

# Real-Time Strategy and Practice in Service Grid\*

Hai Jin, Hanhua Chen, Jian Chen, Ping Kuang, Li Qi, Deqing Zou  
Cluster and Grid Computing Lab  
Huazhong University of Science and Technology, Wuhan, 430074, China  
Email: hjin@hust.edu.cn

## Abstract

*The emerging service grids bring together various distributed application-level services to a 'market' for clients to request and enable the integration of services across distributed, heterogeneous, dynamic virtual organizations. However, there are a number of applications with the requirement of time constraints. We propose a real-time strategy in service grid architecture. We also extend the OGSi grid service semantics for fault-tolerant. The real-time and fault-tolerant strategies seem efficient through experiments.*

## 1. Introduction

Service oriented computing is often seen as a natural progression from component-based software development, and as means of dynamic and flexible integration of services. A service in this context may be allowed to be a part of more than one distributed system and simultaneously serve different applications. According to [1], the service abstraction may be used to specify access to computational resources, storage resources, and networks, in a unified way. How the actual service is implemented is hidden from the user through the service interface.

Real-time and reliability are requirements of some crucial application fields and they are the foremost requirements in our grid project. *Globus Architecture for Reservation and Allocation* (GARA) [2] provides a flexible architecture to make it possible for the application to make advance reservation for different types of resources, including networks, CPUs, disks, and graphic pipelines. Although it

\* This paper is supported by Nature Science Foundation of China under grant 60273076, ChinaGrid project from Ministry of Education, and National 973 Basic Research program under grant 2003CB316903.

is possible for the user to find a time when all the QoS

constraints will be able to be simultaneously met in the future. The failure of reservation is potentially a common case because of competition for resources among applications. This is a flaw of GARA to be implemented in a real-time grid environment. We propose a more assured high-level strategy for a given service to indicate the real-time capacity it can offer. We partition the application time cost into three categories: service computing time, network cost, and grid middleware cost. Base on the cost model different scheduling policies can be designed.

In a real-time service grid application, individual services may fail without the entire application being terminated. The potential of faults is a significant factor in real-time grid architecture. In general a fault-tolerant mechanism must provide two capacities, namely fault-detecting and fault-recovery. Until now, unreliable failure detection technology [3] is a perfect theory on distributed component failure detection with the consideration of the complication in the distributed system. Paul et al. [4] have provided a two-level fault detection architecture to detect the faults of both a computer and any monitored processes on that computer. At the description layer, WSDL provides a mechanism by the way of <wsdl:fault> for applications to specify the error characteristics. This is similar to the way we define exceptions raised by the methods in Java interface. Similarly the underlying SOAP messaging layer, provides a <soap:fault> for applications to communicate the error information. Based on this, [5] introduces their architecture for fault-tolerant of web services. They enable a web service to handle faults by extending the service interfaces with methods such as checkpoint and rollback for fault-tolerance. They also designed their distributed fault-tolerant components and specific communication between them. The error mechanisms of SOAP and WSDL help support errors raised by an application, however no mechanism exists for handling framework failures and system errors. Service oriented grid architecture should unify the concept of faults including both system level and

service level, and need a more flexible mechanism to define, detect and handle the fault. Based on *Open Grid Service Infrastructure* (OGSI) we propose a more flexible and general fault-tolerant framework in section 2.4.

In the real-time grid environment, service customers must be given some form of commitment and assurance, such as service response time, network capacity, middleware cost time and also fail-over policies. These terms of commitment need to be agreed upon before using and manifested in form of a SLA. A real-time and fault-tolerant oriented *Service Level Agreement* (SLA) schema is designed for negotiating contracts among service customer, *service provider* (SP), grid middleware and *Internet service provider* (ISP). We also designed a *Service Level Management* (SLM) framework to manage the status of a service level agreement.

