CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. (2011)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cpe.1742

Planning data transfers in grids: a multi-service queueing approach

Kashif Munir[1,*,†], Renato Lo Cigno[2], Pascale Primet Vicat-Blanc[3] and Michael Welzl[4]

[1]*National University of Computer and Emerging Sciences, Islamabad, Pakistan*
[2]*DISI, University of Trento, Italy*
[3]*LIP Laboratory, INRIA, University of Lyon, France*
[4]*University of Oslo, Norway*

SUMMARY

Grid applications move large amounts of data between distributed resources, and the efficiency of a Grid depends on their timely delivery within given bounds (deadlines). In most cases, the data volume and deadline are known in advance, allowing for both network planning and connection admission control (CAC). We formally define the problem and, based on this formalization, describe the operation of a feasible procedure for network reservations of deadline-constrained bulk data transfer requests. The procedure guarantees a minimum bandwidth to meet the deadlines and allows for opportunistic utilization of residual network capacity. We propose a novel analytical model based on the solution of an $M/M(nc)/1/k(s)$-RPS queue. The analytical model is validated against ns-2 simulations taking into account network level details (IP and TCP protocols), showing remarkably good coherence even under heavy loads. The model is orders of magnitude faster than simulation, which enables its application to plan the capacity of Grid networks, and to enforce CAC under the hypothesis of a dominating bottleneck on the transfer route. Copyright © 2011 John Wiley & Sons, Ltd.

Received 5 December 2009; Revised 5 January 2011; Accepted 1 March 2011

KEY WORDS: Grid; modeling; bulk data transfer; TCP; admission control; processor sharing

1. INTRODUCTION

Grid computing enables the virtualization of distributed computing; resources such as processing, storage, capacity, and network bandwidth jointly provide the end user with a seamless, powerful computing system. Grid computing might be the ultimate overlay network, hiding from users all the details and pains about configuration, resource management and choices, and leaving them only with the satisfaction of the service.

For the time being, Grid computing is mainly used within the scientific community to tackle stiff problems that require humongous amounts of computational, storage, and data transfer capabilities. These requirements are normally predictable, if not known entirely in advance. Besides, requirements are correlated so that bad network scheduling will affect the usage of expensive CPU clusters and petabytes of data will linger in storage systems if they cannot be transferred to the processing units on schedule. Thus, a fully satisfactory behavior of Grid computing can only be achieved if reservations of Grid resources include the network to allow data transfers within the required time limits.

As mentioned above, network reservations in Grids are normally predictable, besides, they can sometimes be done in advance, and, being file-based, they are elastic in nature, meaning that they

*Correspondence to: Kashif Munir, National University of Computer and Emerging Sciences, A. K. Brohi Road, Sector H-11/4, Islamabad, Pakistan.
†E-mail: kashif.munir@nu.edu.pk

can exploit additional bandwidth resources (above the minimum requirement) if available, and they need to be transferred completely to be useful. The different nature of admission control and call rejection in the presence of elastic traffic, in contrast to the traditional circuit-oriented traffic, has been recognized and discussed in [1–3], while Bonald and Roberts [4] and Boyer *et al.* [5] discuss how per-flow admission control can improve the bandwidth sharing efficiency in such conditions. Admission control preserves the performance of flows already in progress, avoiding starvation phenomena and instabilities in the network, which becomes a necessity if guarantees are required for deadline-constrained data transfers in Grids.

Given the heterogeneous nature of Grid systems, the structure of network paths between two end hosts in a Grid may vary wildly; in the most extreme case, it can be an arbitrary path across the Internet which passes through the networks of several ISPs. At any given point of time, the smallest-capacity link along this path forms the bottleneck that is the focus of network capacity and traffic planning. The location of this link may change as flows come and go, and the path itself can change as a result of link failures or traffic engineering. In such scenarios, there is not much control over what happens along the path, but measurement-based systems for shared bottleneck detection such as [6, 7], combined with network mapping tools such as [8], could be applied to obtain a notion of where a path's bottleneck is and which Grid flows share it. However, such a combined entity is, to the best of our knowledge, not yet available in any Grid system.

In more favorable scenarios, we can have a more controlled environment, e.g. the French Grid'5000 [9], where the infrastructure is well known and under control of its users, or in various Cloud-like systems, where the underlying network belongs to the same entity that intends to sell a service. In these cases, it is possible to keep track of all traffic on all paths of the network, and deduce which links will be bottlenecks for how long; an admission control or capacity planning procedure could then essentially regard the path as a single link, and carry out the calculation for every link along the path, to determine all transmission schedules in advance. This is what the BDTS system [10] does; this system has been successfully used in the Grid'5000 environment and was successfully demonstrated on various occasions, e.g. at reviews of the European Project Europe–China Grid InterNetworking. The system has meanwhile been transferred to a spin-off created out of INRIA called LYaTiss[‡].

The early version of this work presented in [11] modeled the deadline-constrained Grid bulk data transfers using an $M/M/1/N\text{-}RPS$ queue, considering only the average value of the minimum rate requirements of different requests. The queueing model $M/M(nc)/1/k(s)\text{-}RPS$ presented in Section 4.2 describes the system's characteristics better than the earlier model, as in this model requests are divided into different classes on the basis of their different minimum rate requirements. Our contribution in the framework of Grids development and deployment is twofold:

1. the formal modeling and analysis of the connection admission control (CAC) and network capacity sharing mechanism proposed in [12];
2. the definition of a planning procedure for both capacity allocation and traffic class design for overlay networks empowering Grid computing applications.

The paper addresses the need of network dimensioning for bulk data transfers in Grids. Traditional dimensioning for telephone circuits uses the Erlang blocking model for multi-rate circuits [13]. These formulas assume constant rate connections suitable for the traditional circuits. Our proposed capacity planning procedure is useful for a Grid network provider because if the shared links are not quantitatively dimensioned, they can either result in the waste of precious bandwidth resources or in high blocking of deadline-constrained Grid applications like inter-site load balancing, scientific data distribution, etc.

The remainder of the paper is organized as follows: Section 2 briefly describes the literature closely related to our work. Section 3 presents the problem from a formal point of view, and Section 4 introduces the multi-class queueing model, which is the base of our performance evaluation and design framework. Section 5 presents the performance evaluation, computing blocking

[‡]http://www.lyatiss.com/.

probabilities and validating the results through simulations; besides, capacity and class design guidelines are presented. The paper closes with some final remarks in Section 6.

2. RELATED WORK

We split the literature discussion into two parts: the first one related to models of elastic traffic and system performance, and the second one related to reservations, planning, and CAC procedures in Grids.

2.1. Modeling elastic and multi-service traffic

In [14], performance modeling of elastic traffic is studied. However, the elastic traffic consists of flows that do not have deadlines. All flows therefore share the bandwidth equally. This is different from our problem, in which the flows have heterogeneous rate requirements, determined by their volumes and transfer durations, and they equally divide *only* the residual network capacity.

