Dynamic Load Balancing and Network Adaptive Virtual Storage Service for Mobile Appliances

Ivy Ong* and Hyotaek Lim**

Abstract—With the steady growth of mobile technology and applications, demand for more storage in mobile devices has also increased. A lightweight block-level protocol, Internet Advanced Technology Attachment (iATA), has been developed to deliver a cost-effective storage network solution for mobile devices to obtain more storage. This paper seeks to contribute to designing and implementing Load Balancing (LB), Network Monitoring (NM) and Write Replication (WR) modules to improve the protocol's scalability and data availability. LB and NM modules are invoked to collect system resources states and current network status at each associate node (server machine). A dynamic weight factor is calculated based on the collected information and sent to a referral server. The referral server is responsible to analyze and allocate the most ideal node with the least weight to serve the client. With this approach, the client can avoid connecting to a heavily loaded node that may cause delays in subsequent in-band I/O operations. Write replication is applied to the remaining nodes through a WR module by utilizing the Unison file synchronization program. A client initially connected to node IP A for write operations will have no hindrances in executing the relevant read operations at node IP B in new connections. In the worst case scenario of a node crashing, data remain recoverable from other functioning nodes. We have conducted several benchmark tests and our results are evaluated and verified in a later section.

Keywords—iATA Protocol, Load Balancing, Network Monitoring, Storage Network Solution, Write Replication

1. INTRODUCTION

Mobile technology is the wave of future favoring a new communication revolution. The old days of being tied to desktops have been replaced by the widespread popularity of mobile devices converged with Internet service. While mobile devices make it easier for users to communicate and exchange information flexibly, there are inherent constraints that come along with these small size devices such as limited storage capacity and computing resources. Take for example, a typical personal digital assistant (PDA) may have a few hundred megabytes of internal storage and additionally attached with a few gigabytes of memory card. This amount of storage capacity is insufficient to sustain the increasing volumes of data and mobile applications.

※ The preliminary version of this paper appeared on the 4th International Conference on Multimedia and Ubiquitous Engineering (MUE). This is the extended version of that paper.
This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology(2010-0016831)
Manuscript received September 6, 2010; accepted January 11, 2011.
Corresponding Author: Hyotaek Lim
* Dept. of Ubiquitous IT, Dongseo University, Korea (Ivy_ong2000@yahoo.com)
** Dept. of Computer and Information Engineering, Dongseo University, Korea (htlim@dongseo.ac.kr)

An alternative way to obtain more storage is through the use of storage network technology, such as deploying the existing Internet Small Computer System Interface (iSCSI) application protocol to facilitate data transmission between a mobile client and a stationary server over the Internet. With the inspiration taken from routable iSCSI protocol and inexpensive Advanced Technology Attachment (ATA) disk drives, we implement a new iATA protocol [1] to transmit ATA commands and data over the TCP/IP network between a mobile client and a stationary server. Users may flexibly access the unutilized storage of a remote machine any time anywhere, as though it is a local storage medium for the mobile device.

In this paper, we propose a new architecture with three main modules to improve the protocol's scalability and data survivability. The first LB module is designed to keep track of node workload, analogous to the idea of system load balancing. Essentially, load balancing serves to balance a cluster system's workload by transferring processes to idle or more lightly loaded nodes in order to maximize the throughput and minimize the total response time [2]. In *distributed clustering,* users are directly connected to one of the processing nodes, where each node itself is responsible to make task scheduling decisions. Whereas in *centralized clustering,* users communicate with the cluster through a master node that is endowed to make all task scheduling decisions. We apply the centralized concept to the LB module to achieve the same goal. A referral node is tasked to direct clients' connection request to associate nodes based on their current workload. Among the many load balancing algorithm policies found in the literature, we refer to a set of policies [2] in our implementation: the *load estimation policy* that specifies types of load data to be collected and analyzed; *information exchange policy* that defines ways to exchange load information among nodes; *process transfer policy* that examines if a node is lightly loaded or heavily loaded based on pre-defined or dynamic thresholds; *selection policy* that decides which process is to be transferred; *location policy* that chooses the most appropriate node with minimum workload to run the process.

