International Journal of Smart Home
Vol. 1, No. 1, January, 2007

SMART : High Performance Internet Server for High Definition Streaming Service

Yuhyeon Bak[1], Kapdong Kim[1], Youngju Lee[1], Songwoo Sok[1], Changsoo Kim[1], Hagyoung Kim[1], Myung-Joon Kim[1], Kyongsok Kim[2]
[1] Electronics and Telecommunications Research Institute
e-mail : {bakyh, kdkim71, youngju, swsok, cskim7, h0kim, joonkim}@etri.re.kr
[2]Department of Computer Science & Engineering, Pusan National University
e-mail : gimgs@asadal.cs.pusan.ac.kr

Abstract In the current life, the Internet is playing a key role in transferring diverse information. As the performance of the Internet has been growing, transferred information has the form of multimedia streams such as video and audio. However, in current computing and internet environment, multimedia streaming service can't be accomplished smoothly due to the various factors such as internet bandwidth, computing power and so on. In this paper, we are proposing an efficient system, called SMART (Server for Multimedia Applications for Residence community) including hardware and software in terms of the design concept and system architecture for the multimedia streaming service. We also present the implementation result for the system.

Keyword: Media Server, Streaming, Multicast, Digital Broadcasting, NS card

1. Introduction

As computers and high speed networks have been popularized, the Internet becomes a main means of transferring information over the world in our lives. As the communication networks to home will be upgraded to 10 ~ 100Mbps bandwidth level in the near future, high quality multimedia services such as internet broadcast, remote medical services and internet video services will be more generalized [1]. For example, a FTTH (Fibre To The Home) village is planned to be constructed in Korea. Those high quality multimedia services require a system that can handle the multimedia streaming service over network fast.

In general, networking operation has high cost in computer system in that it requires several data copies from one place to another and processing steps whenever data passes from one protocol layer to another [2][3]. In order to provide efficient multimedia streaming service over the network, we need to minimize the cost of networking operation. Also, service providers should deliver multimedia contents to end users with stability and efficiency over network in real time [4][5][6][7]. As service requests are increased, the traffic of a back-bone network increases, and as a result, the construction cost for the back-bone network increases. This problem should also be resolved.

In this paper, we are proposing an efficient system including hardware and software in terms of the design concept and system architecture for the multimedia streaming service. We also present the result of the implementation and performance evaluation for the system.

2. Design Concepts and Goals

2.1 Design Concepts

The proposed system is a system specialized to internet-based multimedia streaming services with high-performance communication infrastructure. In order to provide high quality multimedia streaming service, high network bandwidth is required. Also, efficient management of infrastructure and distribution of service requests are required to reduce back-bone traffic and server loads

Our system hierarchically distinguishes the services into global and local services that are processed separately to use network bandwidth efficiently. A global server is a computer system that provides overall multimedia contents and located at Internet Data Center (IDC), e-government and so on. While a local server is a computer that provides multimedia streaming services for local communities such as apartments, companies, universities that are based on local networks. It transparently communicates with a global server. The global server should be designed to provide various functionalities for contents service and to guarantee high scalability as well as high availability. And, the local server should provide not only single-node server solution to achieve cost-effectiveness and high-performance but also multiple-node cluster server solution to build scalable and high available system.

2.2 Design Goals

As first phase, before overall system including the global server is designed, we are focusing at local server solution. Therefore, in this paper, we use the term "server" for a local server. In this case, the role of the global server as contents provider must be simulated. For this role, we will introduce FSN (Files Server Node) node in our system infrastructure. In order to satisfy the requirements described at section 2.1, the server has following design goals.

- The server supports up to 200 clients and provides up to 4 Gbps of concurrent streams, which is equivalent to MPEG streaming service with the bit rate of 20 Mbps.
- The server provides not only clustering solution to build scalable and high available system but also efficient content placement mechanism for distributed

local servers in a cluster without shared storage.

- The server provides an efficient content distribution mechanism for storing the right content at the right time.
- The server provides an optimized multimedia file system.

In order to satisfy these goals, the server should be designed to improve performance in storage including efficient multimedia file system and networking system. Also it should support efficient clustering and content distribution features. In storage and file systems, it is needed to support continuous disk access at high speed. In networking systems, stream data stored at disks should be delivered to clients through the network in a zero-copy fashion. In clustering solution, the server should provide single system image for the cluster and efficient load balancing features between participating local servers. And, in content distribution solution, the server should provide an efficient content distribution mechanism from FSN to local servers.

2.3 Design Issues

In general, normal computer has high level of CPU utilization for networking operations with gigabit level. The reason is that complicated communication protocols are handled in host CPUs. Therefore, using a separate processor for handling networking operations in offload fashion will be useful [2][3]. Another reason of degradation of system performance for multimedia streaming services is several data transfers and copy operations in disk-to-network path. These several copy operations are the main performance bottlenecks [3]. Therefore, several data copy avoidance architectures such as integrated layer processing [4] and direct I/O to eliminate memory-to-memory data copy are suggested trying to minimize data transfers and copy operations. We will design a special PCI card that integrates SCSI controller, TOE engine and Memory module in order to reduce CPU utilization and copy operations.

In addition, because large volume of data storage is needed for multimedia streaming services, it is also required to use striping technique such as RAID (Redundant Array of Inexpensive Disks) for fast disk reading speed [3]. The size of a streaming data is relatively large and the access pattern has continuous form. So, handling streaming data requires developing a multimedia file system that can consider the characteristics of continuous form and large size of streaming data.

According to design goals, a local server is designed to support up to 200 concurrent streams with 20 Mbps bit rate. The regional community where proposed system provides multimedia streaming services has several range of concurrent user size. From this, the server should support scalable solution. General scalable solution is clustering technique. There are several clustering related works [8][9][10]. The proposed server has two-level cluster architecture in that the cluster consists of several server nodes and each server node can have several NS cards which will be described at section 3.1. Also it is formed like shared nothing architecture between server nodes, and in turn between NS storages of each server node. The clustering solution should support these shared nothing and two level features.

Even though the server provides large capacity of storage, all of the multimedia contents to be serviced cannot be stored at the server. As described at section 2.1, the global server plays the role as contents provider. It is important to store the right content in the right place at the right time to satisfy client's requests. Also, when a requested content is not locally available, it should be obtained from the global server or the FSN node as fast and reliably as possible. In order to support these requirements, the server should provide an efficient content distribution mechanism.

