Polimatica: An Implementation of Policy Automated Provisioning Grid — Foundation of Dynamic Collaboration —

By Takashi KOJO,* Yoshiharu MAENO† and Yoshiki SEO†

ABSTRACT Dynamic resource allocation is one of the critical characteristics required for the back-end systems implementing Dynamic Collaboration. In this paper, we first discuss the requirements on such systems and point out that the policy automation is the key technology. Then, scalable models for dynamic resource brokering of complex large distributed systems are discussed. We implemented basic functions of the models and successfully demonstrated feasibility of the model.

KEYWORDS **Grid, Provisioning, Policy automation, Dynamic resource allocation, Dynamic Collaboration**

1. INTRODUCTION

Dynamic Collaboration is supported by a collective of various types of applications. Some of them are user access oriented interactive applications such as mobile applications or Web portals. Some of them require very high network traffics such as stream video applications. Some may be database oriented while others may require computation-intensive tasks. Implementation of the dynamic collaboration systems requires various unique characteristics on its back-end systems as well as the front-end and network.

In such applications, for example, the system load varies drastically from time to time as user access surges unpredictably, while some applications might have very periodic daily, weekly or monthly patterns of systems load. Respective applications require their own hardware and software configurations based on performance, security or other requirements. For its constantly changing system load, it would not be realistic to assume maximum load on the system configuration of each application. Although dynamic change of the system configuration and resource allocation as its change of system load is strongly required for entire system resource utilization, it usually includes many manual operations, time consuming and difficult tasks. Conventional system administration speed does not meet the constantly changing environment.

Provisioning is a technology based on business process management that can pipeline the operations concerned with those kinds of changes of system configuration and deploys all the software required for the application. Although the technology reduces many aspects of the administration overhead and reduces the total cost of the operations, it still requires significant portions of manual operations that prevent a serious cut down of the lead time for system configuration changes.

Provisioning Grid discussed in this paper is an emerging grid technology which focuses on policy-based automation of dynamic resource allocation, resource sharing and utilization, as opposed to large portion of the grid technology focused on IT resource sharing in the collaborative working environment among multiple organizations. Provisioning grid enables fully automated change of its hardware and software configuration as the system environment varies. It can provide an optimal environment for each application for constantly changing the system execution environment so that the application can meet its own service level requirements from the users, while the entire system can realize maximum resource utilization or very high availability.

This paper describes the required characteristics of the provisioning grid, the internal models the system is based on, and its implementation undergoing by NEC.

2. REQUIREMENTS

In this section, we discuss a variety of requirements for the provisioning grid.

1) Manageable Resources

To ensure a full range of optimization regarding the system configuration and resource allocation,

*System Platform Software Development Division
†Internet Systems Research Laboratories

manageable resources of the grid have to include various layers of the system resources from hardware to application software. At the bottom layer, it has to manage allocation of various types of server hardware, storage, firewall, load balancer and network switches. Appropriate type and version of the operating system has to be installed onto the allocated servers. Middleware such as Web server, application server or DBMS also need to be deployed as the application required. For executing the application, the application program along with its initial data, contents or database schema may also be needed on the allocated system configuration.

All those entities are regarded as manageable resources.

2) Resource Virtualization

The resources should have their interfaces for management. In the provisioning grid context, the interface is virtualized into Web services which expose service ports for the management so that the provisioning grid middleware can manage the resources in a consistent manner as it manipulates other grid services (**Fig. 1**).

The scalability introduces heterogeneity in its nature. A large-scale system distributed among organizations cannot assume homogeneity of the hardware, operating systems or middleware. Standardization of the services is being discussed in several organizations. The most recent attempt is Web Services Distributed Management in OASIS Open.

3) Autonomy and Policy

An application should have its autonomy and a certain level of independence in terms of the resource allocation and its behavior. Those characteristics have to be predefined by business processes and policies so that the application can adequately perform its required task regardless of system environment changes.

The business process is a sequence of the activities that can perform certain aspects of management. The processes are either infinite cycle of activities or a finite sequence that is supposed to be started by events.

The policy is a pair of event and action. An event is any kind of phenomena in the system such as user access, change of the system load or any kind of exception or a familiar's action. An action can be defined as a business process which appropriately handles the event.

