This paper was presented as part of the High-Speed Networks 2011 (HSN 2011) Workshop at IEEE INFOCOM 2011

Dynamic Scheduling for Workflow Applications over Virtualized Optical Networks

Jingyu Ding, Yan Wang, Jiajin Le and Yaohui Jin

***Abstract*—Joint scheduling of both computation and communication resources for workflow based distributed computing application over optical networks has been studied recently. Most algorithms proposed in previous work are mainly based on static scheduling strategies with assumption that detail resource information and accurate performance prediction is available. In this paper, we propose to employ shared virtualized optical network (VON) for the task scheduling problem. Both customers and carriers can benefit from such architecture with better flexibility and scalability. Based on a new Scheduled Result Graph (SRG) concept, we propose a computation and communication delay aware rescheduling (C^2DAR) scheme to deal with the dynamics from shared VON resources. We evaluate the performance of dynamic scheduling scheme over shared VON in comparison to the static scheduling over dedicated VON and entire optical network respectively. Simulation results also show that C^2DAR scheme outperforms traditional computation delay aware rescheduling (CDAR) scheme under shared VON scenario.**

***Index Terms*—optical network, virtual optical network, workflow, joint scheduling algorithm, rescheduling**

I. Introduction

Distributed computing over optical network has been recently recognized as a new paradigm for data-intensive peta-scale science [1][2][3]. These distributed applications are often modeled as workflows by specifying a set of interdependent tasks. The inter task communication will generate or transfer large datasets frequently. One of the hot research issues is optimal co-scheduling of computing and network resources, which maps tasks to computing facilities while considering lightpath establishment to satisfy the job's requirements with some optimization goals [4][5][6][7]. Most algorithms proposed in the existing works are mainly based on static scheduling strategies with two assumptions: 1) the scheduler can acquire detail resource information or accurate resource performance prediction; 2) the resource manager provides in-advance resource reservation function to ensure each task or communication execution as scheduled in dynamic shared environment. However, these algorithms are difficult to apply in the practical scenarios. The reasons are as follows. First, it is not an easy task to make accurate performance prediction in dynamic run-time environment. Second, carriers are not willing to expose detail network resource information to their customers due to business interests and confidentiality consideration. Moreover, the in-advance resource reservation still lacks industrial standard support.

Network virtualization can be a feasible solution [8][9]. Both customers and carriers benefit from such approach. Carriers can expose virtualized network information to their clients. Customers can make co-scheduling of their own virtualized computing and networking resources with better isolation and scalability.

The VON provisioning can be managed in dedicated or shared manner. When both the computing and VON resources are strictly assigned for an application, the scheduling issue is equivalent to the previous static joint scheduling problem [4]. Although dedicated resource provisioning is easy to implement and has guaranteed service performance, it is inefficient for the resource utilization. Therefore, it is profitable to provide shared VON service in which each VON is assigned a minimal guaranteed bandwidth on the associated links, while residual capacity is shared by all the VONs on those links. However, this sharing mechanism results in communication performance variation because the established lightpath within shared VON may not be allocated bandwidth as scheduled due to the traffic contention from other VONs. Such case is also same to the computation resource management running in shared manner.

In this paper, to deal with the resource dynamics under the shared VON scenario, we propose a low cost computation and communication delay aware rescheduling (C^2DAR) mechanism which generates a near optimal schedule using a static scheduling algorithm for the job before it starts execution and then changes the initial schedule dynamically during the execution when the task or communication is delayed more than a threshold. The key for the C^2DAR mechanism is how to make rescheduling decision. For this, we build on a Scheduled Result Graph (SRG) concept to measure the task or communication delay tolerance for the rescheduling decision.

The remainder of this paper is organized as follows. Section II introduces the overview of the scheduling architecture. Section III describes the scheduling system model and problem statement. In section IV, we present our proposed C^2DAR rescheduling scheme and the Schedule Result Graph (SRG) concept. Simulation results are presented in section V. Finally we conclude the paper in section VI.

