논문 98-23-11-15 한국통신학회논문지 '98-11 Vol.23 No.11

주문형 비디오 시스템에서 스트림 연계를 이용한 요구 스케쥴링 기법

정회원 한 금 희*, 김 종 훈**, 원 유 헌***

A Request Scheduling Strategy Using Stream Relay in VoD Systems

Kum Hee Han*, Jong Hoon Kim**, Yoo Hun Won*** *Regular Members*

요 약

주문형 비디오 시스템에서 시스템의 효율을 높이고, 많은 사용자들의 요구를 수용하기 위하여 한번의 디스크 검색으로 여러 사용자들의 요구를 만족시키기 위한 브릿징, 피기백킹, 배칭 등의 기법들이 연구되고 있다. 본 논문에서는 한번의 디스크 검색으로 더욱 많은 사용자들의 요구를 서비스하는 스케쥴링 기법을 제안하였다. 본 논문에서 제안한 요구 스케쥴링 기법에서는 서버가 모든 서비스를 처리하는 것이 아니라 해당 비디오를 저장하고 있는 클라이언트에게 작업을 분담하게 하여 서버의 부하를 감소시킨다. 이 논문에서 제안한 기법과 기존의 배칭 기법의 성능을 시뮬레이션으로 비교·실험한 결과 제안한 기법이 요구당 초기 지연시간과 서버의 처리량에 있어서 효율적인 성능을 나타낸다.

ABSTRACT

In the VoD system, several request scheduling strategies like Bridging, Piggybacking and Batching methods are known as a common way of utilizing the system resources. In this paper, we introduce a new request scheduling method, the Stream Relay Scheme, which processes multiple viewer's requests with the same stream retrieval. The SRS utilizes the client's buffer space in order to reduce the server's stream capacity and buffer consumption. Simulation are conducted to evaluate and compare the new scheme with the Batching scheme. The results have shown that the SRS decreases the initial latency and improves the server throughput.

I. 서 론

최근 오디오와 비디오 데이터를 처리할 수 있는 멀티미디어 워크스테이션이 출현되고 ATM[1]과 Myrinet [2]와 같은 초고속의 네트워크들이 개발됨에 따라 주문형 비디오 시스템이 고안되고 있다[3]. 주문형 비디오 시스템은 영화, 비디오 게임, 방송 프로그램 등 각종 영상물을 컴퓨터에 데이터 베이스화한 다음 일반 가입자들에게 네트워크를 통하여 원하는 멀티미디어 정보를 제공하는 시스템이다. 그러나 이와 같은 기술을 현실화하기 위해서는 대용량의 데이터를 디스크 기억장치에 효율적으로 저장하고 있다가 보다 많은 사용자들에게 그들이 원하는 서비스를 제공하는 주문형 비디오 서버 기술의 지원이 있어야 한다[4]. 이와 같은 지원을 위한 노력은 여러 방면에서 시도되고 있는데 브릿징, 피기백킹, 배칭 등과 같은 요구 스케쥴링 정책[5]이 이에 해당된다.

주문형 비디오 시스템에서는 일반적인 텍스트 데이터와는 달리 동일한 비디오에 대한 요구가 많이 발생할 것이므로 특정 비디오에 대한 요구가 발생했을 때 해당 비디오의 내용이 다른 클라이언트의 버퍼에 유지되고 있을 확률이 높을 것이다. 그러므로 이를 효율

* 가톨릭대학교 컴퓨터공학부(hkumhee@www.cuk.ac.kr) 정회원, ** ETRI 운영체제연구팀(jhkim@computer.etri.re.kr) 정회원
*** 홍익대학교 컴퓨터공학과(won@cs.hongik.ac.kr) 정회원
논문번호 : 98171-0416, 접수일자 : 1998년 4월 16일
※본연구는 1997학년도 가톨릭대학교 교비 연구비로 수행되었음.

적으로 활용하는 연구가 필요하나 주문형 비디오 시스템에서는 이를 고려한 연구는 진행된 바가 없다.

