Expert Systems with Applications 38 (2011) 8913–8920

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Joint congestion control and processor allocation for task scheduling in grid over OBS networks

Yahong Yang [a,*], Guiling Wu [b], Wei Dai [b], Jianping Chen [b]

[a] *Shanghai Institute of Space-Power Supply, 2965 Dongchuan Road, Shanghai 200245, China*
[b] *The State Key Laboratory on Advanced Optical Communication Systems and Networks, Shanghai Jiao Tong University, 800 Dongchuan Road, Shanghai 200240, China*

A R T I C L E I N F O

Keywords:
OBS
Grid
Cross-layer design
Congestion control
Processor allocation
Load balancing

A B S T R A C T

This paper presents a task scheduling algorithm that fulfills the coordination of congestion control in conjunction with processor allocation in the grid over optical burst switching networks. Two requirement criteria, namely deadline and payment, are considered and formulated as a user utility maximization function. The optimization problem is decomposed into two parts: congestion control for network transmission rate adjustment and processor allocation for computational capacity adjustment. The parameters from the resource layer are abstracted and provided to a cross-layer optimizer to maximize user's utility function. The effectiveness and performance of the algorithm are evaluated via simulations. Comparison with other algorithms shows the efficiency of the proposed algorithm.

1. Introduction

Grid is becoming attractive for its ability to provide super computing capacities and better sharing of distributed resources (Foster, Kesselman, & Tuecke, 2001). With the expansion of grid applications and the development of large-scale computation and data-intensive traffic, the network becomes data transmission bottleneck between resources. Hence, integrating grid resources with emerging high-performance optical network technologies, including fast optical switching and dense wavelength division multiplexing (DWDM), appears to be the natural choice (Mambretti et al., 2003). Optical burst switching (OBS) is considered as a promising grid underlying network solution because of its low delay, variable burst length and separate control (Qiao & Yoo, 1999). In grid over OBS (GoOBS) environment, resources are widely distributed and owned by many different organizations. A good resource management system is essential to exert the full advantage of GoOBS. The management system is responsible for resource discovery, resource selecting, task scheduling and resource maintenance, where the scheduling algorithm is one of the most important issues (Tseng, Chin, & Wang, 2009). Most of the existing grid resource allocation and scheduling algorithms mainly focus on isolated layers of the grid architecture. The inflexibility of the rigid layered structure results in an inefficient utilization of resources (Baker, Buyya, & Laforenza, 2002). Especially, in OBS networks, because of the optical switch's bufferless property, burst loss mostly induced by contention may be mistaken for heavy congestion so that a time out event will trigger unnecessary TCP congestion control resulting in significant throughput degradation (Yu, Qiao, & Liu, 2004). Thus, the rigid layered scheme of traditional grid does not suit to the unique characteristics of GoOBS and it is very important and necessary to deal with GoOBS system as a whole to achieve the optimal performance. Cross-layer design is based on information exchange and joint optimization among the multiple layers so that it allows us to propagate ambient parameter changes quickly throughout the multiple layers (Shakkottai, Rappaport, & Karlsson, 2003). This idea does not absolutely deny the layered model but blur the strict layered limits by integrating the characteristic parameters distributed in the sublayers of grid. Therefore, it is well suited to the dynamic, autonomous and heterogeneous GoOBS environment (Yang, Wu, Li, & Chen, 2009).

This paper proposes a joint congestion control and processor allocation algorithm for GoOBS task scheduling. Two requirement criteria, namely deadline and payment, are considered and formulated as a user utility maximization function. A utility model is a simple and general means for users to specify their preference (Lee, Lee, & Sohn, 2009), which is suitable for dynamic and heterogeneous GoOBS. In order to maximize user utility, the congestion control for network transmission rate adjustment and processor allocation for computational capacity adjustment is studied. Congestion control mechanisms, such as those in transmission control protocol (TCP), regulate the allowed source rates according to the network congestion so that the total traffic load on any link does not exceed the available capacity. At the same time, the attainable grid user utility also depends on the processor allocation that partitions a certain number of system processors to jobs in order to avoid system saturation (Dussa, Carlson, Dowdy, & Park,

* Corresponding author.
E-mail addresses: yuer1224@gmail.com, yuer1224@tom.com (Y. Yang).

1990). Simulations are carried out to verify the performance of the proposed algorithm. Deadline and budget constrained (DBC) scheduling algorithm is a widely used algorithm, which employs economy-driven method to allocate resources to application jobs in such a way that the grid user's requirements are met (Buyya, Murshed, & Abramson, 2002). However, it does not take into account the changes of grid resources and user requirements and might be unable to allocate resources efficiently. The algorithm we proposed adopts the idea of cross-layer design and aims to dynamically meet the requirements of the upper layer applications by adjusting the resources and parameters in underlying layers.

The rest of the paper is arranged as followings: Section 2 presents GoOBS task scheduling modeling and optimization solutions. In Section 3 the simulations are conducted to validate our proposed algorithm. Section 4 concludes the paper.

2. GoOBS task scheduling modeling and optimization solutions

2.1. Fundamental

In OBS networks, due to the bufferless nature of optical switch, when multiple bursts contend for the same link at about the same time, only one burst can be successfully switched in the optical domain (Wang, 2003). All other overlapping bursts need to be dropped. This results in low link utilizations and high burst drop rates. Some existing contention resolution schemes such as fiber-delay lines, wavelength conversion and deflection routing may be used in OBS networks. However, these existing schemes have their own drawbacks. On the other hand, although they can reduce the degree of contention and thus the packet drop rate some what, they are ineffective when the offered load is excessively high (Chen, Kuo, Yan, & Liao, 2009; Wang, 2003). It is thus very necessary to use a congestion control mechanism to control the load offered to OBS networks. Congestion control dynamically regulates the transmission rate of each connection using feedback information from network so that congestion is controlled or even avoided. It is therefore especially suitable for data transfer service (Ohsaki, Murata, Suzuki, Ikeda, & Miyahara, 1995). In the Internet, TCP plays the role of network congestion control and end-to-end rate allocation. TCP uses sliding windows to adjust the allowed transmission rate in each source based on implicit or explicit feedback of the congestion signals. Different versions of TCP adopt different congestion measures, such as by packet loss (TCP Reno) (Low, Paganini, Wang, Adlakha, & Doyle, 2002), by queuing delay (TCP Vegas) (Brakmo & Peterson, 1995) and by queue length (TCP RED) (Floyd & Jacobson, 1993). In this paper, we use TCP Reno at the sources because of its popularity (Low et al., 2002).