## 2. Real-time Service Grid Model

As acquiring access to remote resources is complicated by the competing needs of the clients and services, it is difficult to guarantee QoS of remote services. However, there are a number of grid applications with the requirement of time constraints, and the users care about the overall capacity of a service, for example the response time or the deadline of a remote service access. We propose a real-time service grid model on base of the conception of *Service Virtualization*. In our model *Service Virtualization* is defined as the ability to schedule the service requests with QoS constraints transparently onto appropriate native implementations of the common service semantics. We propose a quantitative approach for real-time service scheduling. The time cost of an access to grid service is classified into 3 levels: (1) computing time cost  $t_{com}$ , (2) middleware time cost  $t_{mid}$ , and (3) network time

cost  $t_{net}$ . The service providers in our model are assured to specify the computing capacity of the service. We developed an Xml Schema for capacity description. We adopt the framework of *IntServ/RSVP over DiffServ Network* (IS/DS) [6] to deal with network QoS. In our architecture *Network Resource Manager* (NRM) serves as a kind of network QoS Broker [7][8] within a domain. The NRM determines a communication path with stable bandwidth and latency between the source and the destination. We also suggest evaluating the time cost of the service grid middleware.

### 2.1. Service Virtualization

Service-oriented view simplifies *service virtualization* through encapsulation of diverse implementations behind a common interface. [9] introduces a service grid prototype which was built using the Legion wide-area computing infrastructure to explore the design of application and service based replica selection policies in a wide-area network. However they use only random and round robin policies. We partition the *Service Virtualization* problem into two sub-problems: the definition of service interfaces, which defines the semantics of a common and service scheduling among multiple implementations of the service interface with different *Quality of Service* (QoS) separately.

Our grid architecture provides an extensible set of *virtual service* (VS) to users.

$$Grid = \{VS\}$$

We adopt gwsdl (defined in OGSI) to define operations and associated attributes of a virtual service. We call an implementation of VS a *physical service* (PS). Thus, each VS is implemented as a set of redundant PSs with different

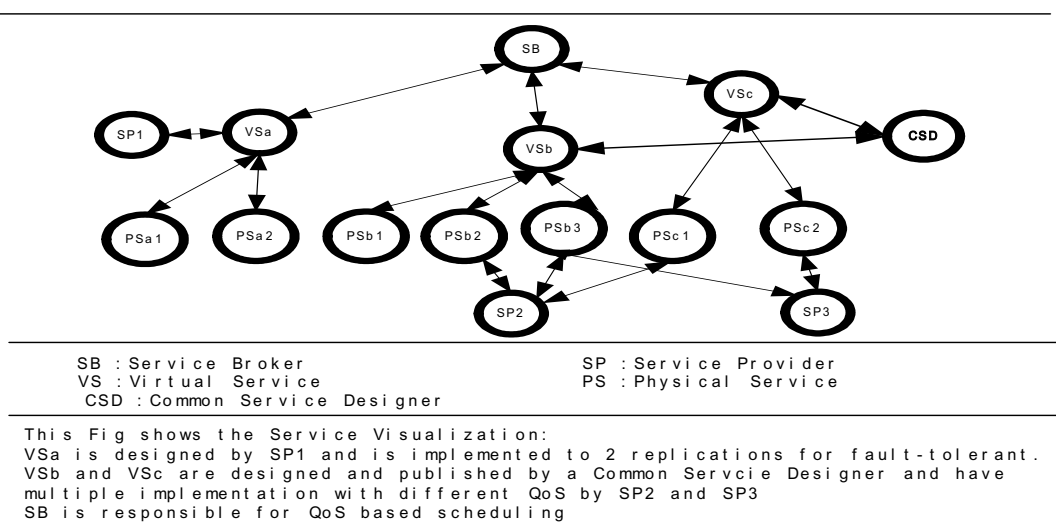


Figure 1. Service Virtualization Model

QoS. A single service provider can publish a VS and deploy multiple PSs of it for fault recovery in the case of failure or for the load balancing of requests. Multiple service providers can implement the same common VS at different response time cost.

## 2.2. Time Cost of Grid Service

**2.2.1. Computing Time Cost  $T_{com}$ .**  $T_{com}$  expresses the capacity of a service to response. Here, it only refers to the total computing time of all the operations involved in once service access. The service providers in our model are assumed to specify the computing time of every operation as the metadata of the service. And the service providers ultimately guarantee the  $T_{com}$  of the service access in the form of SLA.

To guarantee the real-time capacity of a given service we adopt two strategies. For those crucial services, which have much strict time constrains, we adopt real-time operation system in a grid node to guarantee every service execution in the sequence of priority in single machine. For requests to service with less stricter time constrains, we require the service providers to limit the number of service connections, evaluate the time cost of service operations and give the upper limit. In both cases the capacity is specified as metadata of the service and to be negotiated into an SLA.

