비디오 서버에서의 효율적인 대역폭 스케줄링 지원

이 원 준†

요 약

멀티미디어 어플리케이션들은 기존의 화일 서버 기법으로는 쉽게 제공하지 못하는 보장된 검색 및 전송률을 요구한다. 본 논문에서는 CM(Continuous Media : 예를 들어 비디오) 서버를 위한 동적 협상 수락 제어(dynamic negotiated admission control)와 자원 스케줄링(resource scheduling) 기법을 제안한다. 이는 두 부분으로 구성되는데, 예약 기반 수락 제어 방법(reserve-based admission control mechanism)과 동적으로 자원을 할당하는 스케줄러가 그것이다. 정적 스케줄러와 비교할 때, 제안된 기법은 경쟁하는 스트림들에게 이용가능 한 자원들을 효과적으로 공유할 수 있도록 해주며, 전반적인 QoS(Quality of Service)를 향상시키기 위해 자원의 재 할당을 요구하는 스트림들을 위한 스케줄러 초기화 협상(scheduler-initiated negotiation)을 통해 높은 이용률을 얻을 수 있다. 본 논문에서 제안한 기법을 사용하여 동시에 실행시킬 수 있는 클라이언트의 수를 증가시킬 수 있으며 혼잡한 트래픽 상태에서도 좋은 응답 비율과 향상된 자원 이용률을 얻을 수 있다.

Efficient Support for Adaptive Bandwidth Scheduling in Video Servers

Wonjun Lee†

ABSTRACT

Continuous multimedia applications require a guaranteed retrieval and transfer rate of streaming data, which conventional file server mechanism generally does not provide. In this paper we describe a dynamic negotiated admission control and disk bandwidth scheduling framework for Continuous Media (CM : e.g., video) servers. The framework consists of two parts. One is a reserve-based admission control mechanism and the other part is a scheduler for continuous media streams with dynamic resource allocation to achieve higher utilization than non-dynamic scheduler by effectively sharing available resources among contending streams to improve overall QoS. Using our policy, we could increase the number of simultaneously running clients that could be supported and could ensure a good response ratio and better resource utilization under heavy traffic requirements.

키워드 : 수락 제어(Admission Control), 자원 스케줄링(Resource Scheduling), QoS(Quality of Service), 비디오 서버(Video Server), CM(Continuous Media)

1. 서 론

최근 멀티미디어의 I/O 문제와 특히 CM(Continuous Media)에 대한 관심은 증가되고 있다. 기존의 디스크 스케줄링 정책의 목표는 탐색 비용을 줄이고, 높은 처리양을 얻는 반면, 서비스를 탐색해야 하는 모든 프로세스에 공정한 접근을 제공하는 것이다. 연속적인 매체를 위한 디스크 스케줄링의 목표는 요구 비율에 맞게 스트림 관리자가 생성하는 주기적인 I/O 요청의 데드라인을 만족시키는 것이며, 부가적인 목표로는 버퍼 요구를 최소화하는 것이다. CM 서버[7, 13] 상에서 비디오 전송은 연속적이고, 엄격한 실시간 제약을 위해 디스크 대역폭, 버퍼 용량, 네트워크 대역폭 등과 같은 여러 요소들을 신중히, 효과적으로 처리할 수 있도록 고려해야 한다. 이러한 자원 요소들의 예약은 만족할 만한 재생의 질적 수준을 제공하며, 정기적으로 비디오 스트림들을 전송할 수 있는 제약을 지원한다. 특히, 디스크 대역폭 제약은 가장 중요한 요소일 수 있는데, 디스크 상에서 각 스트림들을 위해 예약된 대역폭은 대기 오버헤드 시간, 전송 시간, 한 사이클 길이, 그리고 다수의 스트림 간의 경쟁 상태에 의존한다. 따라서, 각 스트림의 요청은 좋은 디스크 이용률과 서버 비용 성능과 함께 공정하게 지원 받을 수 있도록 보장되어야 한다.

CM에 대한 디스크/서버의 목표는 요청된 QoS를 만족시키는 것이다. 이것은 서버에 상주해 있던 일부의 스트림 매니저가 생성하는 주기적인 I/O 요청들의 데드라인에 만족시키는 것으로, 스트림 매니저는 최소한의 버퍼와 공정한 스케줄링 알고리즘[8]을 요구한다. 시각적인 특성을 가지는 비디오

※ This work was supported by Korea Research Foundation Grant(KRF-2001-003-E00219) and was also partially sponsored by U.S. Air Force contract number F30602-96-C-0130.
† 종신회원 : 고려대학교 컴퓨터학과 조교수
논문접수 : 2001년 10월 11일, 심사완료 : 2001년 11월 26일

스트림들은 데이터 전송의 정확도 및 실시간 전송을 요구하지만, 짧은 시간동안 일부의 데이터가 손실되어도 대부분의 사용자들은 감지하지 못하거나 견딜 만 하다. 따라서, 서비스 질의 저하(degrading)를 수락할 수 있는 방법은 간단히 CM 응용프로그램이 요청한 자원들을 감소시키면 된다. 이러한 soft-QoS로 높은 이용률을 얻는 방법은 네트워킹 분야에서 더 이상 새롭지 않다. 광대역 네트워크 상에서의 soft-QoS에 관한 연구는 비디오 트래픽과 그것이 네트워크 이용률에 미치는 영향과 관련해 많은 참고문헌[5, 12, 2]들이 있다. 본 논문에서 제안한 기법은 네트워크 상에서의 수락 제어 및 동적 대역폭 관리 기법들을 비디오 서버와 같은 시스템 계층에 적용한 것이다.

