

# Dynamic Storage Resource Management Framework for the Grid

Wong Han Min, Wang Yonghong Wilson, Yeo Heng Ng\*, Wang Donghong,  
Li Zhixiang, Leong Kok Hong, Yong Khai Leong  
Network Storage Technology Division, Data Storage Institute, Singapore  
{wong\_han\_min, wang\_yonghong, yeo\_heng\_ngi, wang\_donghong,  
li\_zhixiang, leong\_kok\_hong, yong\_khai\_leong} @dsi.a-star.edu.sg

## Abstract

*In this paper, we consider the design of a dynamic storage resource management framework for the Grid. The framework is service-oriented and conforms to the Open Grid Service Infrastructure (OGSI) specifications. The higher-level framework design is inspired by the Grid and the Peer-to-Peer network architecture, which have been proven to be very effective in distributed environments. The design guarantees a robust way of maintaining the storage resources as it does not rely on any supporting infrastructure. In our design, we included QoS performance metrics as part of our service interface to cater for high-performance Grid applications. At the lower level, we consider the usage of Storage Management Initiative Specifications (SMI-S), which enhance the flexibility and management of heterogeneous storage systems/devices, of various vendors.*

## 1. Introduction

Network based storage systems have proven to be indispensable in various companies and organizations. Unlike direct attach storages, network based storages such as Storage Area Network (SAN) offer a robust and easy way to control, scale, and access large amount of storage resources. However, with the ever increasing number of storage resources and users, future resource management system needs to be much more dynamic and automated. As the industry is embracing the Grid concept, where aggregation of resources is one of the major concerns, storage resource management systems have to take into consideration the issue of scalability and inter-operability. Further, there is also the issue of quality of service (QoS), which is an essential factor for quality sensitive applications such as media streaming and storage utility services.

In this paper, we consider the design of a storage resource management framework, built on top of the Grid foundation that is highly dynamic, easy to administer, scalable and inter-operable. The framework is service based, designed as recommended by the Open Grid Services Architecture (OGSA) standard and conforms to the Open Grid Service Infrastructure (OGSI) specifications. The design separates the management and serving components of the service such that large amount of servers can be administered within a domain without any issue of performance bottleneck or massive network overhead. On a broad perspective, the framework consists of multiple domains with generally independent administration. Each of these domains may be owned by the same or different service providers which may choose to work independently or cooperatively. The framework design is inspired by the Grid as well as the Peer-to-Peer network architecture, which have been proven to be very effective in maintaining inter-operability in distributed environments. Further, the design guarantees a robust way of maintaining the storage resources as the framework utilizes distributed control mechanisms and does not rely on any supporting infrastructure. Finally, at the lower layer of our design, we consider the usage of Storage Management Initiative Specifications (SMI-S) [1] of the Storage Networking Industry Association (SNIA) with the objective to standardize interoperable storage management. The SMI-S based components of our framework simplify the management of heterogeneous storage systems/devices, of various vendors. These devices can be added, replaced, or even migrated from one server to another with minimal or no reconfiguration at all.

In our design, we consider both the highly dynamic environments of the Grid as well as the performance centric environments of storage resource providers. Upon requesting a storage resource, users specify their respective requirements such as storage capacity, preferred transfer protocol, and QoS performance metrics. The service manager will then locate the appropriate

---

\* Primary contact author

resource to match the client requirements. As mentioned, although the domains can be administered independently, multiple domains may choose to operate cooperatively. In the cooperative mode, user from a domain may utilize any resource within the Grid. Further, users and servers are allowed to 'migrate' from one domain to another, based on pre-determined policies, to further improve utilization efficiency and performance. Depending on the availability of the underlying connections, our design also allows storage systems/devices (i.e. fiber-channel connected disk array) to be 'migrated' (utilizing the SMI-S features) from one server to another for any performance related issue.

The organization of the paper is as follows. First we present our work in Section 2. Some of the feasible applications of our dynamic storage management system are highlighted in Section 2.1. Then, the overall architecture and services are discussed in Section 2.2 and Section 2.3 respectively. The details on individual components of our design are discussed in Section 2.4. As an important factor to be considered in all resource management related topic, we present our resource discovery scheme in Section 2.5. After this we present some related works in Section 3. Finally the conclusions of our work and future work are discussed in Section 4.