Bandwidth dimensioning for elastic traffic in a single link's case is studied in [15] to satisfy a performance objective based on the mean probability of the per-connection bandwidth in high-speed data networks. The issue of dimensioning Internet access lines for elastic traffic is discussed in [16] using an M/G/R-PS model characterizing TCP traffic at flow level. In our work, we take the deadline constraint of Grid bulk data transfers into account.

In [17, 18], $M/M(a,b)/c/PR$ priority queueing is used. The authors of [17] consider two product classes having different priorities to evaluate semiconductor manufacturing operations, whereas we have the same priority for all classes using a processor sharing queue. The authors of [18] consider four traffic classes, two for real-time traffic and two for non-real-time traffic, to analyze admission-control schemes for 3G wide-band CDMA (Code Division Multiple Access) wireless networks. Three services are used in [19] for rigid, adaptive, and elastic traffic classes to provide blocking probabilities and throughput guarantees for Internet traffic classes. In our problem, all classes consist of elastic traffic. CAC for elastic traffic and fairness issues are discussed by J. Roberts and L. Massoulie in a number of publications, see [1] for a survey. Our work here is in line with these papers. We extend them by proposing a computational model to predict the deadline-constrained elastic data transfers in Grids.

2.2. Grid network reservations

Grid applications need *QoS* guarantees [20]; network resource reservation [21] is a fundamental issue within Grid to support data transfer deadline enforcement and QoS. An example for a Grid toolkit that supports such mechanisms is *Globus* with its *GARA* resource allocation component [22].

For malleable requests in [23], the method of [24] or [25] is used to adjust the bandwidth or duration to satisfy the requester. However for a fixed request in [23], the only way to avoid being rejected is to adjust the bandwidth of admitted malleable requests. The trouble with this scheme is the extra overhead in finding and adjusting the admitted reservations which may be modified. The multi-interval scheme, presented in [26], avoids this problem. Rather than allocating a constant rate to each request, the *BDTS* multi-interval approach allocates a bandwidth profile, defined as a step function. The proposed *LP*-based algorithm provides an optimal solution in terms of congestion factor minimization.

A time-slot-based approach for scheduling the elastic and streaming requests in Lambda Grids is described in [27]. In this approach, a flow sends at a fixed rate during a time slot and hence does not make use of unreserved bandwidth in that time slot. In this approach, the fixed rate can only be computed between a low rate and a high rate specified by a request, whereas in our approach described in [12], flows can opportunistically utilize the available capacity. In [28], periodic re-optimization is used to determine new bandwidth allocation in optical networks. However, it is not assumed that users make advance reservations with requested end times. As a result [28] does not have the admission-control step. A deadline and budget-constrained scheduling algorithm, focussed on minimizing cost and execution time of a job, for eScience applications on data Grids is presented in [29, 30].

3. PROBLEM FORMULATION

The problem we tackle here is the modeling and representation of an overlay network empowering Grid computing which includes management, so that resources can be reserved globally ensuring that the Grid applications meet their requirements.

An *overlay network* is represented by a connected graph $G(V, E)$, consisting of node set V and edge set E, with edge capacity $\mu(e): E \to \mathbb{R}^+ - \{0\}$, where $\mathbb{R}^+$ is the set of non-negative real numbers. A *path* on the overlay network is a finite sequence of nodes $\phi = (v_0, v_1, \dots v_h)$, such that for $0 \le i < h$, $(v_i, v_{i+1}) \in E$. Table I summarizes the symbols we use throughout the paper.

Definition 1
A *data transfer task* $r = (v_r, \omega_r, \Phi_r)$ is a triple, where v_r is the *volume* of the data to be transferred, $\omega_r = [\eta_r, \psi_r]$ is the *life interval* of r (from *arrival time* η_r to *deadline* ψ_r; $|\omega_r| = \psi_r - \eta_r$ is the life time of r) and Φ_r is the path connecting the source s_r with destination d_r of r.

Since requests are predictable, the CAC mechanism is standard: Request r is accepted at time $\sigma_r = t$ and it is added to the set $Q(t)$ of active requests, only if path Φ_r can devote to it at least MRR_r capacity (out of its total capacity β_{Φ_r}) from time σ_r to time $\psi_r = \sigma_r + \frac{v_r}{MRR_r}$. However, being elastic, request r can use more resources than MRR_r if available and finish before ψ_r.

We evaluate the blocking probability BP as the ratio of rejected requests to offered requests:

$$BP = \frac{1}{|R|}\left(|R| - \sum_{r \in R} x_r\right).$$

The admission-control and request constraints can be stated formally as follows:

$$MRR_r * (\psi_r - \sigma_r) = v_r \quad \forall r \in Q(t), \tag{1}$$

$$\eta_r \le \sigma_r \quad \forall r \in Q(t), \tag{2}$$

$$\sum_{r \in Q(t)} MRR_r \le \beta_{\Phi_r} \quad \forall t. \tag{3}$$

Table I. Summary of the symbols used.

Symbol	Meaning
r	A data transfer task
v_r	Volume (bytes) of r
η_r	Arrival time of r
ψ_r	Deadline of r
$\omega_r = [\eta_r, \psi_r]$	Life interval of r from η_r to ψ_r
$\|\omega_r\| = \psi_r - \eta_r$	Length of the life time of r
s_r	Source of r
d_r	Destination of r
Φ_r	Path from s_r to d_r
σ_r	Scheduled start time of r
τ_r	Finishing time of r
MRR_r	Minimum Required Rate of r obtained by dividing v_r with $(\psi_r - \sigma_r)$
β_{Φ_r}	Capacity of bottleneck link of path Φ_r
R	Set of all submitted requests r
$Q(t)$	Set of active requests at time t
$C_r(t)$	Residual capacity along Φ_r at time t
$\gamma_r(t)$	Effective rate of r at time t
x_r	$x_r = 1$ if r is accepted and $x_r = 0$ if it is not

Equation (1) gives the volume constraints, Equation (2) gives the starting-time constraints, and Equation (3) gives the path capacity constraints. The bottleneck defining the residual capacity along the path can be on any physical link composing the path and may change in time; formally the residual capacity of any path is

$$C_r(t) = \beta_{\Phi_r} - \sum_{r \in Q(t)} MRR_r.$$

Accepted requests opportunistically grab more bandwidth during execution by dividing $C_r(t)$ equally among the requests, ideally implementing a max–min fair criterion. The actual capacity $\gamma_r(t)$ exploited by r is in the interval $[MRR_r, \beta_{\Phi_r}]$ for all t in $[\sigma_r, \psi_r]$. Thus, the actual finishing time of a request r is $\tau_r \leq \psi_r$. When request r is finished, it is removed from $Q(t)$, the set of active accepted requests.

