G-lambda: Coordination of a Grid Scheduler and Lambda Path Service over GMPLS

Atsuko Takefusa[1]	Michiaki Hayashi[2]	Naohide Nagatsu[3]	Hidemoto Nakada[1]
Tomohiro Kudoh[1]	Takahiro Miyamoto[2]	Tomohiro Otani[2]	Hideaki Tanaka[2]
Masatoshi Suzuki[2]	Yasunori Sameshima[3]	Wataru Imajuku[3]	Masahiko Jinno[3]
Yoshihiro Takigawa[3]	Shuichi Okamoto[4]	Yoshio Tanaka[1]	Satoshi Sekiguchi[1]

[1] Grid Technology Research Center, National Institute of Advanced Industrial Science and Technology (AIST),
1-18-13 Soto-Kanda, Chiyoda-Ku, Tokyo 101-0021, Japan
+81-3-5298-4724 (Telephone), +81-3-5298-4502 (Fax), atsuko.takefusa@aist.go.jp
[2] KDDI R&D Laboratories Inc.
[3] NTT Network Innovation Laboratories
[4] National Institute of Information and Communications Technology (NICT), Japan

Abstract

At iGrid2005, we conducted a live demonstration where our Grid scheduling system co-allocated computing and network resources with advance reservation through Web services interfaces using the Grid Resource Scheduler (GRS), the Network Resource Management System (NRM), which is capable of GMPLS network resource management, and a GMPLS-based network test-bed, for the first time. The goal of the G-lambda project is to define a standard Web services interface (GNS-WSI) between GRS and NRM that is acceptable for both application service providers and commercial network operators, and which can be used as a tool for realizing new and emerging commercial services.

Keywords

Grid scheduling, Optical network, GMPLS, Web services, Advance reservation

1. Introduction

The maturity of the Grid infrastructure technology now allows deployment of large-scale high performance computing applications over distributed computing resources from different organizations. However, when application users actually perform computation over the Grids, they encounter many problems. The causes are as follows:

- The current, best-effort Internet is easy to use, but it is unstable.
- While optical network technology and dedicated optical paths can provide high quality communication, the provisioning of network resources is not automated. It often requires 'human negotiations' using e-mail or fax before the computation can take place.
- A 'Superscheduler,' capable of co-allocation of computing resources from different organizations, is premature. Users have to allocate resources by negotiating with computing resource managers, again, by e-mail or fax.

To address these issues, we propose and demonstrate a Web services-based Grid scheduling system that is capable of co-allocation of computing and network resources with advance reservation, leveraging a GMPLS-controlled optical network. GMPLS [2] control plane enables the automated network provisioning over the multi-layers using is a GMPLS protocol suite that unifies the control plane of various types of network equipments, such as optical cross-connects (OXCs), time-division multiplexing cross-connects (TDM-XCs), IP routers, and so on. Our Grid scheduling system consists mainly of the Grid Resource Scheduler (GRS) and the Network Resource Management system (NRM). GRS calls on NRM to request optical paths on a

GMPLS-controlled network, and coordinates the required network and computing resources with advance reservation.

Our contribution is that we actually conducted an experiment of coordination of both computing and network resources with advance reservation automatically using real clusters and a real network. Six clusters in southern part of Japan which are connected by GMPLS network are used for this experiment. In the experiment at iGrid2005; (1)a user submits a requirement on Grid resources to GRS, (2)GRS searches all the computing and network resources for available timeframe of required resources and co-allocates the resources with advance reservation by negotiating NRM and CRMs, (3)NRM provides abstraction of physical GMPLS-based lambda paths resources for GRS and allows automatic provisioning of the resources at the reserved time, and (4)the user program is executed over the actual provisioned resources. Our proposed Web services-based Grid scheduling system is able to provide a suitable set of dedicated computing and network bandwidth resources automatically for each user requirement through the GUI with easy operation, showing the feasibility of the system and its architecture. We also point out some important issues for future resource reservation services, including a messaging protocol between GRS and NRM, debugging, dynamic routing of lambda paths, and fault tolerance.

The goal of the G-lambda project[1] is to define a standard Grid Network Service - Web Services Interface (GNS-WSI) between GRS and NRM, which is acceptable for both application service providers and commercial network operators, and which can be used as a tool for realizing new and emerging commercial services.