3. System Architecture

In this section, we present the design and implementation of a streaming server, called Server for Multimedia Applications for Residence communiTy (SMART). The proposed server is a system specialized to Internet-based multimedia streaming services with high-performance communication infrastructure. In order to provide high quality multimedia streaming services, high network bandwidth is required. Also, efficient management of infrastructure and distribution of service requests are required to reduce back-bone traffic and server loads respectively.

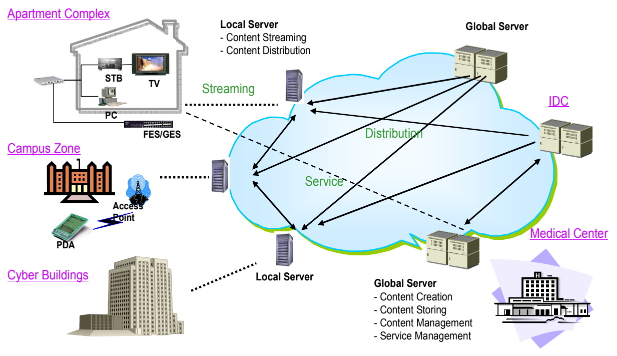


Fig. 1. Service Model of NGIS

Fig. 1 shows the service diagram of the system. Our system hierarchically distinguishes services into global and local services that are processed separately to use network bandwidth efficiently. A global server is a computer system that provides overall multimedia contents and located at Internet Data Center (IDC), e-government and so on. While a local server is a computer that provides multimedia streaming services for local communities such as apartments, companies, universities that are based on local networks. It transparently communicates with a global server. The global server should be designed to provide various functionalities for contents service and to guarantee high scalability as well as high availability. And the local server should provide not only single-node solution to achieve cost-effectiveness and high-performance but also multiple-node cluster server solution to build scalable and high available system.

3.1 Hardware Configuration[11][12]

SMART provides not only multiple node cluster server solution to build a scalable and highly available local server but also single node server solution to achieve a cost-effective and high performance local server. And it has one or two processors, a system memory, a memory controller hub (or bridge), an I/O controller hub, several

PCI/PCI-X controller hubs, a local hard disk. For efficient storage, retrieval, and transmission of large amount of video data, the server employs a new H/W device called Network/Storage card, or NS card for short. NS card includes both a disk storage interface and two Gigabit Ethernet interface all together. It also includes a buffer memory (PMEM: PCI Memory) to temporarily store the retrieved video data to be transmitted and TOE (TCP/IP Offload Engine).

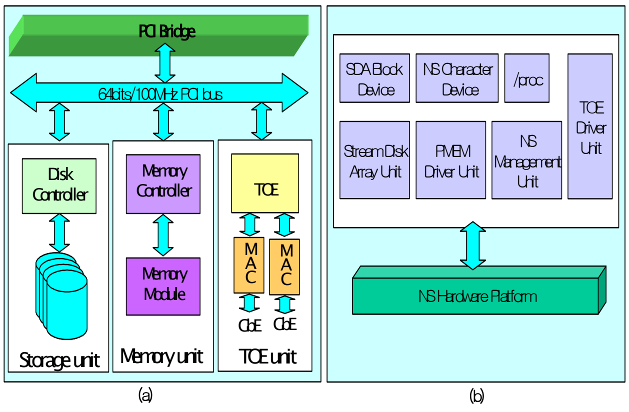


Fig. 2. Architecture of the NS Card's Hardware (a) and Software (b) unit

NS card is devised to rectify some limitations of existing file I/O for networked multimedia, such as unnecessary data copying, lack of guaranteed storage access and overhead of network protocol processing.

Each NS card has unique IP address like a server, so the single server with several NS cards do works like a traditional cluster system; i.e., the performance of a cluster system with 3 different servers might possibly be the same as that of a single server with 3 NS cards, in brief. So, if we use NS card, we can configure a low cost system.

Also, NS card can support high speed disk I/O and send to network compared with general system as following reasons:

Zero-Copy. Generally, from initially designed computer to now, the computers of von Neumann model are used. This model has many internal data copies between kernel memory and user memory when a computer reads data from disk and sends it to network. It is useful to general usage, but it is not inevitable for dedicated streaming service. So, we remove this additional overhead. Fig. 3 and Fig. 4 show the difference of data flow with zero-copy and without it.

Large Block Size. General Linux system use a small block size compared with multimedia files. The block size of Linux is 1Kbytes ~ 4Kbytes. NS card support a large block size such as 128Kbytes, 256Kbytes, 512Kbytes, 1Mbytes and 2Mbytes.

Block Split. When a user request a block, NS card split this block into n-small partitions. Here the value of the n is the same with the number of disks connected to NS card. It is similar operation to S/W striping in RAID. So, NS card can access to disk with parallel among disks.

Pipelined I/O. When NS card read data from multiple disks, the waiting time exists because of different disk performance. So, in the worst case, the read speed depends on the latest disk. NS card driver don't wait the whole I/O complete. As soon as a small partition I/O is completed from a disk, NS card read the next partition at the disk. As this pipelined I/O, all disks operate non-blocking I/O.