Jingyu Ding and Jiajin Le with School of Computer Sci. & Tech. Donghua Univ., China, Yan Wang is with Huawei Technology, Yaohui Jin is with State Key Lab of Advanced Optical Communication System and Network, Shanghai Jiao Tong University, China(e-mail: jinyh@sjtu.edu.cn)

II. SCHEDULING ARCHITECTURE

As shown in Fig. 1, each application has its own associated application-level scheduler which achieves task-resource mapping and job lunching via collaboration with computation and network resource service plane. The computation and network resource service planes achieve resource management and control functions, provide application programming interface (API) to users, and hide the resource technical details. The scheduler consists of three main components: resource information manager, resource allocator and task executor.

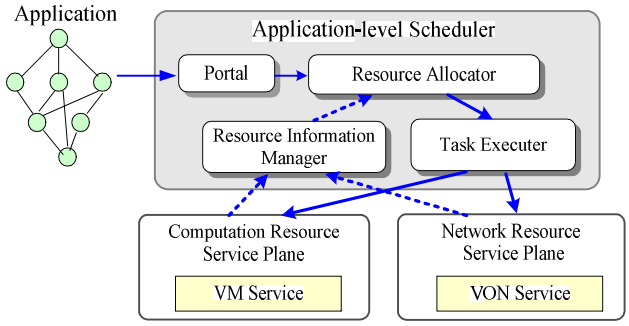


Fig. 1. Scheduling architecture

The functions of the three scheduling components are defined as below:

Resource Information Manager collects and filters available computation and network resource through computation and network resource service plane respectively.

Resource Allocator inquires the resource information from *Resource Information Manager* and estimates the communication and computation cost, it then decides mapping of tasks to computing resources as well as inter-task communication to lightpaths over optical networks, with goal of achieving optimal performance of entire workflow, and submits the schedule to the *Task Executor*.

Task Executor launches ready tasks onto the computing resources and setup lightpaths for the ready communications as scheduled. Optionally, it supports in-advance resource reservation for QoS guarantee. In a dynamic run-time environment, it is also in charge of performance monitoring. If the current execution will delay the entire workflow, it will inform *Resource Information Manager* to update resource information base and trigger the *Resource Allocator* to make rescheduling.

Under the resource virtualization scenario, the computing resource service plane can provides Virtual Machine (VM) service, while the network resource service plane can provide VON service. From the scheduler perspective, the virtualized resources and the real physical resources are identical.

III. SYSTEM MODEL AND PROBLEM STATEMENT

Fig.2 is the overview of task scheduling under VON scenario. A workflow application can be modeled as a directed acyclic graph (DAG). We formulate the task model as $G_t = (\mathbf{V}, \mathbf{E})$, where $\mathbf{V}$ is a set of v tasks and $\mathbf{E}$ is a set of e edges between the tasks. Each edge $e_{mn} \in \mathbf{E}$ represents the precedence constraint such that task v_n can not start execution until v_m finishes. The weight $w(e)$ denotes the data volume transmitted on the edge e. In a given DAG, the set of all direct predecessor of task v is denoted as **pred**(v) and the set of all direct successors of v is denoted as **succ**(v).