2. A Web Services-based Grid scheduling system architecture

Fig. 1 shows an overview of our Web services-based Grid scheduling system, where not only computing resources, but also network resources that connect computing resources, are recognized as part of the managed resources, and are dealt with integrally. In other words, the required network bandwidth is reserved through the co-functioning of the GRS, which reserves overall resources, and the NRM, which flexibly establishes optical paths, with adequate bandwidth between defined locations, all using the defined interface.

For network control, GMPLS technology, which is currently undergoing standardization for the control of optical paths, is applied. GMPLS makes it possible to connect geographically distributed computers and/or storage devices freely on an as-necessary basis, using the appropriate transmission bandwidth. Furthermore, the potential for cost reduction, as well as drastic enhancement of both computation efficiency and usability, is anticipated.

Since resources to be provided and managed in a Grid environment are geographically distributed, the network connecting these resources is likely to extend over several regions and countries, and thus may be provided by multiple network operators. Therefore, it is important to have a common interface among network operators to enable the exchange of information that is required for co-functioning between the GRS and the NRM. We have defined the basic functions of such an interface in the Grid Network Service Web Services Interface (GNS-WSI). The details of the Grid scheduling system architecture are shown in Fig. 2.

2.1. The Grid Network Service Web Services Interface (GNS-WSI)

As a preliminary interface, we defined the following polling-based operations using WSDL: *netResourceReservation* allocates paths requested by GRS to physical links between OXCs with advance reservation. GRS sends a requirement for network resources such as the site ID of each endpoint, bandwidth, reservation time, delay, and the levels of fault recovery for each path. *netResourceModification* modifies the specifications of previously reserved paths such as bandwidth and reservation time. *netResouceRelease* releases reserved paths. *netResourceReservationStatusQuery* returns the current status of requested paths. This query

operation notifies GRS if the reservation, modification, or release request succeeded or if the reserved path connections are actually established at the reservation time. *netResourceQuery* and *netAvailableResourceQuery* returns the availability of paths during a period specified by GRS.

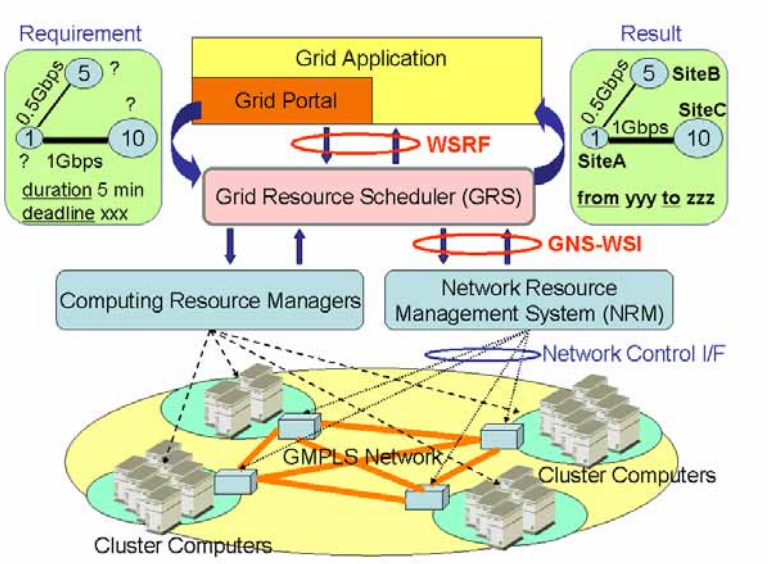


Fig. 1 Overview of the Grid Scheduling system.