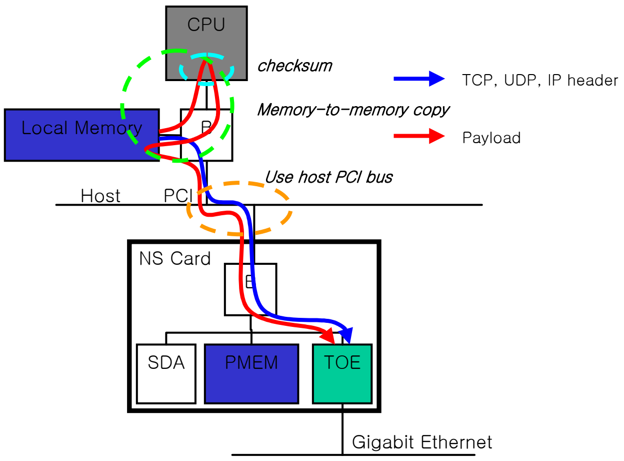


Fig. 3. Data flow of general TCP/IP transmission

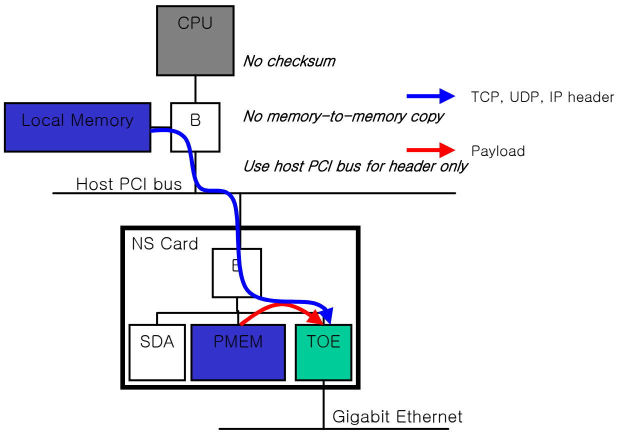


Fig. 4. Data flow of zero-copy

3.2 EXT3NS file system[13]

For high-performance multimedia streaming services, we need a multimedia file system that can utilize the facilities of the NS card and supports large volume of data. Initially, we developed dedicated file system to support NS card. But it has inter-operability problem, so we have developed new multimedia file system with general interface like ext3 file system. The new file system is called EXT3NS file system. EXT3NS provides applications with the standard read and write system call interfaces to use NS card. And in addition to fast-path I/O operation with NS card, legacy buffered I/O is support as well. Metadata is stored and accessed on a 4KB basis, but the unit of allocation for data blocks depends on the configuration NS card.EXT3NS manage NS. So, EXT3NS support a large data block size (128K ~ 2MB).

The EXT3NS is a file system built on top of NS card. It enables applications to access fast-path I/O of NS card via standard read and write system call interfaces. It provides legacy VFS layer interface for existing file system related applications as well. The file system can easily accommodate large block size defined by disk striping driver of NS card. It has a disk layout to efficiently store large numbers of multimedia files as well.

Fig. 5 describes the location of EXT3NS file system in the Linux kernel and how it interacts with the kernel. When the application read data from disk via read system call, EXT3NS determines whether it is read operation to PMEM area of NS card or to the system main memory using the argument value of user buffer address. If it is to PMEM area, EXT3NS performs the fast path I/O by using NS device driver. If it is to main memory, EXT3NS performs the legacy page cache I/O. In case of legacy page cache I/O, EXT3NS just read from disk to main memory by calling generic read function provided by the memory management module of Linux kernel. In this case, requested data block is read from disk to the page cache of Linux kernel through the system PCI bus and copied to the user buffer. Data in page cache can be reused if request for the same data occurs in the near feature. In case of SDA fast-path I/O, EXT3NS calls the read method of SDA device driver. At this time, requested disk block is read to the PMEM block of NS card. Data path of this operation only includes local PCI bus of NS card connecting SDA disk array and PMEM, which does not use system's main PCI bus. From the viewpoint of overall system it could be called as true zero-copy operation, because it does not include memory-to-memory copy and does not use the bus cycle of main PCI bus.

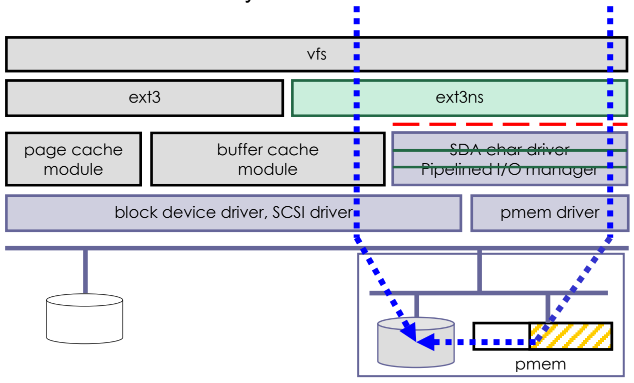


Fig. 5. EXT3NS file system in Linux Kernel

The biggest obstacle for EXT3NS in coexisting with other file systems under the VFS layer is that disk block size of SDA is much larger than that of generic block devices. x86 architectures allows only 512, 1024, 2048 and 4096 bytes as block size, whereas block size permitted by SDA driver is from 128Kbytes to 2Mbytes. So a solution must be made to handle this discrepancy problem properly. We solve it as follows. Metadata is stored and accessed on a 4KB basis. There are six kinds of metadata in EXT3: superblocks, group block descriptors, inodes, blocks used for indirect addressing, data bitmap blocks, and inode bitmaps blocks. They are allocated in 4KB unit and accessed via a buffer cache. This can be easily accomplished due to the fact that most of metadata reside in the head of EXT3NS partition and are kept strictly apart from the data and is collected in a separate structure for each file. For six metadata, indirect block must be taken special care since it resides in the data block area, not in the metadata block area. The unit of allocation for data blocks depends on the configuration of SDA driver. When reading and writing data blocks, applications can use either buffered read/write or fast-path read/write. Buffered read/write is the same as the existing mechanism using main memory cache. On the other hand, fast-path I/O utilizes PMEM as a buffer when reading and writing SDA device.

3.3 Media Streaming Engine

This module can do streaming service. It uses RTSP (Real-Time Streaming Protocol) as a control protocol between streaming server and clients, and can select a data transmission protocol among UDP, TCP, and RTP (Real Time Protocol). The server is divided in to main process and several NS processes. The main process has responsibility to control all of connections to the streaming server and a NS process is generated respected to each NS card of SMART system. Each NS process includes streaming data plane and several services.

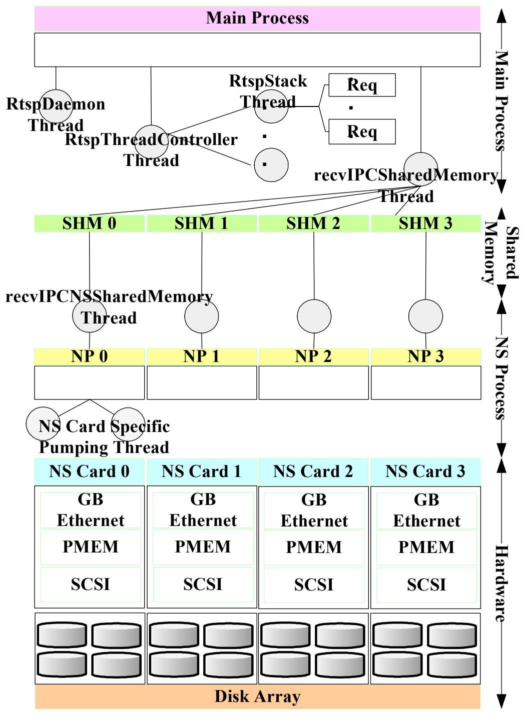


Fig. 6. MP/NP ASrchitecture

As shown in Fig. 6, a media streaming engine consists of a main process (hereafter "MP") and an NS process (hereafter "NP") which number is the same as number of NS card, for example 4, in given figure. An RtspDaemon thread of the media streaming engine creates an RtspThreadController thread which controls the creation and assignment of an RtspStack thread and has a FIFO which keeps a great number of RTSP connections. The RtspThreadController thread also passed a RTSP connection to the RtspStack thread through PIPE. The RtspStack thread manages maximum number of RTSP connections, manipulates RTSP messages and stores it to the specific shared memory area.