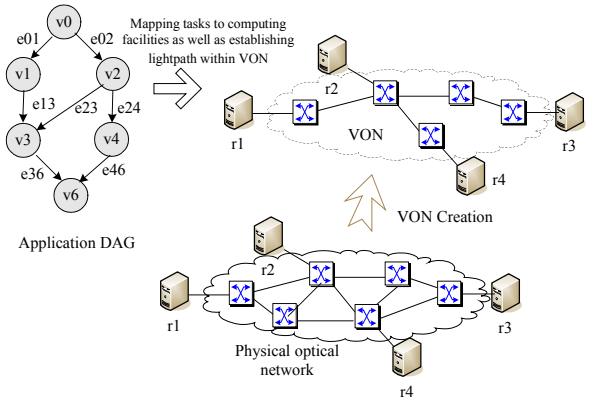


Fig. 2.DAG scheduling under VON scenario

The optical network with attached computational resources can be modeled as a graph $G_{ON} = (\mathbf{N}, \mathbf{L})$, where $\mathbf{N} = \mathbf{R} + \mathbf{S}$ is a set of network nodes, a node $r \in \mathbf{R}$ represents a computing resource and a node $s \in \mathbf{S}$ represents an optical switch node. $\mathbf{L} = \mathbf{L_A} + \mathbf{L_T}$ is a set of links, where a link $l \in \mathbf{L_A}$ represents the access link between a computing resource and an optical edge switch, and a link $l \in \mathbf{L_T}$ represents the transmission link between two optical switches. Each link has a finite bandwidth denoted as B_{ON}^{l}, $l \in \mathbf{L}$.

The ith VON can be represented as $G_{VON}^{i} = (\mathbf{N}^i, \mathbf{L}^i)$ which is a sub-graph of G_{ON} where $\mathbf{N}^i = \mathbf{R}^i + \mathbf{S}^i$, $\mathbf{R}^i \subseteq \mathbf{R}$, $\mathbf{S}^i \subseteq \mathbf{S}$, and $\mathbf{L}^i = \mathbf{L}_\mathbf{A}^i + \mathbf{L}_\mathbf{T}^i$, $\mathbf{L}_\mathbf{T}^i \subseteq \mathbf{L_T}$, $\mathbf{L}_\mathbf{A}^i \subseteq \mathbf{L_A}$. $B_{VON}^{i,j}$ denotes the assigned bandwidth for the ith VON on link $j \in \mathbf{L}'$, and

$$\sum_i B_{VON}^{i,j} \le B_{ON}^{j}, \quad \forall j \in \mathbf{L}.$$

The VON resources can be provisioned and managed in dedicated or shared manner. The dedicated VON assumes that the network resources (wavelength, timeslot, port) are exclusively assigned to a given VON user. The shared VON can then be employed to improve the link resource utilization during task scheduling. Only a minimum guaranteed capacity $B_{VON}^{i,j}$ is assigned to VON i, while residual capacity is shared by all the VONs on that link.

Dynamic scheduling includes just in-time dynamic scheduling and prediction-based rescheduling. Just in-time dynamic scheduling is a task level scheme which only makes scheduling decision at the time of task execution and do not take into account the whole structure of workflow dependencies during the scheduling process. The prediction-based rescheduling is a workflow level scheme

which predicts the performance of task execution on resources and generates a near optimal schedule using a static scheduling algorithm for the job before it starts execution. It changes the initial schedule dynamically during the execution.

In this paper we study prediction-based rescheduling scheme for DAG scheduling over dynamic shared VON scenario. The key problem of rescheduling is how to make rescheduling decision. Previous efforts make rescheduling decision only based on the performance evaluation of task execution or the computing resource dynamics [10][11][12]. When the schedule objective is to minimize the schedule length, the reschedule is triggered if the delay of current task will impact the complete job finish time [11]. We refer to this scheme as computation delay aware rescheduling (CDAR) scheme. The CDAR scheme is implemented over the packet switched networks without QoS guarantees. While in the optical networks, the communication is deterministic in that the lightpath is established before the real data transmission and the bandwidth is guaranteed during communication. So the communication time can be estimated before communication start. However, under the shared VON scenario, only a small portion of bandwidth is guaranteed, so the real established lightpath may not be allocated with bandwidth as scheduled. If the allocated bandwidth is less than the scheduled, the finish time of communication will be delayed. We call this scheme as computation and communication delay aware rescheduling (C^2DAR) scheme.

