QoS Provisioning Framework for an OSD-based Storage System*

Yingping Lu
University of Minnesota
lu@cs.umn.edu

David H. C. Du
University of Minnesota
du@cs.umn.edu

Tom Ruwart
IOPerformance Inc.
tmruwart@ioperformance.com

Abstract

Quality of Service (QoS) is crucial for certain applications such as multimedia. As the Object-based Storage Device (OSD) protocol emerges as the next generation storage technology, QoS provisioning for OSD-based systems has also received a great deal of attention. In this paper, we propose a QoS framework for OSD-based storage system that integrates both the network QoS and storage QoS. We examine the existing OSD specification and analyze the QoS requirements for applications on OSD clients. Based on the QoS requirements analysis, we propose a three-level QoS specification. We further elaborate on extensions to the existing OSD and iSCSI protocol to support our QoS specification. These extensions are then incorporated with the current OSD reference implementation. Finally, we discuss both the implementation structure and issues encountered as part of this study.

1. Introduction

With the ubiquity of TCP/IP networks and the increased popularity of network applications, Quality of Service (QoS) has been extensively studied. QoS is crucial for real-time applications running on a network, such as streaming video and voice over IP, which demand certain guarantees of network bandwidth and delay variance. Generally speaking, QoS is an end-to-end issue, i.e. from application to application [8]. The QoS enforcement involves transport, network and end system. There has been plethora of QoS research in the TCP/IP network transmission domain [9-14], as well as disk scheduling in end-systems [15-25] to satisfy different QoS requirements in storage data transfers. However, these two aspects of QoS (network QoS and storage QoS) are often studied separately [8]. For example, the storage scheduling schemes rarely consider the effect of network's condition.

* This work is supported by StorageTek, Veritas, Engenio, and Sun Microsystems through their memberships in the University of Minnesota Digital Technology Center Intelligent Storage Consortium (DISC).

Recent advances in storage technology have brought about the Object-based Storage Device (OSD) protocol [1][2]. The Storage Networking Industry Association (SNIA) OSD Standard [1] and the Lustre Project [3] represent two OSD-centric efforts. The first deals with the development of a standard OSD protocol that is part of the SCSI standard and implies interoperability with any standard SCSI disk drive that implements the OSD SCSI command extensions. On the other hand, Lustre, is focused on the development of a file system that communicates with object devices rather than traditional block devices over an OSD-like protocol. This is an important aspect of the evolution of OSD as a protocol as Lustre represents an "application" that can fully utilize an OSD hence making it a good platform for testing and validating OSD concepts such as QoS.

Compared to traditional *block-level* data access, OSD offloads storage management functions (such as space allocation) from a traditional file system to the object device, and consequently offers *object-level* data access to its clients. As a result, the OSD protocol imposes new challenges and opportunities for QoS provisioning to its clients.

In most of the previous QoS studies it was assumed that the server had a *dedicated* storage system with a direct I/O connection. In this study, we assume the OSD device is *shared* by a number of clients connected through network rather than directly attached to a single server. Furthermore, not only will each server have *different QoS requirements* for their object access, but each server and/or object access may also have quite *different network access characteristics*, For example, each server and/or object device connection may have a different data path, thus implying different available network bandwidth, variable delay characteristics, … etc. The *intelligence* embedded on object devices can be used to assess the QoS (and other) requirements of its data objects, monitor the current condition and capacity of storage devices, and to measure the existing networking conditions on its interface to its clients. The focus of our study is to explore the *integration of both the* storage *QoS and network QoS* for QoS provisioning and guarantees in this new environment.

Proceedings of the 22nd IEEE/13th NASA Goddard Conference on Mass Storage Systems and Technologies (MSST 2005)
0-7695-2318-8/05 $20.00 © 2005 IEEE

In this paper, we propose a QoS framework for this OSD-based storage system. We first analyze the QoS requirements in an OSD-based storage system. Based on the analysis of these QoS requirements, we present a three-level QoS specification. We then propose extensions to the existing OSD and iSCSI protocol to address the QoS specification requirements. After addressing the QoS specification, we investigate integrating the QoS framework within an OSD. We discuss the framework components and challenges encountered in this integration effort. To demonstrate the effectiveness of the proposed framework, we are working on a reference implementation of the SNIA OSD protocol with the incorporation of the proposed extensions and framework. Finally, we discuss both the implementation structure and issues encountered in this study.

