2009

# Joint Computing and Network Resource Scheduling in a Lambda Grid Network

Vaidhehi Lakshmiraman
*University of Nebraska - Lincoln*, vlakshmi@cse.unl.edu

Byrav Ramamurthy
*University of Nebraska - Lincoln*, bramamurthy2@unl.edu

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2009 proceedings

# Joint Computing and Network Resource Scheduling in a Lambda Grid Network

Vaidhehi Lakshmiraman and Byrav Ramamurthy
Department of Computer Science and Engineering
University of Nebraska-Lincoln
Lincoln NE 68588-0115 U.S.A.
Email: {vlakshmi, byrav}@cse.unl.edu

*Abstract*— **Data-intensive Grid applications require huge data transfers between grid computing nodes. These computing nodes, where computing jobs are executed, are usually geographically separated. A grid network that employs optical wavelength division multiplexing (WDM) technology and optical switches to interconnect computing resources with dynamically provisioned multigigabit rate bandwidth lightpath is called a Lambda Grid network. A computing task may be executed on any one of several computing nodes which possesses the necessary resources. In order to reflect the reality in job scheduling, allocation of network resources for data transfer should be taken into consideration. However, few scheduling methods consider the communication contention on Lambda Grids. In this paper, we investigate the joint scheduling problem while considering both optical network and computing resources in a Lambda Grid network. The objective of our work is to maximize the total number of jobs that can be scheduled in a Lambda Grid network. An adaptive routing algorithm is proposed and implemented for accomplishing the communication tasks for every job submitted in the network. Four heuristics (FIFO, ESTF, LJF, RS) are implemented for job scheduling of the computational tasks. Simulation results prove the feasibility and efficiency of the proposed solution.**

*Keywords: Lambda grid, lightpath, job scheduling, joint scheduling.*

## I. Introduction

Today, the demand for computational, storage and network resources continues to grow. At the same time, a vast amount of these resources remains under-used. To enable the utilization of these vast amounts of resources the tasks can be executed using shared computational and storage resources and which communicate over a network. Imagine a team of researchers performing a job which contains a number of tasks. Each task demands different computational, storage, and network resources. Distributing the tasks across a network according to resource availability is called Distributed Computing. Grid computing is a recent phenomenon in distributed computing. We envision that a future wavelength division multiplexed (WDM) network will provide efficient support for many distributed computing applications that require both execution by multiple geographically separated computing nodes and data transfers across them.

In this work, we consider the problem of efficiently scheduling such jobs over a Lambda Grid network that uses WDM wavelength routing. Thus we jointly schedule both network and computing resources. A job is divided into a number of modules that can be dependent on each other. Each job request uses a set of nodes and a lightpath for a preferred time. When the connection leaves the network, the released resources are utilized for future requests. We assume that the nodes and links in the network are homogeneous and use the first fit wavelength assignment algorithm for allocating lightpath requests. Our study establishes the effective resources utilization for the considered network topology using simulation experiments.

## II. Related Work and Motivation

Scheduling methods in both grid computing and optical networks are often too restrictive to be directly used to achieve optimal grid network scheduling. Previously, many algorithms have been proposed for network scheduling [1], [2]. Most of these assumed an ideal communication system where all the resources are fully connected and communication between any two resources can be used whenever needed. A few recent studies [3] [4], [5] have proposed the framework where the optical network has been considered as a resource in a similar way to a computing node, or storage resource and investigated the joint scheduling model for optical grid applications.

Banerjee *et al.* [4] propose a hybrid approach that combines online and offline scheduling. Based on the transfer time of the file, the network resources are reserved. They proposed various heuristics to schedule the tasks submitted by the user. Task scheduling is defined as determining which job has to be executed first rather than identifying and scheduling resources at a computing node.