IV. Computation and Communication Delay Aware Rescheduling

A. Robustness of Static Schedule Result

The key issue to achieve efficient rescheduling is how to define a metric for rescheduling decision. Fig. 3 illustrates a simple scheduling example. A four node DAG is scheduled onto 3 computing resources connected by a VON, as shown in Fig.3 (a). After scheduling all the vertex and edges in the DAG are mapped to corresponding resources with assigned start and finish time. The Gantt chart of the schedule result is given in Fig.3 (b). Based on the schedule result, we build on and extend two fundamental quantities to measure the robustness of scheduled result: the *spare time*, and the *slack* of a scheduled task or a scheduled communication.

The *spare time*, defined as the time interval between dependent scheduled task nodes and/or communication edges, shows the maximal time that the source of dependence can execute without affecting the start time of the sink of the dependence. As shown in Fig.3 (b), the *spare time* between v0 and v1 is 0, e01 and e13 is 10, v2 and e23 is 15.

The *slack* of a scheduled task node or a scheduled communication edge is defined as the maximum delay that can be tolerated in the execution time of the task or communication without affecting the overall schedule length. If the *slack* value is zero, the node is called critical; any delay on the execution time of this node will affect the complete finish time of the application. As shown in Fig. 3 (b), the *slack* of v1 is 0, e01 is 10, and v2 is 15.

Clearly, if the execution of a task or communication will start at a time which is delayed more than its estimated *slack*, the overall schedule length (assuming the execution time of all other tasks and communications that follow remains the same) will change. In this paper we use *slack* value as the metric to measure the delay tolerance of the schedule for rescheduling decision.

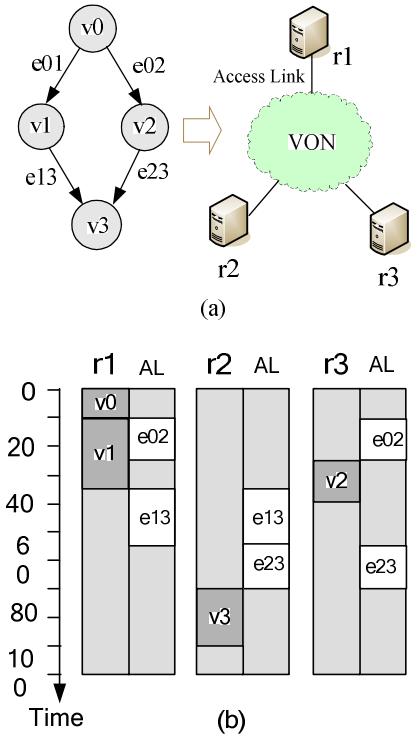


Fig. 3. A simple scheduling example. (a) DAG and resource; (b) Gantt chart of schedule result

B. Schedule Result Graph

From the schedule result (as shown in Fig. 3[b]) we can see there are four kinds of *spare time*, i.e., *spare time* between two adjacent task nodes on the same computing resource (e.g. v0 and v1), between adjacent task and edge (e.g., v1 and e13, e23 and v3), between two adjacent edges (e.g., e13 and e23). There are two kinds of *slack*, i.e., the *slack* of a task node and the *slack* of an edge.

To facilitate calculate all the kinds of *spare times* and *slacks*, we build on a schedule result graph concept.

Definition III. 1: **Schedule Result Graph (SRG)**

From the schedule result, we can construct a schedule result graph $G_{SR} = (\mathbf{V}_S, \mathbf{E}_S)$, where $\mathbf{V}_S = \mathbf{V} \cup \mathbf{E}$, node $v' \in \mathbf{V}_S$ is either task node or edge of the DAG. $e'_{mn} \in \mathbf{E}_S$ denotes dependency between adjacent scheduled node v'_m and $v'_n \in \mathbf{V}_S$. Each node $v' \in \mathbf{V}_S$ is associated with its scheduled result. $ST(v')$ and $FT(v')$ denote the start and finish time of node v' respectively. The weight of e'_{mn}, $w(e'_{mn}) = ST(v'_n) - FT(v'_m)$,

represents the spare time between adjacent $v'_m, v'_n \in \mathbf{V}_S$. ■

The construction of SRG consists of two steps:

Step 1: Put all the DAG nodes **V** and edges **E** in different list according to assigned computing resource and access link, then sort them in increasing order in terms of their scheduled start time.