In the parallel computing system, the resource requirements may vary significantly among the jobs and the demands on the resources may be unpredictable (Yu & Zhou, 2010). Additionally, parallel applications may not be able to efficiently utilize all the processors in the system because the efficiency of parallel jobs generally decreases as their processor allocation increases (Dussa et al., 1990). The efficient job scheduling that maximizes throughput while maintaining job load balancing has always been a critical issue. The various studies have shown the processor allocation for parallel computing system is a good solution, which ensures both that no processors are needlessly idle and that jobs exhibit good load balancing (McCann & Zahorjan, 1994). Processor allocation is to partition the system processors into disjoint sets that are allocated to individual jobs, with the objective of maximizing throughput over many jobs. There are three policies. Static scheduling allocates fixed number of processors to each job (Majumdar, Eager, & Bunt, 1991). It is simple to implement at the cost of lower performance. Dynamic scheduling allows the number of processors to vary during its execution (Dussa et al., 1990). The overhead for reallocating processors during execution may be considerable. Adaptive scheduling adjusts processors according to the workloads by calculating a job's partition size (Ghosal, Serazzi, & Tripathi, 1991). The partition size does not change until the job execution completes. The overhead is minimal. Adaptive scheduling is regarded as the best approach for processor allocation (Ghosal et al., 1991; McCann & Zahorjan, 1994) and we adopt it in our algorithm.

Augmenting the utility maximization framework to include layers other than transport layer leads to a general methodology for cross-layer design, which jointly optimizes the parameters of all layers based on information exchange. It can improve the throughput and load balancing of the grid.

2.2. Model and solutions

Consider a GoOBS environment consisting of Q nodes, where some nodes are edge nodes and some nodes act as core nodes. A part of the edge nodes connecting F client hosts to submit user's requirement. The others connecting M servers acting as computational resource (denoted as $R_1, R_2, \ldots, R_M$). Each computational resource includes a certain number of processors, which are different in terms of computational capacities and prices of CPU time. OBS nodes are interconnected by links. Let C be the capacity of link. Suppose a grid user has a task consisting of N jobs (denoted as $J_1, J_2, \ldots, J_N$) to be processed in GoOBS environment. It is desired to complete as many jobs as possible under a certain time (denoted as T_0) and budget (denoted as E_0) limits. The objective of the task scheduling is to assign qualities and resources so that the grid user utility is maximized subject to the resource and user preferences constraints. Without losing the generality, we made the following simplifications: (1) Jobs are inter-independent. (2) Once a job is initiated, it will be completed without preemption. The notations used in the following sections are listed in Table 1.

The user utility function we proposed is as follows:

$$\begin{aligned}
Max\Bigg\{&\omega_1\left(T_0-\sum_{s=1}^{F}\sum_{k=1}^{M}\sum_{i=1}^{N}\frac{\phi_i^{sk}d_i}{y_s^k}-\sum_{k=1}^{M}\sum_{i=1}^{N}\frac{\varphi_i^k b_i}{Z_k}-D\right)\\
&+\omega_2\left(E_0-\upsilon\times\sum_{s=1}^{F}\sum_{k=1}^{M}\sum_{i=1}^{N}\frac{\phi_i^{sk}d_i}{y_s^k}\right.\\
&\left.-\sum_{k=1}^{M}\sum_{i=1}^{N}c_k\frac{Z_k}{P_k}\times\frac{\varphi_i^k b_i}{Z_k}\right)\Bigg\}\\
\text{s.t.}\quad &\sum_{s=1}^{F}y_s^k\leqslant C\quad(k=1,2,\ldots,M)\\
&\sum_{k=1}^{M}Z_k\leqslant\sum_{k=1}^{M}\alpha_k P_k\\
&\phi_i^{sk}=0\text{ or }1,\ (i=1,2,\ldots,N;\ s=1,2,\ldots,F;\ k=1,2,\ldots,M)\\
&\varphi_i^k=0\text{ or }1,\ (i=1,2,\ldots,N;\ k=1,2,\ldots,M)\\
&y_s^k>0,\quad Z_k>0
\end{aligned}\tag{1}$$

where ω_1 *and* ω_2 *stand for the weights of deadline and payment, respectively. They provide the clients with the right to flexibly specify their weights for money and time (Feng, Song, Zheng, & Xia, 2004). The deadline is the overall time spent by a task in the grid. The task deadline can be broken into two parts: computational time and transmission time. Payments include those for computational resource and network bandwidth resource.* d_i *and* b_i *are the data size and computational quantity of the ith job, respectively.* y_s^k *denotes the TCP transmission rate in the path from the client node s to the destination resource node k. If* d_i *value of job i is really transmitted from source node s to*

Table 1
The description of notations.

Notations meanings	
Q	The total number of OBS nodes
F	The number of client nodes
M	The total number of computational resources
C	The capacity of link
N	The number of jobs
T_0	The time limit given by a task
E_0	The budget limit given by a task
w_i	The priority weight assigned by the grid
b_i	The computational quantity of the ith job
d_i	The data size of the ith job
y_s^k	TCP transmission rate in the path from the client node s to the destination resource node k
Z_k	The required computational capacity from resource k
D	The delay time
v	The cost per unit time of OBS network
c_k	The cost of single processor per unit CPU time of the kth computational resource
P_k	The computational capacity of individual processor
α_k	The total number of processors of the kth computational resource
λ_s^k	The burst loss ratio of the TCP path from the client node s to the destination node k
ρ_l	The burst loss ratio of link l in the TCP path from the client node s to the destination node k
η_k	The difference between the load on computational resource k and the average load
L_k	The time spent on the computational resource k
β_k	The assigned number of processors belonging to computational resource k
RLD	The criterion to measure the load balancing degree of GoOBS system, which is the sum of computational RLD and network RLD

destination node k, then the value of ϕ_i^{sk} will be 1 otherwise ϕ_i^{sk} is 0. If we really assign b_i value to job i, then the value of φ_i^k will be 1 otherwise φ_i^k is 0. Z_k denotes the required computational capacity from resource k. D denotes the delay time. v is the cost per unit time of OBS network. c_k is the cost of individual processor per unit CPU time of the kth computational resource and P_k is the computational capacity of individual processor. c_kZ_k/P_k indicates the required cost of processors per unit CPU time when the computational capacity Z_k is assigned to jobs. The constraint condition implies that the aggregate network flow rate does not exceed the capacity C and the aggregate computational capacity does not exceed the total computational capacity α_kP_k. α_k is the total number of processors of the kth computational resource.