그러므로 본 논문에서는 이러한 점을 고려한 요구 스케쥴링 정책을 제안한다. 제안한 정책에서는 요구가 발생했을 때 무조건 서버가 서비스를 제공하는 것이 아니라 해당 비디오를 저장하고 있는 클라이언트가 있는지를 검색해서 있을 경우에 작업을 클라이언트에게 넘겨 동작하게 된다. 제안한 정책의 효과를 검증하기 위해서 시뮬레이션을 통해 배칭을 적용한 서버와의 성능을 비교한다. 성능 비교를 통해 본 논문에서 제안한 정책이 요구당 초기 지연시간과 서버의 부하에 있어서 효율적인 성능을 나타내는 것을 확인한다.

본 논문의 구성은 다음과 같다. 우선 Ⅱ장에서는 기존의 요구 스케쥴링 정책들에 대해 살펴보고 문제점을 지적한다. Ⅲ장에서 본 논문에서 제안한 스케쥴링 정책에 대해 설명하고 Ⅳ장에서는 성능 평가를 위한 실험 환경을 기술하며 실험 결과와 분석 내용을 설명한다. 그리고 Ⅴ장에서 결론과 향후 연구를 논의한다.

II. 관련연구

본 장에서는 본 논문과 관련된 연구로 기존의 요구 스케쥴링 정책들[5]에 대해 간단하게 살펴본다.

브릿징은 동일한 비디오에 대한 요구가 있을 때 가장 먼저 온 요구에 대해서는 디스크를 검색하지만 이 검색한 데이터를 메모리에 버퍼링 함으로 뒤에 발생한 요구들에 대해서는 디스크가 아닌 메모리에서 서비스해 주는 정책이다. 그러나 이 기법은 매우 많은 양의 메모리 공간을 필요로 하는 단점이 있다. 피기백킹[6]은 비디오의 상영속도가 5%정도 빨라지거나 느려지는 것은 사용자가 감지하기 힘들다는 점을 이용한 방법으로 서비스 중에 동일한 비디오에 대한 요구가 들어오면 앞서고 있는 비디오의 속도를 늦추고 새로 시작하는 비디오의 속도를 빠르게 조정하여 두 사용자에게 동일한 스트림으로 서비스하는 기법이다. 그러나 서버가 검색해야 하는 데이터의 양의 변화가 매우 심하게 되어 디스크 입출력 양이 큰 폭으로 변하고, 각기 다른 속도로 검색하는데에 서버의 능력이 요구되는 문제점이 있다. 또한 널리 이용되는 압축 기법인 MPEG 형식의 파일에는 적용하기가 힘들다는 문제점도 있다. 배칭[7, 8]은 배칭 간격 동안 동일한 사용자 요구들을 모아 하나의 입출력 스트림으로 서비스하는 방법이다. 이러한 배칭은 브릿징이나 피기백킹처럼 큰 메모리의 요구도 없고 서버가 데이터의 검색 속도를 빠르거나 느리게 하는데 능력을 소모하지 않아도 된다는 장점이 있다. 하지만 모든 비디오에 동일하게 배칭 간격이 적용되므로 클라이언트가 불필요하게 기다리는 일이 생길 수 있다. 즉 인기 없는 비디오를 요구한 사용자 또는 평일 새벽시간대처럼 사용자가 거의 없는 시간에 요구한 사용자가 이에 해당된다.

브릿징, 피기백킹, 배칭등의 요구 스케쥴링 정책은 한 번의 디스크 접근으로 보다 많은 사용자들에 대한 서비스를 제공하려는 방법이다. 최근 클라이언트의 성능이 놀라울 정도로 발전하여 일반 분산 시스템 환경에서는 클라이언트 자원을 효율적으로 활용하는 연구[9, 10]가 진행되고 있는데 주문형 비디오 시스템 환경에서는 이에 대한 연구 결과나 진행이 미진한 상태에 있다. 앞서 언급한 스케쥴링 정책들도 이를 간과하고 예전의 중앙집중식 컴퓨팅과 유사하게 서버의 능력에만 업무의 대부분을 의존하고 있다. 그러므로 본 논문에서는 클라이언트 자원을 활용하는 새로운 스케쥴링 정책을 제안한다.