Step 2: Converge **V** and **E** to $\mathbf{V}_S$, and then generate edges between nodes in $\mathbf{V}_S$ according to their sorted order.

It is easy to prove that the schedule result graph is also a DAG. Based on the G_{SR}, the *spare time* between a node $i \in \mathbf{V}_S$ and an immediate successor $j \in \mathbf{V}_S$ is defined as

$$Spare(i, j) = w(e'_{ij})$$

The *slack* of a node $i \in \mathbf{V}_S$ is computed as the minimum *spare time* on any path from this node to the sink node in the graph G_{SR} . This is recursively computed, in an upwards fashion as follows:

$$Slack(i) = \min_{\forall j \in D_i}(Slack(j) + Spare(i, j))$$

where D_i is the set of the tasks that includes the immediate successors of node *i* in G_{SR}. The slack can be determined in $O(|\mathbf{V}|+|\mathbf{E}|)$ using a depth first search (DFS), where |**V**| and |**E**| are the number of task nodes and edges in the DAG.

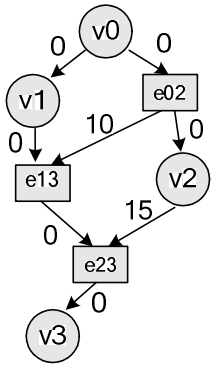


Fig. 4. Schedule Result Graph

Fig. 4 illustrates the schedule result graph (SRG) constructed based on the schedule result in Fig. 3 (b). The schedule result specifies the execution sequence between tasks and edges in the DAG. The circle represents the scheduled task node; the rectangle represents the scheduled communication edge.

C. C^2DAR Rescheduling Algorithm

By using the constructed SRG from the scheduled results, we can compute the slack values of each task node or communication edge in the scheduled DAG.

The C^2DAR algorithm is described in Fig. 5. The input of this scheduler is a DAG modeled application, the VON resource information and an initial static schedule computed by any previous static joint scheduling algorithm (e.g., the ELS algorithm proposed in [4]).

D. Complexity analysis

For the C^2DAR algorithm, the *while*-loop will take (|**V**|+|**E**|-1) times to track each task and communication, where |**V**| and |**E**| are the number of task nodes and edges in the DAG. During each loop, the *slack* value will be calculated once with complexity of $O(|\mathbf{V}|+|\mathbf{E}|)$. The worst case of C^2DAR scheme is that the scheduler will reschedule all remaining non-executed tasks and communications each time a task or communication is about to start execution, that means rescheduling will occurs (|**V**|+|**E**|-1) times. So the complexity of C^2DAR under worst case is (|**V**|+|**E**|-1) $[O(|\mathbf{V}|+|\mathbf{E}|) + O(|\mathbf{V}|\log|\mathbf{V}|+|\mathbf{E}|\log|\mathbf{E}| +|\mathbf{V}||\mathbf{E}|)]$ $= O[(|\mathbf{V}|+|\mathbf{E}|)^2+(|\mathbf{V}|+|\mathbf{E}|)(|\mathbf{V}|\log|\mathbf{V}|+|\mathbf{E}|\log|\mathbf{E}|+|\mathbf{V}||\mathbf{E}|)]$