2. Data Access QoS Requirements in OSD Storage System

In this section, we discuss the data access QoS requirements in an OSD storage system. We begin with a brief description of the OSD Storage Architecture and follow with a more in depth description of the requirements as they relate to QoS.

2.1. OSD Storage Architecture

Figure 1 shows a typical OSD-based storage configuration that includes both an OSD hardware infrastructure and OSD-based file system software. The system in general consists of three main components: Clients (a.k.a servers and/or initiators), a Metadata server, and OSDs. Clients initiate access requests to object storage devices. The Metadata server makes the connection between human-readable file names and the objects that compose those files. Its primary responsibility is to maintain metadata information (such as creation time, size, location, security, authentication and access control etc.) for the files and the associated objects in the OSDs. It decouples the metadata processing (control information) from regular data access to improve client data access performance. In other words, no file data travels through the Metadata server. All data transfers are directly between the client and the OSD containing the object of interest. The OSD stores data objects and provides object data access (read/write) to its client. Normally, these components are connected through a TCP/IP network but they can also use other networks as required (i.e. IB, FC, …etc.)

2.2. QoS Requirements for OSD

As an emerging storage access protocol, OSD provides a generic framework to store and deliver data objects for diverse applications. Data can be accessed by applications on the client through a TCP/IP network. Considering the diversity of clients and applications running on clients, we have defined three-levels of QoS requirements:

Level 1: Object-level QoS, (i.e. the QoS attributes pertaining to a particular object). This is largely driven by real-time applications that require QoS for data access (i.e. playout). For example, an MPEG-4 video object requires 4Mbits/s bandwidth with very low jitter when streaming the data from the storage device to the client. These attributes (BW and jitter) are characteristics of the *object* that describe *how* the object should be delivered to a client unless otherwise stated. In the event that these access characteristics are not required for a particular access, they can be over-ridden on a per-session or per-operation basis as described in the following levels.

Level 2: A QoS session. A "session" is defined as the amount of time that a particular client needs to access the contents of any particular object (generally between the time the object is opened and closed). For example, this is important within the context of data storage consolidation where a corporation may place its data remotely but needs to maintain the QoS requirements for access to objects on the remote storage system. Paper [15] proposed a virtualized storage with QoS attributes to address these QoS requirements. In this environment, the QoS goals should be applied to all requests through out this session. Another example of session-level QoS is related to the MPEG-4 example given for Level-1 above. Even though an MPEG-4 object may have an access bandwidth of 4Mbits/sec during playout, it may also be necessary to access that object at a much higher bandwidth during a copy operation. Therefore, a copy "session" would override the default access attributes with temporary access attributes that exist only for the life of the session and only for that particular client and application program.

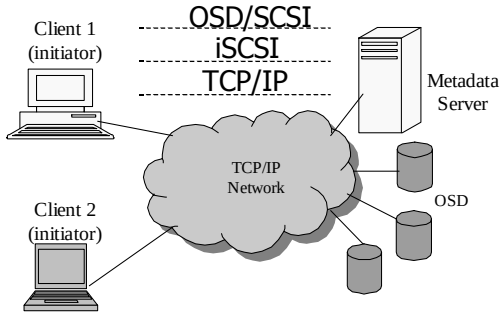

Figure. 1 OSD-based system

Level 3: A QoS *operation*. At this level a QoS attribute only applies to one specific operation. For example, when an application on a client requires *immediate* access to a certain amount of data within an object, a QoS attribute can be specified for this single request.

These three scenarios represent QoS at three different levels, with the object-level QoS most generic and operation-level QoS most specific. In practice most object access will have only one level of QoS specified. If there are QoS requirements at multiple levels associated with an object session, the *operational-level* QoS takes precedence, followed by the *session-level*, and then the *object-level* QoS.

It is important to note that the QoS levels are independent of the QoS attributes. For example, the QoS attributes for an MPEG-4 data stream (BW and jitter) are different than the QoS attributes for a gaming object (latency). QoS-levels can be applied to any set of access attributes.

