실시간 VOD 서버 시뮬레이터 설계*

A Design of Real-time VOD Server Simulator

정지영**, 김성수**
Ji Yung Chung, Sungsoo Kim

Abstract

In recent years, significant advances in computers and communication technologies have made multimedia services feasible. As a result, various queuing models and cost models on architecture and data placement for multimedia server have been proposed. However, these analytical techniques use only probabilistic models to represent the behavior of a system, and then they have several limitations like accuracy. Simulation is a viable alternative to analytical model. It avoids many of the limitations associated with analytical techniques, allowing for more precise representation of system attributes like workload in program code. In this paper, we propose a simulation test bed that can evaluate performance of real-time multimedia server by using simulation model.

* This work is supported in part by the Ministry of Information & Communication of Korea. (Supported Project of University Foundation Research<2000>" supervised by IITA)

** 아주대학교 정보통신전문대학원

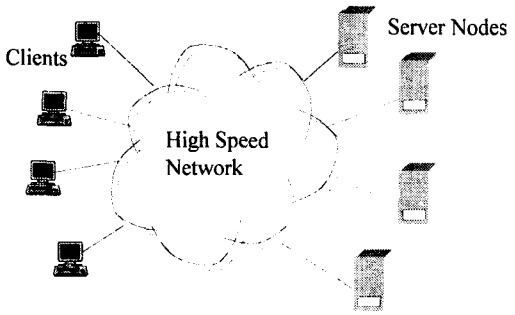
1. 서 론

컴퓨터 기술의 발전과 데이터 압축 기법의 향상은 다양한 멀티미디어 서비스를 가능하게 하였으며 통신 기술의 비약적인 진보는 이러한 서비스의 실시간 제공을 가능하게 하고 있다. 실시간 VOD(Video On Demand) 응용의 경우 수천에서 수만에 이르는 고객의 요구를 수용하기 위해서는 대용량의 디스크와 더불어 데이터의 전송률 및 대역폭이 보장되어야 한다. 이와 관련하여 시스템의 확장성 및 신뢰도를 증가시키기 위한 서버의 구조와 스트라이핑(striping) 방식에 관한 연구가 활발히 진행되고 있으며 가능한 많은 고객을 수용하기 위해 디스크 스케줄링의 개발이 진행되어 왔다[1][2][3].

실시간 시스템의 경우 제한시간을 어겼을 경우의 피해 정도에 따라 경성 실시간 시스템(hard real-time system)과 연성 실시간 시스템(soft real-time system)으로 나눌 수 있는데 멀티미디어 시스템의 경우 제한시간을 어겼다 할지라도 큰 피해가 없으므로 연성 실시간 시스템의 범주에 속하나 제한시간을 만족하지 못한 데이터는 의미가 없으므로 연성 실시간 시스템 분류 중 다소 엄한 시간 제약성을 갖는다. 특히 오디오나 비디오 매체는 시간 제약 조건을 만족하여야만 의미를 가지는 오디오 샘플이나 비디오 프레임들로 구성되어 있기 때문에 멀티미디어 서버의 설계는 전통적인 파일서버와는 달리 실시간 특성이 고려되어야 한다. 즉, 이 서비스를 제공하는 정보시스템은 동시에 많은 고객을 처리할 수 있는 고성능의 저장 서버를 필요로 한다[4][5][6].

서버의 구조는 크게 초병렬 대용량 컴퓨터 시스템과 중소형 컴퓨터에 의한 분산 시스템으로 분류할 수 있으며, 데이터의 배치방법에 있어서는 스트림을 각 노드에 독립적으로 배치하는 방법과 시스템의 모든 노드에 데이터를 스트라이핑하는 방식이 있다. 각각의 경우는 모두 장단점이 있으며 응용 범위와 분야에 따라 적합한 구조가 선택 될 수 있다. <그림 1>은 일반적인 멀티미디어 서비스 시스템의 구조를 보여주고 있다.

여러 대학과 연구기관에서 멀티미디어 서버의 구조 및 데이터 배치에 따른 수학적인 모델로서 다양한 큐잉 모델이 개발되었으며 비용과의 관계를 고려한 비용모델이 제시되기도 하였다.



<그림 1> 멀티미디어 서비스 시스템

이러한 수학적인 모델링은 비용 면에서 장점이 있으나 전통적으로 고 수준에서 시스템을 추상화하는데 비해 시뮬레이션 모델링은 저 수준에서 시스템을 추상화하여 시스템 성능을 평가한다. 본 논문에서는 실시간 멀티미디어 서버의 시뮬레이션 모델링을 통해 수학적인 모델링보다 더욱 정확하게 시스템의 성능을 측정하는데 그 목적이 있다.

논문의 구성으로는 2장에서 멀티미디어 서버의 구조와 데이터 배치에 따른 기존의 연구를 분류하여 살펴보고 몇몇 수학적인 모델링의 한계를 살펴본다. 3장에서는 개발된 시뮬레이터를 구성하는 시뮬레이션 모델과 프로그램 구조에 대하여 설명한다. 4장에서는 본 논문에서 개발한 시뮬레이터를 이용한 다양한 시스템 성능평가 결과를 보여주고 5장에서 결론을 맺는다.