본 논문에서 제시한 전략은 CM 응용프로그램들이 QoS 파라미터 상에서 일정한 변화를 견딜 수 있다는 전제 하에, 경쟁하는 스트림들 사이에서 효과적으로 자원을 공유할 수 있도록 서버 상에 공유 처리 자원들을 위한 알고리즘으로 발전시켰다. 본 논문에서 제안된 알고리즘은 전반적인 QoS를 개선시키기 위한 방법으로, 자원의 재할당을 요구하는 스트림들을 위해 자원을 재분배/조정(즉, 스케줄러 초기화 협상)한다. 높은 성능을 얻을 수 있는 CM 서버를 설계하기 위해서는 CM 스트림의 특성뿐만 아니라 자원 제약들도 고려해야만 한다. 이러한 CM 스트림들은 자체의 특징들과 특별한 QoS 장점(metrics)을 가진다. 비디오나 오디오에 대한 사람의 지각을 평가해보면, 사람은 비디오 프레임 손실의 총합이 23%, 오디오 손실의 총합이 21%가 될 때까지 견딜 수 있으며, 연속적인 비디오, 오디오의 손실에 대해서는 대략 2개의 LDU(Logical Data Units)까지 받아들일 수 있다[11]. 또 비디오의 약 20%, 오디오 편차의 7%까지 견딜 수 있다고 한다. 이러한 사실을 고려해 볼 때, 스트림 상에서 운영되는 일부의 자원들을 제약하거나 독점할 수 있다. CM 서버는 주어진 요청의 수준에 적합한 QoS 파라미터들을 보장할 필요가 있는데, CM 서버의 부하가 낮을 때 이 같은 요구들을 만족시키게 된다. 그러나 CM 서버 상에 동시에 많은 스트림들이 발생하게 되면 모든 QoS 파라미터들을 보장하기가 어려워진다. 일정한 양의 자원들이 주어져 있을 때 많은 CM 스트림들의 QoS 파라미터들 값이 가능하면 모두 수락되기를 원한다. 그러나 클라이언트들이 더 많은 CM 스트림들을 요구하면 할수록 CM 서버의 QoS는 떨어지게 될 것이다. 물론 QoS를 적당히 떨어뜨리는 것은 좋은 방법일 수 있다. 따라서 클라이언트들이 적당한 품질의 서비스를 받도록 보장하기 위해서는 수락 제어가 반드시 필요하다.

본 논문에서는 공정한 스케줄링과 더 나은 비디오 스트리밍 성능을 제공해주는 동적/적응성 수락 제어 방법을 제안한다. 분산 환경 하에서 특정한 대역폭을 가진 비디오 스트리밍의 연속적인 전달은 다수의 클라이언트들에게 효과적인 서비스와 실시간 요구를 만족시킬 수 있다. 이러한 새로운 스트림 스케줄러는 능숙한 수락 제어 기능을 제공해 주는데, 이것은 특히 과부하 상태에서 클라이언트의 요청에 대한 응답 비율을 높일 수 있도록 디스크 이용률을 최적화시킬 수 있다. 본 논문에서 제시한 알고리즘은 기존의 탐욕 수락 제어 알고리즘과 비교해 볼 때 다중 CM 스트림 입력에 대해 향상된 수락 제어 비율을 보인다. 다시 말해서, 입력 거부 비율(rejection ratio)은 총 입력 스트림 대 거부되어지는 스트림의 수로 계산되므로, 입력 수락 비율과 거부 비율은 서로 반대되는 개념이 된다. 낮은 입력 거부 비율은 곧 높은 제어 비율을 의미한다. 스트림의 재생 품질은 비슷한 수준을 보인다. 본 논문에서는 기존의 탐욕 수락 제어 방법과 새롭게 제시된 수락 제어 및 스케줄링 알고리즘의 동작을 시뮬레이션 결과를 통해 비교할 것이다.

본 논문의 구성은 다음과 같다. 2장에서는 가정한 CM 서버의 설계와 수락 제어 제약들에 대해 간단히 기술하며, 3장에서는 본 논문에서 제시한 수락 제어와 스케줄링 알고리즘의 세부 사항에 대해 기술한다. 4장에서는 실험 평가의 양적인 결과를 분석하며, 실험 설계와 파라미터들에 대해 기술한다. 5장에서는 관련 연구에 대하여 알아보고, 마지막으로 6장에서는 결론과 향후 과제에 대해 기술한다.

2. 비디오 서버 상의 수락제어 요구조건

이 장에서는 CM 서버 시스템과 CM 서버의 설계에 대해 알아본다. CM 서버를 통해 네트웍 상에서 클라이언트 요청에 대해 다중(또는 싱글) 스트림을 재생하게 된다. 또한 기본적인 수락 제어 방식과 제약들을 제시한다.