## 2. Our work

In this paper, we propose an intelligent dynamic storage resource allocation service framework for the Grid environment. The base of our framework is supported using the existing Globus environment, which covers the all the basic functions such as identification, and secure communication between our framework components. At the bottom level, we utilize SMI-S based agents for the management of the storage resources attached. SMI-S provides the essential tools for the management of heterogeneous storage resources with features such as health monitoring, zone configuration, and logical volumes administration. These and other features considerably simplify the management and increase the flexibility in utilizing the storage resources.

In our framework, we consider utilizing the peer-to-peer (P2P) network model at the higher-level framework design. The P2P system approach provides various advantages in terms of flexibility, scalability, reliability and efficiency. Domain may choose to function cooperatively or independently by joining or leaving a Grid. Further the design also increases the fault tolerance level of the overall system functionality, as the system is not affected by any means of single point failure.

At this stage, it may be noted that, in this paper, the resources that we are taking into consideration is of the storage space resources and not the data stored (within the storage) as resource. Although it may be important to have

a concrete management system for managing data (as resource), our storage space resource management adopt a much flexible approach. By having servers that support multiple protocols, our framework essentially serves as the basic foundation for any kind of data management system. Further information on our multi-protocol supporting servers is presented later section of the paper.

### 2.1. Applications

In this section, we shall highlight some of the feasible applications of our dynamic storage management system. The primary application for our framework is of personal or private usage as such applications are able to exploit the highly dynamic nature of our framework. However, it does not prevent the framework to be used as a platform for data sharing purposes in a public storage pool fashion.

In personal or private usage, the proposed storage framework maybe most useful in aid of Grid computation applications. Although, currently most Grid computation nodes usually come with relatively enough storage space for majority applications, having a large dynamic storage has its advantages. For example, in the field of Bioinformatics, it maybe required to download data from public databases, which is required for the bioinformatics computation. However, such databases may be located geographically far from these computational nodes. At the same time, there may have large number of users accessing the databases. In such cases, the computation process may be greatly affected due to the lack of data supply from the public databases. As such, it is necessary to have a storage (for the date) that is dedicated and has relatively good performance. Hence, as the computation process is scheduled to run, large amount of storage space can be requested to store all necessary data. These storage spaces should be relatively close to the computing nodes and it can cache the contents of the databases in advance as to avoid any computing node being left idle while waiting for input data from the database.

The dynamic nature of our framework can also be used as a supporting service. For example, in a Video-On-Demand (VoD) service, probably one of the most difficult problems is to efficiently utilize all supporting infrastructure or hardware. This is due to the fact that the number of customers and the popularity of any particular video fluctuate heavily over time. To overcome such problems, the VoD service providers may require the use of temporary storages. Temporary storages, managed by the proposed framework in this paper, may solve the problem in various different ways. One of the possible solutions is to generate replicas of the popular videos on the temporary storage (preferably near a large cluster of customers), which will be used to serve the customers. When customers log into the primary VoD server, the

servers may then redirect the customers to the most suitable temporary storage to retrieve the video. Other than able to provide temporary storages dynamically, our multi-protocol enabled servers can be of beneficial to such services as most of them may utilize proprietary transfer protocol. Such protocol maybe installed manually in the storage servers or can be done dynamically by means of agents on the servers.

Although the dynamic nature of our framework contributes mostly on temporary storage management, our system can also be used as permanent data storage by requesting indefinite usage period. Having online storages has various advantages, such as a hassle-free data backup storage solution, pervasive storage and single data-image. However, online storage will only be on its full potential only if it is able to share data. Although our framework does not consider any structure data sharing features such as content discovery and meta-data indexing, our framework approach the data sharing requirement in a much flexible manner.

Data sharing architecture such as the SRB [2] has it's predetermine algorithms and methods in data contents discovery and sharing. However, in our framework, we consider the essential foundation where sharing applications can be installed, i.e. the multi-protocol supported servers. Consider the case where a group of engineer went abroad to troubleshoot a client's massive production line that demands an efficient interaction between hardware and software. Each of these engineers has their very own expertise and responsibilities. However, due to the scope of the job, they are required to work closely together, where every modification made are required to be known by all members. The engineers may choose to work in small (closely related) groups and meet up to exchange vital information on prescheduled meetings. However, with our proposed framework for online storages, the engineers can then work independently on their free time and have all vital information stored online. These engineers may choose to use their familiar GridFTP to share data among the members as it only requires one time authentication for both their computation and sharing needs. On the other hand, if more than one engineer is required to modify the same data or software code, they may choose to use other protocols such as network file system (NFS) that provides file locking mechanism to avoid data inconsistency or data lost due to data overlap by other engineers.