2. 멀티미디어 서버 구조

2.1 병렬 비디오 서버

실시간 멀티미디어 시스템을 구성하는데 있어 단일 서버에 의한 접근 방식은 몇 가지 고려요소가 있는데, 첫 번째 요소로서 시스템의 확장가능

성(scalability)을 들 수 있다. 멀티미디어 서버 구조는 고객의 요구가 시스템의 용량을 초과하는데 대비하여 확장가능성이 용이하여야 한다. 두 번째 요소로는 결함 허용의 문제를 들 수 있는데 단일 서버는 서버가 실패했을 경우 전체 고객이 서비스를 받을 수 없게 된다.

이러한 확장가능성과 결함 허용의 문제를 동시에 효과적으로 해결하기 위하여 시스템이 다수의 프로세싱 노드의 집합으로 구성되어 있고 각각의 노드에 디스크 배열이 연결되어 있는 병렬 비디오 서버 구조가 연구되어 왔는데 이러한 구조는 단일 서버에 비해 확장성이 용이하며 서버 노드들 중에 하나가 다운되었을 경우, 다운된 노드에 연결되어 있는 고객들만이 서비스를 받지 못하게 된다[7]. 이러한 구조의 예로서 각각의 노드를 데이터의 전달을 담당하는 전달 노드와 데이터의 저장을 제공하는 저장 노드로 나누어 서비스하는 이단 구조(two-tier)가 제안되었다[8][9].

마이크로 소프트의 타이거 비디오 파일 서버는 컵(cub)이라 불리는 각각의 컴퓨터들이 고속의 네트워크에 의해 연결된 실시간 분산 시스템으로 데이터 미러링(mirroring) 방식을 이용하여 결함 허용을 하는 서버 구조이다[10]. 또한 확장가능한 비디오 서버 구조를 위해 제안된 서버 어레이는 여러 개의 서버 노드에 각각의 비디오를 스트라이핑하여 저장하는 방식으로 디스크 배열 개념을 노드수준에서 고려하였다. 만일 스트림이 단일 노드 서버에 저장되어 있다면 서버의 실패는 곧 그 서버에 연결되어 있는 모든 연결을 잃어버리지만 이러한 구조에서 고객은 단지 적은 양의 품질 손실만을 경험할 뿐이며 단일 서버 노드가 가지고 있는 시스템 버스 대역폭의 한계를 벗어날 수 있다.

이러한 구조에서 신뢰도 향상을 위하여 노드 실패를 허용하는 방법이 제안되었는데 노드들 중의 하나가 실패하면 패리티 정보를 가지고 있는 노드가 실패한 노드에 대한 패리티 정보를 클라이언트에게 보내고 클라이언트에서는 XOR연산을 통하여 데이터를 복구한 뒤 재생하게 된다[11].

비디오를 배치하는 방법에 있어서도 각각의 노드에 독립적으로 배치하는 방법은 구현이 용이하고 관리하기에 좋은 장점이 있기는 하나 고객의 요청이 인기 있는 비디오에 집중 될 경우 노드들의 부하가 일정하지 않게 된다. 인기 있는 비디오를 중복 배치함으로써 해결하는 방법이 있기는 하나 기억장소의 낭비가 생기게 되고 인기 있는 비디오는 시간에 따라 변한다는 점을 고려하면 효과적인 대안이 될 수 없다. 이와 같이 만일 비디오가 여러 디스크에 스트라이핑 되어 있지 않다면 동시에 서비스 받을 수 있는 고객의 수는 비디오를 저장하고 있는 디스크의 대역폭에 의해 영향을 받는다.

이에 반해 시스템의 모든 노드에 비디오를 배치하는 방법은 특정 비디오에 인기가 집중된다 할지라도 모든 노드가 균일한 작업부하를 가지므로 실제적으로 높은 대역폭을 가질 수 있는 장점이 있다[7]. 또한 서버 수준에서의 스트라이핑 방법은 서버노드가 실패하더라도 나머지 서버들에 저장되어 있는 패리티 정보를 이용해 복구가 가능하며 최악의 경우라 할지라도 단지 품질의 저하만 가져오게 된다. 물론 이 경우 디스크 배열과 마찬가지로 실패된 서버는 가능한 빨리 복구되는 것이 요구된다[11]. 본 논문에서는 병렬 비디오 서버 구조를 대상으로 하였으며 비디오가 시스템의 모든 노드에 스트라이핑 되어 있는 시스템을 시뮬레이션 한다.

2.2 관련연구

시뮬레이션은 시스템을 표현하기 위해 확률적인 모델은 물론 행위적인 모델링도 사용할 수 있으므로 매우 유용하며 더욱 정확히 표현 가능하기 때문에 수학적 모델링의 많은 한계들을 피할 수 있게 한다[12][13].

