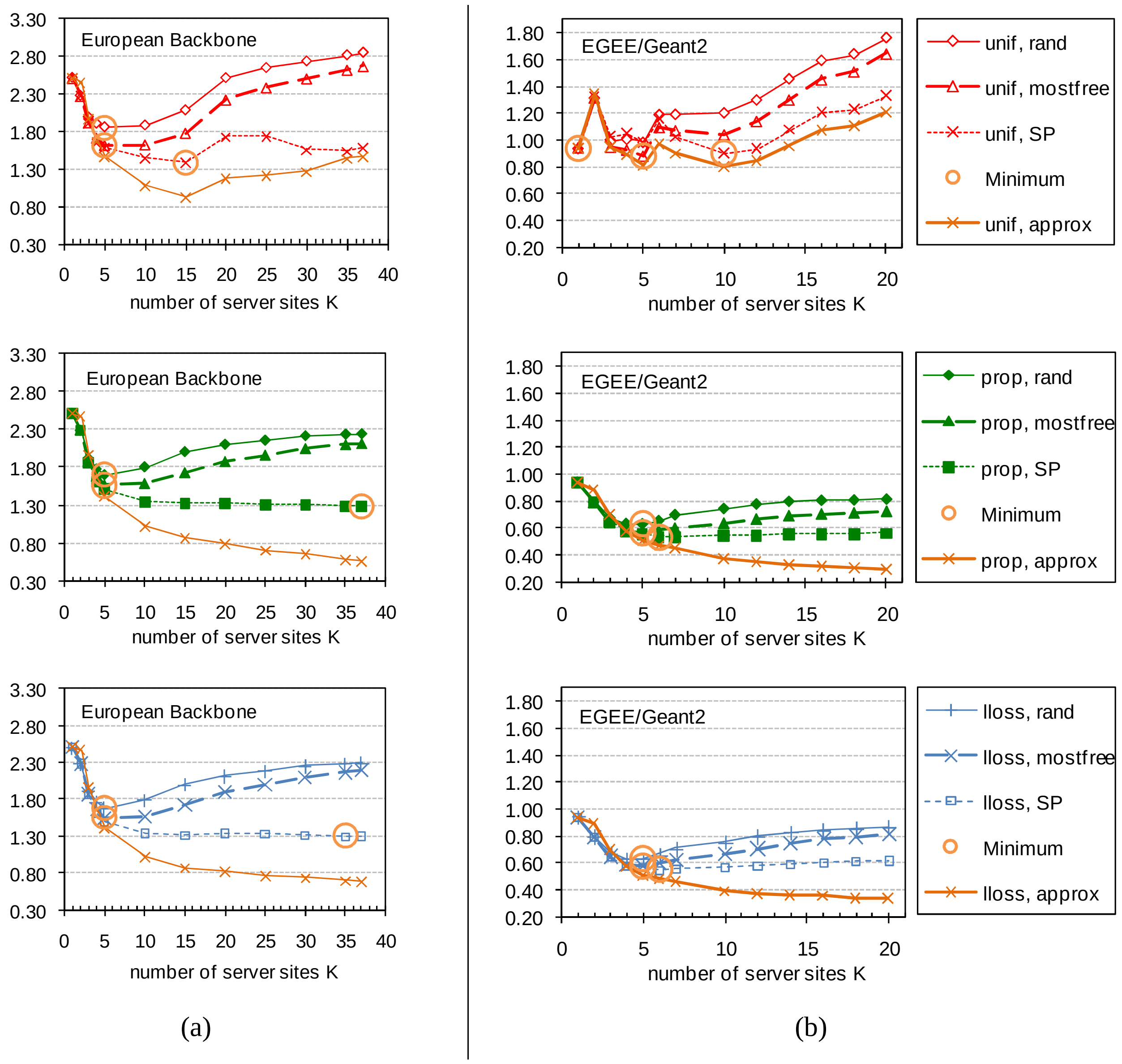


Fig. 5 – The required network capacity, which is proportional to the average job hop count, is minimized by adopting shortest path routing and scheduling (SP), intelligently positioning server capacity (prop vs unif), and deploying a reasonable number of server locations: average hop count over all jobs for (a) the European backbone network, (b) the Geant2 network. The analytical fixed point approximation (approx) deviates from simulation results because it does not accurately capture site inter-dependencies.

Comparing the various combinations of dimensioning and scheduling alternatives, the relative influence of the scheduling algorithm seems to be important. The reasonably large fraction of traffic sent to non-closest server sites—recall the 'local processing rates' from the previous section—has an important influence on the network load. Hence, by adopting shortest path driven scheduling (SP), the lowest average job hob count is reached.