The second NM module is designed to monitor the network status of each node from its communication link. Due to the highly varying nature of network conditions, nowadays, many Internet applications are built with network-aware mechanisms. These applications are sensitive to network variation and able to deal with it. Network variation is commonly referred to as changes in network performance attributes such as bandwidth, throughput, packet loss, delay or latency and jitter [3]. These attributes can be measured through active or passive monitoring approaches. Active monitoring provides easier access to query data from more different types of networks and allows explicit control of the generation of test packets being used in the measurement process. However, it creates extra network traffic. Passive monitoring overcomes the problem by making measurements with actual user packets, yet it raises security concerns on how to retrieve and protect data gathered as it reviews all packets as they flow across the network. It is also more costly and information received may not be up-to-date [3]. In NM module, we employ the active monitoring approach to measure network performance attributes such as the *percentage of packet loss* and *average round trip time (RRT)*. The third WR module is designed to replicate write data across associate nodes on top of a tasking node. In the event of a node crashing, data are still retrievable from other functioning nodes. The objective is to help users save their data from unexpected failure reliably, while reducing the cost of downtime.

The rest of this paper is organized as follows: Section II discusses related research work. Section III describes our new architecture and its implementation strategy. The experiment test and performance evaluation are shown in Section IV. We conclude the paper in the last Section V.

2. RELATED WORK

Traditionally, many load balancing algorithms use CPU queue length to determine the workload of a node. Without considering other important elements this does not sufficiently reflect the exact conditions. Paul Werstein et al. [2] develop a new load balancing algorithm to measure CPU queue length, CPU utilization, memory utilization and network traffic from a homogenous distributed cluster environment. Dynamic thresholds of these elements (except memory utilization) are calculated and used to classify nodes into categories of idle, high, low or normal. CPU-bound, network-bound and memory-bound applications are applied in tests to compare the performance between the new algorithm and the CPU queue length based algorithm. It demonstrates that the former gets higher accuracy and better performance in finding idle nodes and when the cluster has mixed types of application requests.

Besides, there are a lot of load balancing methods available for iSCSI such as Round-Robin (RR), Weighted Round-Robin (WRR), Least Connection (LC) and Weighted Least Connection (WLC). These existing methods do not consider much if an established connection fails. They will only wait for the recovery of the failed connection and retransmit the data. Jung Hun Kang et al. [4] have come out with a new mechanism that efficiently moves data from failed connections to other valid connections in a large scale data transmission. The *Q-Chained Clustering Algorithm* is the core module that distributes loads to other connections equivalently. In contrast to the multiple network card and multiple connections techniques that cannot assign every task appropriately, Li Chen et al. [5] present a dynamic load balancing algorithm based on multiple connections with a feedback mechanism (DLFC) to enhance iSCSI system performance and scalability. The DLFC periodically checks resources parameters' values, combines recent job response time and average response time stored in an historical table to compute the dynamic weight and weight factor of every connection. The system will not allocate tasks to a connection if it is overloaded based on the comparison results between weight factor and the initialization threshold. It provides the system with a relatively optimal state by dynamically adjusting the threshold value.

On the other hand, the unpredictability of network behavior, especially in mobile environments has caused the performance degradation of Internet applications. A lot of research has been conducted towards development of network-aware applications that allow an application to be sensitive to its network environment changes and adapt accordingly. Jürg Bolliger and Thomas Gross [6] introduce a general framework to the construction of network-aware programs. They evaluate its practicability with an adaptive image server, called Chariot. The backbone of this framework is a feedback control loop composed of three parts: *Monitor and React* that repeatedly collects network service quality information from the lower layer and decides if an adaptation for a certain object is required to account for a performance degradation; *Prepare* where the adaptation process takes place to transform the available version of the object to the desired object of quality in order to meet the Quality of Service's (QoS) goal; *Transmit* the prepared object to the receiver. There are several ways of obtaining network status, either through application-level monitoring, transport-level monitoring or network-level monitoring. In [7], Jürg and Thomas further compare and analyze the measurement of bandwidth with application-level and transport-level monitoring mechanisms. The empirical result concludes the latter method is more preferable to get timely and accurate feedback in response.