Finally, some clients may have *statistical* QoS requirements rather than explicit requirements as described above. For example, when a client accesses a storage device at a remote data center the QoS requirements may be specified as guaranteed bandwidth averaged over some period of time (i.e. 1 day), a lower average response time, or an extremely low level of data loss (i.e. dropped frames). In this paper we examine the following two performance metrics:

- Bandwidth: the number of bytes transmitted per second. The data delivery system (i.e. the network) and storage subsystem should provide the required bandwidth to the client.
- Delay: the elapsed time between when a command is issued and the first byte of data is received. The bandwidth will determine when the last byte is received in any particular transmission. Delays *within* a particular transmission are represented in the delivered bandwidth.

3. Proposed QoS Enhancements to the OSD and iSCSI Specifications

Based on the above analysis of QoS requirements, we propose several extensions to the current SNIA OSD specification and the iSCSI standard to address the needs of these requirements.

3.1. Object-level QoS Specification

The first extension we propose is the addition of a QoS attributes page. This page defines the fundamental QoS attributes and values used on a per-object basis. Since these attributes apply to a specific object independent of which client accesses it, this page defines the *Object-level QoS* for the associated object. Therefore, an object that requires specific QoS access characteristics should have its QoS attribute values set in its QoS attributes page. The QoS attributes page is defined as follows:

Table 1. The extended QoS attributes Page

Attribute Number	Length (bytes)	Attribute	Client Settable	OSD logical unit
0h	40	Page Identification	No	Yes
1h	20	Bandwidth	Yes	No
2h	20	Delay	Yes	No
3h to FFFFFEh		Reserved	No	

We define two attributes in this page: bandwidth and delay. The bandwidth attribute can be either a hard guarantee or soft guarantee. A hard bandwidth guarantee ensures that each operation will achieve the bandwidth specified in the bandwidth attribute. A soft bandwidth guarantee is more flexible. It ensures that the average bandwidth over some period of time will meet the bandwidth specified in the bandwidth attribute. In addition, the bandwidth requirement can also be specified at several levels. Each level specifies a range of bandwidth. For example, three levels of bandwidth can be specified as high, medium, and low.

The delay attribute can also specify a particular value for a specific delay requirement or for a range of delays. Table 1 shows these attributes and associated parameters. Compared to other types of attributes, the QoS attributes will be enforced by the object storage device runtime environment when an object with a QoS attribute page is delivered.

For our implementation we use 8000h as the page number for QoS attributes page for a user object. In the numbers allocated for use with user object attributes pages, 8000h-EFFFh are reserved for other standard attribute page definitions like the QoS attribute page. We expect QoS requirements for future applications to allow this attribute page to be included in a future version of the OSD protocol specification much like security requirements.

3.2. Transient QoS Attribute Specification and Object Sessions

In addition to the QoS attribute page, which is stored permanently in the object device, we propose the concept of “transient QoS attributes”. In contrast to permanent attributes, transient attributes only affect the behavior of the I/O operations for a given period of time. We define this “period of time” as an object “session”. In addition to

the QoS parameters (like bandwidth and delay) previously described, we define a parameter to specify a "session ID" that is used to distinguish different applications either running on one client machine or on several client machines. Figure 2 shows the proposed QoS attributes. To define an object session to the object storage device, one of three methods can be used:

1) A Time To Live (TTL) value can be used to specify the length of time that session-level QoS attributes of the object should be observed for I/O operations associated with a specific session ID. The TTL starts upon the receipt of the Transient QoS Attribute for an object. Once the specified time expires, the default QoS parameters for the object are used.
2) An Operation Count (OC) can be used to specify the number of I/O operations that the object device should apply the Transient QoS Attributes to. After this number of operations has been executed on the associated object, the default QoS parameters for the object are used.
3) An explicit method of starting and ending a QoS session involves a parameter within the Transient QoS Attribute called "SessionActive". To denote the start of an object QoS session, SessionActive would be set to a non-zero value. Conversely, to end an object QoS session the SessionActive would be set to 0.

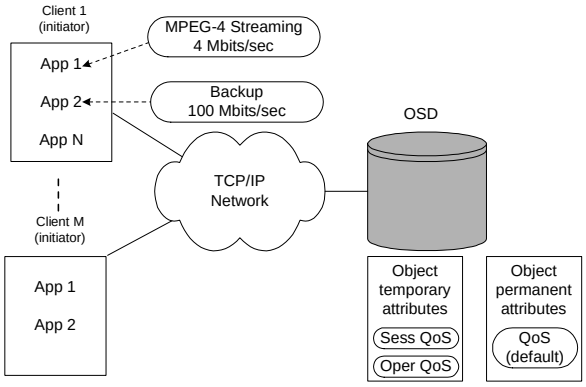

Figure 2. The proposed QoS attributes

Table 2. Get/Set CDB Format values (offset 11)