2.1 개 요

CM 서버 시스템은 전형적인 클라이언트-서버 응용 프로그램이다. 이것은 여러 파일 시스템을 기반으로 하는데, 예를 들면, Presto File System(PFS)[6]과 Unix File System(UFS)들이다. CM 서버의 핵심 부분으로는 네트웍 관리자, QoS 관리자, I/O 관리자, 프락시 서버들이 포함된다. CM 서버 상에서 서브 모듈들은 싱글 프로세스 상에서 운영되는 스레드들이다. 시스템은 클라이언트-서버 모델을 기반으로 한다. 클라이언트와 서버는 같은 로컬(local) 기계 혹은 네트웍 상에 위치할 수 있다. 프락시 서버는 새로운 클라이언트에 대한 요청을 받은 후에 클라이언트와 통신한다. 네트웍 관리자는 나중에 작동하게 될 프락시 서버에게 연결 관리 기능을 위임해 준다. 클라이언트의 요청은 일반적인 구조를 가지며, 모든 타입의 스트림에 사용될 수 있어야 한다. 그 구조는 다음과 같은 요소를 포함한다. *데이터 전송 속도(fps), 샘플링 비율/속도, 클라이언트의 pid, 화일 시스템* 종류(UFS/PFS), *스트림 종류*(AU, MJPEG, etc.) 등. 네트웍 관리자는 QoS 관리자 모듈과 밀접한 관련이 있는데, 이는 QoS 매니저 모듈이 현재 얼

마나 많은 연결이 활성화되어 있으며, 스트림 타입과 크기가 무엇인지 알고 있기 때문이다. 프락시 서버는 각 클라이언트 연결 당 만들어진다. 네트웍 관리자는 새로운 클라이언트 요청을 가진 프락시 서버를 생성한 후, 다른 새로운 클라이언트의 요청이 있을 때까지 기다린다. 각각의 프락시 서버는 어떤 일정한 클라이언트 요청과 관련하여, 네트웍 상에서 CM 서버와 클라이언트 사이의 데이터 통신을 위한 송수신(send/receive) 작동을 처리한다. 또한 프락시 서버는 *QoS 관리자*와 프락시 서버에 의해 만들어지는 하나의 스레드인 *I/O 관리자* 사이에서 수락 제어, 버퍼 관리, I/O 동작을 처리하기 위해 상호작용한다. 프락시 서버는 I/O 관리자들과 소비자/생산자(producer/consumer) 방식으로 서로 협력한다. 각 연결은 프락시 서버와 I/O 관리자와 관련이 있다. 프락시 서버는 일반적인 스트림을 다루도록 설계되는데, 스트림 타입에 대한 고려는 하지 않는다. 반면, I/O 관리자는 모든 작업에 대해 스트림 타입을 고려하며, 프락시 서버와 클라이언트가 상호작용할 수 있도록 각각의 프레임들을 공동의 버퍼에 건네준다.

입력 CM 스트림을 위한 수락 제어는 QoS 관리자에 의해 다루어지는데, 이는 수락 제어자와 QoS 핸들러 두 부분으로 구성된다. 수락 제어자는 두 가지의 제약 테스트를 한다 : (1) I/O 대역폭 테스트, (2) 이용 가능한 버퍼 테스트. 각각의 요청된 스트림은 일정한 비율(예, 재생 속도 : 즉 초당 프레임)로 도착하며, 수락 제어자는 그것을 받아들일지의 여부를 판단한다. QoS 핸들러 모듈은 클라이언트 요청에 의해 제공되는 입력 파라미터 속도에 따라 데이터 속도를 처리한다. 재생 시간의 경우 시스템의 오버헤드로 인해 지연이 발생하는데, 이 모듈은 데이터 속도를 유지하기 위해서 적당한 양의 일부의 프레임을 제거한다. 이 모듈은 또한 동적 QoS 협상과 관리를 제어한다.

수락 제어는 네 가지 경우에 적용된다. (1) 새로운 클라이언트의 요청이 도착하거나, (2) 속도 제어 작동시(Set_Rate, Pause, Resume, 및 Fast_forward와 같은), (3) 스트림의 재생이 끝나거나, (4) 예약된 자원들이 일정 시간동안 동작하지 않을 때, 수락 제어 요구를 위해 다음 두 제약을 확인해 봐야 한다.

2.1.1 I/O 대역폭 제약

수락 제어에서 디스크 I/O 대역폭 이용가능성을 확인하기 위해서 I/O 대역폭 테스트를 사용한다. 식 (1) :

$$(t_s + \sum_{i=1}^{n+1}[\frac{Tsvc \cdot ri}{b}] \cdot t_r) + \sum_{i=1}^{n+1} \leq \frac{Tsvc \cdot ri}{R} \leq T_{svc} \quad (1)$$

즉, 대기시간 오버헤드 타임(= 탐색 시간 + 회전 대기 시간) + 전송 시간 ≤ 사이클 길이

t_s : 탐색 시간(msec), T_{svc} : cycle length(msec),
r_i : 소모 비율(Byte/msec), b : 디스크 블록 크기(Byte),
t_r : 회전 시간(msec), R : 전송 비율(Byte/msec),

첫 번째 요소(t_s)는 하나의 사이클에서 최대 디스크 탐색 대기시간의 오버헤드를 기술한다. 두 번째 요소는 주어진 하나의 사이클 길이(T_{svc})동안 각각의 디스크 블록(요청된 각 r_i를 위한)들을 검색하는데 걸리는 회전 대기 시간의 총합을 보여준다. 마지막 요소는 한 사이클 동안 전송되는 시간의 합을 기술한다. 끝으로, 주어진 사이클 길이 동안 디스크가 접근할 수 있는 총 시간은 서비스 사이클 길이(T_{svc}) 안에 이루어져야 한다. 결국 식 1은 한 사이클 주기 안에 디스크 접근을 통하여 필요한 멀티미디어 데이타를 탐색, 읽어 들여 전송하여야 하는 디스크 입출력 대역폭 제약 조건을 의미한다.