Input: a DAG, VON resource information and a schedule S produced by a static scheduling algorithm A

```
1.  Construct schedule result graph SRG for S
2.  Mark all nodes in SRG as unexecuted
3.  E ← the real, current execution result (initially empty)
4.  while (DAG is not finished)
5.    n ←first node in SRG which is unexecuted and whose all
        predecessors are all finished execution
6.    if n is task node in DAG
7.      EST ← expected start time of n in schedule S
8.      RST ← real start time of n in E
9.      delay(n) ← RST - EST
10.   else if n is edge in DAG
11.     EFT ← expected finish time of n in schedule S
12.     DFT← determined finish time of n in E
13.     delay(n)← DFT-EFT
14.   endif
15.  Calculate Slack(n) from SRG
16.   if delay(n)>Slack(n)
17.     Based on the current computation and VON resource
        information I, and the executed result E, make a
        reschedule with algorithm A. S←A(I, E, DAG)
18.     Reconstruct SRG for S
19.   else
20.     execute n
21.     mark n as executed in SRG
22.     E ← E□{n}
23.   endif
24.  endwhile
```

Fig. 5. C^2DAR algorithm

V. Performance Evaluations

In this section, we present the comparative evaluation from 2 aspects. We first evaluate the performance of rescheduling scheme over shared VON (SVON) in comparison to the static scheduling over dedicated (DVON) and static scheduling over entire optical network (EON). And then we evaluate the improvement of C^2DAR scheme against CDAR scheme under SVON provisioning.

A. Simulation Settings

1) Application Task Graphs

We use randomly generated DAG in the following simulation. Three main parameters are used for DAG generation: the number of tasks in the DAG (|**V**|), the average edges per task node ($\overline{\delta}$) and the communication-computation ratio (CCR). CCR is defined as sum of communication cost

divided by the sum of task execution cost. We choose a set of fixed parameters |**V**|=40, $\overline{\delta}$ =2, and CCR=10. All the results presented below are averaged over 100 randomly generated DAGs.

2) VON Resource Graph

We employ 4 computing resources for task execution, which interconnected by a 16-node NSFNET network with each transmission link assigned 64-unit bandwidth and each access link assigned 7-unit bandwidth, as shown in Fig. 6. The bold lines in the figure show an instance of the VON topology with assigned bandwidth on each link of the tree for the worst-case traffic distribution computed by minimum guaranteed bandwidth of 2 units.

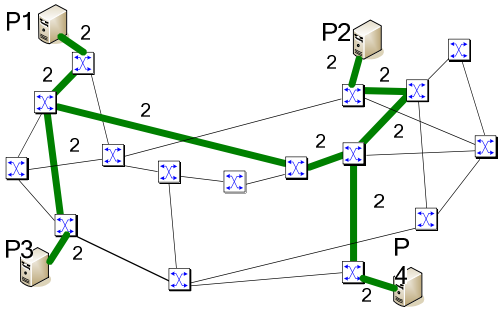


Fig. 6. VON configuration

3) Background Traffic Setting

Background traffics are generated to simulate network performance fluctuation. These traffics are all bidirectional connection requests with one unit bandwidth, and modeled using random exponentially distributed holding and inter-arrival time distributions, with the specific mean values being adjusted according to desired loading.

4) Evaluation Metrics

We use 3 metrics in the following evaluation: schedule length, network resource occupation, and reschedule times. The Network Resource Occupation (NRO) during VON lifecycle has two dimensions: the occupied bandwidth and the bandwidth provisioning time, which is defined as follows:

$$NRO = \sum_{i \in \mathbf{E}} \sum_{j \in \mathbf{L}} T_i B_i \delta_{i,j}$$

where **E** is the set of DAG edges and **L** is the set of network links, T_i is the communication time of edge $i \in \mathbf{E}$ on the network and B_i is its assigned bandwidth, $\delta_{i,j}$ means edge i has been executed on the link $j \in \mathbf{L}$.

B. Performance of Rescheduling Scheme over Shared VON

As for the SVON and DVON cases, when the application arrives the VON is first established, and then the scheduling is done based on the assigned VON resources. After the application is finished, the VON is released. If the VON cannot be setup before scheduling due to inadequate resources, the application is blocked. While for the EON case, if the reservation of one of the scheduled lightpaths cannot be done, the application is blocked.