Value	Description
00b	*Attributes involved only affect the current operation*
01b	*Attributes involved only affect the current object session*
10b	Get an attributes page and set an attribute value
11b	Get and set attributes using list

To distinguish permanent attribute from transient attributes, we propose to extend the semantics of field Get/Set CDBFMT in the CDB (at offset 11). This field contains two bits. The values 10b and 11b were already defined. We propose a value 00b for this field to indicate that the included QoS attributes are used only for the current operation, and a value 01b that indicates the included QoS attributes be used for object session. The Table 2 shows the format of this field and its semantics.

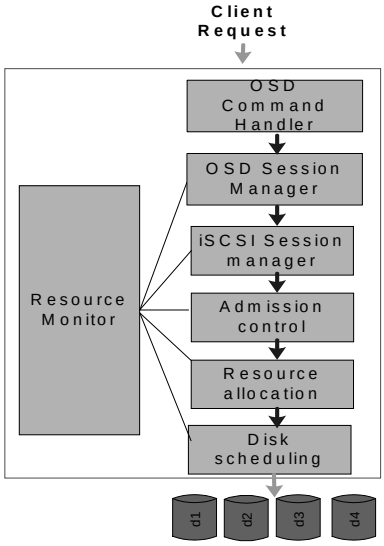

Figure 3. QoS framework in OSD

3.3. iSCSI Session-level QoS Specification

In addition to the proposed OSD extensions, we propose extending the iSCSI specification [4] to accommodate QoS for storage-centric network traffic. The reason for this is because OSD will be run over iSCSI and deliver data through a TCP/IP network. In order to support the QoS specification we have proposed, the underlying iSCSI transport needs to be extended.

In iSCSI, there is a rich set of predefined QoS-related parameters that are related specifically to the network (as opposed to the storage device). For example, MaxConnections defines the maximum number of connections that can be used for a session. ImmediateData specifies whether data is allowed to be attached to an iSCSI command Protocol Data Unit (PDU). In order to support QoS for a session, we propose adding storage-centric QoS parameters to the iSCSI specification. The proposed parameters are as follows:

- MaxBandwidth: This is the maximum bandwidth required for this iSCSI session. The unit is Kilobits/second. The bandwidth is the data rate as seen by the initiator.

- MinBandwidth: the minimum bandwidth required for this iSCSI session. The unit is also Kilobits/second. The bandwidth is the data rate as seen by the initiator. The Target must provide a delivered bandwidth between MaxBandwidth and MinBandwidth. If the target cannot provide the required bandwidth, the returned status will report a denial of the QoS requirement with the reason "*insufficient resources in the target*".
- MaxDelay: the maximum response time for requests in the iSCSI session. The unit is nanoseconds. This delay includes the two-way network transmission time and the delay within the target. The MaxDelay time is the time from when the client issues the command until the first byte of data is received by the client.
- MinDelay: the minimum response time for requests in the iSCSI session. The unit is nanoseconds. Parameters MaxDelay and MinDelay designate the expected response time range for requests in this session. The MinDelay time is the time from when the client issues the command until the first byte of data is received by the client.

These QoS related parameters are exchanged and negotiated during the iSCSI session setup phase. In iSCSI, all communications (command request, data transferred and response of execution status) occur in the context of session. Each iSCSI session has at least one TCP connection. Initially, when a client connects to a target, an "iSCSI session setup" phase starts. During the iSCSI session setup phase, an initiator goes through a discovery phase to locate the target. Once the target is located, a "login" phase is initiated. In the login phase, security parameters are negotiated and the authentication and authorization methods are determined. After a successful login phase completes, the operational parameters are negotiated during the "operational parameter" phase. During this phase the iSCSI QoS parameters would be negotiated between the client and target. Once the iSCSI QoS parameters are determined, the target will enforce these QoS requirements.

4. QoS Enforcement in OSD

The previous section addresses the specification of QoS requirements from a client's point of view. In this section, we discuss how to enforce the network and storage QoS requirements by the object storage device.