2.1.2 버퍼 제약

이용가능 한 버퍼 테스트는 다음과 같다 :

$$2(\sum_{i=1}^{n+1} r_i) \cdot T_{svc} \leq B_{avail} \quad (2)$$

B_{avail} : 이용 가능한 버퍼 크기

총 버퍼 요구의 크기는 이용가능 한 버퍼 크기보다 작아야 한다. 여기서 이중 버퍼 방식을 가정해본다. 따라서 수락 제어 요구는 식 (1)과 식 (2)를 동시에 만족해야 한다. 만약 그들 중 어느 쪽이든 만족되지 않으면, 입력 요청은 거부된다.

3. 적응성 수락 제어

이 장에서는 동적 적응성 수락 제어와 스케줄링 알고리즘에 대해 기술한다. 이것은 시스템 자원 이용률을 높이기 위한 QoS 감소/적응(degration/adaptation)을 포함한다. 이 장에서는 탐욕 수락 제어와 스케줄 방식에서 공통적으로 사용되는, 수락 결정의 상태에 따르는 첫 번째 제약 식 (1)에 대해 고려해 본다. 대부분의 기존 수락 제어 접근 방식들은 탐욕 방식을 사용한다. 이것은 서버가 클라이언트들이 요청하는 자원들을 모두 할당 해줄 수 있을 때만 새로운 응용 프로그램(비디오 스트림)을 받아들인다. 탐욕 방식의 가장 큰 문제점은 대역폭이 최고점일 때를 기반으로 하는 보수적인(conservative) 접근 방식이거나, 통계적 방법[4, 19]에 초점을 맞추고 있다는 것이다. 통계적 방법은 디스크와 네트워크 서브 시스템에 의해 성능의 만족할 만한 수준을 결정하는 것이다. 이 같은 방식은 너무나 보수적이어서 아주 적은 수의 스트림만을 허락할 수 있으며, 그로 인해 서버 자원 이용률이 낮아진다. 비록 확률적인 방법이 응용 프로그램의 실패에 따르는 손실을 줄여 줄 수 있을지라도, 많은 경우에 이 방식 역시 바람직하지 않다.

본 논문에서는 남아있는 자원의 양에 따라 자원들을 재조정할 수 있는 확장된 수락 제어 알고리즘을 제안한다. 이 알고리즘의 기본 아이디어는 예약에 따라 자원의 비율을 할당하고, 응용 프로그램이 이 예약된 자원을 사용하기 시작할 때 다

른 정책이 사용되도록 한다. 대다수의 클라이언트들의 요청이 공평하게 서비스되기를 원할 때 과부하 상태(예, 총 디스크 대역폭 이용률이 70%를 넘는 경우)라고 가정해 보자. 만약 남아있는 이용 가능한 자원들이 임계치(threshold : $T_{reserve}$)보다 작다면, 각 스트림 상에서 서비스된 자원의 비율 대 요청된 자원의 비율인 done_ratio 와 이용 가능한 자원의 양에 따라 새로 요청된 자원들 중 일부만 할당해준다. 이용가능 한 자원의 양을 기반으로, 강등된(degraded) 스트림들은 자원 협상(resource-negotiation)을 요구한다. 자원 협상(reclamation)은 다음과 같은 상황에서 일어난다. (1) 실행 중이던 스트림 재생이 중단되고 시스템에게 자원을 다시 반환할 때 ; (2) 고속 전진(Fast_Forward)과 같은 전송 속도 조정(set_rate) 비디오 기능이 받아들여 질 때, (3) 자원들이 더 이상 사용되지 않은 채 일정 시간이 경과되었을 때 등이다. 수락 제어의 자연스러운 접근은 탐욕 알고리즘을 사용하는 것으로, 새로 요청되는 비디오 스트리밍은 할당 할 수 있는 자원들이 존재하는 한 수락되어 진다[9]. 응용 프로그램이 도착할 때 이용가능 한 자원들이 요청한 자원들보다 크면, 응용 프로그램을 수락하는 정책이 사용되며, 응용 프로그램의 재생이 끝나게 되면 응용 프로그램이 사용하던 자원들까지 사용가능한 풀(pool)에 반환하여 주는 정책이 사용된다. <표 1>에서는 알고리즘에서 사용된 속성들을 설명한다.

기본적인 탐욕 수락 제어 알고리즘의 가장 큰 장점은 알고리즘의 단순성에 있다. 그러나 탐욕 수락 제어 알고리즘은 단순한 반면 자원 효율성에 있어서는 다소 문제가 많다. 본 논문에서는 다른 전략을 생각해보기로 한다. 이를 위해 비디오 서버에 동시에 더 많은 스트림들을 실행시키기 위해서 새롭게 도착한 스트림들이 요청한 QoS 값을 조정하거나, 강등된 스트림들에게 반환된 자원들을 재할당하는 방법이다. 그러나 강등되는 자원의 양을 결정할 수 있는 완벽한 알고리즘을 구현하기란 매우 어려운 일이며, 또한 미래의 트래픽에 대해 정확한 측정을 할 수 없다. 그러나 본 논문에서 제시한 새로운 soft-QoS 구조는 거부 비율을 몇 배 감소시켜 주며, 시스템 상에 더 많은 입력 스트림들을 실행시킬 수 있다.