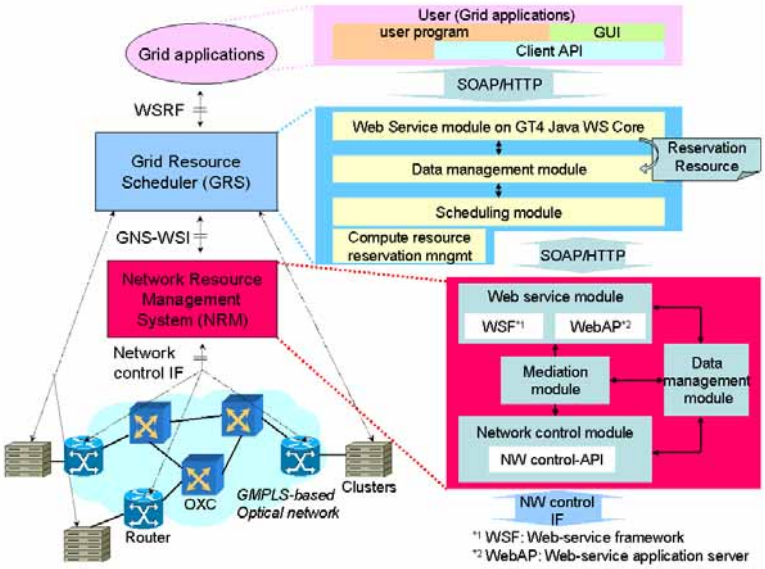


Fig. 2. The details of the Grid scheduling system architecture.

2.2. The Grid Resource Scheduler (GRS)

We have developed a WSRF (Web Services Resource Framework)-based Grid Resource Scheduler (GRS) that is capable of co-allocation of computing and network resources with advance reservation. WSRF, standardized by OASIS[7], is an interface for an open framework for modeling and accessing stateful resources using Web services. Computing resources are reserved through Computing Resource Managers (CRM), and network resources are reserved through the Network Resource Management System (NRM), as shown in Fig. 1. To

co-allocate computing and network resources, GRS negotiates with the CRM of the required computing resources and the NRM provided by network operators and co-allocates the resources for each user requirement.

We have implemented a prototype of GRS using Globus Toolkit 4 (GT4)[4][5], a WSRF reference implementation written in Java. As shown in Fig. 2, GRS consists of the following modules: The *Client API and GUI* are programming and graphical user interfaces. The *Web Service Module* provides a reservation service interface based on WSRF. This module creates a *Reservation Resource* instance for each user reservation request, stores the instance in the *Data Management Module*, and returns the end point reference (EPR) to the user. EPR is a reference to access each *Reservation Resource* and contains the URI of the GRS Web service module and the reservation ID for the request. The user accesses the EPR to get the current scheduling status and the result. The *Reservation Resource* is a reservation service entity made available for each user reservation request. The *Data Management Module* stores the state of each reservation process persistently, and guarantees recovery from failures. The *Scheduling Module* negotiates with CRM and NRM and co-allocates suitable computing and network resources for each user requirement. If the *Scheduling Module* cannot find resources to meet the user specified deadline, the *Web Service Module* returns the "Reservation Failed" status to the user and the user re-submits a new reservation request. The *Computing Resource Reservation Management* module manages the status of advance reservations on each site managed by CRM.

A scheduling scheme for co-allocation with advance reservation of both computing and network resources is a serious issue for Grids. A scheduler has to find when and where resources are available for each user requirement such as the number of clusters, the number of CPUs for each cluster, network bandwidth between the clusters, duration and deadline for reservation timeframe. Developing a effective scheduling scheme is future work. At this point, we employ a depth-first search scheme to find required resources and the earliest timeframe to meet the user-specified deadline by using both information on the number of available CPUs during the specified period provided by CRMs and network resource availability for the specified network bandwidth during the specified period provided by NRM.

2.3. The Network Resource Management System (NRM)

To enable a lambda-based grid network service (GNS) over a GMPLS optical network infrastructure for grid applications, NRM, developed by KDDI R&D Laboratories, is configured with three key functional modules. In the *Web Service Module*, NRM provisions the GNS-WSI defining service conditions and procedure based on the WSDL. By using indirect invocation with the GNS-WSI, and RPC-based SOAP/HTTP messaging between NRM and a global grid brokering system, GRS achieves discovery of network resources, reservation of dedicated lambda paths, and modification of the network service. In the *Mediation Module*, NRM provides resource abstraction and local scheduling functionalities to mediate between the requirement from GRS and the network resource. In the mediation process, NRM virtualizes GMPLS-controlled dynamic optical networks as lambda paths between cluster sites for grid applications. Scheduling of lambda paths is also conducted in the mediation module. In the *Network Control Module*, NRM controls GMPLS routers as network edge devices through the network control interface to activate required lambda paths and to monitor their states and performance. Through the network control module, NRM can synchronize the signaling and operational states of the lambda paths in the actual GMPLS network. Those statuses of the lambda path are retrieved when the lambda path is activated and deactivated.