This task scheduling model is a nonlinear optimization problem. We can apply the Lagrangian method to solve such a problem. Consider the Lagrangian multipliers λ_s^k and η_k as, respectively:

$$\lambda_s^k = 1 - \Pi(1-\rho_l) \tag{2}$$

$$\eta_k = \left| L_k - \sum_{k=1}^{M} L_k/M \right| \tag{3}$$

where λ_s^k represents the burst loss ratio of the TCP path from the client node s to the destination node k, ρ_l is the burst loss ratio of link l in the TCP path. η_k represents the difference between the load on computational resource k and the average load. It reflects the load balancing of each computational resource and the less the difference the more balanced the resource. L_k is the time spent on the computational resource k. M is the total number of computational resources.

Then the Lagrangian form of this optimization problem is as follows:

$$\begin{aligned} L(y_s^k, Z_k; \lambda_s^k, \eta_k) = {} & \omega_1\left(T_0 - \sum_{s=1}^{F}\sum_{k=1}^{M}\sum_{i=1}^{N}\frac{\phi_i^{sk}d_i}{y_s^k} - \sum_{k=1}^{M}\sum_{i=1}^{N}\frac{\varphi_i^k b_i}{Z_k} - D\right) \\ & + \omega_2\left(E_0 - v \times \sum_{s=1}^{F}\sum_{k=1}^{M}\sum_{i=1}^{N}\frac{\phi_i^{sk}d_i}{y_s^k} - \sum_{k=1}^{M}\sum_{i=1}^{N} c_k\frac{\varphi_i^k b_i}{P_k}\right) \\ & - \left(\sum_{s=1}^{F}\sum_{k=1}^{M}\lambda_s^k y_s^k - C\right) - \sum_{k=1}^{M}(\eta_k Z_k - \alpha_k P_k) \end{aligned} \tag{4}$$

From Karush–Kuhn–Tucker theorem, we know that the optimal solution is given $\partial L(y,Z)/\partial y = 0$ for $\lambda_s^k \geqslant 0$, i.e.,

$$\frac{\partial L(y_s^k, Z_k)}{\partial y_s^k} = (\omega_1 + \omega_2 \times v) \times \frac{\sum_{i=1}^{N}\phi_i^{sk}d_i}{(y_s^k)^2} - \lambda_s^k = 0 \tag{5}$$

Then we can obtain the unique optimal transmission rate $(y_s^k)^*$ that maximizes the grid user utility

$$(y_s^k)^* = \sqrt{\frac{(\omega_1 + \omega_2 \times v) \times \sum_{i=1}^{N}\phi_i^{sk}d_i}{\lambda_s^k}} \tag{6}$$

In TCP layer, the source rate x_s^k is adjusted by TCP Reno since it uses packet loss as signals of congestion, which is matched with our link prices λ_s^k. The basic rate adjustment mechanism of TCP Reno is as follows (Low, Paganini, & Doyle, 2002):

$$x_s^k(t+1) = x_s^k(t) + \frac{1-\lambda_s^k}{(D_s^k)^2} - \frac{1}{2}\lambda_s^k\left[x_s^k(t)\right]^2 \tag{7}$$

$$(y_s^k)^* = \sqrt{\frac{(\omega_1 + \omega_2 \times v) \times \sum_{i=1}^{N}\phi_i^{sk}d_i}{\lambda_s^k}} = x_s^k(t) + \frac{1-\lambda_s^k}{(D_s^k)^2} - \frac{1}{2}\lambda_s^k[x_s^k(t)]^2 \Rightarrow x_s^k(t) = \begin{cases} \frac{1}{\lambda_s^k}\left\{1 + \sqrt{1 - 2\lambda_s^k\left[\sqrt{\frac{(\omega_1+\omega_2\times v)\times\sum_{i=1}^{N}\phi_i^{sk}d_i}{\lambda_s^k}} - \frac{1-\lambda_s^k}{(D_s^k)^2}\right]}\right\} & \frac{1-\lambda_s^k}{(D_s^k)^2} \geq \sqrt{\frac{(\omega_1+\omega_2\times v)\times\sum\limits_{i=1}^{N}\phi_i^{sk}d_i}{\lambda_s^k}} \\ \frac{1}{\lambda_s^k}\left\{1 - \sqrt{1 - 2\lambda_s^k\left[\sqrt{\frac{(\omega_1+\omega_2\times v)\times\sum_{i=1}^{N}\phi_i^{sk}d_i}{\lambda_s^k}} - \frac{1-\lambda_s^k}{(D_s^k)^2}\right]}\right\} & \sqrt{\frac{(\omega_1+\omega_2\times v)\times\sum_{i=1}^{N}\phi_i^{sk}d_i}{\lambda_s^k}} - \frac{1}{2\lambda_s^k} \leq \frac{1-\lambda_s^k}{(D_s^k)^2} \leq \sqrt{\frac{(\omega_1+\omega_2\times v)\times\sum_{i=1}^{N}\phi_i^{sk}d_i}{\lambda_s^k}} \end{cases} \tag{8}$$

$$x_s^k = W_s^k / D_s^k \Rightarrow W_s^k = \begin{cases} \frac{D_s^k}{\lambda_s^k}\left\{1 + \sqrt{1 - 2\lambda_s^k\left[\sqrt{\frac{(\omega_1+\omega_2\times v)\times\sum_{i=1}^{N}\phi_i^{sk}d_i}{\lambda_s^k}} - \frac{1-\lambda_s^k}{(D_s^k)^2}\right]}\right\} & \frac{1-\lambda_s^k}{(D_s^k)^2} \geqslant \sqrt{\frac{(\omega_1+\omega_2\times v)\times\sum_{i=1}^{N}\phi_i^{sk}d_i}{\lambda_s^k}} \\ \frac{D_s^k}{\lambda_s^k}\left\{1 - \sqrt{1 - 2\lambda_s^k\left[\sqrt{\frac{(\omega_1+\omega_2\times v)\times\sum_{i=1}^{N}\phi_i^{sk}d_i}{\lambda_s^k}} - \frac{1-\lambda_s^k}{(D_s^k)^2}\right]}\right\} & \\ & \sqrt{\frac{(\omega_1+\omega_2\times v)\times\sum_{i=1}^{N}\phi_i^{sk}d_i}{\lambda_s^k}} - \frac{1}{2\lambda_s^k} \leqslant \frac{1-\lambda_s^k}{(D_s^k)^2} \leqslant \sqrt{\frac{(\omega_1+\omega_2\times v)\times\sum_{i=1}^{N}\phi_i^{sk}d_i}{\lambda_s^k}} \end{cases} \quad (9)$$

where D_s^k is the round trip time.