주어진 평균 탐색 시간(ts), 사이클 길이(T_{svc}), 디스크 블록 크기(b), 회전 시간(t_r), 그리고 디스크 전송 속도(R), 각 클라이언트 요청이 소모되는 속도(q_i)는 I/O 대역폭 제약 식 (1)에서 사용하는 변수이다.

본 연구에서는 T_{svc}(= T_{total})을 이용하여 총 이용할 수 있는 속도를 초기화시키고, congested_bit를 체크하기 위해 사용되는 기준 값인 $T_{reserve}$ 설정한다. congested_bit는 0으로 초기화한다. 만약 자원의 사용(T_{used})이 (T_{svc} - $T_{reserve}$)보다 크다면 자원(여기서는 I/O 대역폭) 혼잡이 발생한다. 자원의 QoS를 떨어뜨려서 더 많은 요청을 수락하기 위해 자원의 양 $T_{reserve}$를 정할 수 있다. 여기서 T_{usage} + T_{avail} = T_{svc} 이다. T_{avail} 이 $T_{reserve}$ 보다 작을 때 congested_bit는 1로 설정된다. 다른 방법으로 기본적인 I/O 대역폭 제약을 적용하는 것이 있다(즉, 새로운 요청이 강등 없이 수락된다). congested_bit를 설정한 다음, 새로운 요청을 받아들이기 위해 자원들의 할당 수를 제약한다. 보다 상세한 내용은 알고리즘 2에 나타나 있다.

〈표 1〉 알고리즘에 사용된 속성들

Attributes	Descriptions
s_i	스트림 i
q_i	초기 요청된 자원 s_i
v_i	현재 서비스된 자원 s_i : = q_I * d_i
RS	실행중인 모든 스트림들의 집합 : U_i s_i, where $\{s_i \mid F_{min,i} < d_i \leq 1.0\}$
DS	강등된 스트림의 집합 : RS - U_i s_I, where $\{s_i \mid d_i = 1.0\}$
s_{min}	DS에서 d_j의 값이 가장 작은 스트림 j : $s_{min} = \{s_j \mid \forall s_j, s_k \in DS d_j \leq d_k\}$
m	RS상에서의 스트림들의 수
done_ratio	스트림에 대해 서비스된 자원의 비율 대 요청된 자원의 비
remaining_ratio	스트림에 대해 이미 서비스된 자원의 비 대 요청된 자원의 비(= 1.0 - *done_ratio)*
remaining_rate$_i$	스트림 i에서 이미 서비스된 자원의 양 I : (= q_i - v_i)
sum_remaining_ratio	강등된 스트림들 중 남아있는 총합 : $\sum_k e_k$, where $s_k \in DS$
MIN_FRACT	스트림에 할당가능한 자원의 최소 양
MIN_RESERVE	DRA 모드를 유지하는 데 필요한 예약 가능한 최소 양
d_i	s_i의 done_ratio
e_i	s_i의 remaining_ratio : (1.0 - d_i)
$F_{min,i}$	s_i의 MIN_FRACT
T_{total}	초기에 주어지는 이용가능 한 자원들
$T_{reserve}$	예약에 할당가능 한 자원들의 양
T_{free_res}	예약에 이용가능 한 자원 : {T_{total} - T_{used} if $T_{used} > T_{total} - T_{reserve}$; 0 otherwise
$T_{free\_non\_res}$	외부에서 예약할 때 이용가능 한 자원 : {T_{total} - $T_{reserve}$ - T_{used} if $T_{used} \leq T_{total} - T_{reserve}$; 0 otherwise
T_{alloc}	자원을 요청하는 스트림들에게 할당된 자원들
T_{avail}	스트림들에게 할당가능 한 자원 : (= T_{total} - T_{used})
T_{used}	현재 사용되고 있는 자원들의 총 합 : $\sum_i vi$, where $s_i \in RS$
$\Psi(T_{free\_res})$	수락 테스트에서 예약에 대한 휴리스틱 함수 : e.g., where $k = k_1^{(Tused - Treserve) \cdot k2}$
$\Phi(T_{free\_res})$	스트림 예약을 위한 휴리스틱 함수 : e.g., where $k = k_3 \cdot (T_{used} - T_{reserve}) + 1$
congestBit	혼잡/비혼잡 상태 확인에 대한 비트 플래그 {1 if $T_{used} > T_{total} - T_{reserve}$; 0 otherwise
$SU(t)$	$\forall_k s_k \in RS$ 인 t 시간동안 $\sum_k$ dk에 의해 정의된 총 시스템 이용률
$QT(t)$	t 시간동안 $SU(t)$/m에 의해 정의된 총 비디오 품질