If a recvIPCNSSharedMemory thread in specific NP receives a RTSP request, it is added by a session manager to the session manager's map table. The session manager gets information of content ID(cid), content file name, for example /ns0/mpeg_ts.mpg in NS card 0, streaming type like MPEG-1/2/4, H.264 and suitable bit rate in metadata DB table and loads a suitable service library. The service library creates a sender instance with protocol type like RTP/UDP/TCP and use a specific NS card. The service thread reads a content file, sends through an NS card at the rate of given bitrate.

3.4 Content Distribution

Even though the server provides large capacity of storage, all of the multimedia contents to be serviced cannot be stored at the server. Also, if services are processed globally, the traffic of the global network infrastructure increases. This increased traffic makes the service to be unreliable or unavailable. To resolve this problem, a service provider generally adopts a CDN (Content Delivery Network) technique that locates many local servers or cache servers at various specific regional areas. In a CDN infrastructure, multimedia streaming services are provided with end users through a nearest local server [4][6][7].

As described at Section 3, the global server plays the role as contents provider, and we introduce the FSN (Files Server Node) node concept for replacing the global server.

It is important to store the right content in the right place at the right time to satisfy client's requests. Also, when a requested content is not locally available, it should be obtained from the FSN node as fast and reliably as possible. In order to support these requirements, the server should provide an efficient content distribution mechanism. The more the server caches streaming media, the more the server can provide services to users. Our server system also adopts prefix caching mechanism for making more services to be available on the spot.

The content distribution software provides following features for versatile transfer and distribution methods, and content usage monitoring.

Content Transfer. The content distribution unit transfers content from the CSN node to a local server. The content transfer job can be executed at promptly (On-Demand Transfer) or at delayed scheduled time (Scheduled Transfer). If a transfer job is scheduled job, the administrator can also modify or delete the scheduled job.

Preloading and Prefix Caching. The content distribution unit provides a mechanism to preload contents from the CSN prior to service requests by using either on-demand or scheduled transfer. For better chance of meeting user requests, it is desired to store as large number of contents as possible. This is a major reason of prefix caching. The content distribution unit also supports preloading with prefix size.

Dynamic Loading. When content is requested to be serviced and if the content is not available completely, the content distribution unit transfers the content dynamically.

Content Purge. When a content transfer job is activated and if the storage has no space to store the content to be transferred, the transfer job will be failed. For preventing this situation, the content distribution unit prepares needed storage space in advance by purging some contents existed in the storage.

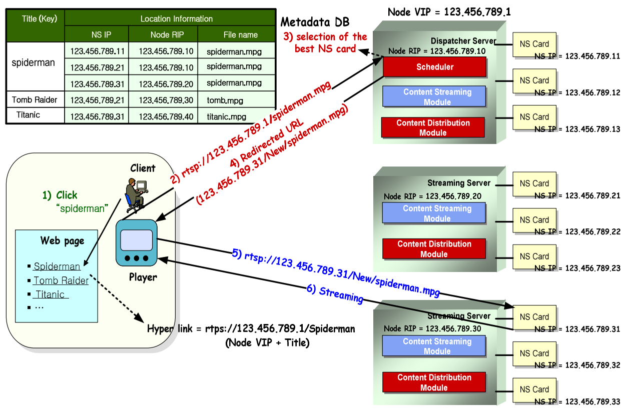
Content Storage Management. For determining the necessity of purge operation, the content distribution unit should know the status of storage.

Content Usage Monitoring. Selection of appropriate contents to cache at the server can affect the cache utilization and thus service performance. The unit should provide information about usage of contents.

Clustering Support. In order to response to user requests fast and correctly without creating unwanted delays, the server should be hosted on a cluster consisting of multiple streaming servers. Actually, the clustering technique is widely recognized to be a viable solution to build a scalable high-performance server system. With related to clustering solution, the clustering unit should distribute all incoming service requests efficiently across the multiple streaming server nodes in the cluster environment. Also, it should place contents effectively across the multiple streaming server nodes for efficient and balanced resource consumption.

3.5 Service Scenario

For the streaming service, we first need to install the content such as MPEG-2, MPEG-4 and H.264 in advance. During the install step, the system makes the auxiliary files for fast forward, fast rewind. And it also makes metadata for the content. Our cluster-based streaming server consists of a file server for installation and several streaming servers. We choose the dispatcher server among the streaming servers. Each streaming server has several (1-4) NS cards. The step of streaming is presented in Fig. 7.



Title (Key)	Location Information		
	NS IP	Node RIP	File name
spiderman	123.456.789.11	123.456.789.10	spiderman.mpg
	123.456.789.21	123.456.789.10	spiderman.mpg
	123.456.789.31	123.456.789.20	spiderman.mpg
Tomb Raider	123.456.789.21	123.456.789.30	tomb.mpg
Titanic	123.456.789.31	123.456.789.40	titanic.mpg

Metadata DB
3) selection of the best NS card
Node VIP = 123.456.789.1
Dispatcher Server
Node RIP = 123.456.789.10
Scheduler
Content Streaming Module
Content Distribution Module
NS Card NS IP = 123.456.789.11
NS Card NS IP = 123.456.789.12
NS Card NS IP = 123.456.789.13
Streaming Server
Node RIP = 123.456.789.20
NS Card NS IP = 123.456.789.21
NS Card NS IP = 123.456.789.22
NS Card NS IP = 123.456.789.23
Streaming Server
Node RIP = 123.456.789.30
NS Card NS IP = 123.456.789.31
NS Card NS IP = 123.456.789.32
NS Card NS IP = 123.456.789.33
2) rtsp://123.456.789.1/spiderman.mpg
4) Redirected URL (123.456.789.31/New/spiderman.mpg)
Client
1) Click "spiderman"
Web page
Spiderman
Tomb Raider
Titanic
…
Player
5) rtsp://123.456.789.31/New/spiderman.mpg
6) Streaming
Hyper link = rtps://123.456.789.1/Spiderman
(Node VIP + Title)

Fig. 7. Steps of Streaming Service

A user can select the content from the web site. When the user selects the content, the information of the content is transmitted to the dispatcher. The dispatcher finds the information of an optimal server and NS card from the metadata tables, and sends the information to the user. This dispatcher chooses the NS card according to the system load. The user connects to the NS card, and then, the NS card sends the data to the user.