4.1. QoS Framework in an Object Storage Device

Figure 3 shows the proposed QoS framework in an Object Storage Device. This framework consists of seven components that reside in the Object Storage Device:

1. OSD Command Handler: processes incoming OSD commands
2. The iSCSI Session Manager: manages QoS in the context of an iSCSI session
3. The Object Session Manager: manages QoS in the context of an individual object session
4. Admission Control: determines when incoming commands will be processed given the current resource availability as determined by the Resource Usage Monitor.
5. Resource Usage Monitor: maintains a current view of resource usage on the object storage device and network channels. It provides this information to the disk scheduler and resource allocator. The Resource Usage Monitor not only receives resource allocation information from other modules, it can also proactively probe the resource usage in certain situations. For example, when a session has been idle for a while, the resource monitor can actively probe the initiator to collect the available network bandwidth and network round-trip time.
6. Disk Scheduling: schedules operations to be performed on the disk storage subsystem on the back-end of the object storage device. The operations are scheduled based on their QoS requirements
7. Resource Allocation: allocates and relinquishes disk storage space, disk bandwidth, disk access windows, and front-end host interface bandwidth based on the QoS requirements.

4.2. Admission Control

Due to limited resources in the object storage device it is possible that not all clients' QoS requirements can be satisfied as the workload increases beyond the capabilities of the object storage device. Admission control is used to govern whether a new request can be satisfied based on the existing resource usage and workload.

For a new incoming request, an admission control check will be applied to determine if adequate resources are available to provide the required QoS. If the required QoS can be satisfied, the request will be accepted. Otherwise, the request will be rejected. The corresponding response message will indicate that insufficient resources were available to satisfy the QoS for this request.

Admission control occurs at two levels: OSD command level and the iSCSI session negotiation phase. At the OSD command level the QoS requirements can be retrieved from either the QoS attribute page of the requested object (implicit QoS), or embedded in the command as described in Table 2 (explicit QoS).

During the iSCSI session negotiation phase the iSCSI QoS parameters will be established between the client and OSD target. At this point, the admission control mechanism needs to check the current available resources and QoS requirements to determine whether the QoS requirements (implicit and explicit QoS) can be met. In addition, for the session-level QoS requirements, the session workload will be tracked so that it will not violate the QoS requirements. If the QoS requirements cannot be met then the request will either be rejected or treated as best-effort request depending on the requestor.

4.3. Disk Scheduling

Disk scheduling is an important mechanism in OSD to enforce the QoS requirements for multiple clients. Our disk scheduling scheme considers the disk characterizations such as disk I/O bandwidth and data placement as well as available host-side network bandwidth for each session and takes network round-trip delay time into consideration. It is very challenging to obtain available network bandwidth for a given path since the network is shared and the traffic/congestion is difficult to predict. There is a great deal of existing research in this area [26-28]. Paper [28] demonstrates an active probing scheme where a sequence of back-to-back packets is sent out and the available bandwidth is discovered based on the gaps between received packets. The OSD environment is similar but more challenging. We have a number of different sessions to maintain, and consequently many different paths exist essentially from the various objects to their associated clients. In our design, we take advantage of the existing approach and adapt them to our OSD environment.

We apply two techniques to disk scheduling in order to maximize the *aggregate* performance. First, we break down large requests to sub-requests to allow more parallelism between disk I/O and network I/O. However, there is tradeoff here: the split of requests allows more parallelism, but may worsen disk utilization due to the added seek time. Secondly, we implement two-level scheduling: logical level and physical level. At the physical level, we take the data delivery order requirement into consideration. This can reduce the resource such as buffer usage and improve the delivery efficiency because this essentially becomes a FIFO pipeline. At the logical level, requests are scheduled based on the QoS requirement of each request, their request size, and available network bandwidth. The details about the scheduling schemes will be presented in a separate paper.

5. Reference Implementation

In order to demonstrate the proposed QoS extensions we are working on an OSD reference implementation (based on Intel version 20) that will incorporate these extensions. We will then write associated drivers and applications that will exploit the QoS capabilities and run a variety of experiments to determine the effectiveness of these extensions.

Figure 4 shows the software architecture of the reference implementation. On the initiator side, there are three drivers:

1. The OSD file system intercepts and processes file I/O requests from the application and sends them to the object storage device driver
2. The object storage device driver builds an extended SCSI command data block for each I/O request to an object and passes the SCSI CDB to the iSCSI driver
3. The iSCSI driver handles session establishment and encapsulates the SCSI commands and data in an iSCSI PDU format and sends them to the object storage device over a TCP connection

On the target side, the OSD Command Handler receives commands from the underlying iSCSI layer and processes the commands. It invokes the storage management component to locate an object and passes the command to disk scheduling to execute the disk I/O operation.