In order to acquire the required transmission rate $(y_s^k)^*$, the source rate x_s^k is modified as:So, the window size W_s^k is updated by:The processor allocation problem is solved in the similar method. Let $\partial L(y,Z)/\partial Z = 0$, then

$$\frac{\partial L(y_s^k, Z_k)}{\partial Z_k} = \frac{\omega_1 \times \sum_{i=1}^{N}\varphi_i^k b_i}{(Z_k)^2} - \eta_k = 0 \quad (10)$$

We can get the optimal computational capacity: $Z_k = \sqrt{\omega_1 \times \sum_{i=1}^{N}\varphi_i^k b_i/\eta_k}$.

Then the unique optimal the number of processors β_k^* maximizing the grid user utility is:

$$\beta_k^* = \left\lceil \frac{\sqrt{\omega_1 \times \sum_{i=1}^{N}\varphi_i^k b_i/\eta_k}}{P_k} \right\rceil \quad (11)$$

By above analysis, a joint congestion-control and processor-allocation (JCCPA) algorithm for GoOBS task scheduling is designed as follows:

(1) At the GoOBS core node, the burst loss ratio $\lambda_s^k(t)$ of each link is periodically collected and sent to a cross-layer optimizer through the control channel.
(2) At the grid computational resource node, the load balancing factor η_k of each resource is periodically collected and sent to a cross-layer optimizer through the control channel.
(3) At the cross-layer optimizer, the user utility, taking into account previously assigned jobs and current grid state (received burst loss and load balancing factor), is calculated for each resource. The TCP path and computational resource with the maximal user utility are selected and assigned to the scheduled job. The message is then sent to the corresponding source node and computational resource node through the control channel.
(4) At the grid source node, after receiving the allocated path, TCP sender measures the round trip time D_s^k of this path and updates its TCP window size W_s^k.
(5) At the grid computational resource node, once receiving the message from the cross-layer optimizer, it updates its number of processors.
(6) Repeat the above steps for each job until the current time or expense is beyond the deadline and budget limits.

3. Simulation results and performance analysis

Simulations are conducted on a centralized GoOBS platform to compare the performance of JCCPA algorithm and DBC algorithm under budget and deadline constraint. A description of the core

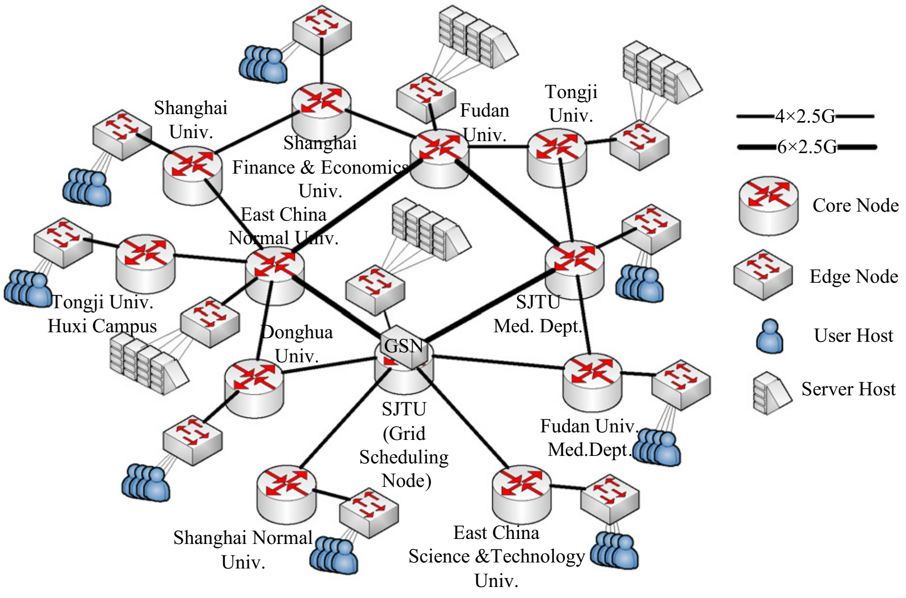

Fig. 1. Simulation Topology.

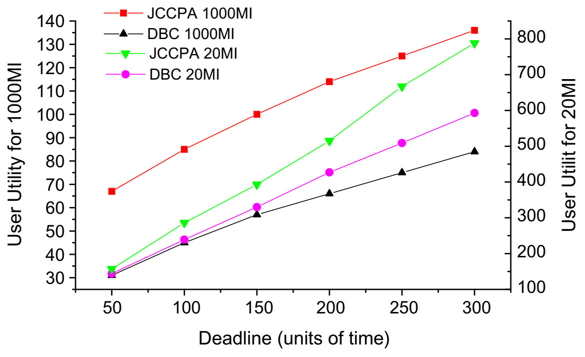

Fig. 2. User utility is a function of the deadline.

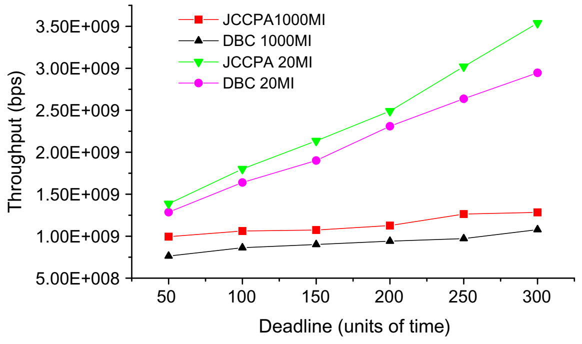

Fig. 3. The throughput is a function of the deadline.

of DBC algorithm follows: (1) For each resource, calculate the next completion time for an assigned job on the basis of taking into account previously assigned jobs. (2) Sort resources by next completion time. (3) Assign job to the first resource for which the cost is less than or equal to the budget limit. (4) Repeat all steps until all jobs are assigned. The OBS based grid topology is shown in Fig. 1, which is from Shanghai Education and Research Network (SHERNET). The details can be referred to Yang, Wu, Dai, and Chen (2010). The characteristics of computational resources simulated can be referred to Buyya et al. (2002).