알고리즘 2 RAC ($T_{reserve}$, *event* E_i)
▷ E_i = {(*START*, q_i), (*CLOSE*, v_i)}
1 **switch** EVENT of
2 **case** "START" :
3 **if** (! congested) **then**
▷ 품질 강등 없이 수락
4 $T_{alloc} \leftarrow q_i$;
5 $T_{used} \leftarrow T_{used} + q_i$;
6 **else** /* 혼잡 상태 */
7 **if** ($\Psi(T_{free\_res}) > MIN\_FRACT$) **then**
▷ 강등 상태로 수락
8 $T_{alloc} \leftarrow$ **min**($\Psi(T_{free\_res})$, q_i) /* $\in (0, q_i)$ */
9 $T_{used} \leftarrow T_{used} + T_{alloc}$;
10 else degraded = 1 ;
11 **else**
▷ 수락하기에는 너무 작은 양
12 Reject s_i ;
13 **case** "CLOSE"
▷ *appl_inst_ID 인자를 통해 …*
14 **if** (degraded)
▷ 자원 v_i를 풀로 반환
15 $T_{used} = T_{used} - v_i$;
16 **switch** MODE of
17 **case** "*DRA*" :
▷ $v_i + T_{avail}$ *자원량 만큼 강등된 어플리케이션에게 재분배*
18 $T_{avail} \leftarrow T_{total} - T_{used}$;
19 **while** (1) **do**
20 $s_{min} \leftarrow$ done_ratio가 가장 작은 요청을 선택 ;
21 **if** ($\Phi(T_{free\_res}) > MIN\_RESERVE$) **then**
22 $T_{alloc} = \min(\Phi(T_{free\_res}), remaining\_rate_{min})$;
23 $T_{used} \leftarrow T_{used} - T_{alloc}$;
24 **end while**
25 **case** "*RWR*" :
▷ 자원 v_i를 강등된 어플리케이션에 재분배
26 $T_{avail} = v_i$;
27 $sum\_remaining\_ratio = \sum_k remaining\_ratio_k$;
28 **for** (강등되었던 모든 k에 대해) *do*
29 $T_{alloc} = T_{avail} * (remaining\_ratio_k / sum\_remaining\_ratio_k)$;
30 $T_{used} \leftarrow T_{used} - T_{alloc}$;
31 **end for**
32 **end switch**
33 **end switch**

본 논문에서는 휴리스틱 기법을 이용하여 협상(reclamation) 알고리즘을 구현하였고, 이들의 성능을 테스트하였다. 본 논문에서는 두 가지 휴리스틱을 이용한다.

3.1 Dynamic Reserve Adaptation (DRA) :

이것은 응용 프로그램이 자원들을 필요로 하나 처음에 요청한 만큼의 자원은 할당받지 못했을 때 적용된다. 스트림 요구가 시작될 때 DRA는 할당된 자원들을 나머지 스트림들이 이용할 수 있도록 반환하고, 총 이용가능 한 자원(이전의 $T_{avail} + q_i$)을 이용해 새롭게 사용할 수 있는 자원 T_{avail}을 새로 계산한다. 아직 완전한 품질로 서비스되고 있지 않은 요청 가운데, 하나를 큐에서 선택해 요청에 맞는 적당한 자원을 할당해 준다. 수행 스트림을 선택하는 방법으로 done_ratio의 값이 가장 작은 것을 택하며, 여기에 maximum $\Psi(T_{free\_res})$-rule을 적용한다.

$$T_{alloc} = \min(\Psi(T_{free\_res}), \text{remaining\_rate}_{min}) \quad (3)$$

선택한 요청(S_{min})에 자원을 할당해주며, 이용가능 한 자원이 더 이상 없거나 T_{alloc}이 너무 작아서 할당할 수 없을 때까지 루프를 계속 돈다.

3.2 Reclamation within Returned Reserve(RWR) :

이것은 DRA와 비슷한 방법이나, 차이가 있다면 재생이 끝나는 스트림들이 반환하는 자원들만 재분배 풀에 넣어 돌린다는 것이다. DRA에서는 자원의 이용을 위해 T_{avail}(v_i 대신에)이 사용된다. 스트림 당 T_{alloc}의 양은 *remaining_ratio* 대 *sum_remaining_ratio*의 비율에 따라 계산된다.

$$T_{alloc} = T_{avail} * (remaining\_ratio_k / sum\_remaining\_ratio) \quad (4)$$

여기서 기본 아이디어는 예약과 상관없이 스트림의 재생 종료로 인해 반환된 자원들만을 이용하는 것이다. 본 논문에서는 이 정책이 DRA 정책과 비교해 더 나은 성능을 제공해 준다는 걸 실험을 통해 검증해 보일 것이다. 예약된 자원들이 장시간 사용되지 않으면 그것의 일부가 재요청된다. 매 k 시간 동안 예약 요청이 없으면 재요청을 위해서 예약된 자원들의 일부를 반환한다. 이것은 예약 적응 방법의 일종이다. 예약기반 수락 제어 및 스케줄링 알고리즘(Reserve-based Admission Control and Scheduling Algorithm : RAC)은 다음과 같다. 다음 장에서는 기본 탐욕 알고리즘과의 성능 비교, 분석을 통한 세부적인 실험 평가 결과에 대해 논의할 것이다.

4. 실험 설계 및 평가

본 논문에서 제시한 수락 제어와 스케줄링 알고리즘의 성능을 검증하기 위해서, 다방면에 걸쳐 시뮬레이션을 수행하였다. 이 장에서는 다양한 부하 조건 하에 이루어진 시뮬레이션의 성능 결과를 제시한다. 이 장에서 제시된 결과는 Seagate Barracuda 4GB disk (ST3437IN)를 장착한 Sun Ultra Sparcstation 환경에서 실험하였다.