4) Scalability

Since the dynamic collaboration tends to extend over a wide range of groups, organizations or enterprises, the back-end system which supports the collaboration should be easily scaled out so that it can meet the service level requirements of the applications. Aspects of the scalability include physical geographic distribution, logical number of organizations, performance to deal with size of users, organizations and a variety of applications.

As the system scales out to a large number of applications, the complexity of the system's behavior and nature also drastically increases. It is usually unrealistic for system designers to predict all the possible system behavior and define the processes and policies that can manage an entire system appropriately on any possible occasion. The grid has to provide adequate means to break down the problem into reasonable pieces. The processes and policies

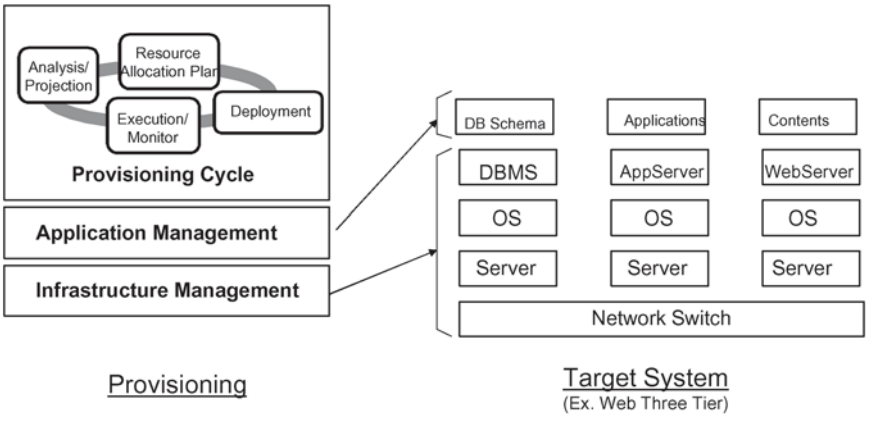

Fig. 1 Provisioning layers.

have to be broken down into local processes and policies so that the system designer can deal with only the local behavior and nature of the system and can break it down to a reasonable complexity of problems.

3. MODELS

This section discusses various models on which the provisioning grid is based.

1) Virtual Organization

VO (Virtual organization) is an isolated virtual work space for applications with which real organizations can share (**Fig. 2**). The model is discussed in OGSA (Open Grid Services Architecture). VO is both a security domain and resource allocation domain. As a security domain, the access from outside to the resources inside the organization is totally prohibited or controlled by policy of the VO.

As a resource allocation domain, VO owns resources allocated from resource pools in Real Organization. The resources are allocated to applications in the VO through appropriate mechanism and policy as it is needed.

2) Network Model

Virtualization of the network devices is a critical issue to minimize the complexity of the resource provisioning in these kinds of systems. VO can also be extended to network virtualization. The provisioning grid implements VO model on top of collection of conventional network devices in which [Globus Tool Kit 1][Globus Tool Kit 2] hides all the details of network device set ups.[Globus Tool Kit 3]

A VO owns isolated network partition. The partition is connected to the outside which is another VOs or public Internet space through a given bandwidth. The partition can be implemented in several different ways depending on the requirement of the network security. The provisioning grid VO is implemented on the partitioned layer two which realizes the virtually most strict isolation from outside of the partition.

In the simple model, the resources inside the partition have equal access each other. However, to address realistic security requirements, we need to implement a more sophisticated model in which cluster of resources have specific connectivity. Three-tier Web configuration is an example of such connectivity.

We have introduced a template model for dynamically creating and managing the configuration. Three-tier Web configuration is a template on which the system can allocate a number of appropriate types of server resources as required.

3) Resource Brokering Model

Resource Broker is a mechanism for dynamically allocating resources owned by the ROs to VOs as they are required. If the system provides a single central brokering mechanism which handles all resource brokering from every RO to every VO, the system scale out brokering policy would become extremely complex and difficult — if not impossible — to design appropriately (**Fig. 3**).