3.6 Applicable Areas

We consider two kinds of area as follows to the applicable field of SMART. In these fields, it certainly needs the high performance streaming server. First is the server for IPTV service. Second is the server for Cable service.

For IPTV service, the server must support a unicast for VoD and multicast for internet broadcasting. In the current internet infrastructure, the perfect multicast streaming service is impossible because all of routers can not support multicast protocol. But, in Korea, the company for internet infrastructure, such as KT, Dacom, Hanaro, can do it in their internet infrastructure.

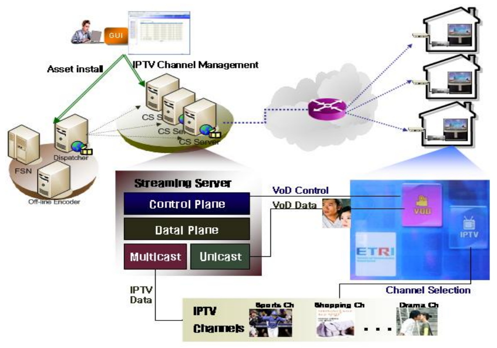


Fig. 8 IP TV Model

In Cable service field, it needs many kinds of equipments like Fig. 9. In left rectangle, there are a few of equipments of Content Provider. These equipments create new contents for service, and distribute them to local cable service companies in middle rectangle area in Fig. 9. SMART can be located in here. Local cable service companies support real streaming service to the client in right rectangle in Fig.9.

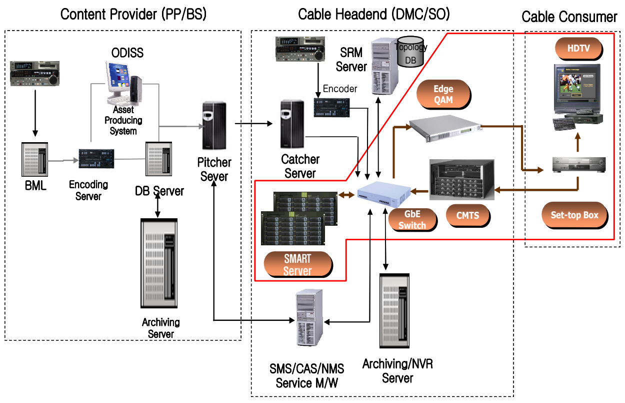


Fig. 9. Cable TV Model

4. Implementation Results and Performance Evaluations

In this section, we will show the SMART and the performance for the normal content.

Fig. 10. The Feature of SMART

There are two models in SMART system. Those are compact and standard model. Compact model can store more contents and support more NS cards than compact model. Fig. 10 shows the standard model.

Each NS card has PCI card, PCI-to-PCI bridge, several disks, PMEM, TOE and two gigabit Ethernet ports. NS card is shown in also.

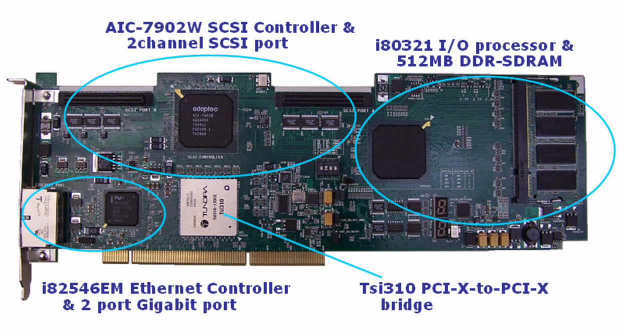

AIC-7902W SCSI Controller & 2channel SCSI port
i80321 I/O processor & 512MB DDR-SDRAM
i82546EM Ethernet Controller & 2 port Gigabit port
Tsi310 PCI-X-to-PCI-X bridge

Fig. 11. NS (Network / Storage) Card

From Fig. 12 to Fig. 21 show the disk I/O performance of our system. Our system is optimal for multimedia data using large block size. So we compare with EXT3[16], XFS[17] and our EXT3NS file system.

To measure the performances, we make 2 volumes with four 15K rpm disks(total disks are 8). There are 32 files in that volumes and each file size is 8GB. All files are allocated sequentially.

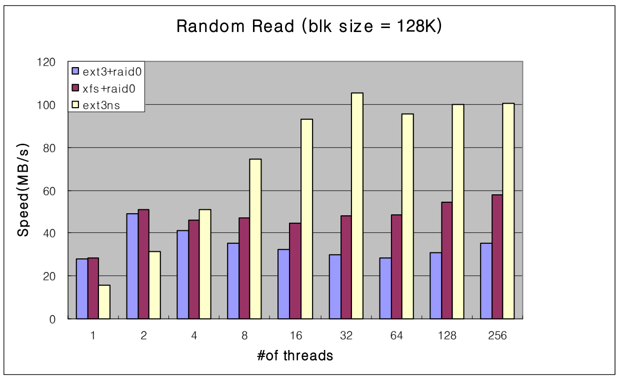


Fig. 12 Random I/O Speed (Block Size = 128KB)

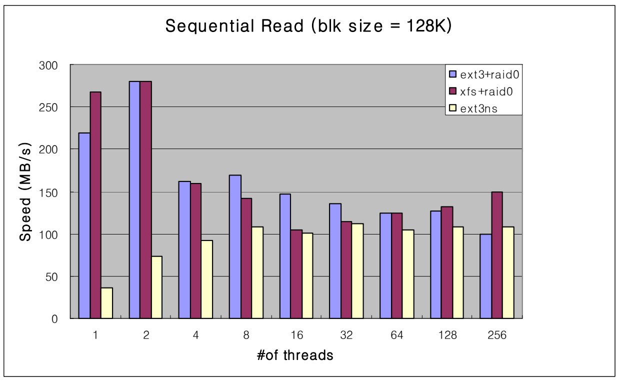


Fig. 13. Sequential I/O Speed (Block Size = 128KB)

In random I/O, our system has best performance than other systems. But in sequential I/O, in case of less-threads, EXT3 and XFS is better than our system. We think that it is due to read ahead operation.