In Fig 2 we developed an XML schema for service capacity description. In addition our prototype provides graphic tools for service providers to publish their services interface with capacity metadata. The element of *ResponseTime* is correspondingly to describe the computing time of every operation in the service.

[10] has introduced an efficient wide-area distributed service discovery strategy. They mainly use caching and propagation of discovery results with client QoS feedbacks in the discovery server hierarchy. We adopt their advice to structure the wide-area information service. In previous paper [11], we have described the domain-divided hierarchical architecture of our grid information service. We built local domain information service on Apache Xindice, an open source XML database system.

It's support for XQuery [12] and Xpath [13] make it easily for us to build the searching engine for service QoS metadata. The API provided by Xindice can also easily update the capacity metadata.

**2.2.2. Network Time Cost  $T_{net}$ .** In a grid environment, networks are essential to the smooth operation of the grid, and they are often the most shared resources among services. Grid application have a wide-variety of network QoS needs, because grid applications use a wide variety of network flows including low bandwidth but low latency control flows, high-bandwidth and low latency data transfers for application such as visualization, and high bandwidth but not necessarily low latency data transfers for moving large amounts of data for analysis and other purposes. If the bandwidth and latency is fixed the  $T_{net}$  during access to a service access can be defined below:

$$T_{net} = (data\_size_{input} + data\_size_{output}) / bandwidth + lantency$$

Network capacity in our framework is primarily concerned with the management of bandwidth and latency. *IETF proposed two major architectures to support Internet QoS: Integrated Services (IntServ) architecture [14] and Differentiated Services (DiffServ) architecture [15][16]. IntServ architecture is a per-flow oriented QoS architecture that uses Resource Reservation Protocol (RSVP) for resource reservation and control. DiffServ networks classify packets into a small number of aggregated flows or classes, based on DiffServ codepoint (DSCP) in the TOS byte of the packet in IP header. At each DiffServ router, packets are subjected to a per-hop behavior. The primary benefit of DiffServ is its scalability. DiffServ eliminates the need for per-flow state and per-flow processing, therefore scales well to large networks.*

Fig.3 shows a simple IS/DS architecture based reference network, which includes a DiffServ region in the middle of a large network supporting IntServ end-to-end and being RSVP unaware. The DiffServ region contains a mesh of