The resource-sharing and request constraints are then stated formally as follows:

$$\int_{\sigma_r}^{\tau_r} \gamma_r(t)\,\mathrm{d}t = v_r \quad \forall r \in Q(t), \tag{4}$$

$$\tau_r \leq \psi_r \quad \forall r \in Q(t), \tag{5}$$

$$\gamma_r(t) : \omega_r \to \mathbb{R}^+. \tag{6}$$

Equation (4) is for opportunistic bandwidth-usage constraints. Equation (5) formulates the finishing-time constraints. Equation (6) gives the opportunistic bandwidth-solution space.

RES-UTI: Under the constraints in Equations (1)–(3), one may maximize the resource utilization ratio, that is, the ratio of granted resources to total resources. The objective function, referred to as RES-UTI, is

$$\textsc{Maximize} \frac{\sum_{r \in R} x_r * MRR_r}{\beta_{\phi_r}},$$

where numerator $\sum_{r \in R} x_r * MRR_r$ is the total bandwidth that has been assigned to requests.

RES-UTIop: The objective function, referred to as RES-UTIop, is

$$\textsc{Maximize} \frac{\sum_{r \in R} x_r * \frac{\int_{\eta}^{\psi} \gamma_r(t)\,\mathrm{d}(t)}{|\tau_r - \sigma_r|}}{\beta_{\phi_r}}$$

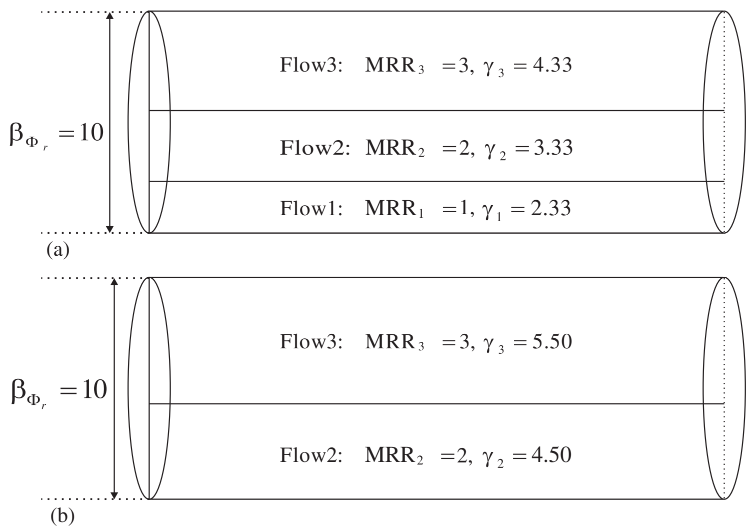
$$= \frac{\sum_{r \in R} x_r * \frac{v_r}{|\tau_r - \sigma_r|}}{\beta_{\phi_r}}$$

where numerator $\sum_{r \in R} x_r * v_r / |\tau_r - \sigma_r|$ is the total bandwidth used by the accepted opportunistic requests within time interval T.

Min-BLOCK: Under the same constraints mentioned above, one may minimize the *BP*. The objective is directly related to the above objectives and can be achieved if the requests can grab the available bandwidth opportunistically.

4. ANALYTICAL MODELING

The maximization objectives defined in Section 3 are met for a single path by the mechanism described in [12]. That work describes a CAC mechanism together with rationales of the path sharing mechanism, based on a simplified assumption of TCP-based transmissions. We are interested here in finding a suitable model for this mechanism that enables network planning and allows for the prediction of the performance of data transfers in Grid computing.



$\beta_{\Phi_r} = 10$
Flow3: MRR$_3$ =3, $\gamma_3 = 4.33$
Flow2: MRR$_2$ =2, $\gamma_2 = 3.33$
Flow1: MRR$_1$ =1, $\gamma_1 = 2.33$
(a)
$\beta_{\Phi_r} = 10$
Flow3: MRR$_3$ =3, $\gamma_3 = 5.50$
Flow2: MRR$_2$ =2, $\gamma_2 = 4.50$
(b)

Figure 1. Rate allocation of the flows: (a) before the termination of the first flow and (b) after the termination of the first flow.

4.1. CAC and path sharing overview

The mechanism guarantees MRR_r to a request r. We consider an ideal sharing of C_r among the flows on the path and we do not consider network details (packet losses, protocol overheads, etc). We assume that there is a Resource Broker (mentioned in [12]) that has complete knowledge of all flows that enter and leave the system.

We briefly explain the basic properties of the system here, referring the interested reader to [12]. Suppose we have a path with $\beta_{\Phi_r} = 10$ Gbps. Suppose that there are three flows in the network. Let the *MRRs* of the first, second, and the third flow be 1, 2, and 3 Gbps, respectively. There is a residual capacity of $10-(1+2+3)=4$ Gbps, which is equally divided among the three flows, which means that each flow will increase its transfer rate by 1.33 Gbps. The transfer rates of the three flows become 2.33, 3.33, and 4.33 Gbps, respectively, as shown in Figure 1. As the first flow terminates, the residual capacity becomes $10-(2+3)=5$ Gbps, which is equally divided between the second and the third flows due to processor sharing, and their transfer rates become 4.5 and 5.5 Gbps, respectively, as shown in Figure 1. Any flow can saturate β_{Φ_r}.

The main performance problem is to find the probability that a new reservation request is rejected. For the sake of simplicity, we restrict results and discussions to a maximum of 10 traffic classes *nc*; however, the model is general and scalable to (almost) any number of classes.

4.2. M/M(nc)/1/k(s)-RPS model

Without loss of generality, we model the path as a constant capacity transmission server with rate β_{Φ_r}, which serves customers (the requests r) in parallel with a guaranteed minimum rate MRR_r. With Markovian assumptions (explained and justified in the following) for arrivals and services, this system is an $M/M(nc)/1/k(s)\text{-}RPS$ queue, whose meaning is explained in Table II.

We assume that requests of each class arrive at the link according to a Poisson process. The process results naturally when a reasonably large population of Grid applications submit requests independently. Since there is no practical evidence that the pattern of arrivals for data transfers in Grids follows any special distribution, but strong indications that Grids support large numbers of users, the Poisson assumption holds.

We also take the volume of flows exponentially distributed with average $V = E[v_r]$. This assumption finds less support in the analysis of the system because *file* transfers in the Internet normally show non-exponential distributions. However, bulk transfers in Grids are supposed to come from computing tasks, and not from stored files, so that we do not find any strong reason not to use an

Table II. $M/M(nc)/1/k(s)\text{-}RPS$ queue.

Symbol	Meaning
M	Markovian arrivals with rate $\lambda = \sum_{\forall c \in \theta} \lambda_c,\ \theta = \{1, 2, \ldots, nc\}$
$M(nc)$	Exponential services with minimum rate MRR_c depending on class c, $\forall c \in \theta$
$k(S)$	Vector of per-state positions in the queue
RPS	Residual Processor Sharing: all customers in the system equally share the residual bandwidth on top of the minimum bandwidth guaranteed to each

exponential distribution. Besides, the model that we propose can be extended to hyper-exponential volume distributions, approximating any kind of distribution that may be measured in the future. Additionally, in Section 5.3, we will show that analytical results match well also when actual transfer sizes are non-exponential.