3. Overview of the iGrid2005 demonstration

At the iGrid2005 showcase in San Diego, we conducted a live demonstration where a user submitted a request

for Grid resources to the GRS from the GUI, the GRS negotiated with the NRM and co-allocated suitable computing and network resources simultaneously with advance reservation, and a scientific application program was executed over the reserved resources. The GUI and the GRS were run on a laptop PC in San Diego, the NRM was located in Tokyo, and all the network and computing resources were located in Japan. We used the GMPLS network deployed over JGN II, an advanced network test-bed in Japan. Clusters distributed over six locations in Japan (Tsukuba, Akihabara, Kamifukuoka, Kanazawa, Osaka, and Fukuoka) were connected over this network, as shown in Figure 3. As an application program executed over the reserved resources, we employed a molecular dynamics simulation implemented with both the grid middleware Ninf-G2[3] developed by AIST, and Globus Toolkit 2 (GT2)[4][6].

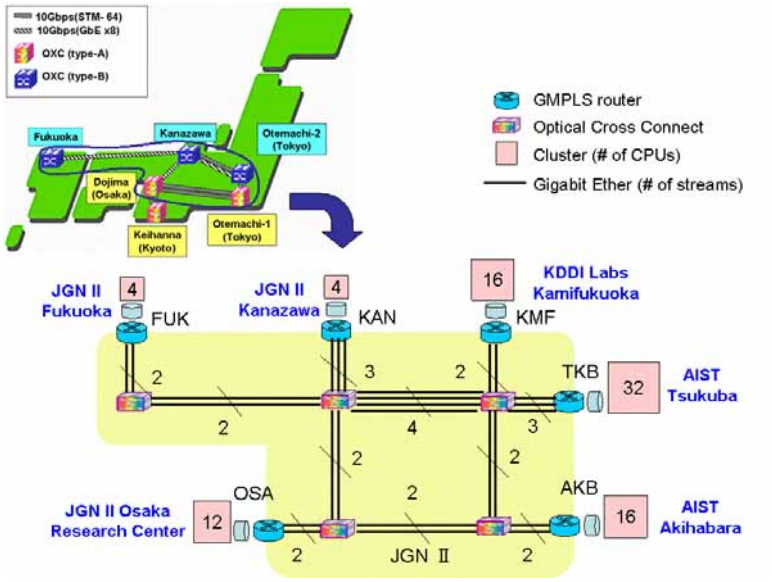
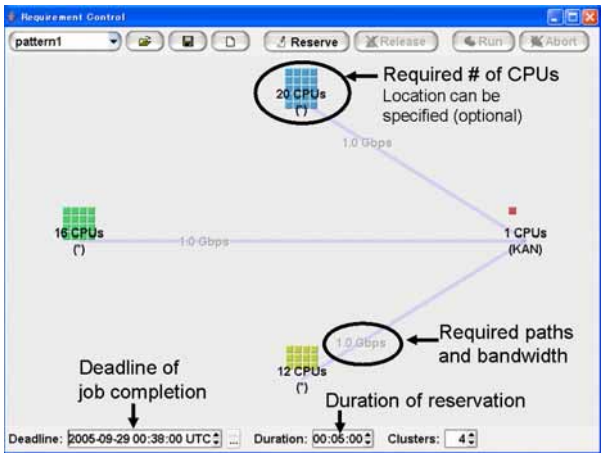


Fig. 3. Network and cluster configuration.

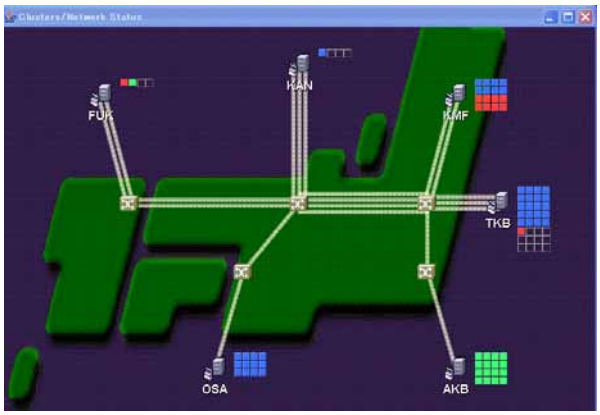
3.1. Experimental environment

We used the IP/photonic GMPLS network deployed over JGN II which consists of optical cross-connects (OXCs) and GMPLS IP routers. Since the GMPLS label switched path (LSP) carrying IP traffic is switched at the photonic layer with optical switches, we define this architecture as IP/photonic GMPLS network. In this paper, we call the GMPLS LSP switched at the photonic layer as lambda path..OXCs and IP routers were distributedly controlled by standard-based GMPLS protocols rather than centralized management plane-based network control.