4.1 실험 설계

본 실험에서는 임의의 순서로 시스템에 도착하는 soft-QoS 요구를 가진 응용 프로그램의 상태를 실험하였다. 응용 프로그램은 재생 속도(요구되는 자원의 수와 일치), 사용자 프로파일과 데이터 내용(데이터 비율 파라미터의 값)에 의존한다. 예를 들어, 원격 사용자에게 다른 타입으로 된 비디오 스트리밍을 제공하는 비디오 분산 응용 프로그램의 집합이 있을 수 있다. 시뮬레이션을 위해 본 연구에서는 다음과 같은 속성을 가정한다(<표 2>는 실험에 사용된 속성을 보여준다) :

응용 프로그램들은 포아송(Poisson) 프로세스의 평균 도착 간격 속도(inter-arrival rate : λ)를 기반으로 실행되며, 지속 시간은 프로세스의 평균값과 표준 편차가 가우시안(Gaussian) 분포를 따른다. 본 실험은 프로세스 파라미터의 생성에 따라

실행되며, 응용 프로그램 트래픽의 기록들이 생성된다. 다음과 같은 내용들을 측정한다 :

〈표 2〉 성능평가에서 사용한 요소들

파라미터	설 명
λ	응용 프로그램들의 도착을 기반으로 하는 포아송 프로세스의 평균 도착 간격 속도 비율
μ_d	응용 프로그램의 지속시간을 의미하는 가우시안 프로세스의 평균값
σ_d	응용 프로그램의 지속 시간 동안 분산된 가우시안 프로세스의 표준 편차 요소

1. 여러 수락된 스트림들의 초과 시간을 누적한다 : 만약 최소한으로 이용 가능한 자원들이 있다면 스트림들은 수락된다($> MIN\_FRACTION_i$). 기본적인 탐욕 알고리즘과 RAC의 두 정책들은 수락된 스트림들의 수가 끼치는 영향을 평가해서 비교한다.
2. 총 시스템 이용률(total system utilization) : 총 시스템 이용률은 스트림을 지원하는 자원들의 done_ratios 총 합으로 계산한다 :

$$total\_system\_utilization = \sum_{Sk \in RS} done\_ratio_k \qquad (5)$$

3. 총 품질(total quality) : 총 품질은 총 시스템 이용률을 현재 실행되고 있는 스트림의 수로 나누어 계산한다 :

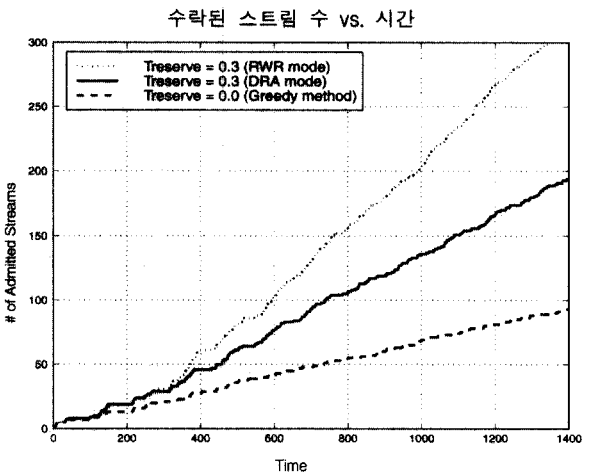
$$total\_quality = \left(\sum_{Sk \in RS} done\_ratio_k \right) / number\_of\_current\_running\_streams \qquad (6)$$

4. 수락 비율 : 수락 비율은(수락된 *스트림 개수/총 입력 스트림 개수*)로 구한다.
5. 디스크 I/O 대역폭 이용률 : 디스크 I/O 대역폭 이용률은 T_{used}/T_{total} 로 구한다.
6. 자원 예약 규모($T_{reserve}$)의 영향 : 시간에 따라 수락된 스트림들의 수에 영향을 미치는 예약($T_{reserve}$) 크기 속성을 조사한다.

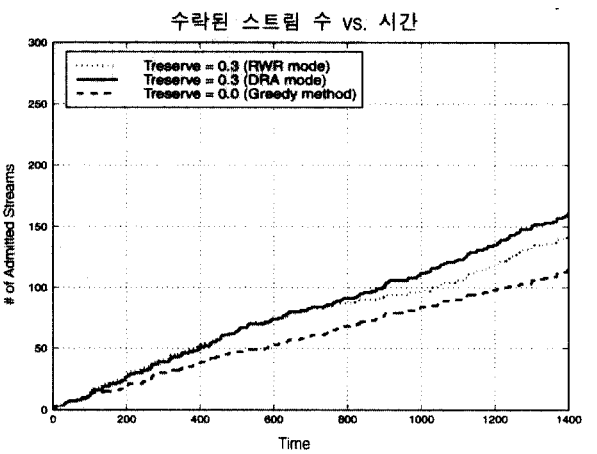
실험을 위해 크게 두 가지의 스트림 데이터 속도(트래픽)을 입력으로 가정하였다. 혼잡한 트래픽에서는 λ = 0.125, μ_d = 90, σ_d = 3.5로 설정되며, 중간 트래픽에서는 λ = 0.125, μ_d = 60, σ_d = 3.5로 설정된다. x축의 값은 정규화된다. 혼잡한 트래픽에서는 중간 트래픽에서 보다 더 자주 자원을 요구한다.