Fig. 2 shows the user utility as a function of the deadline. Two average required computational quantities for each application, namely 20 MI (million instructions) and 1000 MI are adopted, where 20 MI represent the case that computational time at server is much shorter than data transmission delay while 1000 MI represent the case that computational time at server is much larger than data transmission delay. As shown in Fig. 2, all user utilities increase with the deadline since a larger deadline brings out more successfully completed jobs. Furthermore, regardless of DBC or JCCPA, the user utility for the average required computational quantity of 20 MI is higher than that for the average required computational quantity of 1000 MI. The reason is that a larger computational quantity results in the completion time quick to exceed deadline constraint so that many job requests are refused. Under these two required computational quantity, JCCPA is superior to DBC, respectively. This is easily understood because JCCPA has shorter average overall delay (see Fig. 4) that makes the job processed and replied fast so that more job requests can be served.

Under the mentioned required computational quantities, JCCPA achieves higher throughput than DBC illustrated in Fig. 3. This is obvious because JCCPA completes more jobs than DBC. At the same time, no matter what JCCPA algorithm or DBC algorithm is, the case in average required computational quantity of 20 MI achieves higher throughput than the case in average required computational quantity of 1000 MI. This is due to the fact that the former provides higher user utility than the latter (see Fig. 2).

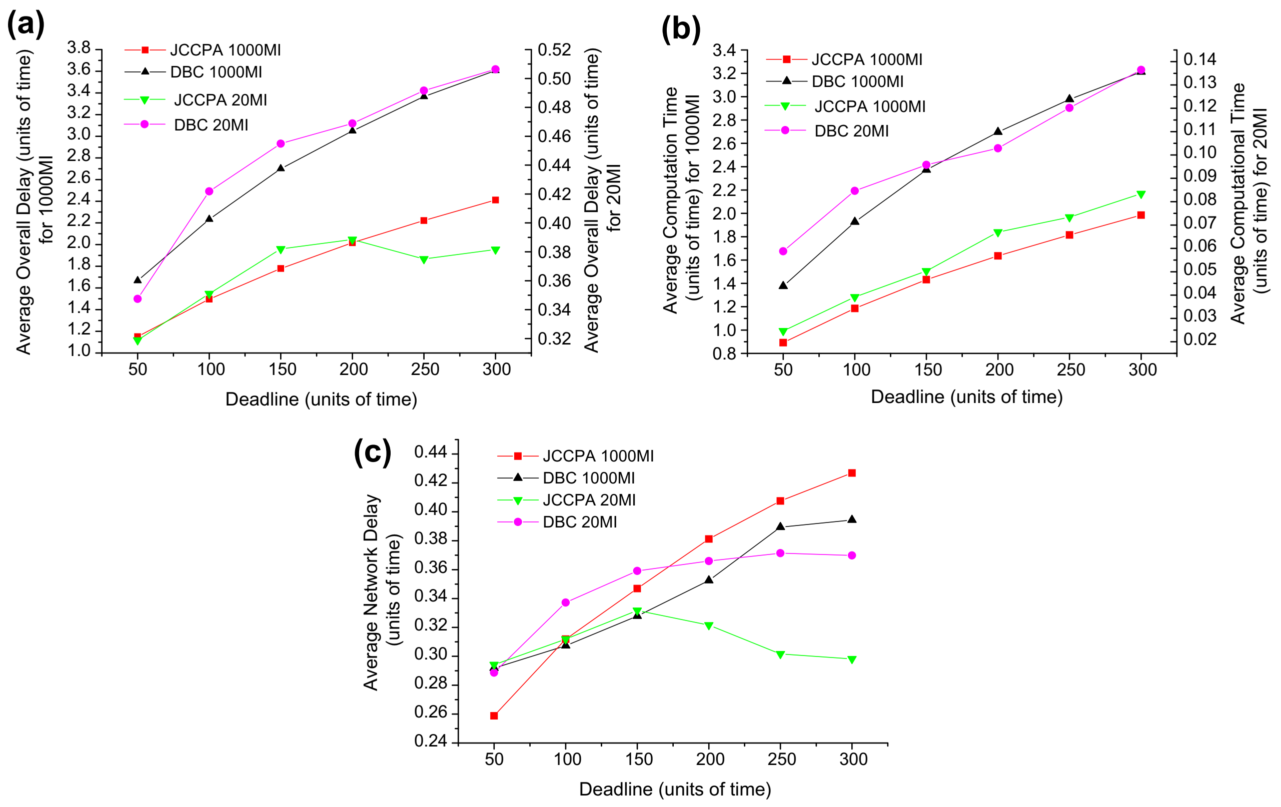

Fig. 4. Effect of deadline on the performance of: (a) average overall delay, (b) average processing time at remote servers and (c) average network delay.