Requests r belong to one of nc classes, where the class of the request identifies its guaranteed minimum service rate MRR_c. The volume of requests V_c can be dependent on the class too.

Let n_c be the number of requests of class c. For the sake of simplicity, we take $C = \beta_{\Phi_r}$ onwards. The residual capacity of the system is

$$C_r = C - \sum_{\forall c \in \theta} n_c \cdot MRR_c. \tag{7}$$

The system can be modeled using an nc-dimensional birth–death process, whose state S is uniquely identified by the number of flows of each class $(n_1, n_2, n_3, \ldots, n_c, \ldots, n_{nc})$.

Given a state $S_i = (n_1, n_2, n_3, \ldots, n_c, \ldots, n_{nc})$ and states $S_j = (n_1, n_2, n_3, \ldots, n_c + 1, \ldots, n_{nc})$, and $S_k = (n_1, n_2, n_3, \ldots, n_c - 1, \ldots, n_{nc})$, and assuming (see Lemma 1 for the proof of it) that service rates are exponential, the birth and death transitions of the continuous time Markov chain (*CTMC*) modeling the birth–death process are

$$\lambda_{i,j} = \lambda_c \tag{8}$$

and

$$\mu_{i,k} = \left\{ \begin{array}{cc} \dfrac{n_c}{V_c} \cdot \left(MRR_c + \dfrac{C_r}{\sum_{h=1}^{nc} n_h} \right), & n_c > 0 \\ 0 & \text{otherwise.} \end{array} \right\} \tag{9}$$

Lemma 1
Service rates $\mu_{i,k}$ are exponentially distributed.

Proof
The volume v_r of any request is exponentially distributed by construction. The service rate for any request r in any given state S_i is constant by construction, so that the dwelling time is exponentially distributed for all requests upon entering the system. Thanks to the memoryless property of the exponential distribution, when the system changes state (new arrivals or departures) from S_i to S_j or S_k, the residual amount of information to be transmitted for all customers in the system is still exponentially distributed with the same average value, and the dwelling time in state S_j or S_k is again exponentially distributed, so that the transition rates defined in Equation (9) are the average values of exponentially distributed RVs.

Lemma 1 proves that indeed the queue we are using to model transfer paths is Markovian, so that a *CTMC* can be used to solve it.

Figure 2 depicts the *CTMC* in case of $nc = 3$, $C = 5$ and $MRR_1 = 1$, $MRR_2 = 2$, $MRR_3 = 3$. We have used dashed transition for the third dimension trying to highlight the 3-dimensional structure of the *CTMC*.

Writing the infinitesimal generator Q (matrix of the transition rates) of the *CTMC* is trivial given the birth–death structure; to find the steady-state solution, a number of well-known direct

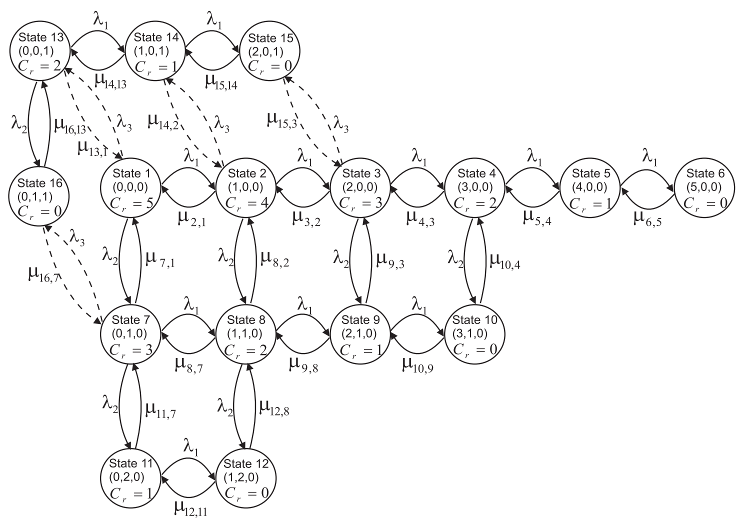

State 1 (0,0,0) $C_r = 5$
State 2 (1,0,0) $C_r = 4$
State 3 (2,0,0) $C_r = 3$
State 4 (3,0,0) $C_r = 2$
State 5 (4,0,0) $C_r = 1$
State 6 (5,0,0) $C_r = 0$
State 7 (0,1,0) $C_r = 3$
State 8 (1,1,0) $C_r = 2$
State 9 (2,1,0) $C_r = 1$
State 10 (3,1,0) $C_r = 0$
State 11 (0,2,0) $C_r = 1$
State 12 (1,2,0) $C_r = 0$
State 13 (0,0,1) $C_r = 2$
State 14 (1,0,1) $C_r = 1$
State 15 (2,0,1) $C_r = 0$
State 16 (0,1,1) $C_r = 0$
λ_1
λ_2
λ_3
$\mu_{2,1}$
$\mu_{3,2}$
$\mu_{4,3}$
$\mu_{5,4}$
$\mu_{6,5}$
$\mu_{7,1}$
$\mu_{8,2}$
$\mu_{9,3}$
$\mu_{10,4}$
$\mu_{8,7}$
$\mu_{9,8}$
$\mu_{10,9}$
$\mu_{11,7}$
$\mu_{12,8}$
$\mu_{12,11}$
$\mu_{14,13}$
$\mu_{15,14}$
$\mu_{16,13}$
$\mu_{13,1}$
$\mu_{14,2}$
$\mu_{15,3}$
$\mu_{16,7}$

Figure 2. Sample CTMC solving the $M/M(nc)/1/k(s)$-RPS queue: $nc=3$, $C=5$ and $MRR_1=1$, $MRR_2=2$, $MRR_3=3$.

and iterative solution techniques exist [31]. We have used a direct method to find the solution as the number of states in the *CTMCs* we explore is up to a few thousand at most. For large state spaces arising from Grids supporting tens of thousands of flows at the same time, any iterative solution technique can be used as in [19].

Of the many performance indices that can be derived from the *CTMC*, we are mostly interested in the transfers blocking probabilities BP_c and the expected number of jobs (transfers) E_c supported by the system for each class. If required, the entire distribution of the number of jobs can be derived from the steady-state distribution $\pi=\{\pi_{S_1}, \pi_{S_2}, \ldots, \pi_{S_N}\}$, where N is the number of states in the *CTMC*.

4.2.1. Computing blocking probabilities. Blocking states of a given traffic class are those states in which a new arrival of that class would result in an infeasible scheduling. An infeasible scheduling for class c has $C_r < MRR_c$. The blocking probabilities of the different classes are obtained from the steady-state probabilities of blocking states.