Recent works [6], [7] considered the problem of jointly scheduling computing and network resources for a Directed Acyclic Graph (DAG). The authors Wang *et al.* [6] formulated the problem assuming that a (SONET/SDH) connection of subwavelength granularity can be used to satisfy the communication requirements of a pair of tasks and proposed a heuristic approach to minimize the completion time of the job. Liu *et al.* [7] formulated the problem for WDM networks. They proposed a fixed routing algorithm to schedule the network and modified list scheduling heuristics to schedule the task submitted by the user for an application specific network. These authors, Banerjee *et al.* [4], Thysebaert *et al.* [8] proposed an Integer Linear Programming (ILP) model for the joint scheduling problem. Similarly Thysebaert *et al.* [8] proposed Ant Colony Optimization, to find a given job the optimal set of computing, network and store resources.

Recent works by Castillo *et al.* [9] considered advanced reservation of resources for both homogeneous and heterogeneous environments. The authors provided scheduling algorithms using best fit strategy to utilize the maximum amount of resources.

IEEE International Conference on Communications, 2009. ICC '09.
Digital Object Identifier: 10.1109/ICC.2009.5199110
Publication Year: 2009 , Page(s): 1 - 5

The shortfalls in the available literature motivates us to attempt to maximize the number of jobs that can be scheduled for a given network topology and traffic. In this work, we consider a problem where 1) lightpaths are required for communicating nodes; 2) multiple jobs may arrive one after another which have to be scheduled on the computing nodes. The objective is to maximize the number of jobs that can be scheduled. We devise an efficient algorithm to schedule the computing and network resources for a general DAG.

## III. Methods and Algorithms

An algorithm is proposed to schedule both the network bandwidth and computing resources jointly in a grid and optical network, in advance, in order to optimize the network performance and to maximize the total number of jobs that can be scheduled. Thus communication contention is incorporated into job scheduling in a Lambda Grid network. We propose an algorithm to implement 1) Network Scheduling, to schedule the bandwidth in a network along the lightpaths and 2) Job Scheduling, to schedule the jobs to various computing resources in the network.

Jobs are submitted to the computing node $n \in N$ in the network. There can be '$j$' ($j \in J$) jobs and each job can be divided into '$t$' ($t \in T$) tasks dependent or independent of each other and hence these tasks may be executed in parallel. And the tasks at next levels are dependent on each other and hence are executed sequentially. We assume that every computing node can execute only one task at a time.

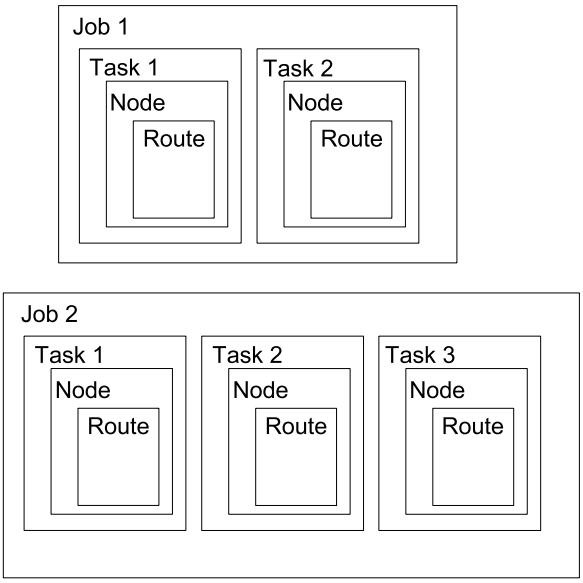



Fig. 1. Lambda Grid network model.

Figure 1 shows the order in which our joint scheduling problem is solved as explained below. Our algorithm acts on the user's input, and as a result the schedule for all the jobs, the task schedules (such as the node on which every task is executed, the computing start time of every task), the routes along which the data transfers corresponding to a task are scheduled, the transfer start time, the duration for data transfer, and the total network time are obtained.

The user provides the topology, jobs that need to be executed and the network status as input to the scheduler. When the user's input is implemented with the proposed algorithm, the scheduler gives the actual schedule time of the jobs and their tasks submitted, the time taken at every node, route and associated wavelengths for each job.