As the concurrent user increases, the access pattern to disks is similar with random access. So, the performance in the random access is more important than sequential access in the area of streaming service. EXT3NS has good performance at many users than other file system.

And as compared with changing the block size, SMART has best performance in large block size such as 512KB or 1MB.

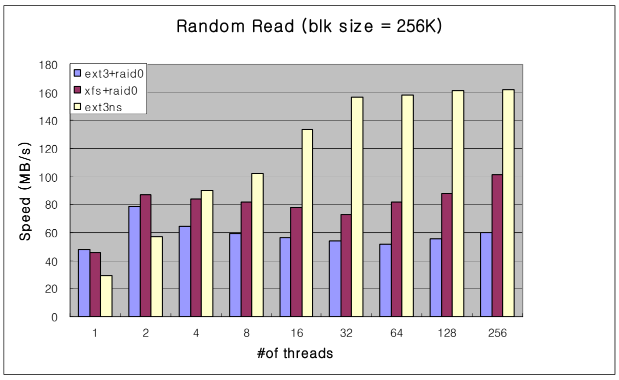


Fig. 14 Random I/O Speed (Block Size = 256KB)

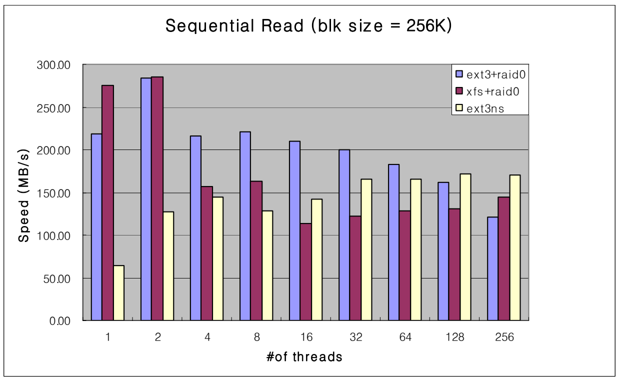


Fig. 15 Sequential I/O Speed (Block Size = 256KB)

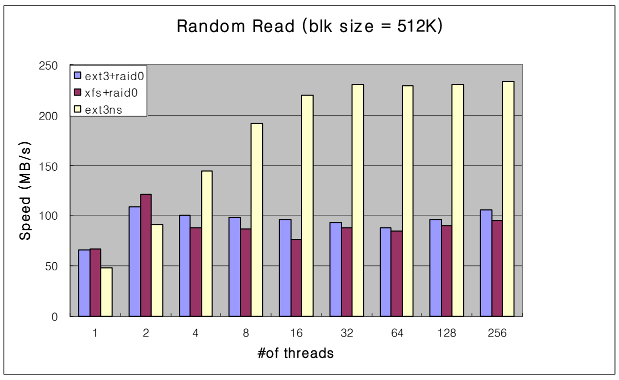


Fig. 16. Random I/O Speed (Block Size = 512KB)

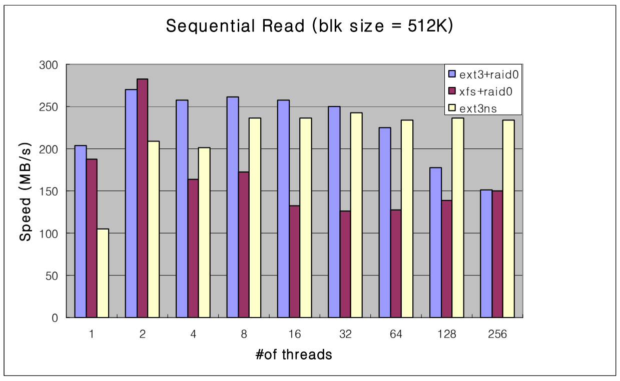


Fig. 17. Sequential I/O Speed (Block Size = 512KB)

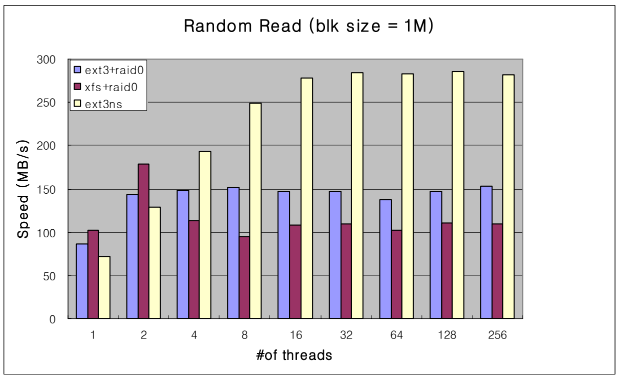


Fig. 18 Random I/O Speed (Block Size = 1MB)

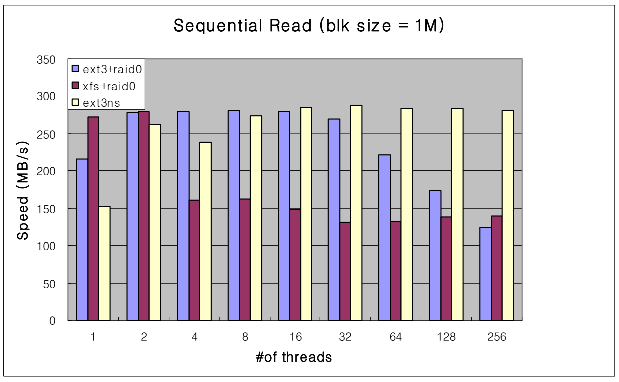


Fig. 19 Sequential I/O Speed (Block Size = 1MB)

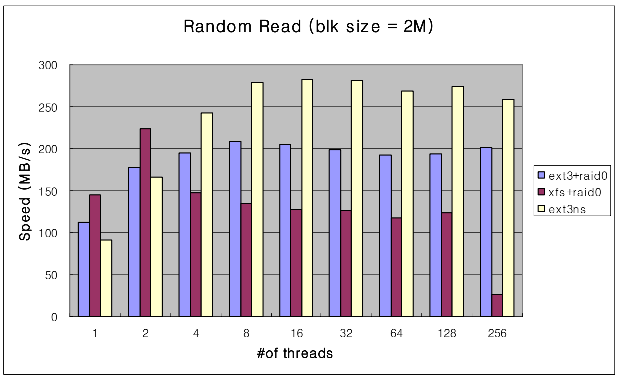


Fig. 20. Random I/O Speed (Block Size = 2MB)