Let S be the set of all states, S_B the set of states which are blocking for at least one class of jobs, and S_{B_c} be the set of states blocking for class c: $S_B = \bigcup_c S_{B_c}$. The blocking probability BP_c of class c and the overall blocking probability BP are given as follows:

$$BP_c = \sum_{\forall S_i \in S_{B_c}} \pi_{S_i}, \tag{10}$$

$$BP = \left[\sum_{c=1}^{nc}\left[\sum_{\forall S_i \in S_{B_c}} \pi_{S_i}\lambda_c\right]\right]\frac{1}{\lambda}. \tag{11}$$

For illustration purposes, consider again the system in Figure 2. For this system, we have:

$$MRR = \{MRR_1=1, MRR_2=2, MRR_3=3\},$$

$$S=\{S_1, S_2, \ldots, S_{16}\},$$

$$S_B=\{S_4, S_5, S_6, S_8, S_9, S_{10}, S_{11}, S_{12}, S_{13}, S_{14}, S_{15}, S_{16}\},$$

$$S_{B_1} = \{S_6, S_{10}, S_{12}, S_{15}, S_{16}\},$$
$$S_{B_2} = \{S_5, S_6, S_9, S_{10}, S_{11}, S_{12}, S_{14}, S_{15}, S_{16}\},$$
$$S_{B_3} = \{S_4, S_5, S_6, S_8, S_9, S_{10}, S_{11}, S_{12}, S_{13}, S_{14}, S_{15}, S_{16}\},$$
$$\pi = \{\pi_{S_1}, \pi_{S_2}, \ldots, \pi_{S_{16}}\}.$$

4.2.2. Expected number of jobs. Let $n_c(S_k)$ be the number of jobs of class c in state S_k; the expected number of jobs E_c of class c and the expected number of jobs E of all classes in the system is

$$E_c = \sum_{k=1}^{N} n_c(S_k)\pi_{S_k}, \tag{12}$$

$$E = \sum_{c=1}^{nc} E_c. \tag{13}$$

4.3. Planning capacity/dimensioning and number of classes

Algorithms 1 and 2 can be used to plan minimum required capacity (*MRC*) and number of classes (*nc*), so that the overall blocking probability is below a certain threshold.

In the following, we apply the model derived so far to compute blocking probabilities and plan the capacity for Grids; results are validated through simulations, both using an *ad hoc* simulator for scalability and realistic ns-2 simulations for small cases to test the impact of TCP and other protocols on the performance.

5. PERFORMANCE EVALUATION AND PLANNING

The main objectives are

- Computation of the blocking probabilities.
- Network capacity planning/dimensioning and number of classes to be supported to keep the blocking probability below a certain threshold.

Algorithm 1 Capacity planning.

```
Procedure MinimumRequiredCapacity
Inputs:
Mean volume of requests: $V$
Number of classes: $nc$
$MRR$ vector: $\{MRR_1, MRR_2, \ldots, MRR_{nc}\}$
Arrival rates: $\lambda_c$
Threshold of overall blocking probability: $BP_{\text{Thresh}}$
Output:
Minimum Required Capacity: $MRC$

$C=0$
$BP=1$
Generate the $CTMC$ of the system
while $BP \geq BP_{Thresh}$ do
    $C=C+1$
    Calculate arrival $\lambda_c$ and service $\mu_{i,k}$ rates
    Generate the infinitesimal generator matrix $Q$
    Calculate $\pi$ by finding steady-state solution of $\pi Q=0$
    Calculate $BP$
end while
$MRC=C$
End Procedure
```

Algorithm 2 Planning the required number of classes.

```
Procedure NumberofClasses
Inputs:
Mean volume of requests: $V$
Capacity: $C$
$MRR$ vector: $\{MRR_1, MRR_2, \ldots, MRR_{nc}\}$
Arrival rates: $\lambda_c$
Threshold of overall blocking probability: $BP_{Thresh}$
Output:
Number of classes: $nc$

$nc = 0$
$BP = 0$
Generate $CTMC$ of the system
while $BP \leq BP_{Thresh}$ do
    $nc = nc + 1$
    Calculate arrival $\lambda_c$ and service $\mu_{i,k}$ rates
    Generate the infinitesimal generator matrix $Q$
    Calculate $\pi$ by finding steady-state solution of $\pi Q = 0$
    Calculate $BP$
end while
$nc = nc - 1$
End Procedure
```

Analytical results are validated through simulations with and without taking network-level details into account.

For the sake of simplicity, we take the average request volume of all classes equal to $V_c = V, \forall c \in \theta$, and we take $MRR_c = c, \forall c \in \theta$, although we stress that any values of MRR_c can be used; also arrival rates are identical for all classes for the same reason and $\lambda = 0.25$ in all calculations and simulations. The actual load of the path is thus $\lambda V/C$.

Some experiments and results are discussed in the following: (i) in Section 5.1, we discuss the blocking probability as a function of the number of classes, which describes the granularity of the requests compared with the available resources; (ii) in Section 5.2, we present planning results for the path capacity C and the number of classes *nc* to be supported; and (iii) in Sections 5.3 and 5.4, we validate the selected results for which the computational burden of simulations is not prohibitive.

5.1. *Effect of nc and $C/\max(MRR_c)$*

Figure 3 presents the overall *BP* as a function of *nc* (varied between 2 and 10) and $C = 25$ Gbps. Figure 3 shows that the higher the *nc*, the greater the overall *BP*. This is due to the reason that, when *nc* is high, a flow with a higher MRR_c finds a smaller chance of being accepted than a flow with a lower MRR_c, which consequently increases the overall *BP*. Figure 4 shows the blocking probabilities of each class when 10 classes are used. The results in Figure 4 show that the higher the C to MRR_c ratio, the greater the BP_c become, which is due to the lower acceptance chances of flows with higher MRR_c values than the acceptance chances of flows with lower MRR_c values.

5.2. *Planning capacity and classes*

Dimensioning a path means finding the *MRC* to support a prescribed maximum BP for each class. *MCR* is calculated for given values of $V s^{-1}$ and BP_{Thresh}. The *nc* is set to 10 for this experiment. The results of the experiment are shown in Figure 5.

In yet another experiment, *nc* is calculated for given values of $V s^{-1}$ and BP_{Thresh}. We take $MRR_c = c; \forall c \in \theta$. C is set to 20 Gbps for this experiment. The results of the experiment are shown in Figure 6.

5.3. *Validation*

We take $MRR_1 = 1$ and $MRR_2 = 2$ for two classes and calculate the overall *BP* as well as the individual *BP* of each class. The simulation code is written in C++. The simulations are run for

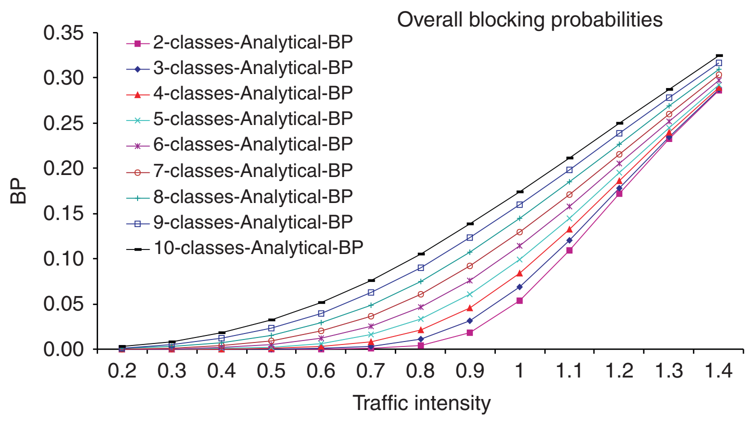


Figure 3. Overall blocking probabilities for: $C = 25$ Gbps; $MRR_c = c$ Gbps, $\forall c \in \theta$; nc is varied from 2 to 10.