실시간 비디오 서버의 성능을 평가하기 위한 연구로 피드백을 갖는 큐잉 네트워크 모델링이 제안되기도 하였다[14]. 이 모델에서 고객은 주어진 도착률로 서비스를 요청한 후 원하는 데이터 패킷을 전송 받을 때까지 서버내의 자원을 할당

받게 되며 주어진 확률에 의해 서비스를 종료한다. 그러나 실제로 고객의 서비스 종료는 확률에 의해 정해지는 것이 아니라 비디오의 모든 패킷이 전달되어야 종료된다. 또한 데이터가 모든 노드를 통해 스트라이핑 되어 있지 않고 각 노드가 독립적으로 서비스를 수행하는 시스템을 대상으로 열린 큐잉 네트워크 모델링을 제안하여 패킷 손실률과 같은 실시간 성능을 측정가능하게 한 연구도 있다[15]. 여기에서 서버노드들간의 부하는 균일하다고 가정하였으나 앞에서 언급한 바와 같이 인기 있는 비디오는 시간에 따라 변한다는 점을 고려하면 실제로 각 노드들의 부하는 균일할 수 없다.

앞서 언급한 이단 구조에서는 모델 설정시 비디오를 저장하고 있는 디스크를 단일 대기열로 가정하였으나 실제로 노드에 연결되어 있는 디스크는 다수인 점을 고려하면 정확도가 다소 떨어지게 된다. 또한 패킷 손실률이나 디스크 이용률 등을 얻는 과정에서 동시에 디스크 스케쥴링 알고리즘의 영향을 고려한 수학적인 모델링은 아직 존재하지 않는다[16]. 이러한 문제점들은 모두 수학적인 접근방식의 한계로 본 시뮬레이터에서는 프로그램의 코드를 통해 정확히 비디오의 패킷 수 만큼을 시스템에 요구하며 여러 개의 디스크가 노드에 연결되어 있는 구조를 표현함으로써 정확성을 높였다.

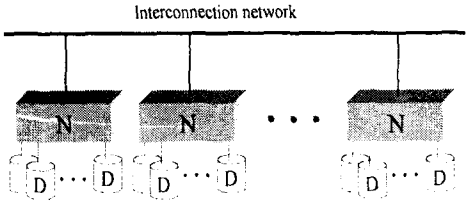
3. VOD 시뮬레이터

3.1 시스템 모델

개발된 시뮬레이터는 실시간 VOD 서버의 성능을 분석하기 위한 도구로서 수학적인 모델링보다 더 정확하고 용이하게 성능을 평가할 수 있다. 특히 디자인 단계에서 시스템을 추상화하여 정확히 그리고 효과적으로 성능을 측정함으로써 실제 개발시의 비용을 줄일 수 있다.

기존의 실시간 시스템을 위한 시뮬레이터나 통신용 시뮬레이터로도 유사한 시뮬레이션이 가능하나 이를 위해서는 고객이 VOD 서버에 대한 상세한 이해가 필요하고 성능 평가를 위해 직접 모델링을 해야 하며 시뮬레이터 사용에 대한 방법을 익혀야 하는 등의 단점이 있다. 이러한 시뮬레이터를 이용할 경우 더욱 다양한 구조의 시뮬레이션이 수행 가능하다는 점은 장점으로 들 수 있다. 제안된 시뮬레이터는 병렬 구조를 가지고 있는 VOD 서버만을 대상으로 하므로 다양한 구조의 시뮬레이션이 어려운 단점이 있으나 VOD 서버만의 특징을 시뮬레이션 할 수 있다는 점에서 유용하다. 또한 고객이 시스템의 상세한 동작을 모른다 할지라도 기본적인 사양만 알면 시뮬레이션이 가능하다.

<그림 2>는 본 논문에서 고려하는 시스템 모델로서 하나 이상의 디스크(D)들이 프로세싱 노드(N)에 각각 연결되고 서로 네트워크를 통해 연결되어 있는 구조를 지니고 있다.
본 논문을 통해 개발된 시뮬레이터는 이러한 모델에 의해 표현될 수 있는 대부분의 VOD 시스템을 시뮬레이션 함으로써 실시간 성능을 측정한다. 시스템의 작업부하는 입력 파라미터에 의해 주어지는 고객의 도착률과 영화 당 패킷 수에 따라 결정되며 시뮬레이션 시간 동안 각 서버노드에 패킷을 요구한다. 이때 비디오의 상영률(playback rate) 또한 서버가 처리해야 할 패킷 수에 영향을 주게 되며 각각의 서버 노드에 요구된 패킷들은 하나의 라운드를 주기로 서비스 되도록 하였다.



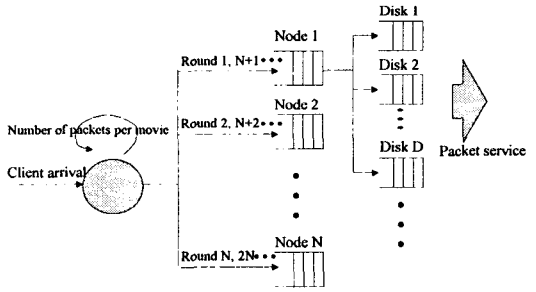
<그림 2> 시스템 모델

비디오 상영의 주기적인 특성으로 인해 서버는 고정된 시간의 라운드마다 요구된 패킷들을 서비스한다[17]. 하나의 라운드에서 디스크로부터

읽혀진 패킷들은 버퍼에 저장되고 그 다음 라운드에서 동시에(concurrently) 서비스된다. 한 편의 비디오는 블럭 단위로 모든 노드에 균일하게 스트라이핑 되어 있다고 가정하였으므로 시스템 내의 노드들은 거의 균일한 부하를 갖게 된다.