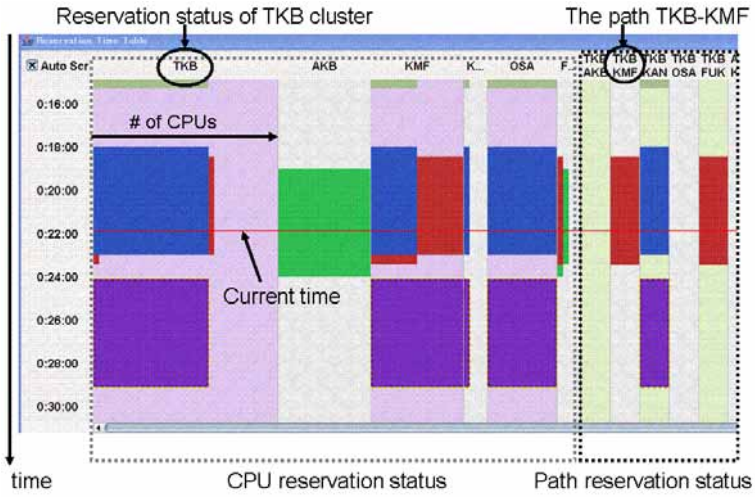
In this demonstration, we did not utilize the E-NNI signaling or routing. Although the network consists of two administrative domains, we utilized single domain-based GMPLS signaling. The route of each LSP was strictly specified at the ingress node. Each GMPLS nodes were connected with Gigabit Ethernet-based Traffic Engineered (TE)-links with the lambda switching type. Single lambda path with 1-Gbit/s bandwidth was configured between routers. In this demonstration, all the lambda paths were managed and controlled by single NRM. To provision lambda paths, NRM activated the lambda path by accessing the initiator GMPLS router, and then the GMPLS router initiated GMPLS RSVP-TE signaling to provision the lambda path.

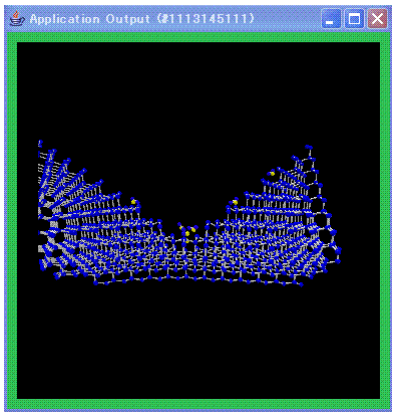


(a) The requirement editor.



(c) The map view.





(b) The reservation timetable (left: cluster status, right: network status). (d) The application panel.

Fig. 4. The GUI windows shown at iGrid2005.

3.2. The flow of the demonstration

In the demonstration, a user specifies the required number of clusters and the number of CPUs for each cluster, the bandwidth required for paths, and the duration and deadline of the computation through the requirement editor shown in Fig. 4(a). The GRS will automatically assign a concrete cluster for each requirement, but users can also specify some of them manually. From the requirement editor, the user of this system has to input detailed requirements on resources such as bandwidth and the number of CPU at this point. For the future, Grid application service providers (ASP) with knowledge about application programs provide “services”, abstraction of computing, for the end-users. The end-user sends abstract requirements on the service to Grid ASP and the Grid ASP transforms the user requirements into detailed requirements and sends them to GRS for the end-user.

The user requirements are sent to the GRS through the editor and the computing resources and the GMPLS network resources are reserved as the result of inter-working between the GRS and the NRM with SOAP on HTTP. After the advance reservation by GRS is completed, the result shows up on the reservation timetable as shown in Fig. 4(b). The table shows the reservation status of cluster CPUs and network paths. The left-hand side indicates the number of reserved CPUs on each cluster and the right-hand side indicates the network bandwidth

of the reserved paths between the clusters. The center horizontal line stands for the current time, and it is moving and approaching the reservation time.