The above scenario highlighted the advantage of our framework in terms of flexibility where framework actually adapts the way the people work rather than having to adapt the system. In large scale data sharing application, similar approach can also be used. A simple approach would be to allocate the storage accordingly and install resource sharing applications such as FreeHaven [3] in the client machines to construct a large data sharing

network. Another much effective way would be to introduce new applications/protocols or modification of existing applications, as part of the transfer protocol that is supported by the storage servers. By having protocol/software at the servers, the proposed framework can then be transformed into a large scale data sharing platform with all the features (such as data indexing and discovery) provided by the protocol itself. With the support of multiple protocols, the system can be used as a platform for sharing data resource, it still able to support on-demand personal storages.

## 2.2. Architecture

In a broad scope, our service framework components can be generally divided into manager and server as illustrated in Figure 1. The manager component is solely responsible for the administrative and management of the storage resources while the server will be directly involved in the management of resource allocation and access. The clients, using the Globus standard authentication service to authenticate themselves, will request the resources from the manager. After reaching an agreement (either manager satisfied the client requirements or client is satisfied with alternatives that the manager offers), the manager will provide instructions to the servers to allocate the resources and serve the respective clients. Any traffic (of the same resource request) there after will bypass the manager and forwarded directly to the server. Such an approach significantly lowers the amount of network overhead, avoids performance bottleneck, and keeps the manager free to perform other administrative functions. This also simplifies the tasks to be performed at the client side. Further, as many existing data transfer protocols do not support Grid environment, clients may use native authentication methods to access the resources directly with the server, rather having to go through different levels of authentication. It may be noted that, the clients considered in this paper can be generally classified into two categories, (a) user and (b) service or application. User is relevant to the person at the final receiving end of the service, while service/application is relevant to the middle person whom utilizes our service to serve another party (user or service/application). In this paper, without lost of generality, we consider both user and service/application as client.

Each manager is solely responsible for its respective group, which comprise of a set of clients and servers. We refer such groups as domains. Clients from a domain are only required to request resources from their respective manager (within the same domain). In a Grid with more than one domain, managers may query each other for resources and clients are allowed to utilize resources from

other domains. Members within a domain may be determined initially by means of geographic location, IP mask or Globus domain. However, at later stages, members (except the manager) of a domain may be 'migrated' to another domain for any performance related issue. This approach introduces another level of flexibility on the resource utilization while taking into account of the overall efficiency. For example, in cases where clients are satisfied with the resources within their domain, overhead network traffic (i.e. inter domain query messages) for the allocation process can be extensively reduced. On the other hand, if the client requests are often satisfied by other domains, the client may be 'migrated' to the domain that has the highest rate of success in order to reduce the respond time as well as network traffics.

It may be noted that at this stage, it can be seen that the manager is the single-point failure for each domain. However, since in our design, the administrative and serving components of our system are discrete in nature, failure on the manager will not affect allocated storage. Failure on the manager will still affect the utilization efficiency of the servers in the domain. As such, as a countermeasure for such scenario, we consider a dual-manager system where each server and client will have an active and backup manager. Upon the failure of the active (default) manager, the server and client will automatically locate the backup manager and have it to be the active manager while the backup manager will recommend another backup manager.

### 2.3. Services

Clients submit their request to the manager by means of descriptions of their requirements such as usage time period, amount of resource, and the data transfer protocol. Advance reservations on storages are also allowed to increase the overall system efficiency. In our framework, clients may also specify QoS parameters such as the data throughput, and seek time/latency. However, we omit the component for satisfying the end-to-end network QoS. This is mainly due to the fact that end-to-end network QoS is generally difficult to implement and satisfy as end-to-end network path usually consist of components from various services/providers. However, in the event that end-to-end network QoS can be guaranteed (i.e. within a relatively small LAN or network utilizing high bandwidth optical links) a network resource broker can be easily integrated into our framework to establish a QoS network path before allocating and accessing any resources.