The QoS requirements are specified by the initiator. An application can specify the QoS attributes of an object or a particular operation through ioctl call. For a session level QoS specification, a client can specify the QoS parameters in configuration files that will be extracted during the object session setup time.

The target is responsible for QoS enforcement. The Admission Control manager is used to check whether the available resource in the object storage device and the network can deliver the required QoS. The Resource Allocation Manager and Resource Usage Monitor are responsible for the resource management. They monitor the I/O bandwidth usage, the network usage and memory usage and allocate these resources as required and as available. Finally, the disk scheduler partitions and schedules the actual disk I/O requests from different sessions based on their individual QoS requirements.

6. Conclusion

As OSD emerges as the next significant storage protocol, the QoS guarantee for applications running on

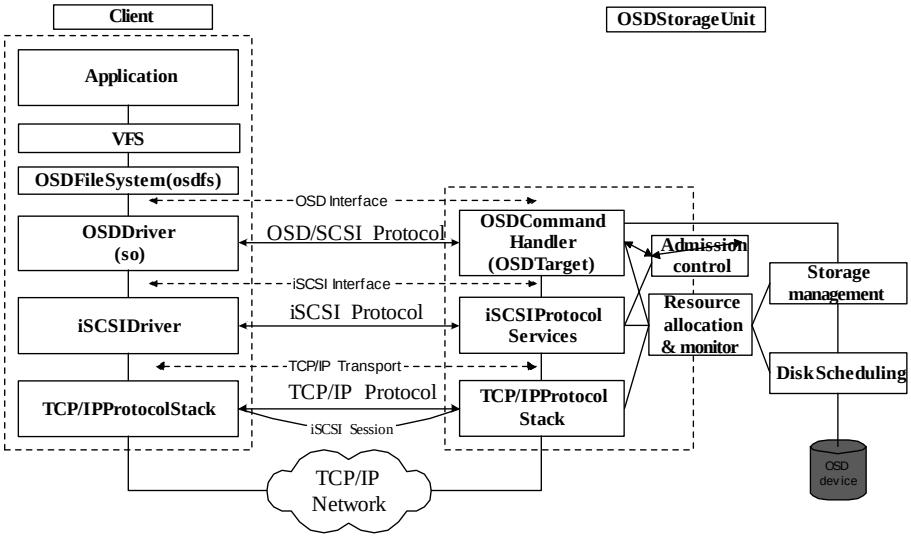

Figure 4. The software architecture

multiple clients accessing shared storage devices is becoming more important than it was when every application had essentially a direct connection to a storage device. In this paper, we have examined the current SNIA OSD and iSCSI protocols and presented a three-level model for QoS requirements that apply to the OSD protocol: object-level, operation-level and session-level. Based on our analysis, we have proposed extensions to the existing OSD and iSCSI protocol specifications to address QoS requirements in a storage-centric application. In addition to the QoS extensions, we have also presented a QoS framework for use within an OSD device. Finally, we are working on a reference implementation that will incorporate our extensions in an iSCSI OSD. We have discussed the architecture of the reference implementation and associated issues. Our future work will involve studying the resource allocation and scheduling in an object storage device that will enforce the QoS requirements.

References

[1] J. Satran, et.al., "Object-based Storage Device Commands", http://www.t10.org/drafts/osd/, Oct. 2004.

[2] M. Mesnier, G. R. Ganger, E. Riedel, "Object-Based Storage," *IEEE Communications Magazine*, Vol. 41, No.8, pp 84-90, August 2003

[3] Lustre project, http://www.lustre.org

[4] J. Satran, K. Meth, C. Sapuntzakis, M. Chadalapaka, E. Zeidner. "Internet Small Computer Systems Interface (iSCSI)," *RFC 3720*, IETF, April 2004.

[5] K.Z. Meth and J. Satran, "Design of the iSCSI Protocol," *IEEE/NASA MSST 2003*, Apr. 2003.