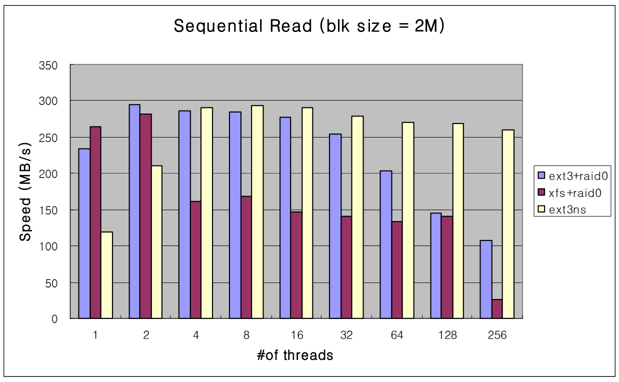


Fig. 21. Sequential I/O Speed (Block Size = 2MB)

Fig. 22 and Fig. 23 show the CPU loads of each system. The more concurrent users, the more CPU load of EXT3 and XFS. But, in case of EXT3NS, the CPU load is only a little. So, SMART can support more sessions for multimedia streaming.

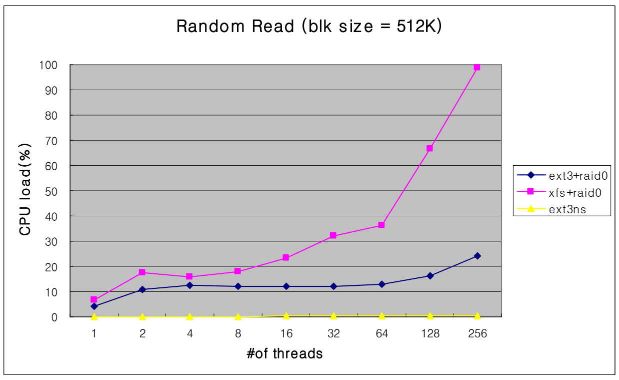


Fig. 22. CPU loads at random I/O

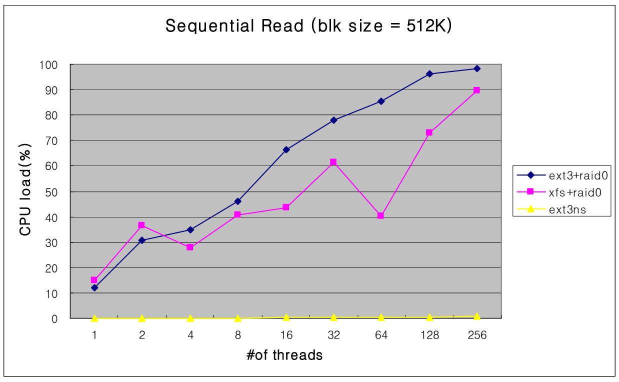


Fig. 23. CPU loads at sequential I/O

From Fig. 24 to Fig. 27 show the streaming performance with NS card and that without NS card. We used the MPEG-2 contents of 4Mbps and 10Mbps for performance evaluation. In the case without NS card, we used general NIC that has two 1GB network ports. As shown in the figure, when we use NS card, the maximum user count is bigger than without NS card due to CPU and network load.

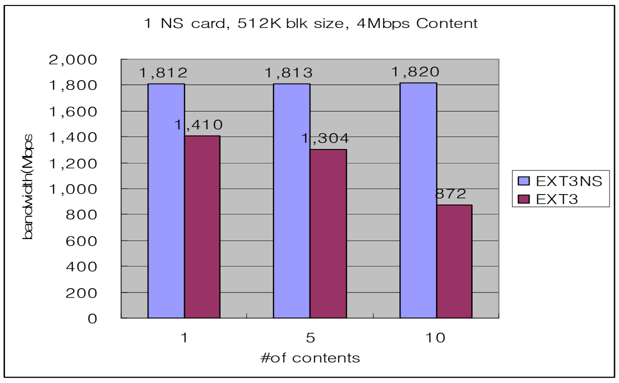


Fig. 24. Streaming Performance(4Mbps, 2-Gigabit port)

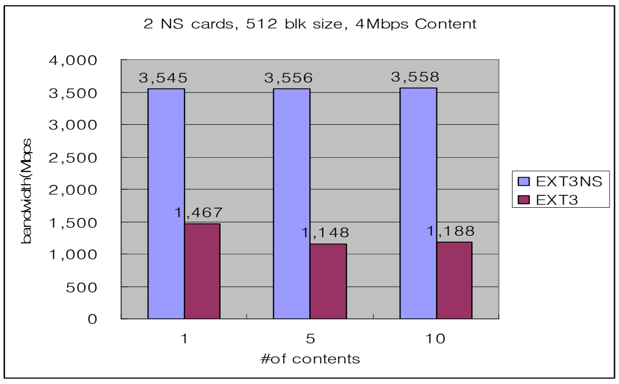


Fig. 25. Streaming Performance(4Mbps, 4-Gigabit port)

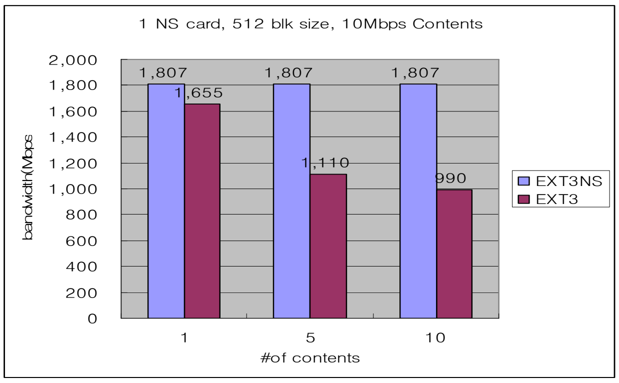


Fig. 26. Streaming Performance(10Mbps, 2-Gigabit Port)

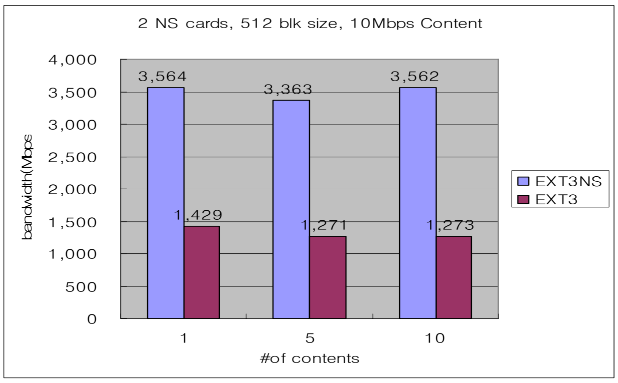


Fig. 27. . Streaming Performance(10Mbps, 4-Gigabit Port)

5. Conclusion

In this paper, we are proposing an efficient system for multimedia streaming services in terms of the design concepts, the characteristics and the system architecture including hardware and software. When designing the system, we had to consider the design goals and several design issues. The design issues for the system are the CPU utilization, zero-copy, disk throughput, multimedia file system, clustering solution and content distribution. Therefore, we used special hardware(NS card), special multimedia file system(EXT3NS), clustering solution and efficient content distribution mechanism.