In our framework design, network QoS is only taken into consideration at the edge of the service by controlling the flow of traffic at the server end. For example, a server may limit the throughput of normal users while allowing privilege users to have higher throughput. In our

framework design, we consider the use of network traffic-monitoring component such as the Network Weather Service (NWS) [4], which will be used to monitor the traffic in between the client and the server. This is mainly for the purpose for providing network traffic information in aid the client making decision on QoS requirement.

### 2.4. Framework Components

In this section, we shall present in details the major components of our framework. Our framework design can be generally divided into 5 major components, client agent, Quality of Storage Service (QoSS) manager, resource catalog, storage resource server, and storage resource. The relations of these components are illustrated in Figure 1. We shall now present each of these components in details.

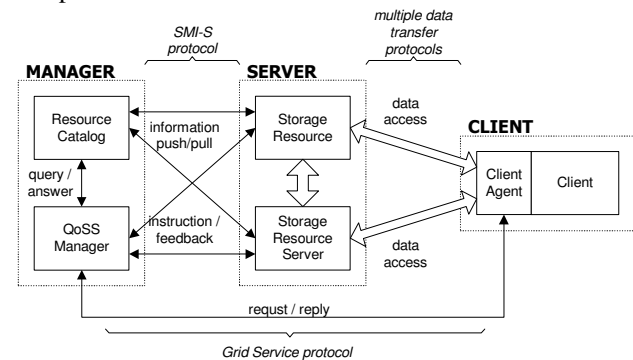


Figure 1. General framework.

#### 2.4.1. Quality of Storage Service (QoSS) Manager.

The QoSS Manager is the major focal point of our overall system framework design. Each QoSS Manager is the manager of their own respective domain, which consists of one QoSS Manager, set of clients and set of Storage Resources Servers. A Grid may consist of more than one domain and clients and storage resources servers within a domain maybe 'migrated' from one (domain) to another for load balancing or other performance related purposes. In the event excessive number of clients/traffics, a domain maybe divided into smaller domains by introducing additional QoSS Managers, to avoid any performance bottle neck issues. On the other hand, lightly loaded domains may be joined together to increase overall utilization efficiency. Each QoSS Manager has a list of all the QoSS Managers within the Grid. They (the managers) are responsible for keeping this list up-to-date throughout the Grid by percolating the information to other managers such that it is sufficient for a new QoSS Manager (or domain) to join in the Grid by registering to any one of the existing QoSS Managers in the Grid. Upon receiving a request, the manager will decide (based on the predetermined policies) the suitable resources based on

the client's requirements. In the events where the client's requirements cannot be satisfied within the local domain, the QoS Manager will then resolve to external resources by invoking the resource discovery process. Depending on the result of the resource discovery process, the manager may list out the available choices for the client or negotiate with the clients and offer other available alternatives. Upon reaching an agreement, the manager will send the appropriate resource and QoS parameters to the respective storage resource server (or external QoS Manager) for resource allocation and QoS management. It may be noted that a client may utilize storage resources from multiple domains but only request for resources from their respective QoS Manager. Similarly, storage resource servers may serve clients from multiple domains but only respond to instructions from their respective QoS Manager. This simplifies the tasks on clients as well as lowers the overall system requirements on the servers.

**2.4.2. Resource Catalog.** Closely attached to each QoS Manager is a resource catalog. The catalog is mainly responsible for registering all the storage resources within the same domain (i.e. storage resources managed by the QoS Manager it is attached to). Nevertheless, it may be used to store replication of catalogs of other domain in aid of storage resource discovery. Such design minimizes the communication overhead (for resource updating) while maintaining high efficiency in resource discovery. Further, by 'migrating' clients to domains with higher success rate in allocating resources, the resource discovery effort as well as communication overhead may be reduced further. In our design, only the QoS Manager and Storage Resource Servers of the same domain can access to the resource catalog. While the QoS Manager only makes queries for available resources, the Storage Resource Servers are mainly responsible for updating the catalog on the available resources and other QoS related attributes such as the average bandwidth usage and system utilization. Future performance prediction may also be updated to the catalog by other components such as NWS. Such information will be used by the QoS Manager in making decision on resource allocation with performance and load balancing issues taken into consideration. We adopt a pull-based approach where the information is "pull" by the catalog from the server in an on-demand basis. This ensures a higher data accuracy while lowering the overall network overhead. However, the Storage Resource Servers may implicitly update the catalog upon any triggering event at the storage resources. For example, in the event of a failure to any storage device, the respective storage resource server would autonomously update catalog.