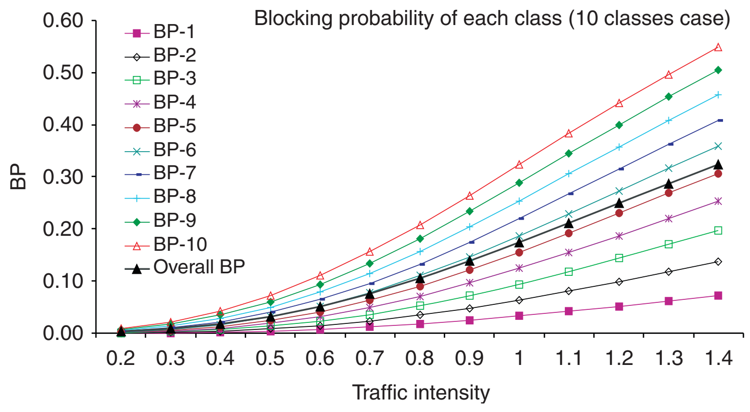


Figure 4. Per-class blocking probabilities for: $C = 25$ Gbps; $MRR_c = c$ Gbps, $\forall c \in \theta$; $nc = 10$.

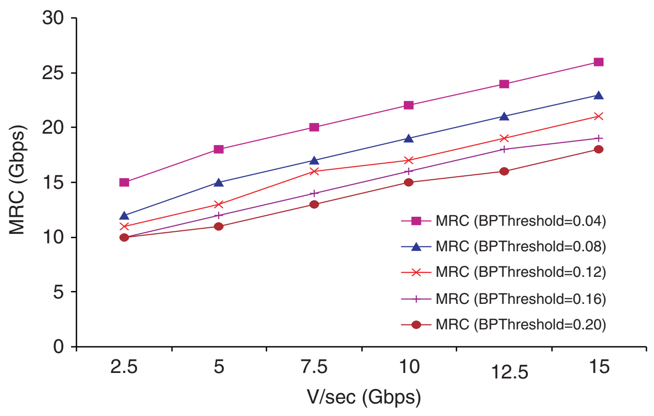


Figure 5. Planning the capacity C.

the same traffic intensities as are used for the analytical results. For each measurement, we run 10 simulations and take the average. Each simulation is run for 10 000 flows. The results in Figure 7 show that the analytical results match the simulations extremely well.

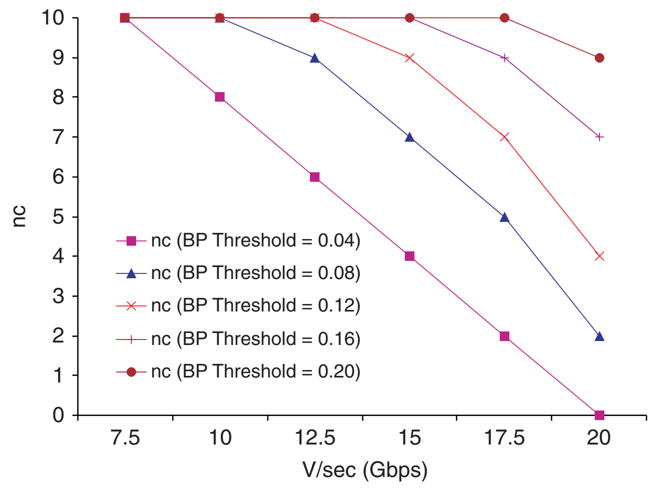


Figure 6. Planning the number of classes *nc*.

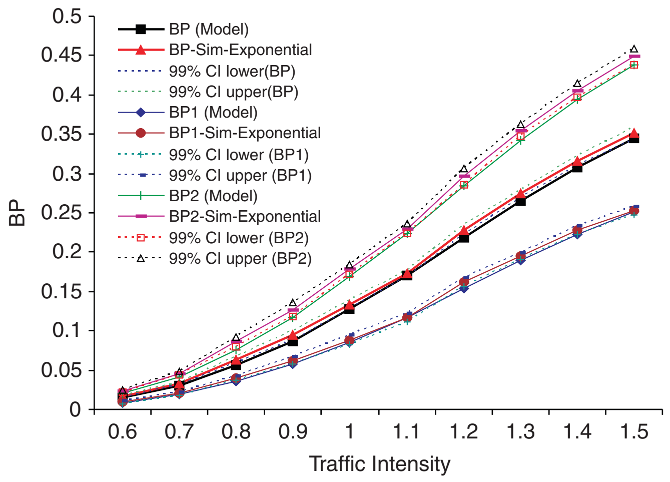


Figure 7. Validation: service times according to exponential distribution.

We have also considered Uniform and Bounded Pareto distributions for volumes of requests. The service times in the case of the Uniform distribution lie in the interval $[1, 2 \times V]$. A Bounded Pareto distribution is characterized by three parameters: α, the exponent of the power law; k, the smallest possible request volume; and p, the largest possible request volume. The probability mass function for the Bounded Pareto $B(k, p, \alpha)$ is taken from [32], and is given as follows:

$$f(x) = \frac{\alpha k^{\alpha}}{1 - \left(\frac{k}{p}\right)^{\alpha}} x^{-\alpha-1}, \quad k \leq x \leq p. \tag{14}$$

We take the parameters for the heavy-tailed distribution, for the service times, as $\alpha = 1.1$, $5 \leq k \leq 15$, and $p = 1500$ to get the desired values of V.

The results in Figures 8 and 9 show that even relaxing the hypothesis of exponential distribution of requests, the CTMC results are still a very accurate approximation of the results obtained with very different distributions. The reason is probably rooted in the processor sharing features of the system, even if applied only to residual resources. We recall that the $M/G/1$-PS queue is insensitive to service distributions as far as the client distribution is concerned. We are also aware

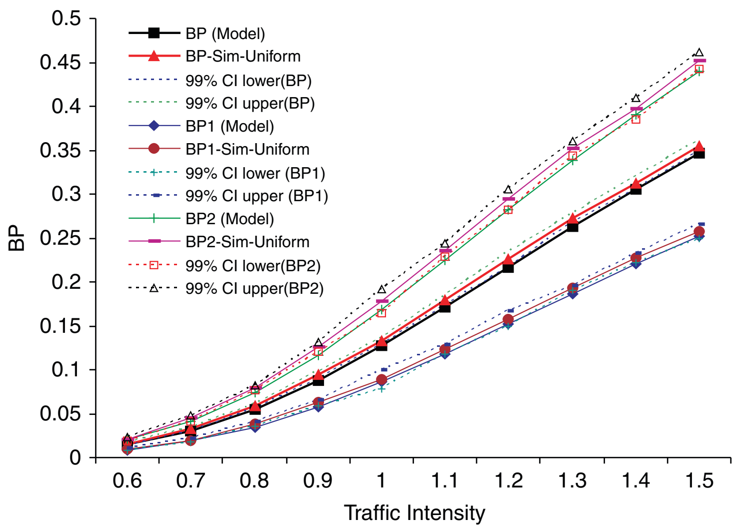


Figure 8. Validation: service times according to uniform distribution.