To break down the design scope locally to each RO and VO and simplify the problem, we introduce a push and pull models of the resource brokering (**Fig. 4**). The push brokering model is a resource provider centric model. Resources provided by a provider are brokered among resource consumers. The provider is an RO and the consumer is a VO, in the grid system. The broker has its brokering policy and the resource provider has its resource providing policy while the

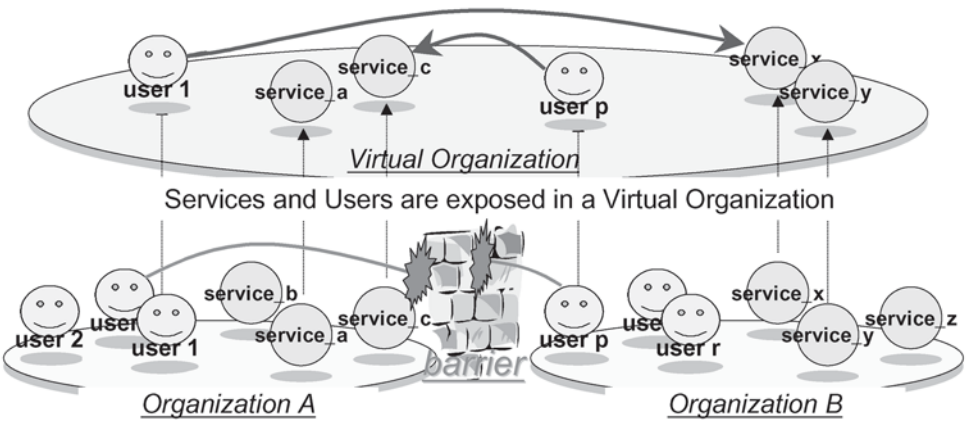

Fig. 2 Real and virtual organizations.

consumers have their requesting policy.

The pull model is the counter model of the push model. Resources required by a consumer are brokered among available resources from multiple providers. As with the push model, each component has its own policy.

A brokering model for multiple requesters and providers can be constructed by a combination of the push and pull model brokers (**Fig. 5**). The advantage of the model is that its policies are localized to VOs or ROs so that the system designer can concentrate on a particular VO or RO for the policy design. Another advantage is that it is purely peer to peer architecture and does not require central mechanism, so that it can avoid scalability bottle neck for large systems that create a large number of VOs.

4) Policy Model

The policy model is another key model for grid implementation. A variety of the models both generic and for specific application are discussed. The event and action policy model is a simple but powerful generic policy model. It defines a set of events and corresponding action to be taken.

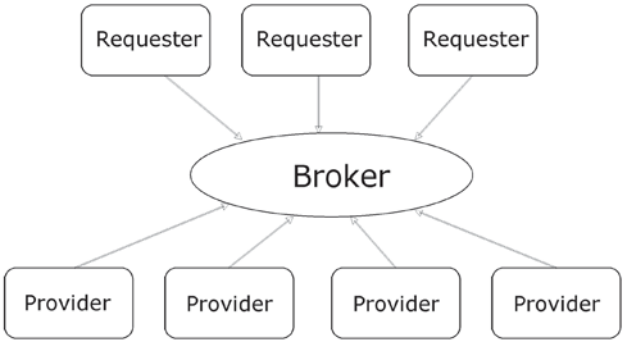

Fig. 3 Single broker model.

4. IMPLEMENTATION

4.1 Functional Configuration

A prototype system has been developed based on the model discussed in the previous section. **Figure 6** shows the functional configuration of the system. The system consists of three layers. The VO management layer implements and manages VOs. The resource provider layer manages resources owned by ROs. The agent layer is for the grid middleware interfacing with physical devices.

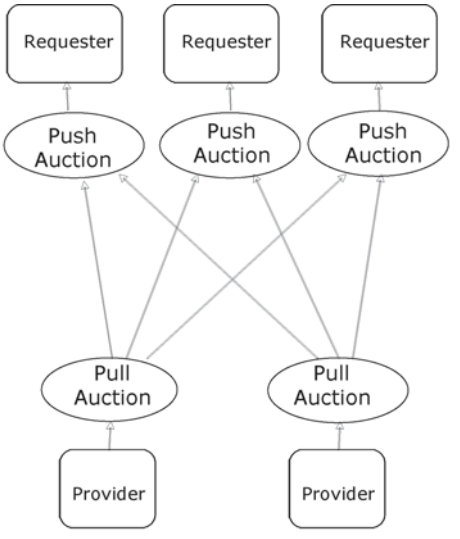

Fig. 5 Combination of push and pull models.