A fast and inexpensive Remos API [8] is designed to collect static and dynamic network in-

formation explicitly through a hierarchical architecture. It has two layers, where a *collector* collects network status relevant to applications and a *modeler* acts as an intermediate library to export collected information to satisfy application requests. The collector can be a *master collector* or *data collector*. To minimize overhead problems, collectors may cache the information and an "age" parameter (specified in seconds) is used to determine whether requests can be served from the cache. Simple Network Management Protocol (SNMP) queries are used to obtain network information from a Local Area Network (LAN), whilst user-level benchmarks (e.g. traceroute) are used for a Wide Area Network (WAN). Moreover, network equipment is available in the market to capture network information. For example, Mizuta Toshiya et al. [9] implement a QoS Monitoring System with two separate components: An *IQ1000 QoS Probe* which is installed at each desired measurement point to collect users' actual packets and transfer them to *QoS Manager Software*, which is installed on a personal computer and equipped with SNMP Management Information Base (MIB) for calculation.

3. ARCHITECTURE DESIGN AND IMPLEMENTATION

3.1 iATA Overview

iATA protocol lies on the application layer of an Internet protocol stack. Each iATA Protocol Data Unit (PDU) is composed of two portions: the *common header segment* and *data segment*. The common header segment occupies a total of 128-bits and indicates the size and type of data segment that the PDU carries. The data segment can either be a *configuration/query message* used for parameter exchange or an *ATA command block* that carries actual data to be read or written to a disk drive.

The protocol supports four operation phases: the *login phase* where both client and server negotiate parameters to be used in a session; the *discovery phase* where the server sends geometry information of its exported disk to the client; the *full feature phase* where all in-band read and write operations take place; the *termination phase* where the client sends a termination request message to the server to disconnect the current session. These phases are handled by *iATA client stream interface drivers* that operate on a Windows CE platform and an *iATA server program* that operates on Linux platform. Once a TCP connection is established successfully between a client and a server, ATA commands or data are encapsulated into PDUs before transmitting to the target server. PDUs received at the target server are then de-capsulated and reassembled to obtain their underlying ATA commands or data for execution. Response messages are generated and replied to in the same way as the client sent request messages.

3.2 iATA New Architecture

Fig. 1 depicts the new architecture with system and network monitoring modules. When the intermediate referral server (rserver) receives an incoming client connection request, it will invoke LB and NM procedures installed at each associate node (e.g. node A, B and C) to collect system resources states and network performance attributes. Based on the collected information, a weight factor is calculated and sent to the rserver. The rserver accumulates and analyzes all factors received to choose the most appropriate node with the least weight (e.g. node B) to serve the client. Node B's IP address is forwarded to the client for the next step of the disk mounting