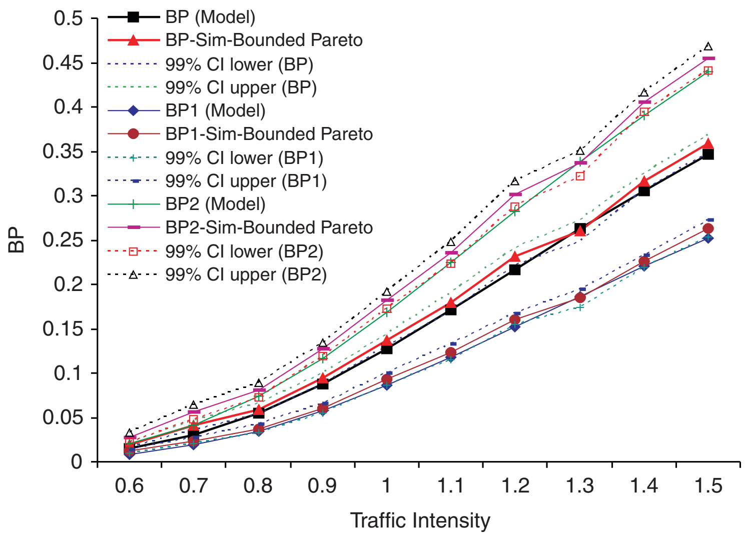


Figure 9. Validation: service times according to bounded Pareto distribution.

that limiting the queue size and having minimum guaranteed resources impair this result in theory; however, the results presented here show that the CTMC model remains a very good approximation even in the presence of non-exponential requests.

5.4. *Validation against ns-2 simulations*

Finally, we take $C/max(MRR_c)=2.5, \forall c \in \theta$ for 10 classes and calculate the overall *BP* taking the network conditions into account. To estimate TCP throughput, we use Equation (13) to calculate the expected number of jobs E in the system which is used in a formula from [33] to estimate the aggregate throughput $x(E)$ of E connections sharing a bottleneck link of fixed capacity C as given as follows:

$$x(E)=C\left(1-\frac{1}{1+3E}\right). \tag{15}$$

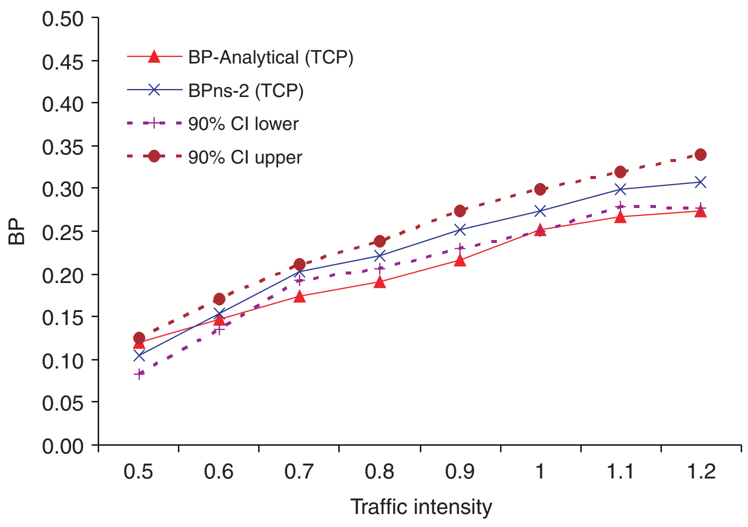


Figure 10. Estimation of BP with standard TCP.

Algorithm 3 Estimation of blocking probability of the mechanism combined with TCP.

```
**Procedure BP Estimation for TCP**
Inputs:
Mean volume of requests: $V$
Number of classes: $nc$
$MRR$ vector: $\{MRR_1, MRR_2, \ldots, MRR_{nc}\}$
Probability vector: $\{p_1, p_2, \ldots, p_{nc}\}$
Arrival rate: $\lambda$
Output:
Blocking Probability: $BP$

$\varepsilon = 1$
$C' = C$
$E = 1$
while $\varepsilon > 0.01$ do
    $E' = E$
    Generate $CTMC(C')$, get $E$, calculate $BP$
    Compute $x(E)$
    $C' = C' \times x(E)$
    $\varepsilon = ABS(E - E')/E$
end while
End Procedure
```

Algorithm 3 estimates the blocking probability when the mechanism is combined with standard TCP.

To validate, we run *ns*-2 simulations using the same traffic loads (intensities) by combining the mechanism with standard TCP. A single bottleneck link dumbbell network configuration is used for the simulations using *ns*-2. The bottleneck capacity is 1 Gbps and the bottleneck delay is set to 50 ms. Drop Tail routers are used. The buffer size of the bottleneck link is set to 100% of the Bandwidth–Delay product. The packet size is set to 1500 B. The capacity of side links is 10 Gbps and the delay of each side link is set to 2 ms. For each measurement, we run 10 simulations and take the average of them. Each simulation is run for 100 flows. The 90% confidence interval curves are plotted along with the simulation curve. The blocking probabilities are shown in Figure 10.

The results show remarkably good coherence even under heavy loads. The analytical curve for TCP is not smooth due to rounding of C' and also due to using an approximation of the TCP throughput. The simulation curve shows that the BP obtained from simulation is generally more than the analytically calculated BP for higher traffic intensities. This is due to the reason that, when traffic intensity increases, TCP performance deteriorates and consequently the BP increases.

6. CONCLUSION

In this paper, we have proposed a novel queueing system, named $M/M(nc)/1/k(s)\text{-}RPS$ to model data transfers in Grid computing applications over a shared path. We have used the model to calculate the blocking probability of requests and to dimension path capacities and requests admission classes. The results proved to be extremely accurate not only in the case of exponential requests, but also if requests are non-Markovian.

We have shown that, given the mean volume of requests per second, the multi-rate model can be used to calculate the necessary capacity of a bottleneck link as well as the number of classes that can be supported in order to have the blocking probability of requests below a certain threshold. This information is useful for a Grid network provider because over-dimensioning wastes precious bandwidth resources, while under-dimensioning generally leads to less satisfactory *QoS* perceived by Grid users.

ns-2 simulations were performed to check whether the model fits reality. We have seen that real rate-control methods like standard TCP combined with the CAC mechanism appear to provide a good approximation of the model, which shows that the model is representative of the actual system behavior.