**2.4.3. Storage Resource.** Each storage resource server is responsible for a set of storage resources. These storage

resources may be a mixture of disk and tape storages from direct attached storage, NAS, SAN, etc. Similar to that of the QoS Manager, the storage resources may be 'migrated' from one server to another depending on the connection availability (i.e. by configuring the NAS or network router), for the purpose of load balancing or other performance related issues.

**2.4.4. Storage Resource Server (SRS).** All storage resources are connected to a storage resource server. These servers provide the storage resources to the clients by means of storage virtualizations. They support multiple data transfer protocols, both file-level (i.e. GridFTP) and block-level (i.e. iSCSI). Any other means of protocol can be installed/setup manually or by means of scripts. The server is also responsible for all the I/O operations in between clients and storage resources. These operations are monitored (in terms of QoS performance metrics) and may be controlled based upon the pre-arranged QoS parameters (by the server), depending on the features of the underlying protocol. The storage resource server is also responsible for allocating/managing storage resources for the client upon receiving instructions from the QoS Manager. In cases where storage resources involved configurable intelligent storage disk arrays, the storage resource server will be able to configure (i.e. build/destroy of RAID storage) the storage to maximize the overall storage utilization. This feature will be performed by equipping the server with SMI-specification (SMI-S) based agent, which is responsible for all management procedures on all SMI-S enabled storages. This further enhances the scalability of the server by allowing unified managements and zero configurations (at the server and storage) for any SMI-S enabled storage devices (from different vendors) to be attached to it. Further, the server may also equipped with CIM object manager, which facilitates the agent to perform more complex operations that may be of vendors' legacy or proprietary functions.

**2.4.5. Client Agent.** Residing at each client is a client agent, which is responsible for any client related operations. These include the communications (with the QoS Manager) and performance (QoS) monitoring. As mentioned, clients can be 'migrated' from one QoS Manager to another in respond to any failure recovery or performance related issues. As such, the client agents are required register/update the attached QoS Manager. Further, upon any successful storage resource allocation, the agents may require to transcribe information provided by QoS Manager in order for the client to access the allocated storage resources. Depending on the data transfer protocol used, the clients may access the data directly from the server with or without going through the client agent. The client agents can also be used for monitoring the client-side QoS performance and may

recommend any QoS related issues to the clients/QoS Manager for the current or future resource allocation. For example, it may not be fair or efficient for a client to request a high throughput storage service when the client side network is not able to efficiently support the given throughput.

## 2.5. Resource Discovery

In this section, we shall describe the resource discovery mechanism, a vital component, which will be used in our resource management framework. It may be noted that our discovery system is designed mainly for the storage resource discovery and not the discovery of data within the storage. However, as the SRS supports multiple transfer protocol, discovery or metadata handling of data within the storages can be done by means of having customized protocol installed within the Storage Resource Servers. For a fast and simple implementation, the storage can be allocated as per normal while sharing software and applications can then be installed to the clients system.

In our framework, the resource discovery mechanism will be used in the event where the client's requirements cannot be satisfied within the local domain. In designing any resource discovery system, it is vital to minimize the overall cost of query in all possible ways such as the cost of supporting system, computational time and network bandwidth. Further, in the context of this paper, the discovery system needs to be scalable and able to handle dynamic systems and resources that may vary over time. Vast amount of works have been done for resource discovery systems. The work done in [5] utilizes a hierarchy structure approach, similar to [6], that requires specialized supporting servers that record/summarize resources available in all the sub-domains. Such systems are generally scalable but require additional supporting servers at different hierarchy levels at the cost of additional hardware or skewed workload on existing servers. Further, consider the dynamic nature of the entire system, any failure or departure (leaving the network) of any important server might temporary impair the system resource discovery system. As such, in the designing of our resource discovery framework, we consider utilizing Peer-to-Peer (P2P) content discovery methods in aid of our resource discovery system.