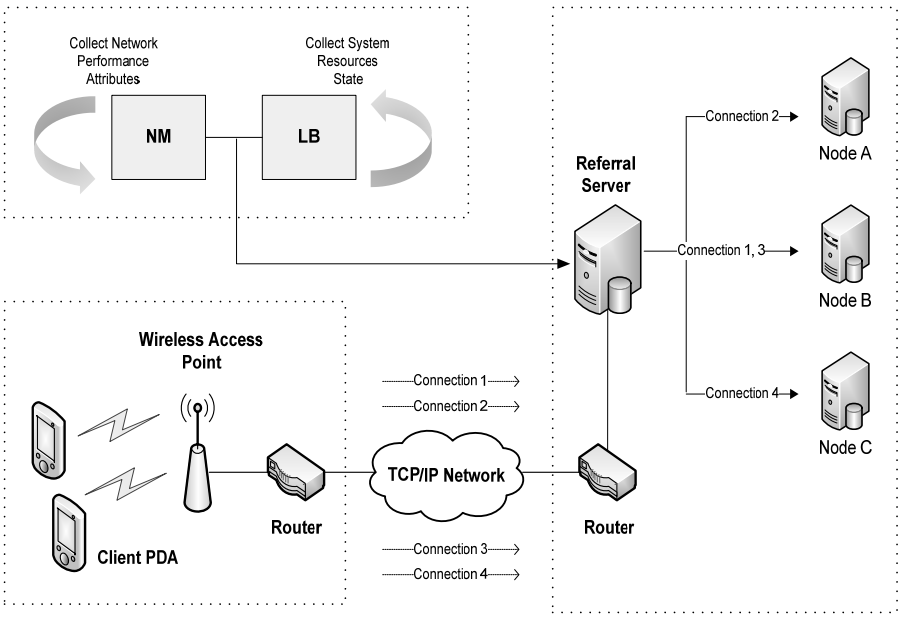


Fig. 1. iATA new architecture with system and network monitoring modules

process. With an additional write replication function, any write operation performed on node B will be duplicated to node A and C by the *Unison file synchronizer* [11]. Each component's details are discussed below.

3.2.1 Load Balancing (LB) Module

This module is written in Java language and installed at each associate node to be invoked by the rsever. A Linux top command is executed to collect system resources states information.

(a) *Linux top command:* The command is useful to monitor which users and processes are consuming the most system resources at present. It displays a list of tasks being managed by Linux kernel and the system's overall status in a summary form, comprising of uptime, load average, process counts, CPU status, memory or swap space utilization statistics and others.

(b) *Load elements:* A parsing technique is used to break up the output strings received and retrieve only targeted load elements such as the CPU user process, CPU system process, total memory and memory used rate (the memory utilization state is obtained by dividing memory used by total memory). A parser will firstly create tokens and follow by matching them with the output strings. The matched pairs are extracted and kept for subsequent calculation with the NM module to generate a weight factor at a given time.

3.2.2 Network Monitoring (NM) Module

Similarly, the NM module is written in Java language and installed at each associate node. A Linux ping command is executed to collect network performance attributes from each communication link in a simple way without necessitating excessive computation to the node.

(a) *Linux ping command:* The command is useful to verify the connectivity between two hosts on a network quickly. It sends Internet Control Message Protocol (ICMP) echo test packets to the destination host and watches for the response. It measures the minimum/average/maximum/ mean of RTT and records if there is any packet loss during a probing process. In our case, each probing attempt will send a set of five test packets and receive the output strings in return.

(b) *Network performance attributes:* NM does parsing to extract only required network attributes such as the percentage of packet loss and average RTT. Packet loss refers to packets that are not reaching their destination. Extreme loss will make it difficult to support real-time I/O operations over the network. RRT is the accumulated time for a packet to be transmitted forward and backward between a source and a destination and is measured in milliseconds (ms). The value of this attribute provides an indication of the presence of congestion in the network. The matched pairs are kept for subsequent calculation with the LB module to generate a weight factor at a given time.

3.2.3 Write Replication (WR) Module

Write replication is a process to duplicate or share data on other devices in addition to the primary device to achieve better data accessibility and reliability. A way to accomplish this is by applying *Unison*, a file-synchronization tool, resilient to failure with simple mirroring and backup utilities. It allows data backup and data synchronization across different operating systems. We utilize the standard functions available in the *Unison Linux version 2.32.52* to make real time backup for iATA write operations. The configuration settings to synchronize files among nodes at every minute are done earlier, which will be called during system booting time. These settings are flexible to change based on users' requirements.