ACKNOWLEDGEMENTS

The authors would like to thank the anonymous referees for their helpful comments.

REFERENCES

1. Roberts JW. A survey on statistical bandwidth sharing. *Computer Networks* 2004; **45**(3):319–332.
2. Casetti C, Lo Cigno R, Mellia M, Munafò Zsóka Z. A new class of QoS routing strategies based on network graph reduction. *Computer Networks* 2003; **41**(4):475–487.
3. Salvadori E, Lo Cigno R, Zsóka Z. Dynamic grooming in IP over optical networks based on the overlay architecture. *Optical Switching and Networking* 2006; **3**(2):118–133.
4. Bonald T, Roberts JW. Congestion at flow level and the impact of user behaviour. *Computer Networks* 2003; **42**(4):521–536.
5. Boyer J, Guillemin F, Robert P, Zwart B. Heavy tailed M/G/1-PS queues with impatience and admission control in packet networks. *Proceedings of the IEEE INFOCOM*, San Francisco, CA, U.S.A., 2003.
6. Yousaf MM, Welzl M, Yener B. On the accurate identification of network paths having a common bottleneck. *ACM SIGCOMM*, poster, 2008.
7. Yousaf MM, Welzl M, Yener B. Accurate shared bottleneck detection based on SVD and outlier detection. *DPS NSG Technical Report 1*, Institute of Computer Science, University of Innsbruck, 2008.
8. Yousaf MM, Iqbal J, Welzl M. A network topology mapping tool for the grid. *Seventh ACM International Conference on Frontiers of Information Technology* (*FIT*), Abbottabad, Pakistan, 2009.
9. The Grid'5000 project. Available at: http://www.grid5000.fr/ [12 April 2011].
10. Pasin M, Primet Vicat-Blanc P. Network resource reservation and virtualization for grid applications. *International Advanced Research Workshop on High Performance Computing and Grids*, Cetraro, Italy, 2008.
11. Munir K, Primet Vicat-Blanc P, Welzl M. Grid network dimensioning by modeling the deadline constrained bulk data transfers. *Proceedings of the 11th IEEE International Conference on High Performance Computing and Communications* (*HPCC 2009*), Seoul, South Korea, 2009.
12. Munir K, Javed S, Welzl M, Ehsan H, Javed T. An end-to-end QoS mechanism for grid bulk data transfer for supporting virtualization. *Proceedings of the IEEE/IFIP International Workshop on End-to-end Virtualization and Grid Management* (*EVGM 2007*), San Jose, CA, U.S.A., 2007.
13. Kaufman JS. Blocking in a shared resource environment. *IEEE Transactions on Communications* 1981; **29**(10):1474–1481.
14. Bonald T, Roberts J. Performance modeling of elastic traffic in overload. *Proceedings of the ACM SIGMETRICS*, Cambridge, MA, U.S.A., 2001.
15. Berger AW, Kogan Y. Dimensioning bandwidth for elastic traffic in high-speed data networks. *IEEE/ACM Transactions on Networking* 2000; **8**(8):643–654.
16. Fan Z. Dimensioning bandwidth for elastic traffic (*Lecture Notes in Computer Science*, vol. 2345). Springer: Berlin, 2002; 826–837.
17. Phojanamongkolkij N, Cochran JK, Fowler JW. Multi-products multi-servers bulk service queue with threshold service size. *Proceedings of the International Conference on Semiconductor Manufacturing Operational Modeling and Simulation*, San Francisco, CA, U.S.A., 1999; 153–156.

18. AlQahtani SA, Mahmoud AS. Performance analysis of two throughput-based call admission control schemes for 3G WCDMA wireless networks supporting multiservices. *Computer Communications* 2008; **31**(1):49–57.
19. Fodor G, Rácz S, Telek M. On providing blocking probability- and throughput guarantees in a multi-service environment. *International Journal of Communication Systems* 2002; **15**(4):257–285.
20. Foster I, Roy A, Sander V. A quality of service architecture that combines resource reservation and application adaptation. *Proceedings of the Eighth International Workshop on Quality of Service* (*IWQoS 2000*), Westin William Penn, Pittsburgh, U.S.A., 2000.
21. Foster I, Fidler M, Roy A, Sander V, Winkler L. End-to-end quality of service for high-end applications. *Computer Communications* 2004; **27**(4):1375–1388.
22. Foster I, Kesselman C, Lee C, Lindell R, Nahrstedt K, Roy A. A distributed resource management architecture that supports advance reservations and co-allocation. *Proceedings of the Seventh International Workshop on Quality of Service* (*IWQoS 1999*), London, U.K., 1999.
23. Wu L, Xing J, Wu C, Cui J. An adaptive advance reservation mechanism for grid computing. *Proceedings of the PDCAT*, Dalian, China, 2005.
24. Burchard L, Heiss H, Rose D. Performance issues of bandwidth reservations for grid computing. *Proceedings of the Computer Architecture and High Performance Computing*, Sao Paulo, Brazil, 2003.
25. Xing J, Wu C, Tao M, Wu L, Zhang H. Flexible advance reservation for grid computing. *Proceedings of the GCC*, Wuhan, China, 2004.
26. Soudan S, Chen BB, Primet Vicat-Blanc P. Flow scheduling and endpoint rate control in GridNetworks. *Future Generation Computer Systems* 2009; **25**(8):904–911.
27. Naiksatam S, Figueira S. Elastic reservations for efficient bandwidth utilization in LambdaGrids. *The International Journal of Grid Computing*: *Theory*, *Methods and Applications—FGCS* 2007; **23**(1):1–22.
28. Bhatia R, Kodialam M, Lakshman TV. Fast network re-optimization schemes for MPLS and optical networks. *Computer Networks* 2006; **50**(3):317–331.
29. Venugopal S, Buyya R. A deadline and budget constrained scheduling algorithm for eScience applications on data grids. *Proceedings of the Sixth International Conference on Algorithms and Architectures for Parallel Processing*. Springer: Berlin, 2005.
30. Venugopal S, Buyya R. A set coverage-based mapping heuristic for scheduling distributed data-intensive applications on global grids. *Proceedings of the Seventh IEEE/ACM International Conference on Grid Computing* (*Grid 2006*). IEEE Computer Society Press: Silver Spring, MD, 2006.
31. Stewart WJ. *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press: Princeton, NJ, 2004.
32. Balter MH, Crovella M, Murta C. On choosing a task assignment policy for a distributed server system. *Journal of Parallel and Distributed Computing* 1999; **59**(2):204–228.
33. Altman E, Barman D, Tuffin B, Vojnovic M. Parallel TCP sockets: Simple model, throughput and validation. *Proceedings of the IEEE INFOCOM 2006*, Barcelona, Spain, 2006.