### A. Job Scheduling Heuristics

Job Scheduling allows for various priority schemes to select which job should be executed first. The chosen job is scheduled at the preferred starting time or at the next available time slot. If a job cannot be fully scheduled within the given time slot, the job is blocked. The performance of the Lambda grid network is compared based on various job selection techniques. The different job selection heuristics are described below.

- First In First Out (FIFO) - The jobs and their respective tasks are chosen to be scheduled in the order in which they are submitted by the end user when the scheduler starts.
- Earliest Start Time First (ESTF) - The jobs that have earlier starting time are chosen first. It is expected to decrease the job blocking rate due to unavailable resources or elapsed time.
- Largest Job First (LJF) - The jobs are sorted based on the size of the job. The largest job is chosen first. It is expected to reduce the resource blocking due to unavailable resources at a node.
- Random Selection (RS) - The jobs are chosen in a random order. It is expected to reduce the request blocking rate when the resource load is high due to its load balancing properties.

### B. Task Scheduling

The tasks are executed in the order in which they are dependent on each other. Let task $t_i$ be parent of task $t_j$, Parent($t_j$) = $t_i$ $\forall i, j \in T$. Thus, $t_j$ is dependent on $t_i$ and the chosen destination node of $t_i$ is the source node for $t_j$.

### C. Node Scheduling

When a task $t$ is selected, from the set of all nodes N, if $n \in N$ is available from the starting time of task $t$ for its execution time (CPU duration), the node is added to the node list, NLIST. For all the nodes $n \in NLIST$ the shortest path is found between the source node and $n$. The node which has the least route cost from the source node is chosen to execute the task $t$.

### D. Route Scheduling

*1) Fixed Routing Algorithm:* Fixed routing approach always chooses the same fixed route from a source to the chosen destination node. It uses Dijkstra's algorithm to find the shortest path. All the routes between two nodes are computed offline. This approach to routing is simple. The disadvantage of this approach is that if resources such as routes or wavelengths are not available, then the task and the associated job are blocked. Thus it leads to high job blocking probabilities and high usage of wavelengths.

*2) Adaptive Routing Algorithm:* The proposed adaptive routing algorithm is implemented to overcome the disadvantages of fixed routing algorithm. The lightpaths between the source node and destination nodes $n_i \in NLIST$ are discovered dynamically. It uses a combination of the layered graph model and uses Dijkstra's algorithm to find the shortest path.

Let our network topology be represented as $Y(N, L, W)$, where $N$ be the nodes, $L$ be the links between the nodes, and $W$ be the wavelengths in the links.

The layered graph model is a graph, $G(V, E)$, obtained from a given network topology $Y$ as follows [10]. Each node $n \in N$ in $Y$, is replicated $W$ times in $G$ forming $W$ layers in the graph. These nodes are denoted as $v_i^w \in V, w \in W$ . If link $l \in L$ connects node $n_i$ to $n_j$, $n_i, n_j \in N$, then $v_i^w, v_j^w$ is connected by edge $e_{ij}^w \in E$ in a layered graph. We have two nodes, source and destination nodes, connected to every layer in the graph. The source node is the node where the execution request for task $t$ is generated and destination node is the node where the task $t$ is to be executed. The number of nodes in $G$ is $|N| \times |W|$ and number of links in $G$ is $|L| \times |W|$. In the layered graph model a lightpath request to execute a task is routed based on the available links as well as on the available wavelengths in these links. It is guaranteed that, whenever a route is found, it meets the wavelength continuity constraint. Thus, in the layered graph model, routing and wavelength assignment steps are tightly integrated.

For every destination node $n_i \in NLIST$, a layered graph is created and Dijkstra's algorithm is used to find the shortest path between the source and destination nodes. Even if one of the links in the shortest path is not available in the given layer $w$, the next layer is checked. If the route is not available on any of the layers, we find the next available shortest path. If the route is not available for any the destination nodes, then we increase the start time by 1 unit and the layered graph heuristic is implemented again. If a possible route is not found within time slot $S$, the job $j$ is blocked and the resources scheduled for other tasks in the blocked job $j$ are released.