3.2 시뮬레이터의 동작

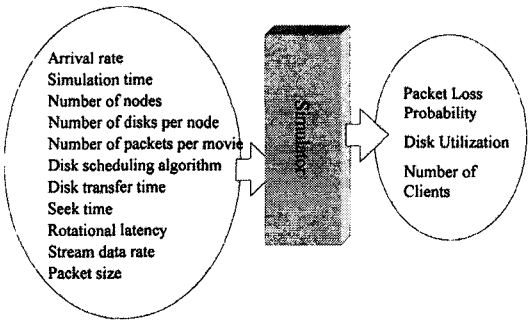
시스템이 N개의 노드로 이루어져 있고 패킷이 노드 단위로 스트라이핑 되어 있는 경우 비디오 요구가 받아들여지면 첫 번째 패킷은 첫 번째 노드에 요구되고 두 번째 패킷은 두 번째 노드에 요구되며 (N + 1)번째 패킷은 다시 첫 번째 노드에 요구된다. 즉 i번째 패킷은 (i modular N)번째 노드에 서비스를 요구한다.



<그림 3> 패킷 요구 과정

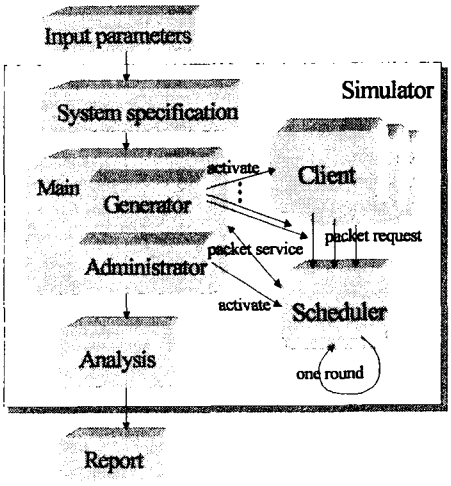
개발된 시뮬레이터는 이와 같은 방식으로 하나의 비디오에 대한 모든 패킷을 서비스하고 서비스가 종료되면 고객은 시스템을 떠나도록 하였다. <그림 3>은 시스템에 한 명의 고객이 도착했을 경우에 라운드에서의 패킷 요구 과정을 보여주고 있다. 노드 측면에서 보면 각각의 노드는 매 라운드마다 다수의 고객로부터 요구된 다수의 패킷을 지정된 디스크 스케줄링 방식을 이용하여 디스크에서 읽어오며 동시에 이전의 라운드에서 읽혀진 패킷들을 고객에게 전송한다. 이러한 과정은 시뮬레이터를 통해 내부적으로 이루어지며 제한시간을 만족시키지 못하는 패킷은 손실되어 고객은 품질의 저하를 경험하게 된다.

시뮬레이터는 프로그램 실행 시에 고객 도착률, 시뮬레이션 시간, 노드의 수, 노드 당 디스크의 수 등 <그림 4>에서 보여지는 입력 파라미터 값을 요구하며 이를 입력받아 패킷 손실 확률, 동시에 요구된 최대 고객 수, 디스크 이용률 등을 출력으로 제공한다.



<그림 4> 시뮬레이터의 입력과 출력

<그림 5>는 시뮬레이터의 개략적인 구조를 보여주고 있는데, 먼저 대화상자를 통하여 입력되어진 파라미터들을 통해 시스템 사양과 작업부하가 결정되어진다. 새로운 고객의 도착 시간 간격은 지수(exponential) 분포를 따른다고 가정하였고 Generator 모듈을 통해 구현되었다. 이 모듈에서는 시뮬레이션 시간 동안 이러한 고객의



<그림 5> 시뮬레이터 구조

요구를 발생시킬 뿐만 아니라 현재 시스템 내에 있는 고객 수와 서비스 받은 총 고객의 수를 계산한다. 또한 고객 한 명에 대한 비디오 요구가 있을 때마다 Client 모듈이 활성화되어 영화 당 패킷 수만큼 서버에 요구하게 된다.

시뮬레이션 시간 동안 Administrator 모듈은 매 라운드마다 주기적으로 Scheduler 모듈을 활성화시키고 이로 인해 활성화 된 Scheduler 모듈은 하나의 라운드 동안 요구된 패킷들을 차례로 처리한다. 시뮬레이션 과정은 윈도우에서 그래픽 화면을 통해 고객에게 보여지게 되며 시뮬레이션이 끝나게 되면 Analysis 모듈은 모든 결과 값들을 보고서로 화면에 출력한다.

전통적으로 FCFS(First Come First Served), SSTF(Shortest Seek Time First)와 같은 디스크 스케줄링 알고리즘이 검색 시간과 회전 지연 시간을 줄이기 위해 서버에 의해 제공되어 왔다. 그러나 멀티미디어 서버는 실시간 제약이라는 개념이 추가됨으로 해서 전통적인 디스크 스케줄링 알고리즘의 직접적인 적용은 사실상 어렵다. 이러한 이유로 실시간 작업들의 스케줄링에 대한 방법들이 많이 연구되어지고 있는데 예를 들어 EDF(Earliest Deadline First) 알고리즘의 경우 제한시간이 가장 빠른 요구부터 먼저 서비스함으로써 실시간 요구를 만족시킨다.