4. EXPERIMENT METHODOLOGY

In the experiment test, hardware components such as a *PDA*, a *Wi-Fi Access Point*, a *Router* and three *ATA-based Personal Computers (PCs)* are interconnected through a Fast Ethernet. Both the PDA and PCs' specifications are listed in Table 1 and Table 2 respectively.

Table 1. Client PDA specifications

Model	HP iPAQ hx2700
Processor	Marvell PXA270 , 624MHz
RAM	64MB
ROM	320MB
Operating System	Microsoft® Windows Mobile® 5.0
Secure Digital (SD) Card	4GB

Table 2. Server PCs' specifications

Processor	Intel(R) Core(TM)2 Duo, 2.53GHz
RAM	2GB
Operating System	Fedora Core 6 (Linux Kernel 2.6.18)
Exported iATA Disk	4GB

4.1 Scenario Test

We assessed the new architecture in a laboratory on an LAN. Summarized below are our steps of verification.

- Two PCs (nodes with IP address xxx.xxx.xxx.76 and xxx.xxx.xxx.77) are connected to a referral rserver. Each node exports 4GB of disk space (abbreviated as D_{76} and D_{77}).
- Once the rserver receives a connection request from a PDA, it invokes LB and NM modules installed at node 76 and 77 to start the querying and probing processes. Each node's system resources states and network performance attributes are collected in a summary form. As shown in Fig. 2, the percentage of packet loss, average RRT, CPU user process, CPU system process, total memory and memory used rate are extracted from the output strings received.
- Each node performs calculations to generate a weight factor and send it to the rserver. Node 76 is found with a lower load compared to node 77. The rserver forwards the node 76 IP address to the PDA.
- The PDA continues to establish a connection with the selected node 76 and starts in-band I/O operations as normal.
- In write operation, for instance, the PDA writes two files namely *Flower.jpg* and *OneDream.mp3* to D_{76} through an iATA Graphical User Interface (GUI) as exemplified in Fig. 3 (1). With the synchronization service provided by *Unison*, both files are written to D_{77} internally. Fig. 3 (2) compares the content of D_{76} and D_{77} on a Linux terminal screen. It proves that both disks have exactly the same content.
- In a new connection, if the PDA is assigned to mount on D_{77} instead of D_{76}, similar content of *Flower.jpg* and *OneDream.mp3* are readable from D_{77}. In the worst case scenario in which one of the disks (D_{76} or D_{77}) crashes or is lost, data can be retrieved from another available disk.

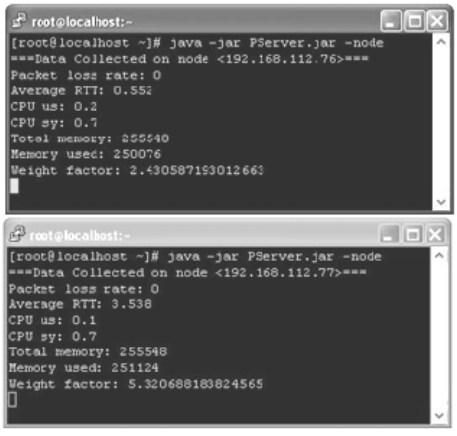
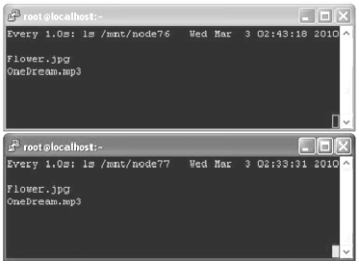


Fig. 2. Collection of system resources states and network performance attributes at nodes 76 and 77

(1) Files written to D_{76} through iATA GUI



(2) D_{76} and D_{77} on a Linux terminal screen

Fig. 3. Both D_{76} and D_{77} have exactly the same content

4.2 Data Reading and Writing Test

We use the award winning Windows Mobile benchmarking tool, *Pocket Mechanic Professional version 2.90* [12] to evaluate iATA performance. The tool auto generates a set of ten test files with sizes ranging from 4kB to 2MB, and calculates the average read and write throughput (the total number of bytes delivered divided by the total elapsed time) in buffered and un-buffered conditions. The Number of Sectors per Command (NSPC) parameter is set at 4096 sectors as the data request size throughout the test. Performance results are analyzed and discussed in the following section.