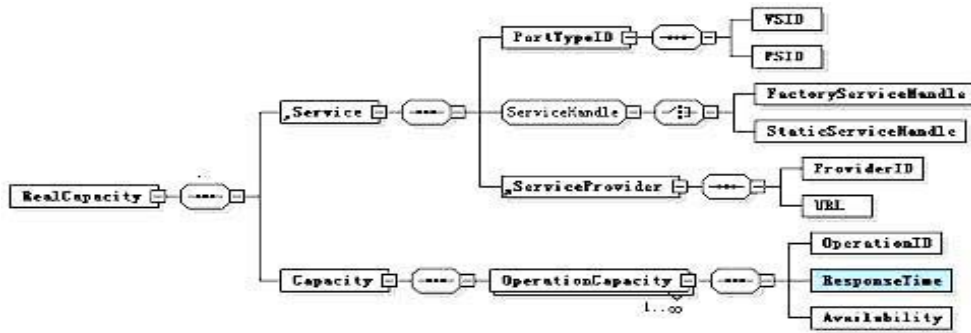


Figure 2. Service Capacity Metadata XML Schema

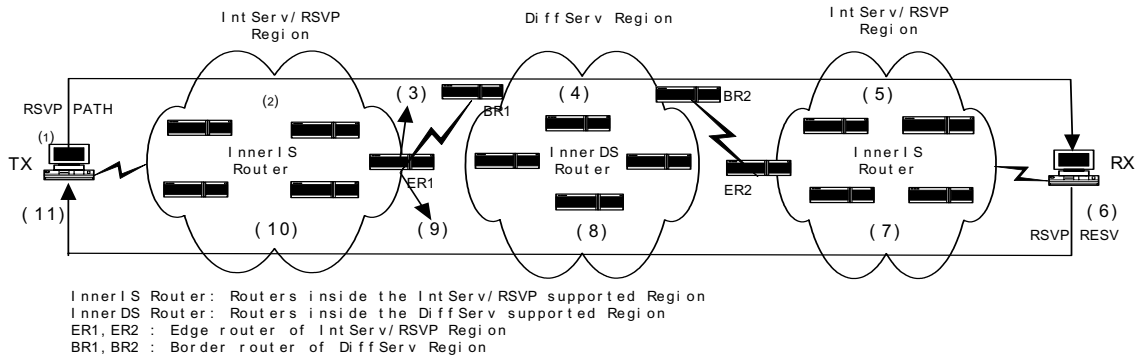


Figure 3. A Simple IS/DS Architecture

routers, at least some of which provide aggregate traffic control. The regions outside the DiffServ region contain meshes of routers and attached hosts, supporting integrated services. The NRM determines a communication path with stable bandwidth and latency between the source and the destination.

The assumptions of the reference network are as follows:

- a) Sending and receiving hosts use RSVP to communicate the quantitative QoS requirements of QoS-aware applications running on the host;
- b) RSVP signaling messages travel end-to-end between hosts Tx and Rx to support IntServ/RSVP reservations outside the DiffServ network region;
- c) Edge routers (ER1 and ER2) act as admission control agents to the DiffServ network. Border routers (BR1 and BR2) act as pure DiffServ routers. Devices in the DiffServ region pass RSVP messages transparently.

The steps (1)~(11) in Figure 3 illustrate the process by which an application obtains end-to-end QoS.

**2.2.3. Middleware Time Cost  $T_{mid}$ .** The middleware time cost is particular important as our prototype is based on existing middleware, namely GlobusToolkit3.0 and other third party software which have no real-time guarantee. For example, in the experiment on our prototype we found that the costs of service selection, service instance creation and the initialization of a grid client etc. are fluctuating.

We simply specify the upper limit of the overall middleware cost  $T_{mid}$  in the form of SLA. It is totally up to the middleware to provide such a QoS guarantee. The Libra project [17] aims to explore middleware QoS in particular developing a QoS-based scheduler for resource management on a homogeneous cluster.

### 2.3. Service Scheduling Policy

In the *Service Virtualization* model, all the PSs of the same VS have the uniform *operation* (O) set and are invoked in the same manner. We define  $C^j(i)$  as the time cost of operation  $O_i$  of jth PS implementation of VS. We define the time cost of a grid service operation  $O_i$  as: (1) computing time cost  $t_{com}^i(i)$ . (2) middleware time cost  $t_{mid}$ , and (3) network time cost  $t_{net}(i)$ .

$$C^j(i) = C_{com}^j(i) + C_{net}^j(i)$$

The overall service access time cost of jth PS  $C^j$  is as follow:

$$C^j = C_{mid}^j + \sum_{i=1}^n c^j(i) \quad (o_i \in \text{the request operation set})$$

We use 2-tuple  $\langle PS_a^j, C^j \rangle$  ( $0 < j < n$ ,  $n$  is the number of redundant ps) to define the response capacity of jth PS during an access to virtual service  $VS_a$ . We use 2-tuple  $\langle VS_a, c \rangle$  to define the time constrain of the request to  $VS_a$ .  $c$  is the time constrain offered by the user. Different

capacity mapping policies can be choose, and the agreement can be negotiated before request dispatching. For example in our prototype we implemented a policy of *economical-selection*, which simply selects  $\langle PS_a^m, C^m \rangle$

from  $\{\langle PS_a^j, C^j \rangle\} (0 < j < n)$  and  $C^m$  is the most economical cost among  $\{C^j\} (0 < j < n)$  to meet requirement of  $\langle VS_a, c \rangle$ . We also provide other policies such as *best-selection* policy to choose the best capacity. The policy set in our prototype is extensible.

## 2.4. Reliability of Real-time Scheduling Policy

OGSI provides mandatory grid service semantics, such as service invocation, lifetime management, service status obtaining, communication mechanism, that ensure fundamental interoperability among all grid services. OGSI introduces the concept of service data, which combined all attributes of grid service instance into a single document. With service data mechanism we can define any kind of attribute of the service in an XML Schema, for example the termination-time of the grid service instance, the medial result of a computation, and even the CPU occupying rate of the computer the service hosted in. OGSI also provides rich pull and push remote operations against service data documentation. In our framework we use service data for faults definition. Service providers can define faults of their service on their demands. For attaining the service status, we also provide a set of libraries bound dynamically to the service code.