본 시뮬레이터에서는 실시간 스케줄링 알고리즘의 가장 기본이 되는 EDF 방식과 실시간 멀티미디어 서비스 시스템에서 비교적 우수한 성능을 나타내는 CSCAN 방식을 지원한다. EDF 방식을 적용했을 때 요구되는 패킷의 각각은 서비스 할 때마다 디스크 암(arm)이 무작위로 움직이게 된다. CSCAN은 SCAN 유형의 디스크 스케줄링 알고리즘으로 디스크 헤드가 한 방향으로 움직이며 요구된 패킷을 서비스한다. 즉 하나의 라운드 동안 디스크 헤드는 최대 두 번 디스크를 탐색하게 된다. CSCAN 방식에서 디스크 탐색 시간(seek time)을 T_{seek}라 하고 회전 지연 시간(rotational latency)을 T_{rot}, 디스크 전송 시간을 T_{tran}이라 하였을 때 n개의 요구를 서비스하기 위해 필요한 시간 $T_{service}$는

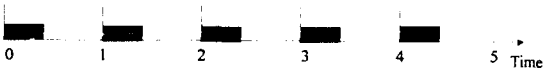
$$T_{service} = (T_{rot} + T_{tran}) \times n + 2 \times T_{seek} \quad (1)$$

로 표현될 수 있다. CSCAN 방식의 경우 최악의 상황에서도 하나의 라운드에 두 번의 탐색 시간을 가지므로 실시간 서비스에 적합한 알고리즘이라 할 수 있다.

시뮬레이터 내에서 디스크로부터 하나의 블럭을 읽는데 걸리는 시간은 탐색 시간, 디스크 회전 지연 시간 그리고 데이터의 전송 시간으로 구성된다. 구현 과정에서 탐색 시간은 평균 탐색 시간을 사용했으며, 디스크 회전 지연 시간은 uniform 분포를 따르는 것으로 가정했다. 평균 탐색 시간은 디스크 모델링에서 주로 사용되는 방식이며 디스크 회전 시간에 있어서 uniform 분포의 가정도 많이 사용된다[6]. 미디어 전송시간은 대부분의 디스크 모델링에서 행해지는 전송크기에 의존한 고정된 상수 값으로 모델링 되었다.

전통적인 파일서버와 달리 실시간 시스템은 단지 고객의 요구에 대해서 제한시간만 만족시키면 되고 응답 시간이 빠를 필요는 없다. 특히 멀티미디어 실시간 시스템의 경우 빠른 응답 시간은 오히려 클라이언트 측에서 미리 저장해야 할 기억 공간을 필요로 하므로 거기에 따르는 비용을 증가시킬 우려가 있다.

(a) 응답 시간을 최소화 한 경우



(b) 제한시간만 만족시키는 경우

<그림 6> 서비스 방식의 비교

<그림 6>에서 (a)는 시스템의 응답시간을 최소로 한 경우를 나타내고 있으며 (b)는 단지 제한시간만 만족시키는 경우를 보여주고 있다. 두 가지 경우 모두 모든 패킷 요구가 제한시간을 만

족하고 있지만 (a)의 경우에는 모든 패킷이 서비스된 Time 2가 되었을 때 앞으로 상영될 비디오를 클라이언트 측에서 모두 저장하고 있어야 한다. 이와 같은 방식에서는 최악의 경우 클라이언트 측의 기억 공간이 영화 한편을 저장할 수 있어야 한다. 이에 비해 (b)의 경우는 최소한의 기억공간만 있으면 된다. 즉 (a)는 (b)에 비해 클라이언트 측의 기억공간을 더 필요로 한다. 이러한 이유로 본 시뮬레이터에서는 (b)의 경우를 대상으로 시뮬레이션 하였다.

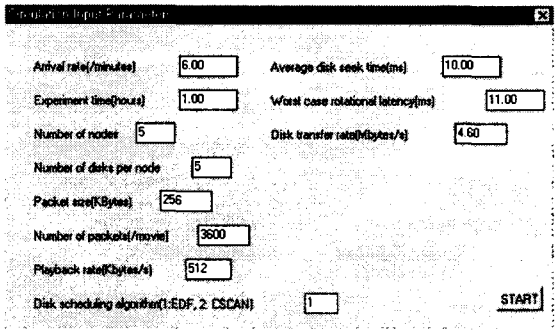
클라이언트 측의 비디오 상영률을 $T_{display}$라 하고 서비스되는 패킷의 크기를 P_{size}라 할 때 서버는 하나의 요구된 비디오 요청에 대해 평균적으로 P_{size} / $T_{display}$ 시간마다 하나의 패킷을 전송해야 한다. 그러므로 처음으로 패킷을 요구한 시간을 T_{first}라 하였을 때 i번째 패킷의 제한시간 T_{dead}는

$$T_{dead} = T_{first} + i \times P_{size} / T_{display} \qquad (2)$$

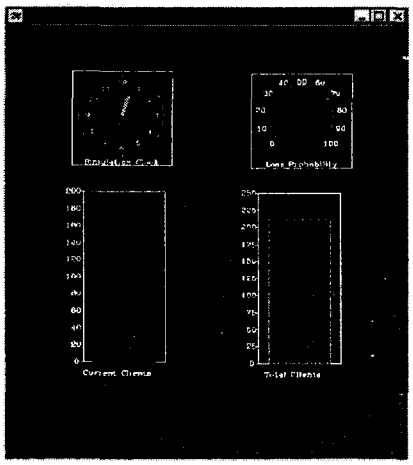
로 볼 수 있다. 시뮬레이터에서는 이러한 제한시간을 만족시키지 못하는 패킷의 요구는 무시되며 고객은 서비스를 받지 못하는 것으로 가정하였다.