A note about the heuristics: when the adaptive routing algorithm is used to route the job chosen using the FIFO job scheduling heuristic, it is called as GreedyA. Similarly, ESTF heuristic combined with the fixed routing algorithm is called ESTFF.

### *E. Simulation Environment*

In our simulation, we study the effectiveness of the proposed job selection heuristics and the adaptive routing algorithm for the NSFNet topology. There are totally 24 computing nodes and 43 bidirectional optical links connecting all the computing nodes. The bandwidth of each optical link connecting the nodes is 40Gbps. Two wavelengths are available on each link and in each direction. The scheduler can schedule the network and computing resources upto 72 hours in advance from the start time of the scheduler.

The number of jobs simulated are varied from 10 to 200. The number of tasks per job is varied from 4 to 6. Three different types of job requests are used for the simulation. The job size was determined based on the number of jobs that were submitted to the scheduler. Group of jobs with average task size of four, five and combination of four, five and six are used. We investigate the effectiveness of the four proposed job selection heuristics (ESTF, LJF, FIFO, and RS) with both adaptive and fixed routing algorithms.

We compare our proposed adaptive routing algorithm with the simple fixed routing algorithm. The performance of the proposed algorithm is evaluated based on metrics such as Job blocking rate, Fairness and Effectiveness.

- Job blocking rate - It is the percentage of jobs blocked divided by the total number of jobs submitted.
- Fairness - It is the metric that shows performance of the heuristics for smaller and larger jobs.
- Effectiveness - It is calculated as the percentage of latest finish time of the job scheduled and the blocking rate to the maximum time slot $S$. The higher the percentage, the more effective the heuristics.

$Effectiveness = [1 - (JobBlockingRate \times LatestFinishTimeOfTheJobsSubmitted)/(100\% \times MaximumTimeSlot)] \times 100$

## IV. Results and Discussions

We implemented the above proposed adaptive routing algorithm along with the proposed heuristics on a WDM network (NSFNET topology). Each edge node is connected to one computing node. The performance of the adaptive algorithm and the heuristics are evaluated below using the metrics such as job blocking rate, effectiveness, fairness.

### *A. Job Blocking Rate*

Figure 2 shows the blocking rate of the proposed heuristics with adaptive and fixed routing algorithms. We can see that the FIFO heuristic with adaptive routing (GreedyA) algorithm followed by RS heuristic with adaptive routing algorithm (RSA) worked better than other algorithms for all task sizes that we simulated.

From Figure 2, we see that the heuristics simulated using the fixed routing algorithm have higher job blocking rates. Upon comparing the heuristics using fixed algorithm, FIFO heuristics using fixed algorithm (GreedyF) are found to perform better than other heuristics using fixed algorithm (RSF, LJFF, ESTFF). ESTF heuristic using fixed routing algorithm (ESTFF) always has high blocking rate compared to other heuristics. Since RSF randomly selects a job to execute, the blocking rate also varies and oscillates. For smaller job size, Largest Job First using fixed routing algorithm (LJFF) performs better than Random Selection using fixed routing algorithm (RSF).

Comparing the fixed and adaptive routing algorithms, we find that adaptive routing clearly outperforms the fixed routing algorithm for all proposed heuristics. RSA performs better than ESTFA since even for small job sizes, the job blocking rate using ESTFA is more than that for RSA. GreedyA has minimal blocking rate compared to other proposed heuristics. It is also evident from Figure 2 that for jobs with average task size of five or more, the blocking rate of almost all heuristics increases dramatically with the increase in job size.

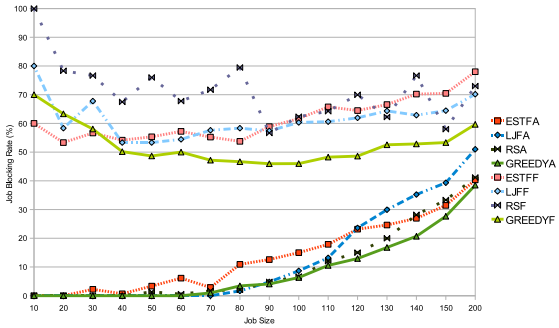



Fig. 2. Average Job Blocking rate.