Fig. 7 compares the scheduling results in terms of scheduling length and network resource occupation. As can be seen, although the DVON case can produce shortest schedule length, it has two problems: one is that it occupied most network resources; the other one is that it is not applicable when the background traffic load is too high. In this simulation when the traffic load is higher than 300, the block ratio is more than 90% (So we omit the schedule results of the DVON when background traffic load >300). The EON case achieves lowest network resource occupation with lowest blocking probability. However, it has difficulty in implementation since scheduler is required to obtain entire optical network resource information. By comparison, the SVON case is applicable and its performance is close to that of the EON case with moderate blocking ratio under low and median traffic load.

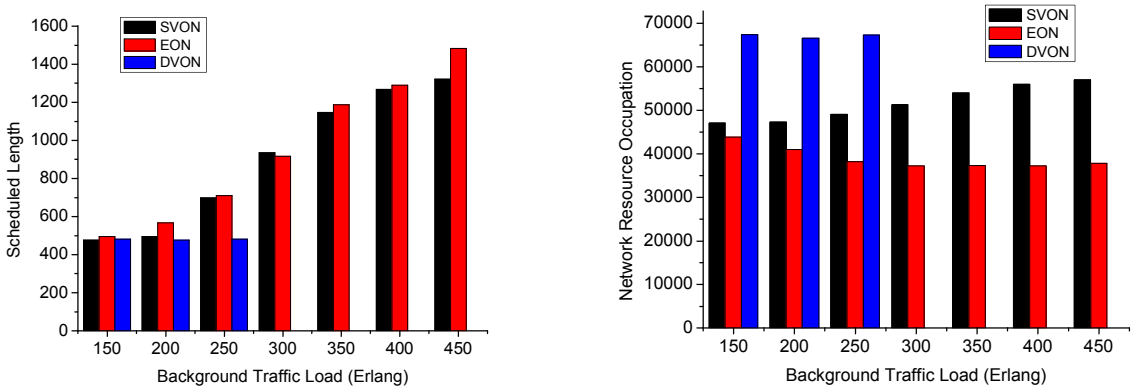
C. Improvements of C^{2}DAR Scheme against CDAR Scheme

Table I gives the comparative scheduling results of C^{2}DAR and CDAR schemes under different background traffic load. When the background traffic load is low (<200), there is almost no rescheduling occurs, C^{2}DAR and CDAR produce the same results in terms of schedule length and network resource occupation. When the traffic load increases to median level (200 ~ 350), reschedule times goes up accordingly. C^{2}DAR outperforms CDAR distinctively since the C^{2}DAR scheme is more sensitive to the bandwidth variance than CDAR and can take reschedule action earlier than CDAR. When the traffic load is high (>350), there is less shared bandwidth to utilize for the application scheduler, which results in deduction of rescheduling times. Consequently both C^{2}DAR and CDAR tend to produce the same schedule result in terms of schedule length and network resource occupation. C^{2}DAR outperforms CDAR at the cost of rescheduling complexity as analyzed in section IV.D.

VI. Conclusions

In this paper we studied the feasibility of VON service for workflow applications. In order to deal with the dynamism from both the computing resource and the shared VON resource, we proposed a computation and communication delay aware rescheduling (C^{2}DAR) scheme based on a new defined Scheduled Result Graph (SRG) concept. Simulation results show that rescheduling DAG applications over shared VON has close performance to static scheduling over entire Optical Network with more applicability. Moreover, our proposed C^{2}DAR scheme outperformed traditional CDAR scheme both in terms of schedule length, network resource occupation and rescheduling times.

In this paper it is assumed that the overheads of lighpath establishment and rescheduling are insignificant and ignored in comparison with the long data transmission time. However, in some cases, data transmission has burst behavior, so that these overheads are comparable to the communication time and cannot be ignored. How to incorporate these overheads into rescheduling scheme is our future work.



(a) (b)

Fig.7. Comparison of (a) schedule length and (b) network resource occupation with respect to background traffic load under DVON, SVON and EON scenarios.