시뮬레이터는 총 12개의 모듈들로 이루어져 있고 CACI의 Simscript Ⅱ.5 언어를 사용하여 구현하였다. 시뮬레이터를 실행시키기 위해 필요한 파일은 모두 4개로서 2개의 DLL(Dynamic Link Library)파일과 하나의 실행 파일 그리고 그래픽 라이브러리 파일로 되어 있다.

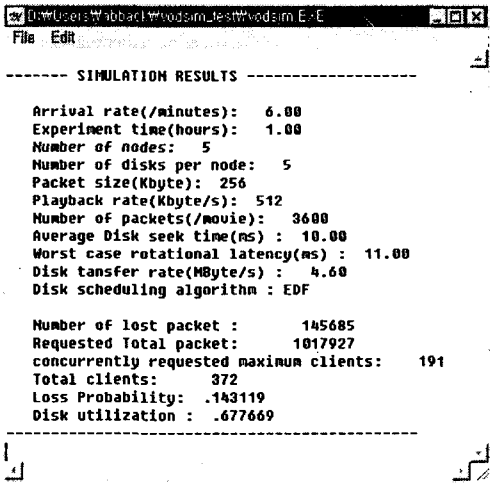
<그림 7>-(a)는 vodsim.exe을 실행시켰을 때 입력값을 요구하는 대화 상자를 보여주고 있다. 이 대화상자에는 처음에 초기 값이 주어져 있으나 고객은 자신이 시뮬레이션 하고자 하는 시스템 사양으로 변경할 수 있다. 입력 과정을 마치고 시작 버튼을 누르게 되면 시뮬레이션이 실행된다. 시뮬레이션이 진행되는 동안에는 고객의 이해를 돕기 위해 <그림 7>-(b)와 같은 그래픽 화면이 보여지며 위쪽에는 시뮬레이션 클럭과 패킷 손실 확률이, 아래쪽에는 현재 서비스를 받고



(a) 입력 파라미터 대화상자



(b) 그래픽 윈도우



(c) 시뮬레이션 결과 윈도우

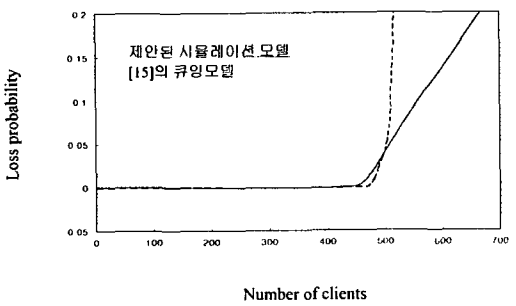
<그림 7> 시뮬레이터의 고객 인터페이스

있는 고객 수와 총 고객 수가 실시간으로 보여지게 된다. 시뮬레이션이 끝나게 되면 <그림 7>-(c)와 같은 결과 값들이 출력됨을 볼 수 있다.

4. 시스템 성능 평가

이 장에서는 3장에서 개발된 시뮬레이터를 이용하여 수학적인 모델링과 비교하여 보고 다양한 입력 파라미터 값에 따른 실시간 VOD 서버의 성능을 평가한다.

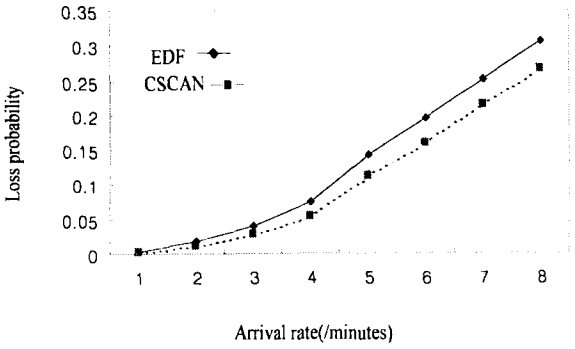
현재까지 제안된 비디오 서버에 대한 큐잉모델은 주로 노드가 독립적으로 고객에 대한 서비스를 수행하는 시스템을 대상으로 하였거나 고객의 도착률에 따른 영향보다는 고객의 수를 고정시켜 놓은 상태의 시스템 성능을 측정하였으므로 개발된 시뮬레이터와의 직접적인 비교는 어려우나 시뮬레이션 시간 동안의 평균 고객수를 적용하여 [15]의 큐잉모델과 비교하였다. <그림 8>은 두 모델간의 비교를 보여 주고 있는데 두 모델 모두 유사하게 고객의 수가 500명에 가까워짐에 따라 패킷 손실이 발생함을 볼 수 있다. 500명 이상 고객이 계속 증가함에 따라 발생하는 패킷 손실률의 차이는 [15]에서는 전통적인 FCFS 알고리즘을 사용했으나 시뮬레이터에서는 실시간 시스템에 적합한 EDF 알고리즘을 사용했기 때문이다. 관심을 가져야할 부분은 패킷 손실이 시작되는 부분으로 이 수치는 실제 시스템이 구현될 때 고객 수용 단계에서 고객의 수용여부를 결정하기 위한 값으로 사용된다. 또한 EDF 알고리즘