III. 스트림 연계를 고려한 스케쥴링 기법

본 장에서는 시스템 구조에 대해서 살펴보고 본 논문에서 제안한 스트림 연계법에 대해 살펴본다.

3.1 시스템 구조

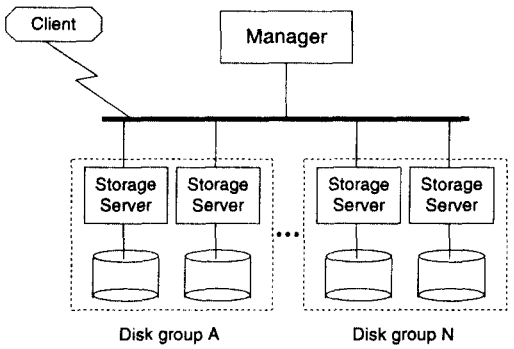

그림 1. 시스템의 구조

그림 1은 본 논문에서 가정하는 주문형 비디오 시스템의 구조를 나타낸 것으로 각 사용자를 클라이언트로 하고 서비스 제공자를 서버로 하는 클라이언트-서버 구조이다. 특히 서버는 저장 서버(storage server)와 관리자(manager)로 구성되어 있다. 저장 서버는 비

디오 데이터를 검색하여 요청한 클라이언트에게 데이터를 보내주는 일을 담당하는 부분으로 디스크 효율을 높이기 위해 그룹화 하여 그룹내 다수의 디스크들에 스트라이핑[11, 12, 13]하여 데이터를 저장한다. 그룹내 요구들에 대한 서비스는 라운드-로빈 방식으로 이루어진다. 관리자는 클라이언트의 요구들에 대한 관리와 처리를 하는 부분으로 요구에 대한 서비스가 가능한지를 검사하여 서비스를 지시하거나 큐잉하여 저장·관리하는데 구체적인 동작은 뒤에서 언급한다.

3.2 스트림 연계법 (SRS:Stream Relay Scheme)

주문형 비디오 시스템에서는 일반적인 텍스트 데이터와는 달리 동일한 비디오에 대한 요구가 많이 발생하므로 특정 비디오에 대한 요구가 발생했을 때 해당 비디오 스트림이 다른 클라이언트의 버퍼에 유지되고 있을 확률이 높다. 이러한 점을 고려하여 스트림 연계법에서는 요구가 발생했을 때 무조건 서버가 서비스를 제공하는 것이 아니라 해당 비디오 스트림을 버퍼에 저장하고 있는 클라이언트가 있는지를 검색하여 있을 경우에 작업을 클라이언트에게 넘겨 동작하게 된다. 그럼으로써 서버의 부하가 감소하게 되고 더욱 많은 사용자들에 대한 서비스를 제공하게 된다. 최근 클라이언트의 성능은 놀라울 정도로 발전하여 자신의 버퍼에 저장되어 있는 비디오 스트림을 다른 클라이언트에게 보내는 단순한 작업을 처리하는 것은 전혀 문제가 되지 않는다.

서비스하는 비디오의 시작 부분을 버퍼에 저장하고 있는 클라이언트가 있는지에 대한 정보를 관리자가 테이블에 유지하고 있다. 관리 테이블은 매우 간단한 구조를 지니고 있다. 만약 200가지 종류의 비디오를 서비스하는 시스템이라면 크기가 200인 두 개의 정수형 배열로 이루어진 테이블 하나만 있으면 된다. 만약 1번 비디오를 10번 클라이언트가 보기 시작하면 TABLE[1][1]에 클라이언트 번호 10을 저장하고 TABLE[1][2]에는 클라이언트가 비디오를 보기 시작한 시간을 저장하면 된다. 그리고 주기적으로 TABLE[i][2]의 내용을 검색하여 비디오 시작 부분이 TABLE[i][1]번 클라이언트의 버퍼에서 쫓겨날 시간이 되면 TABLE[i]에 NULL을 저장한다.

다음은 스트림 연계법의 동작을 요구의 형태에 따라 분류하여 나타낸 것이다.

3.2.1 클라이언트로부터 발생한 요구

클라이언트로부터 요구가 발생했을 때 요구된 비디오의 시작 부분이 다른 클라이언트의 버퍼에 있는지의 여부에 따라 동작이 달라진다.