The simplest would be the flooding technique, which propagate the query to all the servers with a time-to-live (TTL) value to control the scale of search. Such greedy approach is simple but very inefficient. Yet another approach would be the Content-Addressable Network (CAN) [7] where hash tables are used to map contents into network addresses that can be located easily. This method is very efficient in locating contents by one specific 'name' or attribute. Nevertheless, in the context of this

paper, the required resources may be described by more than one attribute. For example, a client may be looking for a storage resource with 10TB capacity, available for period of a week, and supports the GridFTP protocol. A simple way to solve this problem may consider the use of multiple CAN overlapping each other and query each level independently for a complete match. Clearly, such an approach will generate tremendous amount of traffics upon a query or an update.

In our framework, we consider the use of discovery mechanisms that are able to support multiple attributes searching such as the Meteorograph [8] and the content discovery system (CDS) proposed in [9]. However, slight extensions to the existing methods may be required to optimize the discovery of resource with attributes of multiple choices or delimited range of choices. For example, in the context of this paper, a client may request for storage with local cache of more than 2GB and supports either the iSCSI or GridFTP protocols. As such, for our discovery framework design, we consider a P2P like discovery approach with addressable contents that supports query for a set of attributes, which may consist of multiple choices, or delimited range of choices. The discovery system will also maintain the individuality of each domain such that the majority work can be done locally without resolving to resources of external domains.

## 3. Related work

There are various storage resource management related works in the literature with similar objectives. The work in [10] offers an intelligent middleware, Storage Resource Manager (SRM), which handles file transfer operations in between storage resources and clients. The SRM offers a variety of unique file transfer related functions such as dynamic space allocation, data caching and data streaming. While SRM is able to manage user resource quotas, the SRM is not a complete storage resource management system, which has the essential functions such as client management, resource discovery and negotiation. The SRM can be seen as a front-end of resources (similar to that of GridFTP) under a management system such as a request executor, an example given in the paper [10].

The Storage Resource Broker (SRB) concept, proposed in [2], offers a centralized resource management middleware for the data grid. The SRB is a complete set of system that focuses on the issue of data sharing among the grid members. Although SRB still performs functions such as storage allocation, the clients are assumed to utilize the storage in a write-once read-many fashion. As such, the system is more involved in data management such as data storage indexing and replica discovery. SRB is very well suited to perform a digital library function as

it keeps track of all the data and allows clients to perform query to locate the required resource. However, the system does not consider issues that may occur on day-to-day storage resource utilizations such as storage resource discovery, users' quota and access management.

The OceanStore infrastructure [11] provides a maintenance-free global data storage solution that guarantees to scale almost infinitely. Nevertheless, this work are more concerned with the sustainability of the framework rather than the dynamic properties such as on-demand storage allocations, which are essential for maintaining high efficiency in today's dynamic computing environments. A similar work done in [12], presented the Darwin resource management mechanism, which concentrate on the issue of dynamic resource allocation. The mechanism is a service-based system that integrates the management of different resources by utilizing a virtual mesh illustration method to generate a customizable abstract of available resources. However, the centralized management design is prone to single point failure and also inefficient in terms of scalability.

The work done in [13], with scalability issue taken into account, considers the aggregation of different services within grid environments. Similar to the work in [12], this work considered a framework design, dedicated to grid, which provides single virtual end-to-end service by combining different type of resource services. These aggregation services attempt to match the client requirements against all the available resource services. However, these resource services may not attempt to adjust their own resources effectively according the client requests. For example, a storage service with two 10 terabyte (TB) partitions will not autonomously combine these two 10TB partitions to serve a 15TB request. Hence, in order to provide an effective overall service, the individual base resource services have to be more intelligent.

#### 4. Conclusions and Future work

The Grid application supporting layers can be generally divided into the fabric, connectivity, resource, collective, and application layers [14]. In this paper, we presented the design of a dynamic storage resource management framework, which fits in the collective and resource layers. The framework consists of two major components, the manager (at the collective layer) and the server (at the resource layer). The manager is responsible for coordinating the usage of multiple storage resources while the servers are directly involve in serving the storage space via multiple data transfer protocols. This is to ensure the simplicity of the overall resource management while ensuring scalability and interoperability. At a broad scope, our overall design

emphasize on collaborative environments while maintaining the individuality of each domain. Hence the domains are able to manage their own local policies (such as user quota and charge rate) without neglecting the Grid concept of resource sharing and problem solving in multi-institutional virtual organizations [14].

The details of the components of our framework were presented in Section 2.4. In this section, we highlighted the responsibility of each components and how all the components work together to provide a seamless storage management framework. We equipped the Storage Resource Servers with SMI-S based agents, which allow complex tasks to be performed such that it significantly increases the utilization efficiency of the resources.