TABLE I
COMPARISON OF C^2DAR AND CDAR WITH RESPECT TO BACKGROUND TRAFFIC LOAD UNDER SHARED VON, IN TERMS OF SCHEDULE LENGTH, NETWORK RESOURCE OCCUPATION, AND RESCHEDULING TIMES.

Background Traffic Load (Erlang)	Schedule Length			Network Resource Occupation			Rescheduling Times		
	CDAR	C^2DAR	Reduction	CDAR	C^2DAR	Reduction	CDAR	C^2DAR	Reduction
200	229	227	0.9%	23311	23289	0.1%	0.18	0.18	0.0%
250	295	261	11.5%	24075	23215	3.6%	1.76	1.74	1.1%
300	415	373	10.1%	25370	24364	4.0%	2.80	2.62	6.4%
350	529	529	0.0%	26530	26470	0.2%	2.10	1.61	23.3%

ACKNOWLEDGMENT

This work was supported by China 973 program under grant 2010CB328205, China 863 Program under grant 2009AA01334, and NSFC under grant 60736002.

REFERENCES

[1] M. Veeraraghavan, X. Zheng, and Z. Huang, "On the Use of Connection-oriented Networks to Support Grid Computing," IEEE Communications Magazine, vol.44, pp. 118-123, 2006.

[2] D. Simeonidou, C. Nejabati, G. Zervas, D. Klonidis, A. Tzanakaki, and M.J. O'Mahony, "Dynamic Optical Network Architectures and Technologies for Existing and Emerging Grid Services", IEEE Journal of Lightwave Technology, vol. 23, pp. 3347-3357, 2005.

[3] Wei Guo, Yaohui Jin, et al., Distributed computing over optical networks, OFC 2008, OWF1, San Diego, CA, USA (invited)

[4] Y. Wang, Y. H. Jin, W. Guo, W. Q. Sun, W. S. Hu and M. Y. Wu, "Joint Scheduling for Optical Grid Applications", Journal of Optical Networking, vol. 6, pp. 304-318, 2007.

[5] Z. Sun, W. Guo, Z. Wang, Y. Jin, W. Sun, W. Hu, and C. Qiao, "Scheduling algorithm for workflow-based applications in optical grid," J. Lightwave Technol. vol. 26, no. 17, pp.3011-3020, 2008.

[6] X. Liu, W. Wei, C. Qiao et al, "Task Scheduling and Lightpath Establishment in Optical Grids," INFOCOM 2008. The 27th Conference on Computer Communications. IEEE , vol., no., pp.1966-1974, 13-18 April 2008

[7] W. Guo, Z. Liang, Z. Sun, S. Xiao, Y. Jin, W. Sun, and W. Hu, "Task scheduling considering fault probability for distributed computing applications over an optical network," J. Opt. Netw. 7, 947-957 (2008)

[8] N. Chowdhury and R. Boutaba, "Network Virtualization: State of the Art and Research Challenges", IEEE Communications Magazine, 47(7), 2009, pp.20-26

[9] P. Vicat-Blanc Primet, S. Soudan, and D. Verchere, "Virtualizing and Scheduling Optical Network Infrastructure for Emerging IT Services", Journal of Optical Communication and Network, 1(2), 2009, pp.121-132

[10] Z. Yu and W. Shi. "An Adaptive Rescheduling Strategy for Grid Workflow Applications". In Proceedings of the 21st IPDPS 2007, Long Beach, USA, Mar 26 -30 2007. IEEE Computer Society Press.

[11] R. Sakellariou and H. Zhao. "A Low-Cost Rescheduling Policy for Efficient Mapping of Workflows on Grid Systems". Scientific Programming, 12(4), pages 253-262, December 2004.

[12] M. Lopez, E. Heymann, M. Senar. "Analysis of dynamic heuristics for workflow scheduling on Grid systems". ISPDC 2006, West University, Timisoara, Romania, 6-9 July 2006.