The influence of the dimensioning strategy is also obvious, though less significant. Especially for a larger number of server sites *K*, it pays off to intelligently distribute server capacity: *prop* and *lloss* (which hardly differ in resulting average job hop count) result in lower network load than straightforward *unif*orm server distribution.

Comparing the European backbone case with the EGEE/Geant2 case, we note that the major qualitative difference lies in the curves for the *unif*orm dimensioning strategy: this curve is mainly increasing for larger values of *K* in the EGEE/Geant2 case. This can be explained by the larger relative variations in shortest path hop counts in this network: the penalty of unintelligently distributing server capacity is more pronounced.

With respect to the choice of the number of server sites *K*, we observe there is an optimal choice, which tends to lie around $K = 5$ in the studied cases, ranging between 1/7 and 1/4 of the total number of sites. Note that the optimum depends on both the dimensioning strategy and the scheduling approach. When too much server sites are installed, the total server capacity is fragmented too much, resulting not only in low 'local processing rates' (see Fig. 4) but also a lot of jobs sent to remote servers. Indeed, for larger number of server sites, the opportunities of statistical multiplexing diminish and with it the probability of finding a free server at the closest server site for a particular job. This apparently outweighs the fact of lowering the average distance to a server site.

5. Conclusion

Contrary to (rather scarce) earlier work on Grid dimensioning, we proposed a dimensioning methodology fully taking into account the anycast routing principle, that is, without presuming a priori knowledge of (source, destination)-based traffic. The proposed step-wise methodology is suitable for dimensioning both server and network capacities. We outlined it for computational Grids, but extension to also incorporate for example storage capacity is possible.

We used the methodology to evaluate various scheduling algorithms and server dimensioning options with respect to the required network capacity. From two case studies on European topologies, we concluded that placing server capacity where a lot of jobs arrive is important to minimize network bandwidth requirements: the *prop* dimensioning strategy, placing a number of servers proportional to the job arrival rates at its closest Grid sites is most beneficial. With respect to Grid scheduling, a simple shortest path (*SP*) strategy, preferring closer server sites, led to the lowest bandwidth demands. With respect to choosing an appropriate number of server sites *K* (ranging from 1 to the total number of Grid sites N), we found that there is an optimal value. For a larger number of server sites, the total server capacity gets fragmented, reducing opportunities for statistical multiplexing, whereas for smaller server site counts *K* the average distance jobs need to travel is too large. That optimum of K depends on the scheduling algorithm and server site dimensioning strategy, and in the considered case studies was about 1/7 to 1/4 of the total number of sites.

ACKNOWLEDGMENT

Part of this work has been funded by the EU, through the IST project Phosphorus and the IST Network of Excellence (NoE) e-Photon/ONe+, and by the Research Foundation Flanders (FWO, *www.fwo.be*).

C. Develder is supported by the Research Foundation Flanders as a postdoctoral fellow, and thanks the FWO for the travel grant for his stay at UC Davis in the summer of 2007.

REFERENCES

[1] D. Simeonidou, R. Nejabati, G. Zervas, D. Klonidis, A. Tzanakaki, M.J. O'Mahony, *"Dynamic optical network architectures and technologies for existing and emerging Grid services"*, IEEE/OSA J. Lightwave Technology, Vol. 23, no. 10, pp. 3347–3357, Oct 2005.

[2] M. De Leenheer, et al., *"Design and control of optical Grid networks" (Invited)*, Proc. Broadnets 2007, Raleigh, NC, pp. 107–115, 10-14 Sep. 2007.

[3] F. Fahramand, et al., *"A multi-layered approach to Optical Burst-Switched based Grids"*, Proc. 5th Int. Workshop on Optical Burst/Packet Switching (WOBS) co-located with IEEE/CreateNet BROADNETS 2005, Boston, MA, USA, vol. 2, pp. 1050–1057, 3 Oct. 2005.

[4] M. De Leenheer, et al., *"A view on enabling consumer oriented Grids through Optical Burst Switching"*, IEEE Commun. Mag., vol. 44, no. 3, pp. 124–131, Mar. 2006.

[5] K. Christodoulopoulos, N. Doulamis, P. Kokkinos, E. Varvarigos, *"Quality of Service Scheduling of Computation and Communication Resources in Grid Networks"*, in *Grid Computing Research Progress*, Nova publishers, 2008.

[6] T. Stevens, et al., *"Multi-Cost Job Routing and Scheduling in Optical Grid Networks"*, submitted to Future Generation Computer Systems, Special Section on Networks for Grid Applications.