Fig. 4 compares the throughput obtained from the buffered and un-buffered file system read at each file size. The buffered file system read achieves the significant average throughput of 5MB/s compared to the un-buffered condition of 896kB/s. It is peaked when file sizes 32kB and 64kB are read. As file size increases, the throughput decreases substantially. For un-buffered file system read, the throughput continually increases and reaches the peak when file size 32kB is read. It drops dramatically for file sizes that are larger thereafter. It has the lowest throughput when reading file sizes 64kB and 128kB.

Fig. 5 compares the throughput obtained from the buffered and un-buffered file system write

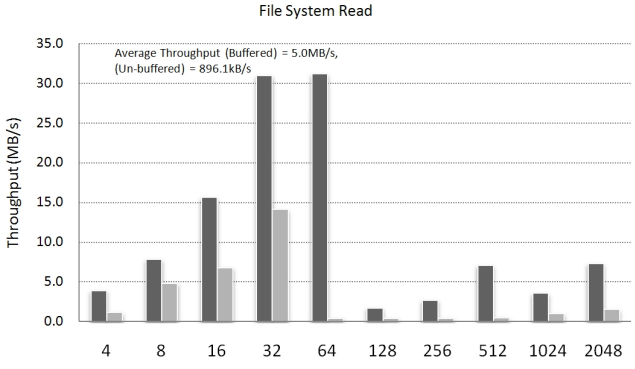


Fig. 4. Buffered and un-buffered file system read performance

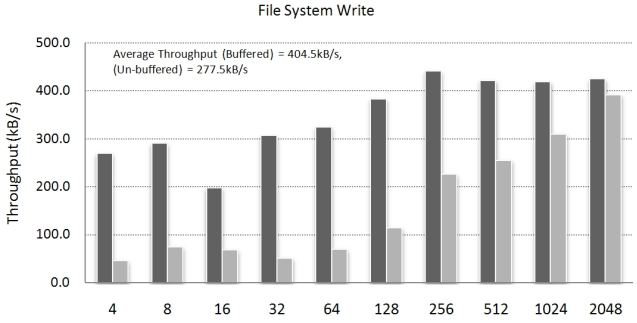


Fig. 5. Buffered and un-buffered file system write performance

at each file size. The buffered file system write has the average throughput of 405kB/s. Except for file size 16kB, the rest shows the throughput of magnitude close to or more than 300kB/s. The average write throughput with buffering bypassed is 278kB/s. It has the throughput below 100kB/s for file size smaller than 128kB. In contrast to read operation, the un-buffered file system write has increasing throughput when file size increases. It reaches the peak when file size 2MB is read.

5. CONCLUSION

iATA, an IP storage network protocol which operates on standard and common network components such as Ethernet, has overcome the insufficient storage issue in mobile appliances. It helps to make use of an unutilized storage machine at home or office wisely for immediate cost savings. Mobile device users are served with a virtual storage medium where the storage capacity can be flexibly changed according to ones need. The proposed architecture with system and network monitoring features has met the goal of delivering a more scalable protocol with better data availability. There are still many spaces of improvement to be worked out to improve the protocol's performance and functionality. We will be exploring more on dynamic data compression algorithms, adaptive error control and data recovery features as part of our future work.

REFERENCES

[1] Chee-Min Yeoh, Yu-Shu They, Hoon-Jae Lee and Hyotaek Lim, *"Design and Implementation of iATA on Windows CE Platform: An ATA-based Virtual Storage System," Proceedings of the WRI International Conference on Communications and Mobile Computing (CMC)*, Vol.3, January, 2009, pp.85-89.