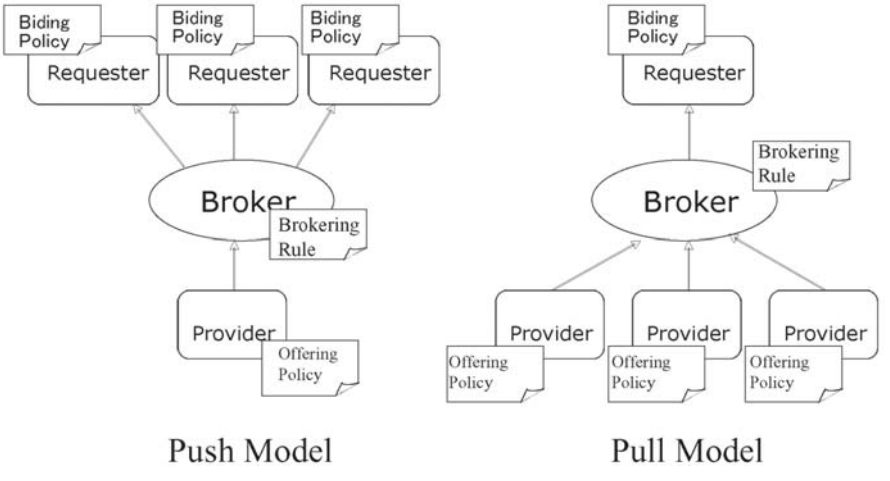

Fig. 4 Push and pull models.

The VO management layer consists of three grid services. Policy automation GS is for the top level process and policy implementing autonomic resource management of the VO. Monitor GS is for monitoring the load of system components. Whenever the load level of a component is changed, it notifies the event to the Policy Automation GS. Broker GS is a requester side broker of the system resources. The current system manages a single RO. The system implements only pull model brokering.

The resource provider layer consists of Resource Monitor GS, Resource Control GS and Mapping GS. Mapping GS virtualizes network device operations into network partitions owned by VOs. As the broker request to add or remove resources in the network partition (VO), mapping GS interprets the request and translates it into a series of commands to the network devices.

4.2 Policy Engine

The policy engine is the core of the policy automation GS. The engine implements event and action type policies in which policy is a pair of event conditions and an action definition. A policy is started if one or more of the event conditions is satisfied and requires corresponding action.

The events have their assigned priority. The engine controls the priority of policy so that, if events are caused by error of a previous action, the recovery policy can override and start the action.

4.3 Resource Monitoring

Resource monitor GS monitor the resource status. They periodically collect the status information from resource control GSs which correspond to respective managed entities. The entities are physical devices or software. VO resource monitor and RO resource monitor are implemented on MDS (Monitoring and Discovery Service). MDS is a service provided by globus tool kit which provides directory services for grid resources. MDS in the VO monitor caches resource information owned by the VO, while that in the RO monitor stores resource information owned by the RO. For implementing the VO based security, a VO monitor is only allowed to access the resources owned by the VO or free resources in the pool.

4.4 Resource Pool

The resource pool is also managed by the resource monitor GS. The GS shows all the available resources in RO to the inquiry by VOs. With current implementation, the resources inquiries are serialized and the resources are allocated to VOs on a first-come first-serve basis.