From the results, it is evident that the FIFO heuristics with Adaptive routing algorithm could be used to reduce the blocking rate. The performance of LJF heuristics with adaptive routing algorithm (LJFA) is found to be effective for smaller job sizes (upto 100 jobs) for a task size of four. As the job size increases, the blocking rate of LJFA increased exponentially. Therefore it is not proposed for larger problems. The performance of ESTF with adaptive routing algorithm (ESTFA) is not consistent for jobs with task size of four. We can see that the adaptive routing algorithm performed better than fixed routing algorithm in all simulations.

GreedyA has an average blocking rate varying from 0% to 38.5%. This heuristic blocked 38.5% of the jobs for a job size of 200. While LJFA blocked upto 51% of jobs for a job size of 200. Thus GreedyA blocks 24.5% lesser than LJFA. When compared to heuristics using the fixed routing algorithm, GreedyA blocks 35.5% less than GreedyF. However GreedyF is better than other heuristics using the fixed routing algorithm.

### B. Fairness

Figure 3 shows the fairness of the proposed heuristics with adaptive and fixed routing algorithms. We can see that the FIFO heuristic with adaptive routing algorithm performed better than the other heuristics. LJF heuristic with adaptive routing algorithm (LJFA) blocked more smaller jobs than larger jobs. We calculated the fairness for a job size of 100.

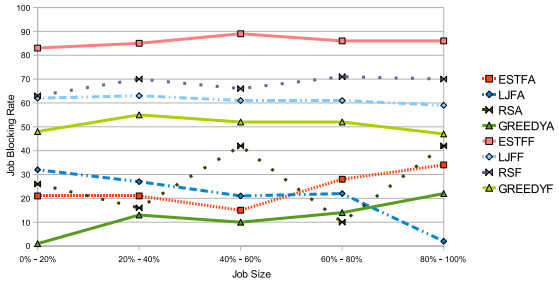
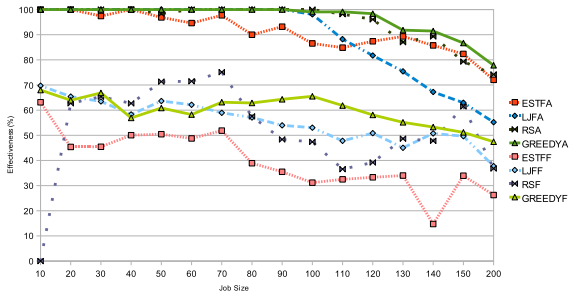



Fig. 3. Fairness for different job sizes.

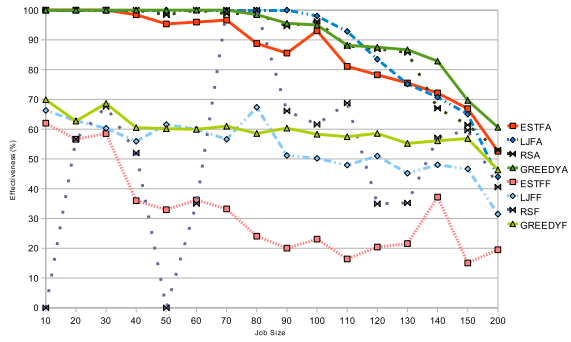
From Figure 3, we note that the heuristics using fixed algorithm have higher blocking rates for jobs of different sizes. The results show that LJFA is more effective for larger job sizes as it has less blocking rate. The simulation results indicate that the adaptive routing algorithm has better effectiveness than the fixed routing algorithm. ESTFA has higher blocking rate for jobs of all sizes. On average for larger jobs, LJFA performed better than GreedyA and for smaller jobs, GreedyA outperformed other heuristics.