We extend the OGSI grid service interfaces for fault-tolerance. *ServiceMonitor* interface is responsible for analyzing and detecting faults of a service instance. *MonitorManager* is a grid component to manage multiple *ServiceMonitors*. Any third party and service provider can implement them to manage potential fault, to realize his or her own fault analysis algorithm and even to specify the policy for service recovery. For the communication mechanisms among the fault-tolerant components we utilized the OGSI Notification mechanism, which support full push and pull methods.

## 3. Contract-Based QoS Guarantee

Negotiating a SLA is an exchange of messages among customer, middleware, ISP, and service provider. The terms in the SLA should be measured and monitored during the life cycle of the grid service instance. The SLA also should indicate what actions should be taken in case any side is unable to meet the objectives in the SLA.

We designed a *Service Level Management (SLM)* framework to manage the status of a service level agreement. Figure 4 shows the status management framework for the SLA instances during the lifetime. The

status set includes *Init*, *Created*, *Running*, *Evaluate*, *Failure* and *Destroyed*. Before once service grid access, a SLA document instance is created after negotiating among all the sides. During the access to a service instance, the status of the service instance is monitored. If any faults of the service instance occur the recovery for the real-time request should be tried in the left time before the deadline.

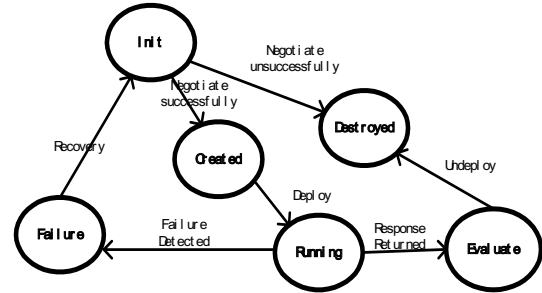


Figure 4. Service Level Agreement Status

At the same time the capacity metric should be monitored according to the *Directives* of the SLA, and capacity of the service is evaluated after the service response. Actions should be taken in case any side is unable to meet the objectives in the SLA.

## 4. Experiment Results

On the prototype of our service grid we published a virtual calculator service as an example. The virtual calculator has three implementations on the sites from different sub-nets of the *Cluster and Grid Computing Lab of Huazhong University of Science and Technology* in China. Services deployed on heterogeneous sites provide different response capacity and network cost. On site 211.69.206.128 an IBM X series 235 server runs, 211.69.206.199 an HP server rx2600, and 202.114.14.252 a PC server equipped with an Intel Xeon 1GHz CPU and 512MB memory.

We have three groups of experiments. In each group, 20 random recodes of requests, which have the same time constrain, are selected. The deadlines specified in the three groups are separately 240ms, 260ms, and 450ms. According to the metadata of the services all the requests are allocated to a site. Each record comprises the cost of computing time, communication time and the middleware time monitored, see Table 1. The data with block show the cases in which the real-time constrain is not met. The preliminary results demonstrate that the strategies have a good role on our service grid prototype. The single scheduling success ratio is above 90%.

## 5. Conclusions and Future Work

Based on service virtualization model of service grid, a framework for real-time service scheduling is proposed. We designed the description language for three kinds of time cost in the service grid environment. The language is used for a service provider to describe the capacity metadata of a service. The IntServ/RSVP over DiffServ architecture satisfies the inter-domain network QoS. We also extended the OGSi grid service interfaces for fault-tolerant. To provide real-time guarantees for service

execution we designed the schema of SLA. A SLM framework was also proposed for SLA status management.

In next steps, we will design a more light-weighted and effective RSVP signaling protocol for inter-domain real-time network communication. We will also analyze the cost of grid middleware and to improve the performance.