요구된 비디오의 시작 부분이 다른 클라이언트 버퍼에 있을 경우 그림 2의 예를 통해 동작을 살펴본다. 클라이언트 A로부터 3번 비디오에 대한 요구가 들어오면 관리자는 관리 테이블을 검색하여 3번 비디오의 시작 부분을 버퍼에 저장하고 있는 클라이언트가 있는지를 점검한다. 만약 해당 비디오를 저장하고 있는 클라이언트(B)가 있을 경우에 관리자가 자료를 전송하라는 명령을 클라이언트 B에게 보낸다. 그러면 클라이언트 B에서 해당 비디오 스트림을 클라이언트 A에게 전송하게 된다. 그리고 관리자는 관리 테이블의 내용을 B에서 A로 갱신하고 시간 정보도 갱신한다.

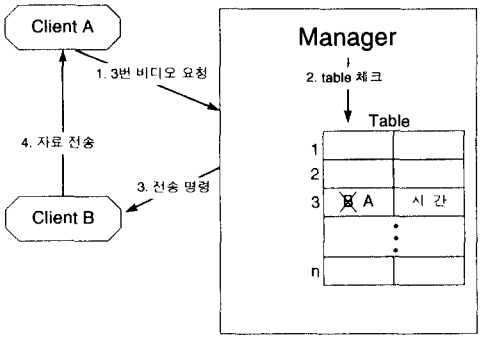

그림 2. 클라이언트 A가 요구한 3번 비디오가 클라이언트 B의 버퍼에 있을 경우의 동작

요구된 비디오의 시작 부분이 다른 클라이언트 버퍼에 없을 경우 해당 요구를 일단 관리자의 대기 큐에 큐잉한다.

그림 3은 클라이언트로부터 요구가 발생했을 때의 동작을 알고리즘으로 나타낸 것이다.

```
while (클라이언트로부터 요구 발생) {
  table check;
  if (요구된 비디오의 시작 부분이 다른 클라이언트(source_client)에
    존재) {
      source_client에게 자료전송 요구;
      table 갱신;
  }
  else {
      enqueue;
  }
}
```

그림 3. 클라이언트로부터 요구가 발생했을 때의 동작 알고리즘

3.2.2 큐잉되었던 요구

주기적으로 대기 큐에서 요구를 디큐하여 처리하는데 다음의 과정을 따른다.

요구된 비디오의 시작 부분이 다른 클라이언트 버퍼에 있을 경우 관리 테이블을 검색하여 해당 비디오의 시작 부분을 저장하고 있는 클라이언트가 있으면 그림 2와 동일하게 처리한다.

저장 서버가 요구를 처리할 수 있을 경우 만약 요구된 비디오의 시작 부분을 저장하고 있는 클라이언트가 없을 경우에는 저장 서버가 요구를 승인할 수 있는지를 판단한다. 승인이 가능하면 요구를 저장 서버로 전송하여 저장 서버에서 비디오 스트림을 클라이언트로 전송하도록 한다. 그리고 관리자는 관리 테이블의 내용을 갱신한다.

처리가 불가능한 경우 만약 저장 서버가 요구를 처리할 수 없을 경우에는 큐에서 대기한다.

그림 4는 이를 그림으로 나타낸 것이고, 그림 5는 큐잉된 요구들을 처리하는 알고리즘이다.

IV. 실 험

본 장에서는 성능 평가를 위한 실험 환경을 살펴보고 실험 결과와 그의 분석 내용을 설명한다.