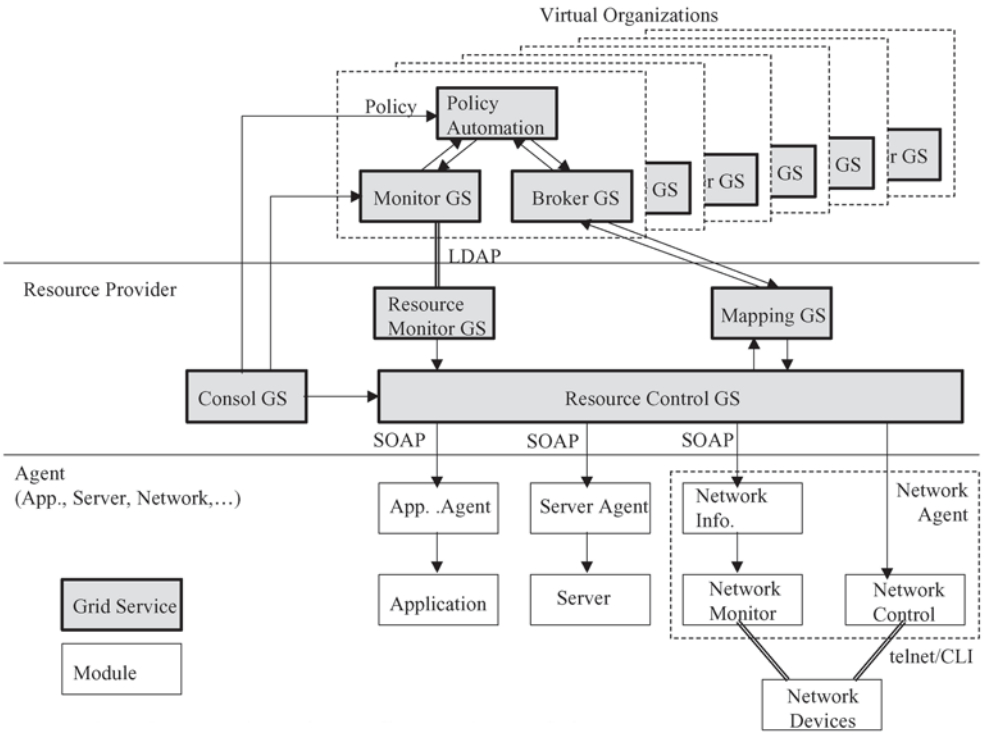

Fig. 6 Functional configuration of the system.

4.5 Technology Foundation

The middleware was developed on top of Globus Tool Kit 3. The grid services are implemented as OGSI services. For availability timing, we used Globus Tool Kit 2 based MDS.

4.6 Concept Proof Demo Applications

To proof the provisioning grid concept, we implemented two applications on top of the grid middleware. The first application is a video stream delivery application. It consumes a number of servers and bandwidth between the servers and client as a request comes from the client. The second application manages the business process. The application does not require a network bandwidth.

Figures 7 and **8** show the physical and logical network configuration of the demo system. The system manages two VOs which have network devices such as packet shaper, firewall and load balancer along with servers. Managing the template configuration for the applications, the system allocates an appropriate number of servers and bandwidths for the applications.

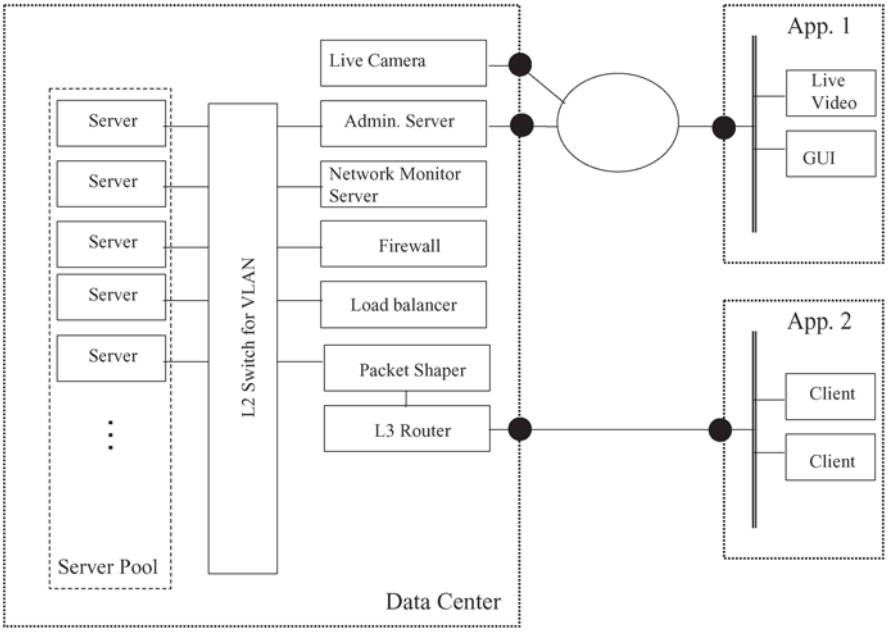

Fig. 7 Physical configuration of the demo system.