[2] Paul Werstein, Hailing Situ and Zhiyi Huang, *"Load Balancing in a Cluster Computer," Proceedings of the 17th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT)*, December, 2006, pp.569-577.

[3] Jinwei Cao, Kevin M. McNeill, Dongsong Zhang and Jay F. Nunamaker, Jr., *"An Overview of Network-Aware Applications for Mobile Multimedia Delivery," Proceedings of the 37th HICSS Annual Hawaii International Conference on System Sciences*, Vol.9, 2004.

[4] Jung Hun Kang, Wonil Choi and Myong-Soon Park, *"Efficient Load Balancing Method for Mobile Applicable iSCSI-based Remote Storage Service," Proceedings of the 4th International Conference*

on Software Engineering Research, Management and Applications, August, 2006, pp.155-161.

[5] Li Chen and Ning Ma, *"Dynamic Load-Balancing in iSCSI Systems Based on a Feedback Control Mechanism," Proceedings of the 4th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM),* October, 2008, pp.1-6.

[6] Jürg Bolliger and Thomas Gross, *"A Framework-Based Approach to the Development of Network-Aware Applications," IEEE Transactions on Software Engineering,* Vol.24, May, 1998, pp.376-390.

[7] Jürg Bolliger and Thomas Gross, *"Bandwidth Monitoring for Network-Aware Applications," Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing,* 2001, pp.241-251.

[8] Nancy Miller and Peter Steenkiste, *"Collecting Network Status Information for Network-Aware Applications," Proceedings of the 19th Annual Joint Conference of IEEE Computer and Communications Societies,* Vol.2, 2000, pp.641-650.

[9] M. Toshiya, N. Kazuo, M. Shouji and M. Hiroyuki, *QoS Monitoring System,* 2002. Yokogawa Technical Report No.34.

[10] Ja-Won Seo, Hae-Sun Shin and Myong-Soon Park, *"Optimizing iSCSI Parameters for Improving the Performance of iSCSI based Mobile Appliance in Wireless Network," Proceedings of the International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies (ICNICONSMCL),* 2006.

[11] Copyright Benjamin C. Pierce, *"Unison File Synchronizer,"* [online], http://www.cis.upenn.edu/~bcpierce/unison/docs.html

[12] Copyright Wizcode LLC., *"Pocket Mechanical Professional Tool,"* [online], http://www.wizcode.com/products/view/pocket_mechanic_professional

[13] Kelvin Ng and Wilson Y.H. Wang, *"Design and Implementation of Algorithm with Multichannel Load Balancing and Failover for Generic Storage Area Networks," Proceedings of the 9th International Conference on Communications Systems (ICCS),* September, 2004, pp.311-315.

[14] Yolanda Villate, Arantza Illarramendi and Evaggelia Pitoura, *"Keep Your Data Safe and Available While Roaming," ACM Trans. Mobile Networks and Applications,* Vol.7, No.4, 2002, pp.315-328.

Ivy Ong

She received her Bachelor degree of IT in Information Systems Engineering from Multimedia University, Malaysia in 2005. She worked as a Software Engineer II in Western Digital (Malaysia) Sdn. Bhd. from 2007 to 2009. Presently, she is pursuing the Master of Science in Ubiquitous IT at Dongseo University, Korea. Her research interest includes hard disk drive technology, reliability analysis, mobile computing and network management.

Hyotaek Lim

He received his Bachelor degree in Computer Science from Hongik University in 1988, a Master degree in Computer Science from POSTECH and a PhD degree in Computer Science from Yonsei University in 1992 and 1997, respectively. From 1988 to 1994, he worked for Electronics and Telecommunications Research Institute as a research staff. Since 1994, he has been with Dongseo University, Korea, where he is currently a professor in the Division of Computer and Information Engineering. His research interest includes computer networks, protocol engineering, storage networking, IPv6 and mobile applications.