4.1 실험 환경

본 논문에서는 스트림 연계법(SRS)과 FCFS 방식을 적용한 배칭 기법의 성능을 비교하였다. 실험에 사용된

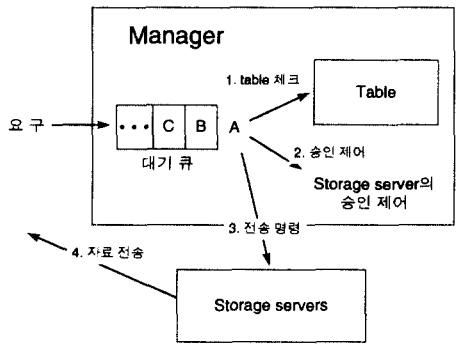

그림 4. 디큐된 요구를 처리하는 동작

```
while (큐에 요구가 존재) {
    dequeue;
    table check;
    if (요구된 비디오의 시작 부분이 다른 클라이언트에 존재) {
        source_client에게 자료전송 요구;
        table 갱신;
    }
    else if (저장 서버가 서비스 가능) {
        저장 서버가 자료 전송;
        table 갱신;
    }
    else {
        큐에서 대기;
    }
}
```

그림 5. 큐잉된 요구들을 처리하는 알고리즘

스트림 연계법에서는 본 정책을 사용함으로 얻어지는 이득을 정확하게 파악하기 위하여 전혀 배칭을 하지 않았다. 그리고 배칭 기법은 배칭 간격을 다양하게 하여 실험한 결과 중에서 가장 효율적인 성능을 나타낸 것이다.

실험은 시뮬레이션을 통해 평가하였다. 사용된 요구 작업부하는 실험의 편리함을 위하여 가상적으로 만들어 사용하였다. 가상 요구 작업부하는 주어진 수만큼의 요구를 생성할 수 있도록 되어있으며 작업부하는 요구하는 클라이언트의 번호와 원하는 비디오 번호로 이루어져 있다. 본 실험에서는 20,000개의 요구로 이루어진 작업부하를 만들어 이용하였다.

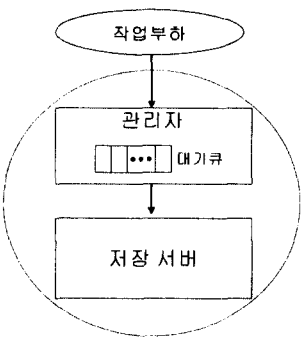

그림 6. 시뮬레이터의 구성

시뮬레이터는 C++를 사용하여 개발하였으며 구성은 그림 6과 같이 작업부하를 입력하는 부분과 관리자와 저장 서버로 구성된다. 작업부하에서 요구들은 주기적으로 서버 시뮬레이터로 입력되며 시뮬레이터에서

관리자는 대기큐를 두어 들어오는 요구들을 저장하며 큐에 들어온 순간부터 시간을 측정한다. 시뮬레이터의 동작은 알고리즘과 동일하게 동작하며 시뮬레이터에서는 각 요구에 대한 초기지연시간과 서버가 동작한 시간을 측정한다.

표 1. 시뮬레이션 파라미터

비디오의 개수	100개
각 비디오의 상연시간	100분
서버 스트림 용량	200개
라운드 시간	1초
요구 발생 간격	0.5초, 1초, 1.5초, 2초
데이터의 양(각 라운드, 각 클라이언트	192KB
192KB 데이터의 네트워크 전송시간	10 miliseconds

시뮬레이션 파라미터는 표 1과 같이 서버가 제공하는 비디오의 개수는 100개, 각 비디오의 길이는 100분, 서버 스트림 용량은 200개로 하였다. 그리고 한 라운드는 1초로 가정하였으며 한 라운드당 한 클라이언트에게 보내는 데이터의 양은 MPEG-1[14]을 기준으로 192KB로 하였다. 192KB 데이터가 네트워크를 통해 전송되는 시간은 ATM/155Mbps[1]를 고려하여 네트워크 오버헤드를 포함하여 9.8 miliseconds인데 10 miliseconds로 고정시키고 실험하였다. 각 정책에 대한 성능 비교 척도로는 요구당 초기 지연시간과 서버의 처리량을 사용하였으며 시뮬레이션에서 클라이언트 요구 발생 간격을 다양하게 하여 실험하였다. 단, VCR과 같은 동작은 고려하지 않았다.