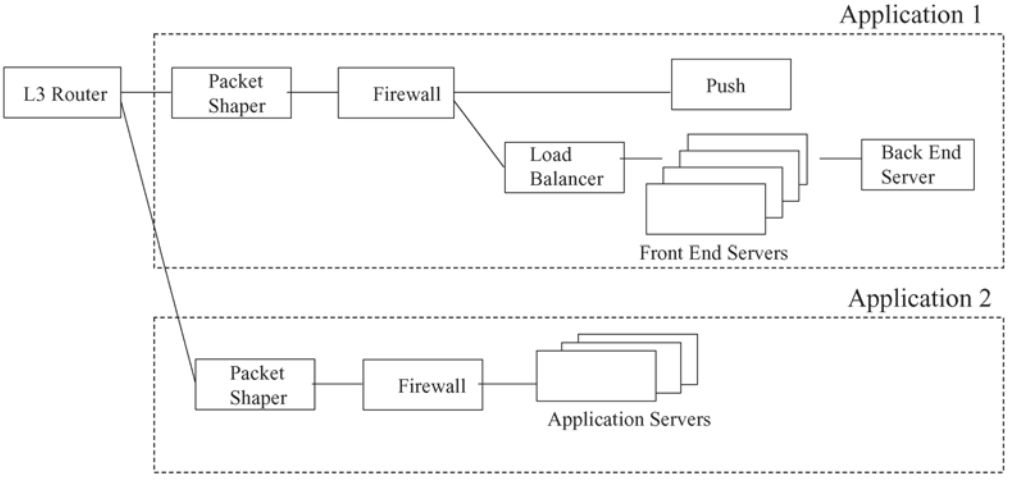

Fig. 8 Logical network configuration of the demo system.

5. CONCLUSION

We have successfully demonstrated that the design concept of the provisioning grid works effectively in a dynamically changing environment. However, the current prototype is a limited implementation, and many of the functionalities remain for future work. The system has been being enhanced toward product level maturity. Mentioned below are some of the future issues we are currently planning to solve.

1) Resource Brokering Policy

In the current prototype, we implemented a minimum set of policies that is just enough to handle simple resource brokering cases. Part of the weakness comes with mediation of the resource request from multiple VOs when the resources are exhausted from the pool. The system requires a more sophisticated policy to judge the priority of the request from multiple VOs.

2) Multiple ROs

Current implementation supports a single RO. It is critical to support multiple ROs for the future deployment of the large scale grid. This will require full implementation of the brokering model discussed in this paper.

3) Dynamic VO Lifecycle

Current Implementation assumes static VOs. For more realistic environment of the dynamic collaboration, VO has to be dynamically created and deleted as required.

4) Resource Reservation

Current implementation assumes resource allocation for immediate use only. It does not support resource reservation. For supporting resource reservation of the future use, ?????(問合せ中)

5) Deployment

Current implementation assumes that all the necessary software is appropriately installed prior to the usage. For large-scale resource allocation, the system will have to support on the fly deployment as the resource is reserved or allocated. The deployment has to support application, middleware and OS layer. The application deployment should include all the contents data, database schema and other registry information as well as application software. It also has to deploy infrastructure software such as OS, DBMS, Application server and Web server.

REFERENCES

[1] I. Foster, D. Gannon and H. Kishimoto, "The Open Grid Services Architecture," Global Grid Forum OGSA working group, http://forge.gridforum.org/projects/ogsa-wg.

[2] S. Tuecke, K. Czajkowski, et al., "Open Grid Services Infrastructure (OGSI) Version 1.0," Global Grid Forum OGSI working group, http://forge.gridforum.org/projects/ogsi-wg.

[3] R. Wolski, J. S. Plank, et al., "Analyzing Market-based Resource Allocation Strategies for the Computational Grid," *The International Journal of High Performance Computing Applications*, **15**, 3, pp.258-281, 2001.

[4] R. Buyya, H. Stockinger, et al., "Economic Models for Management of Resources in Peer-to-Peer and Grid Computing," Proceedings of the SPIE International Conference on Commercial Applications for High-Performance Computing, Denver, 2001.

[5] G. Wasson and M. Humphrey, "Towards Explicit Policy Management in Virtual Organizations," Proceedings of the IEEE 4th International Workshop on Policies for Distributed Systems and Networks, Lake Como, 2003.

[6] "Utility data center: architecture," http://h30046.www3.hp.com/solutions/architecture.html.

Received February 9, 2004

* * * * * * * * * * * * * * * * *

Takashi KOJO is now Chief Manager of System Platform Software Development Division. He is one of the earliest members of Business Grid Project jointly held by NEC, Fujitsu and Hitachi. He is a co-chairi of CDDLM working group in GGF. He has been in charge of many of Web Services related development projects in NEC. He received B.E. and M.E. of EECS, Aoyama-gakuin Univ.

Yoshiharu MAENO joined NEC Corporation in 19xx. He is now Assistant Manager of Internet Systems Research Laboratories.

Yoshiki SEO joined NEC Corporation in 19xx. He is now Senior Manager of Internet Systems Research Laboratories.