**Table 1. Experiments Result of Three Groups**

	211.69.206.128			211.69.206.199			202.114.14.252		
	Tmid	Tnet	Tcom	Tmid	Tnet	Tcom	Tmid	Tnet	Tcom
1	120	70	30	130	90	40	110	111	80
2	100	80	40	120	80	50	130	90	90
3	110	80	30	120	80	81	150	90	70
4	100	80	41	130	70	30	110	100	70
5	120	80	30	331	80	30	110	90	70
6	120	70	40	120	110	30	130	90	70
7	120	85	30	100	71	30	110	90	70
8	100	70	41	110	70	30	110	100	70
9	110	261	40	90	70	30	110	110	70
10	110	70	40	101	70	40	130	100	70
11	110	80	30	120	70	40	110	90	80
12	100	81	40	90	370	40	90	90	70
13	90	60	40	100	60	30	90	90	71
14	140	70	35	100	71	30	100	100	70
15	100	70	30	110	70	30	90	110	80
16	100	80	40	100	60	40	90	100	70
17	110	70	30	90	90	30	90	110	70
18	120	70	30	90	60	30	351	90	70
19	101	80	30	90	70	30	90	100	80
20	111	70	40	110	70	20	100	90	70

## References

- [1] I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke, "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration", DRAFT document of Globus Project, February 17, 2002.
- [2] I. Foster, C. Kesselman, C. Lee, R. Lindell, K. Nahrstedt, and A. Roy, "A Distributed Resource Management Architecture that Supports Advance Reservations and Co-Allocation", Proceedings of International Workshop on Quality of Service 1999, May 31 - June 4, 1999, pp.27-36.
- [3] R. Guerin, A. Orda, and D. Williams, "QoS routing mechanisms and OSPF extensions", Proceedings of Global Telecommunications Conference, 1997, pp.1903-1908.
- [4] P. Stelling, I. Foster, C. Kesselman, and C. Lee, "A Fault Detection Service for Wide Area Distributed Computations", Proceedings of 7th IEEE Symp. on High Performance Distributed Computing, 1998, pp. 268-278.
- [5] V. Dialani, S. Miles, L. Moreau, D. D. Roure, and M. Luck, "Transparent fault tolerance for web services based architectures", Proceedings of Eighth International 12 Europar Conference (EUROPAR'02), Lecture Notes in Computer Science, Vol.2400, pp.889-898.
- [6] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998.
- [7] J. Vollbrecht, P. Calhoun, S. Farrell, L. Gommans, G. Gross, B. de Bruijn, C. de Laat, M. Holdrege, and D. Spence, "AAA Authorization Application Examples", RFC 2905, August 2000.
- [8] B. Teitelbaum, S. Hares, L. Dunn, R. Neilson, V. Narayan, F. Reichmeyer, "Internet2 QBone: Building a Testbed for Differentiated Services", Network, Vol.13, 1999, pp.8-16.
- [9] J. B. Weissman and B. D. Lee, "The Service Grid: Supporting Scalable Heterogeneous Services in Wide-Area Networks", Proceedings of 2001 Symposium on Applications and the Internet (SAINT 2001), Jan 8-12, 2001, San Diego, CA, pp.95-102.
- [10] D. Y. Xu, K. Nahrstedt, and D. D. Wichadakul, "QoS-Aware Discovery of Wide-Area Distributed Services", Proceedings of the First International Symposium on Cluster Computing and the Grid, 2001, pp.92-99.
- [11] H. Chen, H. Jin, M. Zhang, D. Zou, and P. Yuan, "Early Experience in QoS-Based Service Grid Architecture", Advanced Web Technologies and Applications - Lecture

*Notes in Computer Science, Vol.3007, pp.924-927.*

- [12] S. Boag, D. Chamberlin, M. F. Fernández, D. Florescu, J. Robie, J. Siméon, “XQuery 1.0: An XML Query Language”, W3C Working Draft, November 12, 2003.
- [13] J. Clark and S. DeRose, “XML Path Language (XPath) Version 1.0”, W3C Recommendation, November 16, 1999
- [14] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, “Resource Reservation Protocol (RSVP) - Version 1 Functional Specification”, RFC 2205, Sept. 1997.
- [15] R. Braden, D. Clark, and S. Shenker, “Integrated Services in the Internet Architecture: an Overview”, RFC 1663, July. 1994
- [16] K. Nichols, S. Blake, F. Baker, and D. Black, “Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers”, RFC 2474, December 1998.
- [17] J. Sherwani, N. Ali, N. Lotia, Z. Hayat, and R. Buyya, “Libra: An Economy driven Job Scheduling System for Clusters”, Technical Report, The University of Melbourne. July 2002.