4.2 실험 결과

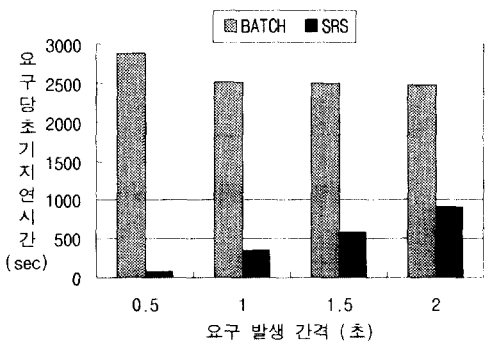

그림 7. 정책에 따른 초기지연시간

그림 7은 스케쥴링 정책에 따른 초기지연시간을 나타낸 것이다. 그림에서 BATCH는 배칭 기법을 나타낸 것이고 SRS는 스트림 연계법을 나타낸 것으로 각 클라이언트 버퍼의 용량은 압축된 데이터 5분 동안의 분량을 저장할 수 있는 크기이다. 그리고 X축은 클라이언트로부터 발생하는 요구 발생 간격을 의미한다. 그림의 결과에서 몇 가지 사항을 관찰할 수 있다. 우선 모든 경우에 있어서 스트림 연계법이 배칭 기법에 비해 매우 효율적인 초기 지연시간을 나타내는 것을 볼 수 있다. 이는 자주 요구되는 비디오에 대한 서비스는 클라이언트가 처리함으로 전체 시스템 입장에서 더욱 많은 클라이언트들에 대한 서비스를 제공할 수 있기 때문이다. 다음으로 스트림 연계법에서는 요구 발생 간격이 커짐에 따라 요구당 초기 지연 시간이 지연되는 것을 확인할 수 있는데 이는 요구 발생 간격이 커지면 커질수록 비디오의 시작 부분을 클라이언트의 버퍼에 저장하고 있는 동안 발생하는 요구의 발생 횟수가 감소하기 때문이다.

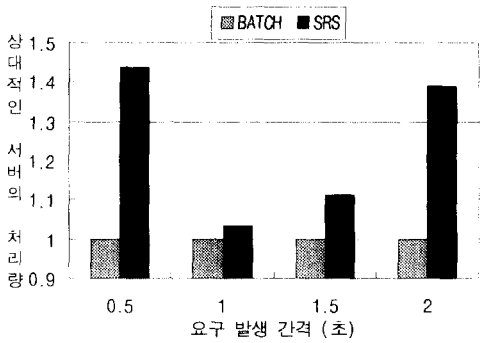

그림 8. 정책에 따른 서버의 상대적인 처리량

그림 8은 정책에 따른 서버의 처리량을 상대적으로 나타낸 것이다. 정책간의 성능에 있어서는 스트림 연계법이 배칭 기법에 비해 4~45% 가량 서버의 처리량이 증가하는 것을 볼 수 있다. 그러므로 본 논문에서 제안한 스트림 연계법은 클라이언트 입장에서 초기지연시간이 짧아짐은 물론이고 서버의 처리량 측면에 있어서도 좋은 성능을 나타내는 효율적인 정책임을 알 수 있다. 특히 본 실험의 스트림 연계법에는 배칭을 전혀 적용하지 않았음에도 불구하고 이와 같이 효율적인 성능을 나타내므로 배칭 또는 피기백킹과 같은 정책을 통합 적용하면 더욱 효율적인 성능을 나타낼 것이다.

V. 결 론

최근 시스템 환경의 발전으로 주문형 비디오 시스템 개념이 시선을 끌고 있다. 이러한 주문형 비디오 시스템에서 보다 많은 사용자들에게 서비스를 제공하기 위해 브릿징, 피기백킹, 배칭 등과 같은 요구 스케쥴링 정책이 제안되었다. 본 논문에서도 한번의 디스크 검색으로 더욱 많은 사용자들의 요구를 서비스하는 스케쥴링 기법을 제안하였다. 주문형 비디오 시스템에서는 일반적인 텍스트 데이터와는 달리 동일한 비디오에 대한 요구가 많이 발생하므로 특정 비디오에 대한 요구가 발생했을 때 해당 비디오의 내용이 다른 클라이언트의 버퍼에 유지되고 있을 확률이 높다. 본 논문에서 제안한 정책은 이러한 점을 고려하였다. 제안한 정책에서는 요구가 발생했을 때 무조건 서버가 서비스를 제공하는 것이 아니라 해당 비디오를 저장하고 있는 클라이언트가 있는지를 검색해서 있을 경우에 작업을 클라이언트에게 넘겨 동작하게 된다. 제안한 정책의 효과를 검증하기 위해 시뮬레이션을 통해 배칭 기법과의 성능을 비교하였다. 성능 비교를 통해 본 논문에서 제안한 정책이 요구당 초기 지연시간과 서버의 처리량에 있어서 매우 효율적인 성능을 나타내는 것을 확인하였다.