### C. Effectiveness of Job Selection Heuristics

Figures 4 and 5 show the effectiveness of the proposed heuristics with adaptive and fixed routing algorithms. We can see that the GreedyA followed by RSA worked better than other algorithms for all task sizes that we simulated.




(a) Effectiveness of job of task size 4.




(b) Effectiveness of job of task size 5.

Fig. 4. Job Effectiveness for jobs of task sizes 4 and 5.

From Figure 4, we note that the heuristics simulated using fixed algorithm have less effectiveness. As explained previously, LJFA is more effective for smaller jobs. The simulation results indicate that the adaptive routing algorithm has better effectiveness than the fixed routing algorithm. Similar results were found for jobs with task size of four (Figure 4(a)) and five (Figure 4(b)). It was also found that for the average effectiveness (average of the effectiveness on jobs with four, five and different task sizes), the adaptive routing algorithm was better than the fixed routing algorithm.

For job size of 4, the effectiveness of GreedyA ranged from 77.8% to 100%. It was more effective (100%) for a job size of upto 90. As the job size increases, the effectiveness reduces by 14% and for larger job size such as 200, the effectiveness reduced upto 32%. As the job size increases, the effectiveness of the heuristics tends to decrease. When compared to other heuristics, GreedyA performs 3.62% better than RSA, 22.61% better than LJFA, and 5.78% better than ESTFA. Also, GreedyA performed 30.39% better than GreedyF.

From Figure 5, we note that for different job sizes, the effectiveness of GreedyA ranged from 60% to 100%. It was more

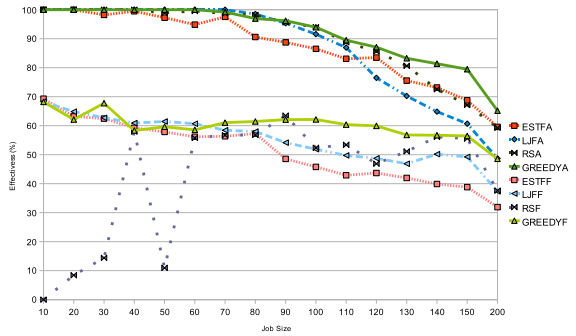



Fig. 5. Average Effectiveness of jobs with different task sizes.

effective for a job size of upto 80. As the job size increases, the effectiveness reduces to 40%. When compared to other heuristics, GreedyA performs 7.68% better than RSA, 16.6% better than LJFA, and 8.05% better than ESTFA. Also, GreedyA performed 14.31% better than GreedyF.

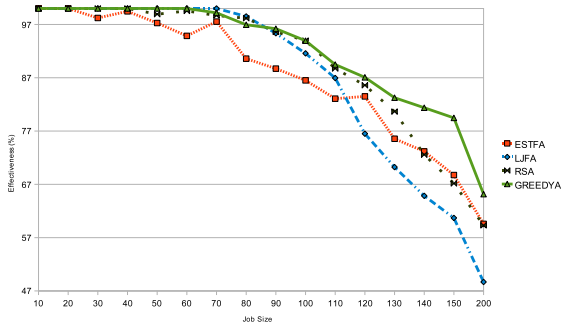



Fig. 6. Average Effectiveness of Jobs using Adaptive Routing Algorithm

The effectiveness of the different heuristics of the adaptive routing algorithm were evaluated (shown in Figure 6). Comparisons indicated that for smaller job sizes the differences in effectiveness across heuristics were insignificant. For job sizes, of the order greater than 80, there was significant variation in the effectiveness.

On average for a job size of 200, GreedyA is 5.85% better than RSA, 5.57% better than ESTFA, 16.48% better than LJFA, and 16.59% better than GreedyF. On average, GreedyA performs better for all task sizes compared to the other heuristics. It is 2.08% better than RSA, 4.71% better than ESTFA, 4.92% better than LJFA, and 32% better than GreedyF.