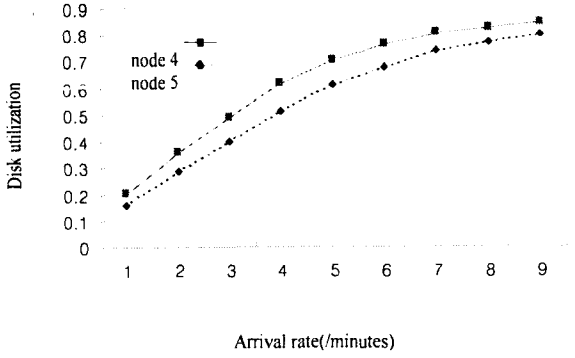
<그림 8> 고객수 변화에 따른 패킷 손실률

을 사용했음에도 불구하고 큐잉모델보다 조금 먼저 패킷 손실이 시작된 것은 [15]에서는 모든 노드가 완전히 균일한 부하를 가진다고 가정하였으나 시뮬레이터에서는 그러한 가정을 하지 않고 실제 고객의 요구에 따른 실험을 하였기 때문인 것으로 분석된다.

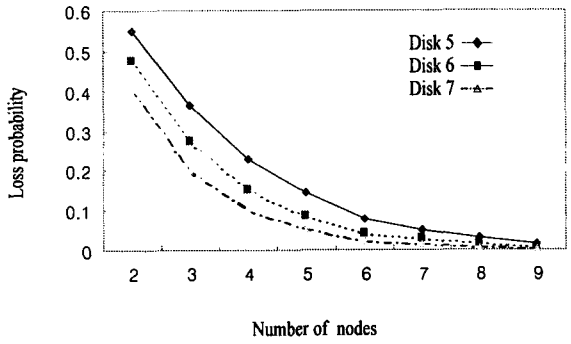
<그림 9>는 <그림 7>-(a)의 입력 파라미터에 고객 도착률을 증가시키면서 패킷 손실률을 측정한 것으로 도착률의 증가에 따라 패킷 손실률이 증가함을 볼 수 있다. 실선은 EDF 알고리즘의 영향을 나타내고 점선은 CSCAN 알고리즘의 영향을 나타내고 있는데 CSCAN 알고리즘을 사용했을 때 더 많은 고객을 수용할 수 있음을 보여준다. 이와 같은 결과는 EDF와 CSCAN을 비교하여 CSCAN이 우수함을 나타낸 타 연구결과와도 일치한다[18].



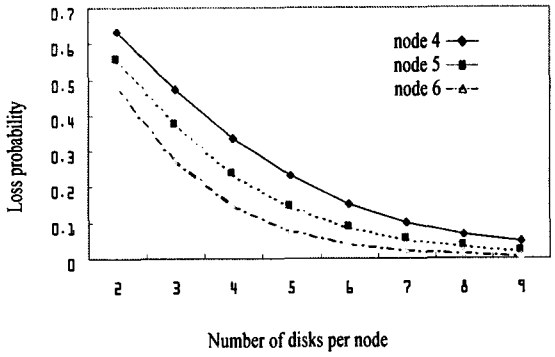
<그림 9> 고객 도착률 변화에 따른 패킷 손실률



<그림 10> 고객 도착률 변화에 따른 디스크 이용률



<그림 11> 노드의 수 증감에 따른 패킷 손실률



<그림 12> 디스크 수 증가시 패킷 손실률 변화

<그림 10>부터 <그림 12> 또한 <그림 7>-(a)에 있는 입력 파라미터들을 기초로 측정한 것으로 <그림 10>은 고객 도착률의 증가에 따라 디스크 이용률이 점차로 증가함을 보여주고 있다. 또한 <그림 11>은 노드의 수를 증가시키면서 패킷 손실률을 측정한 것이고 <그림 12>는 노드당 디스크의 수를 증가시키면서 패킷 손실률을 측정한 그림으로 노드나 디스크 수의 증가에 따라 패킷 손실률이 감소하는 모습을 볼 수 있다.

5. 결 론

최근 컴퓨터와 통신 기술의 비약적인 발전은 대규모 멀티미디어 서비스가 실현될 수 있음을 보여주고 있고 이와 관련한 연구가 다각도로 이루어져 왔다. 그러나 제안되고 있는 많은 수학적 모델링은 시스템의 동작을 표현하는데 있어 그 한계로 인해 정확도 면에서 다소 부족한 실정이다. 본 논문에서는 실시간 멀티미디어 서버의 시뮬레이션 모델링을 통해 수학적인 모델링 보다 더욱 정확하게 시스템의 성능을 측정하였으며 개발된 시뮬레이터를 이용하여 다양한 시뮬레이션을 수행하였다.

개발된 시뮬레이터는 VOD 서버 설계시 시스템을 디자인하는 단계에서 미리 성능을 평가해 봄으로써 구축될 시스템의 사양을 결정하는데 활용될 수 있으므로 실제 개발 시 시행착오로 인한 비용을 줄이는 효과를 기대할 수 있다.