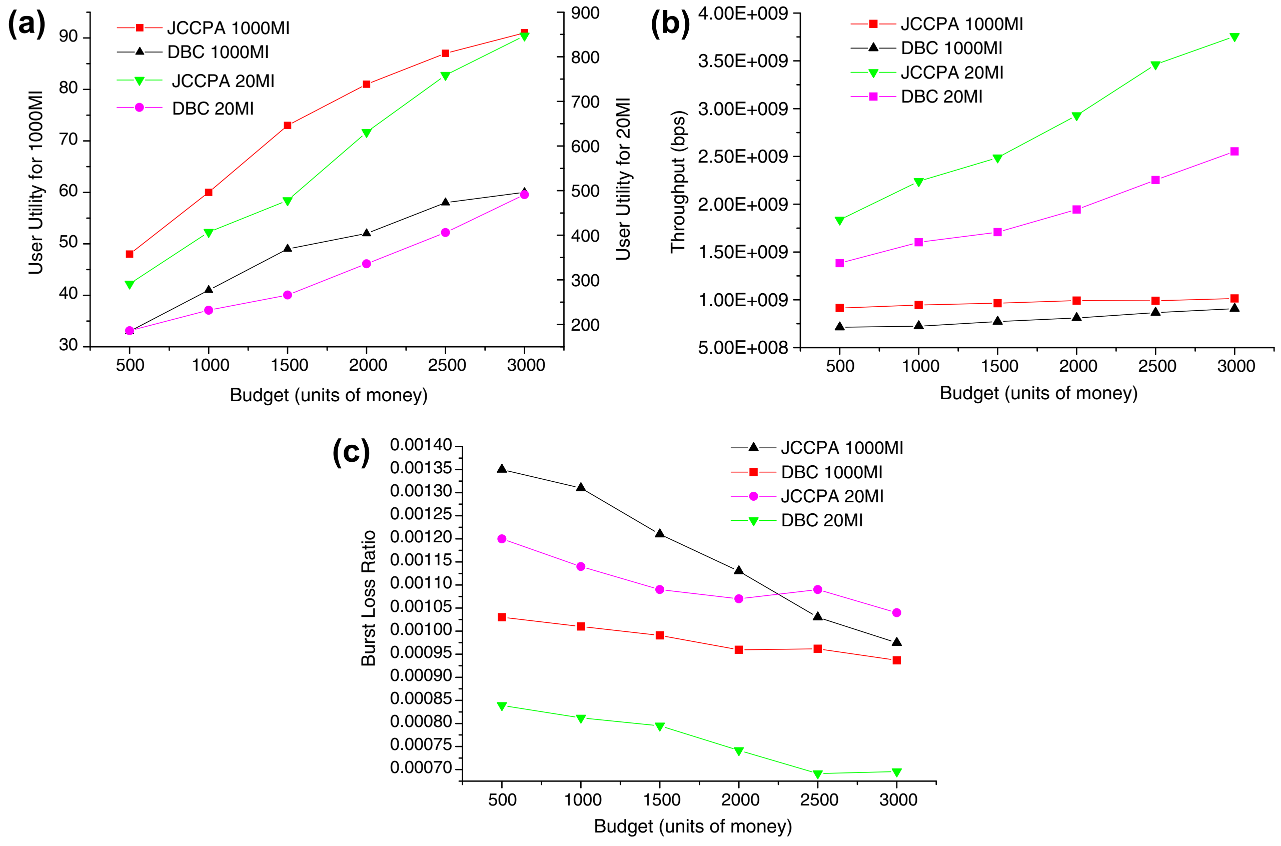

Fig. 5. Dependence of (a) user utility, (b) throughput and (c) burst loss ratio on the budget.

Average overall delay is the average time needed to complete a job from start time sending the request to the end time receiving the processing results. It consists of average computational time and average network delay, which is used to evaluate the performance of GoOBS to a certain extent. Fig. 4 examines the effect of deadline on the average overall delay, average processing time at remote servers and average network delay. From the results in Fig. 4(a), the average overall delay of JCCPA is shorter than that of DBC since JCCPA chooses some suitable grid resources to transmit and process jobs at server. In the case that average required computational quantity is 1000 MI, the average network delay of JCCPA is higher than that of DBC for large deadline as shown in Fig. 4(c). This is reasonable because for the average required computational quantity of 1000 MI, the bottleneck of GoOBS lies in computational resource in this case and then the effect of congestion control mechanism is less determinant. The average overall delay in the case of 1000 MI is longer than that in the case of 20 MI. This reason is that a larger computational quantity has more obvious effect on delay (see Fig. 4(b) and (c)).

The dependence of performance on budget is illustrated in Fig. 5. When the budget is small, the user utility and throughput are low. This is because user cannot afford expensive and efficient resources to process jobs. As the budget increases, both user utility and throughput grow since users can buy more expensive resources to maximize user utility. JCCPA achieves higher throughput than DBC because of more successfully completed jobs (see Fig. 5(a)). In Fig. 5(c), all burst loss ratios decrease with the budget. This can be explained as follows. Since the required grid resources are not released in time, it makes a great number of jobs to be refused in submission so that lots of burst losses mainly occur at the beginning. Therefore, the traffic load that can be successfully sent to OBS networks is not large and to some extent network status is in balance. As the budget increases, the jobs that can be successfully completed also increase and then the burst loss ratio comparatively reduces.

In Fig. 6, we set the average required computational quantity to be 100 MI. The budget is constrained to be 1000 units of money. Average computational payment is the average charge taken to execute a job at server. When the job load (namely the number of job requests) is small, the user utility is high. This is easily understood. There are sufficient resources to share in GoOBS environment so that the scheduler can select some free or light-load resources to transmit and process user jobs resulting in low burst loss ratio and average computational payment. As job load increases, the available resources become less and conflicts rapidly increase, which induce the increasing of average computational payment and burst loss ratio. Before the budget is reached, a large number of job requests are refused and then the user utilities decrease. JCCPA has superior performance than DBC since it properly allocates resources in terms of current GoOBS conditions. It is noted that payment is mainly expended on computational resource. This is due to the fact that computational price per unit time is much more than network bandwidth price per unit time.

Fig. 7 compares the user utility among JCCPA, processor allocation local optimization scheduling (PLOS) and congestion control local optimization scheduling (CLOS). The user utility increases with the increasing of deadline because longer time contributes to the successful completion of more jobs. For the average required computational quantity of 1000 MI, when the deadline is smaller (for example less than 150), the user utility of PLOS is slightly better than that of JCCPA. As the deadline increases, the latter is higher than the former. The reason is that as far as the large computational quantity is concerned, the bottleneck of GoOBS consists in computational resource and the impact of congestion control is