## V. Conclusions

We have defined a joint scheduling problem in the context of providing efficient support for emerging distributed computing applications in a Lambda Grid network. From the findings, we conclude that the adaptive routing algorithm outperforms the fixed routing algorithm. On average, the proposed adaptive routing algorithm is 33% more effective than the fixed routing algorithm. We observed that, heuristics using fixed routing algorithm have higher blocking rate for all job sizes. But heuristics using adaptive routing algorithm have comparatively lower blocking rates. When we compare the heuristics with adaptive routing algorithm, LJFA is more effective for larger jobs as the larger jobs are scheduled first. GreedyA performs much better than the rest of the heuristics for smaller jobs where the job blocking rate was as small as 1%.

## VI. Future Work

The primary limitation of this study is considering a homogeneous network. With most networks being heterogeneous (with different amounts of bandwidth on different links), the proposed algorithms should be extended for such networks. We can also consider finer granularity for both network and computing resources to minimize the scheduling time and maximizing the number of jobs that can be scheduled. The proposed joint scheduling problem should also be mathematically formulated and solved using Integer Linear Programming (ILP) and other techniques. The other limitation of the study is the consideration of only CPU allocation and the network bandwidth reservation for the Grid resources. There are other resources such as storage, scientific instruments, etc. that should also be included in scheduling the Grid resources. We plan to investigate these in our future work.

## References

[1] T. Yang and A. Gerasoulis, "DSC:Scheduling parallel tasks on an unbounded number of processors," *IEEE Transactions on Parallel Distributed System*, vol. 5, pp. 951–967, September 1994.

[2] I. Ahmad, Y. Kwok, and M. Wu, "Analysis, evaluation, and comparison of algorithms for scheduling task graphs on parallel processors," in *Proceedings of 2nd International Symposium on Parallel Architectures(ISPAN 1996)*, (Beijing, China), pp. 207–213, June 1996.

[3] Q. She, X. Huang, N. Kannasoot, Q. Zhang, and J. Jue, "Multi-resources many cast over optical burst switched networks," in *Proceedings of 16th International Conference on Computer Communications and Networks (ICCCN 2007)*, (Honolulu, Hawaii, USA), August 2007.

[4] A. Banerjee, W. Feng, D. Ghosal, and B. Mukherjee, "Algorithms for integrated routing and scheduling for aggregating data from distributed resources on a lambda grid," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, pp. 24–34, January 2008.

[5] D. Adami, S. Giordano, M. Repeti, M. Coppola, D. Laforenza, and N. Tonellotto, "Design and implementation of a grid network-aware resource broker," in *Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Networks, as part of the 24th IASTED International Multi-Conference on Applied Informatics*, (Innsbruck, Austria), pp. 41–46, February 2006.

[6] Y. Wang, Y. Jin, W. Guo, W. Sun, W. Hu, and M. Wu, "Joint scheduling for optical grid applications," *OSA Journal of Optical Networking*, vol. 6, pp. 304–318, March 2007.

[7] X. Liu, W. Wei, C. Qiao, T. Wang, W. Hu, W. Guo, and M. Wu, "Task scheduling and lightpath establishment in optical grids," in *Proceedings of the INFOCOM 2008 Mini-Conference and held in conjuction with the 27th Conference on Computer Communication (INFOCOM 2008)*, (Phoenix, Arizona), 2008.

[8] P. Thysebaert, B. Volckaert, M. De Leenheer, F. De Turck, B. Dhoedt, and P. Demeester, "Dimensioning and on-line scheduling in lambda grids using divisible load concepts," *Journal of Supercomputing*, vol. 42, pp. 59–82, October 2007.

[9] C. Castillo, G. Rouskas, and K. Harfoush, "On the design of online scheduling algorithms for advance reservations and qos in grids," in *Proceedings of 21st IEEE International Parallel and Distributed Processing Symposium*, (Aveiro, Portugal), pp. 769–774, July 2007.

[10] C. Chen and S. Banerjee, "A new model for optimal routing and wavelength assignment in wavelength division multiplexed optical networks," in *Proceedings of the IEEE INFOCOM 1996*, (San Franciso, CA, USA), pp. 164–171, March 1996.