본 논문에서 제안한 정책은 클라이언트의 버퍼만을 활용하는데 앞으로 서버의 버퍼도 함께 고려하여 동작하는 정책으로 확장할 것이다.

참 고 문 헌

1. D. E. McDysan and D. L. Spohn, *ATM : Theory and Application*, McGraw-Hill, 1995.
2. N. J. Boden et al., "Myrinet: A Gigabit-Per-Second Local Area Network," *IEEE Micro*, 15(1):29-36, February 1995.
3. A. Heybey, M. Sullivan, and P. England, "Calliope: A Distributed, Scalable Multimedia Server," In *Proceedings of the USENIX 1996 Annual Technical Conference*, January 1996.
4. D. Gemmell, H. vin, D. Kandlur, P. Rangan, "Multimedia Storage Servers: A Tutorial and Survey", *IEEE Computer*, 1995.
5. L. Golubchik, J. Lui, and R. Munts, "Reducing I/O Demand in Video-On-Demand Storage Servers," *ACM Sigmetrics Conference*, pages 25-36, May 1995.
6. C. Aggarwal, J. Wolf, and P. Yu, "On Optimal Piggyback Merging Policies for Video-on-Demand Systems," *Technical Report, IBM RC 20337*, February 1996.
7. A. Dan, D. Sitaram, and P. Shahabuddin, "Scheduling Policies for an On-Demand Video Server with Batching," In *Proceedings of the 2nd ACM Multimedia Conference*, pages 25-32, 1994.
8. H. Shachnai and P. Yu, "The Role of Wait Tolerance in Effective Batching: A paradigm for Multimedia Scheduling Schemes," *IBM Research Report, RC 20038*, 1995.
9. A. Leff, J. Wolf, and P. Yu, "Efficient LRU-Based Buffering in a LAN Remote Caching Architecture," *IEEE Transactions on Parallel and Distributed Systems*, 7(2):191-206, February 1996.
10. M. Dahlin, R. Wang, T. Anderson, and D. Patterson, "Cooperative Caching: Using remote client memory to improve file system performance," In *Proceedings of the First Symposium on Operating System Design and Implementation*, pages 267-280, November 1994.
11. T. Anderson, M. Dahlin, J. Neefe, D. Patterson, D. Roselli, and R. Wang, "Serverless Network File System," *ACM Transactions on Computer Systems*, 14(1):41-79, February 1996.
12. J. Hartman and J. Ousterhout, "The Zebra Striped Network File System," *ACM Transactions on Computer System*, 13(3):274-310, August 1995.
13. P. Chen, E. Lee, G. Gibson, R. Katz, and D. Patterson, "RAID : High-Performance, Reliable Secondary Storage," *ACM Computing Surveys*, 26(2): 145-185, June 1994.
14. O. Rose, "Statistical Properties of MPEG Video Traffic and Their Impact on Traffic Modeling in ATM Systems," *University of Wurzburg Research Report Series No. 101*, February 1995.

한 금 희(Kum Hee Han) 정회원

1971년 : 성심여자대학교 화학과 이학사

1981년 : 미국 RPI 대학교 computer science 이학석사

1982년~현재 : 가톨릭대학교 컴퓨터공학부 교수

1992년~현재 : 홍익대학교 대학원 전자계산학과 박사과정

<연구분야> 멀티미디어 시스템, 프로그래밍 언어론, 컴파일러 이론

김 종 훈(Jong Hoon Kim) 정회원

한국통신학회논문지 제22권 제6호 참조

원 유 헌(Yoo Hun Won) 정회원

한국통신학회논문지 제22권 제6호 참조