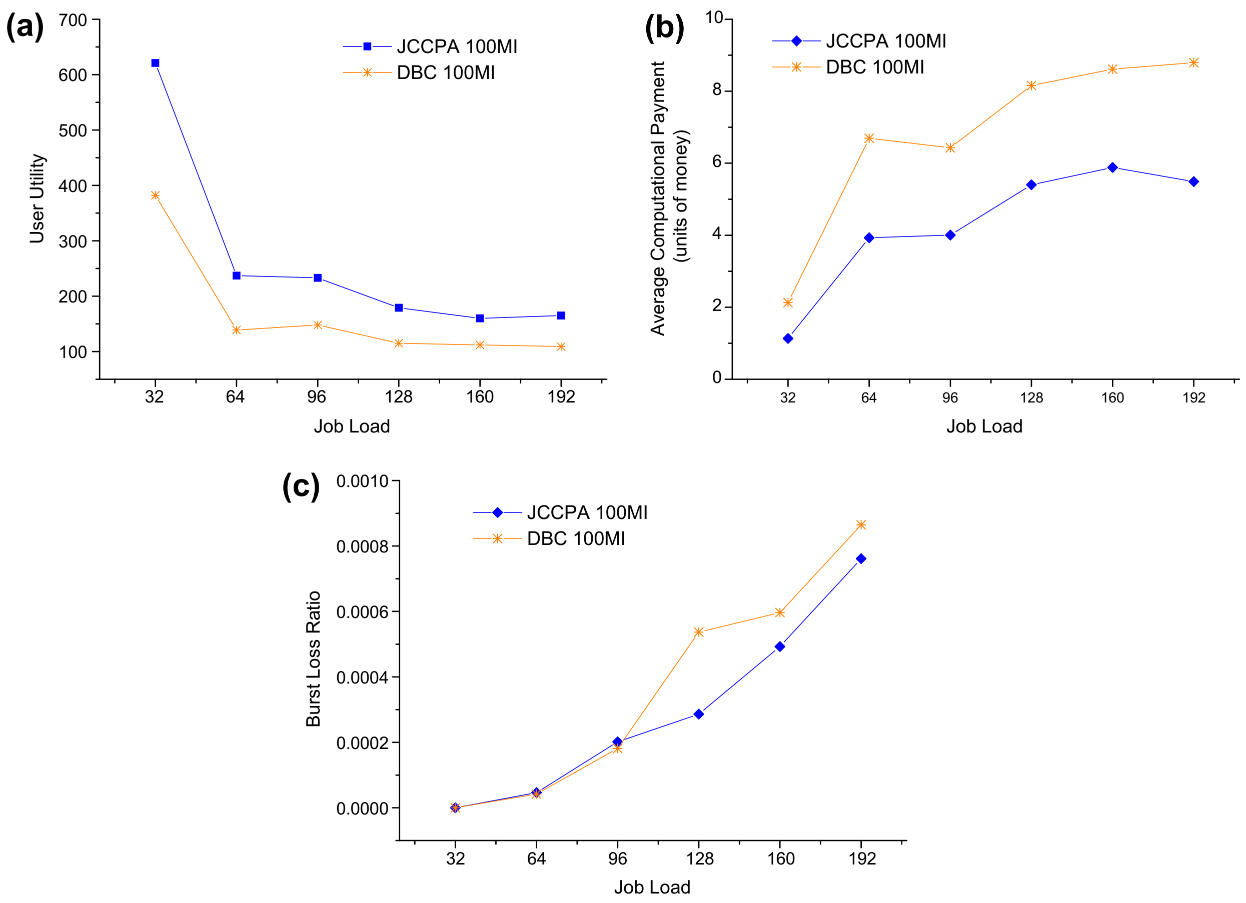

Fig. 6. Dependence of (a) user utility, (b) burst loss ratio and (c) average overall delay on the job load.

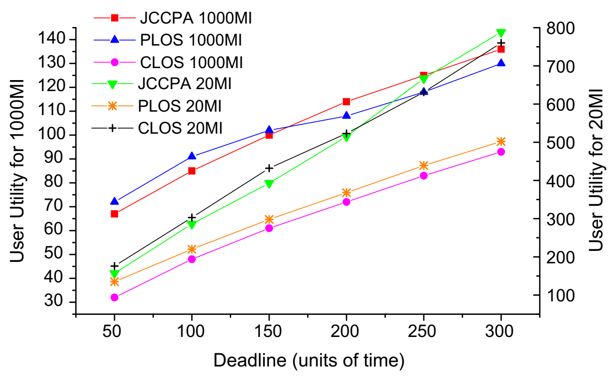

Fig. 7. Effect of deadline on the performance of user utility.

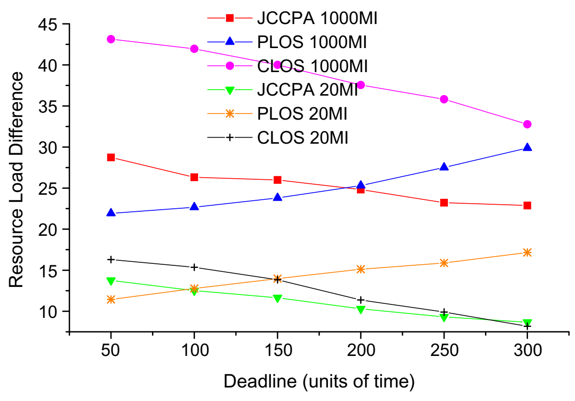

Fig. 8. Effect of deadline on the performance of resource load difference.

not obvious. PLOS successfully completes more jobs by adopting processor allocation policy. However, as the deadline increases, more jobs are completed and the load is also increased so that the effect of congestion control becomes important. JCCPA chooses some servers with unblocked links that are beneficial to more job requests served. CLOS only considers congestion control, thus it provides the least user utility in the case of large computational quantity. For the average required computational quantity of 20 MI, JCCPA and CLOS provide higher user utility than PLOS since the bottleneck of GoOBS is determined by network resource.

Fig. 8 investigates the load balancing of GoOBS system. The balanced load assignment across heterogeneous computing and network infrastructure is critical for both grid resources availability and user/application efficiency since the heavy load of some resource is easy to become the bottleneck of entire gird system and affect the timely completion of all tasks (Simeonidou et al., 2005). We use resource load difference (RLD) to measure the load balancing degree of GoOBS system. In GoOBS environment, the load balancing should take into account network RLD and other grid resource (namely computational resource in this paper) RLD. For the computational resource, we use “workload/processing capacity” as resource load. Actually, the resource load is the time spent by the computational resource to execute all jobs assigned to it. The computational RLD is the sum of the difference between the time spent on each computational resource and the average occupation time. For network resource, the burst loss ratio of TCP path is seen as network load since it reflects the status of traffic load in some sense. The network RLD is the sum of the difference between the burst loss ratio of each OBS link and the average burst loss ratio. RLD is the sum of computational RLD and network RLD, i.e., $\mathrm{RLD} = \sum_{j=1}^{M}|T_j - T_{avg}| + \sum_{l=1}^{G}|p_l - p_{avg}|$. The smaller the RLD value is, the better the system load balancing is. From the results in Fig. 8, the load balancing in the case of average computational quantity 20 MI is better than that in the case of average computational quantity 1000 MI. The reason is that RLD is the sum of network RLD and computational RLD, where network RLD is reflected by burst loss ratio while computational RLD is reflected by computational time and burst loss ratio is often less than computational time. For average computational quantity of 1000 MI, when the deadline is small, PLOS has better load balancing than JCCPA. When the deadline is large, the latter is superior to the former. This is due to the fact that the bottleneck of GoOBS lies in computational resource. PLOS can select some idle or light-loaded computational resource to process job so that the computational RLD is better and it plays a decisive role in RLD. When the deadline increases, the load increases and network congestion control has important effect. JCCPA can properly allocate resource by fulfilling the coordination of congestion control in conjunction with processor allocation. CLOS that only considers congestion control achieves much worse load balancing than JCCPA and PLOS. Because when the computational quantity is large, the network congestion control is less important, furthermore, network RLD is less than computational RLD. For average computational quantity of 20MI, JCCPA and CLOS have better load balancing than PLOS. The reason is that the bottleneck of GoOBS is determined by network resource.