Fig. 4(c) is the map view showing the current status of the network. The path status shown on the map is taken directly from the network itself by snooping the GMPLS control packets. When the reservation time comes, the allocated lambda paths will be established and shown on the map. Finally, a molecular dynamics simulation is executed using the reserved computers and lambda paths, and the result is will show up on the application panel shown in Fig. 4(d).

3.3. Discussion

From the iGrid2005 demonstration, we learned the following:

- It is important to improve the messaging protocol between GRS and NRM. In the iGrid2005 demonstration, high-latency communication between the GRS and the NRM with SOAP was an acceptable performance for the response time in a demonstration. That is, the duration between submission of a resource requirement from the GUI and display of the reservation results on the GUI was acceptable here. However, user requirements on large-scale Grid resources will cause an enormous number of the SOAP messages back and forth, and thus the response time will increase rapidly. To reduce the number of transmissions, we have to develop a messaging protocol suitable for a large set of resources.
- A lock-and-release reservation process is important for future operational needs. We deployed a single GRS and a single NRM in the first experiment, but *N* GRSs and *M* NRMs will be deployed over the Grid in the future. Under such conditions, GRSs send requirements on network resources to NRMs at the same time and compete to secure the same network resources for their users. This competition tends to cause inconsistency between network resource information discovered by a GRS and actual network resource status. Therefore, we have to prepare a suitable lock-and-release reservation process using temporal reservation and confirmation, which avoids overbooking and over allocation of network resources.
- A debugging methodology has to be developed for the scheduling system, especially for GMPLS lambda path configuration. Global co-allocation scheduling systems, like our prototype system, consist of multiple segments, GUI, GRS, NRM, clusters, and GMPLS-controlled networks, and it is difficult to find the cause of problems. The debugging methodology for lambda path level failures, especially, has not been fully developed. While the map view shown in Fig. 4(c) helped us to find which lambda paths had failed to establish connections over GMPLS easily in the just completed experiment, the method to monitor the GMPLS network we deployed was not scalable and reliable. To address the debugging issue, we should improve and standardize a monitoring method for the lambda path level and more detailed fault management and fault notification procedures will be also required for the GNS-WSI specification.
- The GMPLS lambda paths can be dynamically routed using the open shortest path first (OSPF)-based route calculation method. This functionality of the GMPLS routing protocol enhances the survivability and the resource utilization of the lambda networks. However, this functionality was not effectively utilized for the demonstration, since the conditions of the network latency negotiated between GRS and NRM might change after the routing operation because of re-allocation caused by any failures. As a solution of this issue, the route of the lambda path was explicitly specified and fixed in the GMPLS network. To discuss the feasibility of utilizing the dynamic routing functionality, the significance and tolerance of the network delay need to be clarified from the applications' viewpoint.
- Fault tolerance of resource reservation is a serious issue for the next step. Though advance reservation of

the lambda paths succeeded in our experiments, the path connection at the specified reservation time sometimes failed because of errors caused by bugs of composed modules or actual GMPLS messaging. In this case, the processes of our application program implemented with Ninf-G and GT2 were not able to notice those errors and kept on running and waiting for establishment of their communication. To resolve this situation, we set the start time of the application program some dozens of seconds late and implemented a monitor functionality such that GRS asks NRM if the reserved lambda paths are actually established at the specified reservation time and notifies users through the GUI in case of error. In the future, we have to support fault tolerance of resource reservation and mechanisms to notify users in advance of inevitable errors, and find methods to compensate users if required.

4. Related work

The CANARIE project aims to accelerate Canada's advanced Internet development. CANARIE has deployed CA*net4, an educational lambda path network, in Canada, and has developed the UCLP (User Controlled LightPaths) software[10] that allows end-users to treat network resources as software objects, and provision and reconfigure lambda paths. While the UCLP end-user can select a network configuration directly, our scheduling system automatically selects and configures suitable computing and light path resources for each user requirement on available resources, and supports advance reservation.

The VIOLA project has a similar motivation in co-allocating resources on Lambda-Grids[11]. VIOLA proposes a WS-Negotiation/-Agreement standardized by a GGF[9]-based MetaScheduler for UNICORE-based Grids[12]. We have developed a Grid resource scheduler based on WSRF, and a Web services-based NRM capable of GMPLS network resource management, and we demonstrated this Grid scheduling system at iGrid2005.