본 논문을 통해 개발된 시뮬레이터의 추가 사항으로 VOD 서버의 각 노드와 디스크에 대한 더욱 정확한 모델링이 필요하며 네트워크와 클라이언트 측을 고려한 모델링의 확장 또한 필요하다.

참고 문헌

[1] L.A. Rowe, *et. al.*, "A Distributed Hierarchical Video-on-Demand System," Proc. of International Conference on Image Processing, Washington DC, Oct. 1995.

[2] 조진성, 신현식, "디스크 배열을 이용한 실시간 멀티미디어 저장 서버에서의 스케줄링 기법," 한국정보과학회 논문지, 제 21권, 제 11호, pp. 1981-1989, 1994. 11.

[3] K. Mayer-Patel and L.A. Rowe, "Design and Performance of the Berkeley Continuous Media Toolkit," Proc. of Multimedia Computing and Networking, 1997.

[4] B. Ozden, A. Biliris, R. Rastogi, and A. Silberschatz, "A Disk-based Storage Architecture for Movie on Demand Servers," Special Issue on Multimedia Information, Information Systems Journal 20, pp. 465-482, May 1995.

[5] E. Chang and H. Garcia-Molina, "2D BubbleUp - Managing Parallel Disks for Media Servers," Stanford Technical Report, 1998.

[6] B. Ozden, A. Biliris, R. Rastogi, and A. Silberschatz, "A Low-cost Storage Server for Movie on Demand Database," Proc. of International Conference on Very Large Databases, Sep. 1994.

[7] J.Y. Lee, "Parallel Video Servers: A Tutorial," IEEE Multimedia, pp. 20-28, Apr. 1998.

[8] R. Tewari, *et. al.*, "Design and Performance Tradeoffs in Clustered Video Servers," Proc. of the IEEE International Conference on Multimedia Computing and Systems, Tokyo, Japan, pp. 144-150, May 1996.

[9] R. Tewari, D. Dias, R. Mukherjee, and H.M. Vin, "High Availability for Clustered Multimedia Servers," Proc. of International Conference on Data Engineering, Feb. 1996.

[10] W.J. Bolosky, *et. al.*, "The Tiger Video Fileserver," Proc. of the 6th International Conference on Network and Operating System Support for Digital Audio and Video, pp. 97-104, Apr. 1996.

[11] C. Bernhardt and E. Biersack, "A Scalable Video Server : Architecture, Design, and Implementation," Realtime Systems Conference, pp. 63-72, Jan. 1995.

[12] J. Banks, *et. al.*, Discrete-Event System Simulation, 2nd Ed., pp. 4-6, Prentice-Hall, New Jersey, 1984.

[13] J.A. Clark, "Dependability Analysis of Fault Tolerant Multiprocessor Architectures though Simulated Fault Injection," Ph.D. dissertation, Univ. of Massachusetts, Sep. 1993.

[14] 박기진, 정지영, 김용규, 박주용, 김성수, "대규모 주문형 멀티미디어 서비스 시스템의 큐잉네트워크 모델 설계," 98 한국정보과학회 춘계학술발표논문집, Vol. 25, No. 1(A), pp. 237-239, 1998. 4.

[15] 조진성, 신현식, "대규모 주문형 비디오 서버의 큐잉 모델," 97 한국정보과학회 춘계학술발표논문집, Vol. 24, No. 1(A), pp. 383-386, 1997. 4.

[16] 정지영, 김성수, "실시간 멀티미디어 서버 시험환경 개발," 99 한국정보과학회 춘계학술발표논문집, Vol. 26, No. 1(A), pp. 36-38, 1999. 4.

[17] B. Ozden, R. Rastogi, and A. Silberschatz, "Periodic Retrieval of Videos from Disk Arrays," IEEE International Conference on Data Engineering, Apr. 1997.

[18] J. Gemmell, *et. al.*, "Multimedia Storage Servers: A Tutorial," IEEE Computer, Vol. 28, Issue 5, pp. 40-49, May 1995.

저자소개

정지영

1998년 아주대학교 정보 및 컴퓨터공학부(공학사)
2000년 아주대학교 정보통신전문대학원 정보통신공학과(공학석사)
현 재 아주대학교 정보통신전문대학원 정보통신공학과 박사과정
관심 분야: 멀티미디어 시스템, 클러스터 시스템, 시뮬레이션, 결함허용 시스템

김성수

1982년 서강대학교 전자공학과(공학사)
1984년 서강대학교 전자공학과(공학석사)
1995년 Taxas A&M University, 전산학과(공학박사)
1983~86년 삼성전자(주) 종합연구소 컴퓨터연구실(주임연구원)
1986~96년 삼성종합기술원 수석연구원
1991~92년 Taxas Transportation Institute 연구원
1993~95년 Taxas A&M University, 전산학과, T.A. & R.A.
1997~98년 한국정보과학회, 한국정보처리학회 논문지 편집위원
1996년~현재 아주대학교 정보통신전문대학원 부교수/ 정보및컴퓨터공학부 주임교수
관심 분야: 결함허용 시스템, 성능 평가, 멀티미디어, 클러스터 시스템