[6] Y. Lu, D. H.C. Du, "Performance Study of iSCSI-based Storage Subsystem," *IEEE Communications Magazine*, Vol. 40, No. 8, 2003

[7] Y. Lu, F. Noman, and D. H.C. Du, "Simulation Study of iSCSI-based Storage System," *IEEE/NASA MSST 2004*, Apr. 2004

[8] C. Aurrecoechea, A. Campbell, and L. Hauw, A survey of QoS architectures, Multimedia Systems, Vol. 6, No. 3, pp: 138-151, 1998

[9] R. Braden, D. Clark, and S. Shenker, "Integrated services in the Internet architecture: An overview," *RFC 1633*, IETF, July 1994

[10] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource ReSerVation Protocol (RSVP) –Version 1 functional specification," *RFC 2205*, IETF, Sept. 1997

[11] S. Shenker, C. Partridge, and R. Guerin, "Specification of guaranteed quality of service," *RFC 2212*, IETF, Sept. 1997

[12] M. Carlson, W. Weiss, S. Blake, Z. Wang, D. Black, and E. Davies. "An Architecture for Differentiated Services," *RFC 2475*, IETF, Dec. 1998

[13] R. Guerin and V. Peris, "Quality-of-service in packet networks: basic mechanisms and directions," *Computer Networks Journal*, Vol. 31, No. 3, Feb. 1999

[14] I. Stoica and H. Zhang, "Providing guaranteed services without per flow management," *in*

Proc. ACM SIGCOMM 99, (Cambridge, MA), Sept. 1999

[15] E. Borowsky, E. Borowsky, R. Golding, A. Merchant, L. Schrier, E. Shriver, M. Spasojevic, and J. Wilkes, "Using attribute-managed storage to achieve QoS", *in Proc. of 5th Intl. Workshop on Quality of Service*, 1997

[16] P. Shenoy and H. Vin, "Cello: A disk scheduling framework for next-generation operating systems," *in Proc. of ACM Sigmetrics*, 1998

[17] J. Bruno, J. Brustoloni, E. Gabber, B. Ozden, A. Silberschatz, "Disk scheduling with quality of service guarantees," *in Proc. of the IEEE Intl. Conf. On Multimedia Computing and Systems*, 1999

[18] R. Wijayaratne and A.L.N. Reddy, "Integrated QoS management for disk I/O," *in Proc. of IEEE Int. Conf. on Multimedia Computing and Systems (ICMCS'99)*, (Florence, Italy), June 1999

[19] P. Bosch, S. J. Mullender, "Real-Time Disk Scheduling in a Mixed-Media File System," *Sixth IEEE Real Time Technology and Applications Symposium (RTAS 2000)*, (Washington, D.C.), Jun. 2000

[20] J. Wilkes, Traveling to Rome: QoS specifications for automated storage system management, In *International Workshop on Quality of Service* (Karlsruhe, Germany), pp: 75-91, Jun. 2001

[21] W. G. Aref, K. E. Bassyouni, I. Kamel, M. F. Mokbel, "Scalable QoS-Aware Disk-Scheduling," *International Database Engineering and Applications Symposium (IDEAS'02)*, pp. 256-265, (Edmonton, Canada), Jul. 2002

[22] Z. Dimitrijevic and R. Rangaswami, "Quality of service support for real-time storage systems," *in Proc. of Intl. IPSI-2003 Conference*, (Stefan, Montenegro), October 2003

[23] K. Kim, J. Hwang, S. Lim, J. Cho, and K. Park, "A real-time disk scheduler for multimedia integrated server considering the disk internal scheduler," *in Proc. of the International Parallel and Distributed Processing Symposium*, Apr. 2003

[24] S. Brandt, S. Banachowski, C. Lin, and T. Bisson. "Dynamic integrated scheduling of hard real-time, soft real-time and non-real-time processes," *in Proc. of the IEEE Real-Time Systems Symposium (RTSS '03)*, Dec. 2003.

[25] C. R. Lumb, A. Mrchant, G. A. Alvarez, Façade: virtual storage devices with performance guarantees, *In Conference on File and Storage Technology (FAST 03)*, (San Francisco, CA), Mar. 2003

[26] R. Carter and M. Crovella, "Measuring bottleneck link speed in packet-switched networks," TR BU-CS-96-006, Boston University, 1996

[27] K. Lai and M. Baker, "Measuring link bandwidths using a deterministic model of packet delay," *in ACM SIGCOMM*, 2000

[28] C. Dovrolis, P. Ramanathan, and D. Moore, *What do packet dispersion techniques measure*, IEEE INFOCOM, Apr. 2001.