AppLeS[13] proposes a scheduling scheme that a scheduler estimates performance of specific applications using prediction about the status of computing and network resources provided by a Grid monitoring system such as NWS[14]. AppLeS provide non-dedicated resources and the estimated performance is not guaranteed.

The paper [15] described co-allocation requirements on Computational Grids and developed a general resource management architecture for the simultaneous co-allocation of multiple resources. The paper[15] does not argue how their co-allocator provides guaranteed network resources. Our contribution is to define a Web services-based network service interface and to realize co-allocation of guaranteed computing and network resources by interworking with NRM, capable of automatic provisioning of lambda paths network over the actual network test-bed.

5. Conclusions

We have defined a preliminary interface between a Grid resource scheduler and a network resource management system, GNS-WSI, and have developed the prototype system, a Web Services-based Grid scheduling system that allows us to co-allocate computing and network resources over Lambda-Grids. We conducted a live demonstration using GRS, NRM, and a GMPLS-based network test-bed, JGN II, at iGrid2005. Our proposed Grid scheduling system is able to provide a suitable set of dedicated computing and network bandwidth resources automatically for each user requirement through the GUI with easy operation, showing the feasibility of the system and its architecture. We also point out some important issues for future resource reservation services. In the future, we will focus on detailed specifications of the GNS-WSI, aiming to make this interface both open, and a global standard.

Acknowledgements

We would like to thank all the people who supported our iGrid2005 demonstration: Yusuke Tanimura, Hiroshi

Takemiya, and Fumihiro Okazaki from AIST, Munefumi Tsurusawa from KDDI R&D Laboratories, Takuya Ohara and Yukio Tsukishima from NTT Network Innovation Laboratories, Shinji Shimojo and Toyokazu Akiyama from NICT, and Masatomo Kobayashi from the University of Tokyo.

References

[1] G-lambda. http://www.g-lambda.net/.

[2] GMPLS. RFC 3945.

[3] Y. Tanaka, H. Nakada, S. Sekiguchi, T. Suzumura, and S. Matsuoka. Ninf-G: A Reference Implementation of RPC-based Programming Middleware for Grid Computing. *Journal of Grid Computing*, 1(1):41-51, 2003.

[4] I. Foster and C. Kesselman. Globus: A Metacomputing Infrastructure Toolkit. *International Journal of Supercomputer Applications*, 1997.

[5] I. Foster. Globus Toolkit Version 4: Software for Service-Oriented Systems. *IFIP International Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779*, pp. 2-13, 2005.

[6] I. Foster and C. Kesselman. The Globus Project: Status Report. *Proc. IPPS/SPDP '98 Heterogeneous Computing Workshop*, pp. 4-18, 1998.

[7] OASIS. http://www.oasis-open.org/.

[8] iGrid2005. http://www.igrid2005.org/.

[9] Global Grid Forum. http://www.ggf.org/.

[10] R. Boutaba, W. Golab, Y. Iraqi, Tianshu Li, B. St. Arnaud. Grid-Controlled Lightpaths for High Performance Grid Applications. *Journal of Grid Computing, Special Issue on High Performance Networking*. 2004.

[11] Ch. Barz, Uni Bonn. Dynamic allocation of network resources in VIOLA. VIOLA workshop, 2005.

[12] M. Romberg. The UNICORE Architecture: Seamless Access to Distributed Resources. *Proceedings of the Eighth IEEE International Symposium on High Performance Distributed Computing*, pp. 287-293.

[13] F. Berman, R. Wolski, S. Figueira, J. Schopf, and G. Shao. Application-Level Scheduling on Distributed Heterogeneous Networks. *Proc. the 1996 Supercomputing Conference*. 1996.

[14] R. Wolski, N. Spring, and C. Peterson. Implementing a Performance Forecasting System for Metacomputing: The Network Weather Service. *Proc. the 1997 Supercomputing Conference*, 1997.

[15] K. Czajkowski, I. Foster, and C. Kesselman. Resource Co-Allocation in Computational Grids. *Proc. the Eighth IEEE International Symposium on High Performance Distributed Computing*, pp. 219-228, 1999.