4. Conclusion

We present a joint congestion control and processor allocation algorithm for task scheduling in grid over OBS networks. Parameters from resource layer are abstracted and provided to a cross-layer optimizer to maximize user's utility function. Simulations are carried out to evaluate the performance of the proposed algorithm by comparing with the DBC, PLOS and CLOS algorithm, respectively. Results show that the proposed algorithm can dynamically adjust the resource information according to the feedback of GoOBS environment in order to improve the entire performance, which is suitable for the dynamic, autonomous and heterogeneous GoOBS. In the future, we will consider joining the characteristic parameters of OBS networks, such as assembly delay and data channels to study the cross-layer performance.

Acknowledgments

The paper is partially supported by National Science Foundation of China (NSFC) (ID90704002 and 60877012), 863 Project (ID2006AA01Z242 and 2007AA01Z275), Dawn Program for Excellent Scholars by the Shanghai Municipal Education Commission, and the Key Disciplinary Development Program of Shanghai (T0102).

References

Baker, M., Buyya, R., & Laforenza, D. (2002). Grids and grid technologies for wide-area distributed computing. *Software–Practice and Experience, 32*(15), 1437–1466.

Brakmo, L. S., & Peterson, L. L. (1995). TCP Vegas: End to end congestion avoidance on a global Internet. *IEEE Journal on Selected Areas in Communications, 13*(8), 1465–1480.

Buyya, R., Murshed, M., & Abramson, D. (2002). A deadline and budget constrained cost-time optimization algorithm for scheduling task farming applications on global grids. Computing Research Repository cs. DC/023020.

Chen, C. K., Kuo, H. H., Yan, J., & Liao, T. (2009). GA-based PID active queue management control design for a class of TCP communication networks. *Expert Systems with Applications, 36*(2), 1903–1913.

Dussa, K., Carlson, B., Dowdy, L., & Park, K. H. (1990). Dynamic partitioning in a transputer environment. In *Proceedings of ACM SIGMETRICS on measurement and modeling of computer systems* (pp. 203–213). Boulder, Colorado: Academic.

Feng, H., Song, G., Zheng, Y., & Xia, J. (2004). A deadline and budget constrained cost-time optimization algorithm for scheduling dependent tasks in grid computing. *Grid Coop. Comput, 3033*, 113–120.

Floyd, S., & Jacobson, V. (1993). Random early detection gateways for congestion avoidance. *IEEE/ACM Transaction on Networking, 1*(4), 397–413.

Foster, I., Kesselman, C., & Tuecke, S. (2001). The anatomy of the grid: Enabling scalable virtual organization. *International Journal of Supercomputer, 15*(3), 200–222.

Ghosal, D., Serazzi, G., & Tripathi, S. K. (1991). The processor working set and its use in scheduling multiprocessor systems. *IEEE Transactions on Software Engineering, 17*(5), 443–453.

Lee, J. K., Lee, J. H., & Sohn, S. Y. (2009). Designing a business model for the content service of portable multimedia players. *Expert Systems with Applications, 36*(3), 6735–6739.

Low, S. H., Paganini, F., Wang, J., Adlakha, S. A., & Doyle, J. C. (2002). Dynamics of TCP/RED and a scalable control. In *Proceedings of IEEE INFOCOM* (pp. 239–248).

Low, S. H., Paganini, F., & Doyle, J. C. (2002). Internet congestion control. *IEEE Control Systems Magazine, 22*(1), 28–43.

Majumdar, S., Eager, D. L., & Bunt, R. B. (1991). Characterization of programs for scheduling in multiprogrammed parallel systems. *Performance Evaluation, 13*(2), 109–130.

Mambretti, J., Weinberger, J., Chen, J., Bacon, E., Yeh, F., Lillethun, D., et al. (2003). The photonic terastream: Enabling next generation applications through intelligent optical networking at iGrid 2002. *Journal of Future Generation Computer Systems, 19*(6), 897–908.

McCann, C., & Zahorjan, J. (1994). Processor allocation policies for message-passing parallel computers. *ACM SIGMETRICS Performance Evaluation Review, 22*, 19–32.

Ohsaki, H., Murata, M., Suzuki, H., Ikeda, C., & Miyahara, H. (1995). Rate-based congestion control for ATM networks. *ACM SIGCOMM Computer Communication Review, 25*(2), 60–72.

Qiao, C., & Yoo, M. (1999). Optical burst switching (OBS) – A new paradigm for an optical internet. *Journal of High Speed Networks, 8*(1), 69–84.

Shakkottai, S., Rappaport, T. S., & Karlsson, P. C. (2003). Cross-layer design for wireless networks. *IEEE Communications Magazine, 41*(10), 74–80.

Simeonidou, D., Nejabati, R., Ciulli, N., Battestilli, L., Carrozzo, G., & Castoldi, P. (2005). Grid optical burst switched networks (GOBS). Global Grid Forum, GHPN Group, Information track draft.

Tseng, L. Y., Chin, Y. H., & Wang, S. C. (2009). A minimized makespan scheduler with multiple factors for grid computing systems. *Expert Systems with Applications, 36*(8), 11118–11130.

Wang, S. Y. (2003). Using TCP congestion control to improve the performances of optical burst switched networks. In *Proceedings of IEEE ICC* (pp. 1438–1442). Anchorage, Alaska.

Yang, Y., Wu, G., Li, X., & Chen, J. (2009). Joint flow control and processor allocation for task scheduling in grid over OBS networks. In *Proceedings of 15th Asia-Pacific conference on communications* (pp. 511–514). Shanghai, China.

Yang, Y., Wu, G., Dai, W., & Chen, J. (2010). Multi-objective optimization based ant colony optimization in grid over optical burst switching networks. *Expert Systems with Applications, 37*(2), 1769–1775.

Yu, X., Qiao, C., & Liu, Y. (2004). TCP implementation and false time out detection in OBS networks. In *Proceedings of IEEE INFOCOM* (pp. 774–784). Hong Kong.

Yu, K. M., & Zhou, J. (2010). Parallel TID-based frequent pattern mining algorithm on a PC Cluster and grid computing system. *Expert Systems with Applications, 37*(3), 2486–2494.