We measured the performance on various situations and got desirable results. But, more refinement on the system for multimedia streaming services is required to enhance the overall performance and stability of system. Specially, we have modified our system to support the Fibre Channel model. SMART can not share among other volumes in the same node. So, we operated SMART with replication or dynamic loading in real time. After modifying, SMART is able to share the whole volumes within same cluster.

Furthermore, in order to provide total solution for multimedia streaming services, we have to design and implement the global server.

References

[1] Korea National Computerization Agency, Ministry of Information and Communication: 2002 Korea Internet White Paper (2002)

[2] Adaptec Company: Advantages of a TCP/IP offload ASIC. White paper

[3] Thomas Plagemann, Vera Goebel, Pal Halvorsen and Otto Anshus: Operating System Support for Multimedia Systems. The computer Communication Journal, Elsevier, Vol. 23, No. 3 (2000) 267-289

[4] Kasenna: Video Content Distribution: A Hybrid DCS/Caching Architecture for Broadband Video. White paper

[5] Subhabrata Sen, Jennifer Rexford, and Donald F. Towsley: Proxy Prefix Caching for Multimedia Streams. In Proc. of INFOCOM (1999) 1310-1319

[6] Stardust: Content Networking and Edge Services: Leveraging the Internet for Profit. White paper

[7] HCL Technologies: Content Delivery Networks: An Architectural Overview. White paper

[8] Gregory F. Pfister: In Search of Clusters, Prentice Hall PTR (1998)

[9] T. Schroeder, S. Goddard, and B. Ramamurthy: Scalable Web Server Clustering Technologies. IEEE Network, 14(3) (2000) 38-45

[10] Wensong Zhang: Linux Virtual Server for Scalable Network Services. In Proc. of the Ottawa Linux Symposium (2000)

[11] "Next Generation Internet Server Requirement Specifications V1.0", ETRI, 2002

[12] "Next Generation Internet Server Design Specification V1.0", ETRI, 2002

[13] "Media Server Development Report for Digital Cable Broadcast V1.0", ETRI, 2005

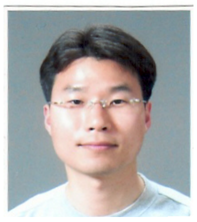
[14] Seok Soo Kim, Minseong Ju and Dae Joon Hwang, "An Algorithm for Formation and Confirmation of Password for Paid members on the Internet-based Telemedicine," Springer, LNCS 2105, pp. 334-340, June 2001.

[15] M.B. Abbott and L.L. Peterson: Increasing network throughput by integrating protocol layers. IEEE/ACM Transactions on Networking, Vol. 1, No. 5 (1993) 600-610

[16] Michael K. Johnson, "White paper : Red Hat's New Journaling File System : ext3", http://www.redhat.com/support/wpapers/redhat/ext3", 2001

[17] "XFS : A high-performance journaling filesystem", http://oss.sgi.com/projects/xfs/

Authors



Yuhyeon Bak

Received the B.S. and M.S. degrees in computer science from Pusan National University, Korea, 1996 and 1998. Since 1998, he has been working toward the Ph.D degree in computer science at Pusan National University, Korea. During 2000, he worked at KIDA (Korea Institute for Defense Analyses) in Seoul, Korea and he joined ETRI (Electronics and Telecommunications Research Institute), Daejeon, Korea in 2001. His research interests include multimedia content delivery, network-storage system, database and so on.

Kapdong Kim

Received the B.S. and M.S. degrees in computer science from the Chungnam National University, Korea, in 1998 and 2000, respectively. He joined ETRI, Daejeon, Korea in 2000. He had been worked in development of IMT-2000 system, Parlay and JAIN. During the years from 2002 to 2005, he currently involve in development of Next Generation Internet Server. Since 2003, he has been working toward the Ph.D degree in computer science at ChungNam National University, Korea. His research interests include high-speed networks architecture, quality of service in Internet and optical subscriber networks, MANET, multicast, network management, multimedia, middleware, and digital cable broadcast system.

Youngju Lee

Received the B.S. and M.S. degrees in computer engineering from Chonbuk National University, Korea in 1999 and 2001, respectively. Since 2001 he is working on ETRI and developed media streaming server. His main interests are computer system architecture such as network-storage system, distributed file system, multimedia database and so on.

Songwoo Sok

Received a B.S. degree in electronic engineering from Kyungpook University , Korea, 1999, and M.S. degree in electronic engineering from Kyungpook University, Korea, 2001 . In 2001 he joined the ETRI.

Changsoo Kim

Received a B.S. degree in computer science from Kwangwoon University, Korea, 1993, and M.S. degree in computer science from Sogang University, Korea, 1995. From 1995 to 1999, he worked at LG Electronics Inc., and from 1999, he is researching at ETRI.

Hagyoung Kim

Received the B.S. and M.S. degrees in electronics engineering from Kyungpook National University, Korea, in 1983, 1985, and Ph.D degrees in computer engineering from Chungnam National University, Korea, in 2003. He joined ETRI in 1988, and he has currently served as the Project Leader of the Media Streaming Research Team of ETRI. His current interests include computer architecture, high-speed networks architecture, multimedia, middleware, and digital cable broadcast system.

Myung-Joon Kim

Received a B.S. degree in computer science from Seoul National University, M.S. degree in computer science from KAIST, Korea in 1978, 1980 and Ph.D degree in computer science from Nansy 1 University, France in 1986. Since 1986 he is working on ETRI. His research interests are database, system software, software engineering.

Kyongsok Kim

Received a B.S. degree in international trade and M.S. degree in computer science from Seoul National University, 1977 and 1979, Korea and Ph D. degree in computer science from University of Illinois at Urbana-Champaign, in 1988. From 1998 to 1992, he was an assistant professor in University of North Dakota. He joined Pusan National University, Pusan, Korea, in 1992. His main research areas are database, multimedia, Hangul/Hanmal information processing, internet computing.