[7] K. Christodoulopoulos, N. Doulamis, E. Varvarigos, *"Joint Communication and Computation Task Scheduling in Grids"*, Proc. 8th IEEE Int. Symp. On Cluster Computing and the Grid (CCGRID 2008), pp. 17–24, 19-22 May 2008.

[8] H. Höller, S. Voß, *"A heuristic approach for combined equipment-planning and routing in multi-layer SDH/WDM networks"*, European J. of Operational Research, vol. 171, no. 3, pp. 787–796, Jun. 2006.

[9] K. Zhu, H. Zang, B. Mukherjee, *"A comprehensive study on next-generation optical grooming switches"*, IEEE J. Selected Areas in Commun., vol. 21, no. 7, pp. 1173–1186, Sep. 2003.

[10] O. Gerstel, R. Ramaswami, G. H. Sasaki, *"Cost-effective traffic grooming in WDM rings"*, IEEE/ACM Trans. Networking, vol. 8, no. 10, pp. 618–630, Oct. 2000.

[11] D. Colle, S. De Maesschalck, C. Develder, P. Van Heuven, A. Groebbens, J. Cheyns, I. Lievens, M. Pickavet, P. Lagasse, P. Demeester, *"Data-centric optical networks and their survivability"*, IEEE J. Selected Areas in Commun., vol. 20, no. 1, pp. 6–20, Jan. 2002.

[12] M. Pickavet, P. Demeester, *"Long-term planning of WDM networks: a comparison between single period and multi-period techniques"*, Photonic Network Commun., vol. 1, pp. 331–346, Dec. 1999.

[13] A. Bley, T. Koch, R. Wessäly, *"Large-scale hierarchical networks: How to compute an optimal architecture?"*, Proc. 11th Int. Telecommun. Network Strategy and Planning Symposium (Networks 2004), Vienna, Austria, 13-16 Jun. 2004.

[14] B. Mukherjee, D. Banerjee, S. Ramamurthy, A. Mukherjee, "*Some principles for designing a wide-area WDM-network*", IEEE/ACM Trans. on Networking, vol. 4, no. 5, pp. 684–696, Oct. 1996.

[15] P. Thysebaert, F. De Turck, B. Dhoedt, P. Demeester, *"Using divisible load theory to dimension optical transport networks for Grid excess load handling"*, Proc. Int. Conf. on Autonomic and Autonomous Systems & Int. Conf. on Networking and Systems (ICAS/ICNS 2005), Papeete, Tahiti, 23-28 Oct. 2005.

[16] M. De Leenheer, C. Develder, F. De Turck, B. Dhoedt, P. Demeester, *"Erlang reduced load model for optical burst switched grids"*, Proc. 3rd Int. Conf. on Networking and Services (ICNS2007), Athens, Greece, 19-25 June 2007.

[17] Z. Rosberg, H.L. Vu, M. Zukerman, J. White, "*Blocking probabilities of optical burst switching networks based on reduced load fixed point approximations"*, Proc. 22nd Annual Joint Conf. of the IEEE Computer and Commun. Societies (Infocom 2003), San Francisco, CA, USA, Vol. 3, pp. 2008–2018, 30 Mar. - 3 Apr. 2003.

[18] J. B. MacQueen, "*Some methods for classification and analysis of multivariate observations"*, Proc. 5th Berkeley Symp. on Mathematical Statistics and Probability, Berkeley, University of California Press, vol. 1, pp. 281–297, 1967.

[19] F. Gagliardi, B. Jones, F. Grey, M.E. Bégin, M. Heikkurinen, "*Building an infrastructure for scientific Grid computing : status and goals of the EGEE project*", Philosophical Trans. Series A Mathematical, Physical and Engineering Sciences, pp. 1729–1742, 15 Aug. 2005.

[20] K. Christodoulopoulos, E. Varvarigos, C. Develder, M. De Leenheer, B. Dhoedt, *"Job demand models for optical grid research*", Proc. 11th Int. IFIP TC6 Conf. on Optical Netw. Design and Modeling (ONDM2007), Athens, Greece; Lecture Notes in Computer Science, vol. 4534, pp. 127–136, May 2007.

[21] E. Van Breusegem, J. Cheyns, D. De Winter, D. Colle, M. Pickavet, F. De Turck, P. Demeester, "Overspill routing in optical networks: A true hybrid optical network design", IEEE J. Selected Areas in Commun., vol. 24, no. 4, supplement, pp. 13–26, Apr. 2006.

[22] S. De Maesschalck, et al., "Pan-European optical transport networks: an availability-based comparison", Photonic Network Commun., vol. 5, pp. 203–225, May 2003.

[23] The GÉANT2 project, *http://www.geant2.net*