In Section 2.5, we proposed the resource discovery mechanism of our framework that is inspired by the robustness of the content discovery methods used in P2P networks. These mechanisms do not rely on any supporting infrastructure and hence the entire system will not be impaired by any means of domain failure. Further, due to the individuality characteristic of the entire discovery mechanism, majority works can be done locally without inducing external network overheads. Finally, the flexibility of the mechanism also allows domains/services to join into or depart from the Grid without affecting the operability of the other domains.

The framework presented in this paper can be used as the foundation layer for a storage resource management service. Any further higher-level features and functions can be easily built on top of it. Further as this and many other works are service based, our framework may be combined with others to enhance the overall functionality. For example, our framework may incorporate the SRM [10] to include advance transport level features such as file streaming or may be incorporated into the G-QoS [5] to collaborate with other resources services such as computing services.

As for future development, we have considered implementing the above design to a Grid environment and may work together with a particular Grid application such as Bioinformatics.

#### References

- [1] Storage Management Initiative – <http://www.snia.org/smi/home/>
- [2] Storage Resource Broker – <http://www.npaci.edu/dice/srb>
- [3] Roger Dingleline, Michael J. Freedman, and David Molnar, "The Free Haven Project: Distributed Anonymous Storage Service", *Proceedings of the Workshop on Design Issues in Anonymity and Unobservability*, July 2000.

- [4] Rich Wolski, Neil Spring, and Jim Hayes, "The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing", *Journal of Future Generation Computing Systems*, Volume 15, Numbers 5-6, pp. 757-768, October, 1999.
- [5] Rashid Al-Ali, Omer Rana, David Walker, Sanjay Jha and Shaleeza Sohail, "G-QoS: Grid Service Discovery Using QoS Properties", *Computing and Informatics Journal*, 21 (4), 2002.
- [6] Dongyan Xu, Klara Nahrstedt, Duangdao Wichadakul, "QoS-Aware Discovery of Wide-Area Distributed Services", *Proceedings of IEEE/ACM Int'l Symposium on Cluster Computing and the Grid (CCGrid 2001)*, Brisbane, Australia, May 2001.
- [7] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker, "A Scalable Content-Addressable Network", *Proceedings of ACM Special Interest Group on Data Communication (SIGCOMM)*, August 2001.
- [8] Hung-C. Hsiao and Chung-T. King, "Similarity Discovery in Structured P2P Overlays", *Proceedings of International Conference on Parallel Processing (ICPP 2003)*, October 2003.
- [9] Jun Gao and Peter Steenkiste, "Design and Evaluation of a Distributed Scalable Content Discovery System", *IEEE Journal on Selected Areas in Communications (JSAC)*, Special Issue on Recent Advances in Service Overlay Networks, 22(1), pp. 54-66, January 2004.
- [10] Arie Shoshani, Alexander Sim, and Junmin Gu, "Storage Resource Managers: Essential Components for the Grid", *Grid Resource Management: State of the Art and Future Trends*, Edited by Jarek Nabrzyski, Jennifer M. Schopf, Jan Weglarz, Kluwer Academic Publishers, 2003.
- [11] John Kubiawicz, David Bindel, Yan Chen, Steven Czerwinski, Patrick Eaton, Dennis Geels, Ramakrishna Gummadi, Sean Rhea, Hakim Weatherspoon, Westley Weimer, Chris Wells, and Ben Zhao, "OceanStore: An Architecture for Global-Scale Persistent Storage", *Proceedings of the Ninth international Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2000)*, November 2000.
- [12] Prashant Chandra, Yang-Hua Chu, Allan Fisher, Jun Gao, Corey Kosak, T.S. Eugene Ng, Peter Steenkiste, Eduardo Takahashi, and Hui Zhang, "Darwin: Customizable Resource Management for Value-Added Network Services", *IEEE Network*, Number 1, Volume 15, January 2001.
- [13] Rashid. J. Al-Ali, O. F. Rana, and D. W. Walker, "G-QoSM: A Framework for Grid Quality of Service Management", *Proceedings of the UK e-Science Programme All Hands Meeting*, Nottingham, UK, 2-4 September 2003.
- [14] Ian Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations", *International Journal on Supercomputer Applications*, 15(3), 2001.