4.2 실험 1 : 수락된 스트림의 수

*RAC*의 주요 장점은 다수의 스트림들을 수락할 수 있도록 해주며, 동시에 유사한 수준의 QoS로 동시에 많은 스트림들을 실행시킬 수 있도록 보장하는 것이다. 이 장에서는 수락된 스트림들의 누적 수를 가지고, *RAC*(*RWR*과 *DRA*) 알고리즘과 탐욕 알고리즘의 효율성을 비교한다. (그림 1)은 혼잡한 트래픽과 중간 트래픽에 대해 시간에 따라 수락된 스트림의 누적 수를 보여준다. 트래픽이 부하 상태에 있을 경우, 이용 가능한 자원들보다 더 많은 자원들을 요구하게 되면 기존의 방법과 본 논문에서 제시한 방법 모두 수락 비율(거부비율 증가)이 감소된다. 그러나 *RAC*는 전반적으로 기존의 탐욕 알고리즘과 비교해 볼 때 수락 비율 면에서 향상된 성능을 얻을 수 있으며 특히 *RWR* 모드의 경우 최고의 성능을 얻을 수 있다. 혼잡한 트래픽 상태에서도 *RAC* 알고리즘은 기존의 알고리즘보다 더 나은 성능을 얻을 수 있다.



(a) 혼잡한 트래픽

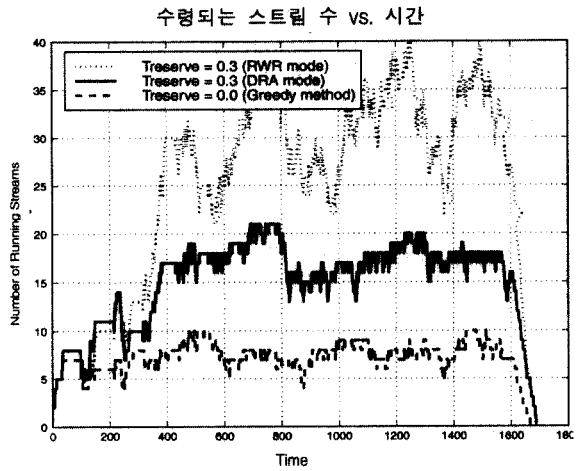


(b) 중간 트래픽

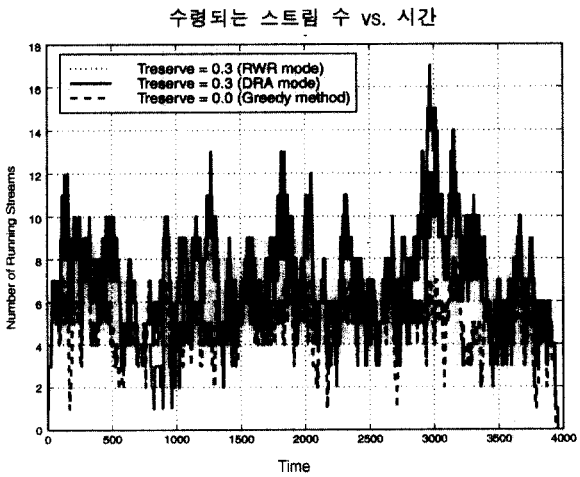
(그림 1) 수락된 스트림 수 vs. 시간

4.3 실험 2 : 동시 수행 스트림의 수

다른 관점에서 살펴보면(그림 2), *RWR* 알고리즘을 이용하여 서버에 의해 동시에 서비스할 수 있는 스트림의 수가 200~300% 증가하였으며 또한 DRA 알고리즘의 경우 약 100% 증가되었다. *RAC* 알고리즘(*RWR*과 *DRA* 모두)은 기존의 탐욕 알고리즘과 비교해 볼 때 거부 비율(rejection ratio)을 상당히 감소시킬 수 있다.

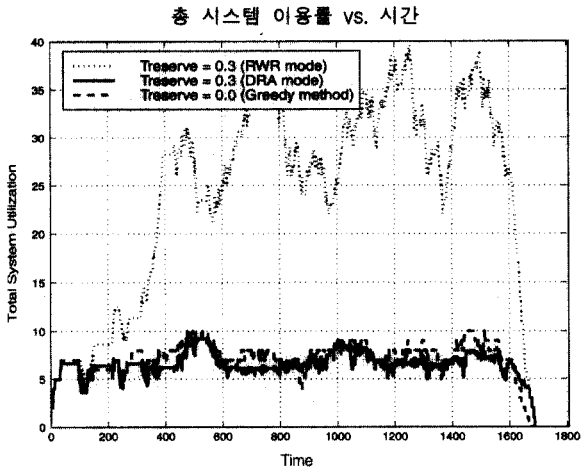


(a) 혼잡한 트래픽

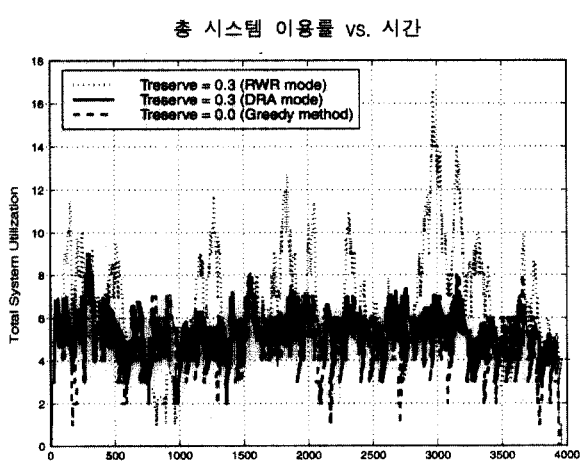


(b) 중간 트래픽

(그림 2) 동시에 수행되는 스트림 수 vs. 시간



(a) 혼잡한 트래픽



(b) 중간 트래픽

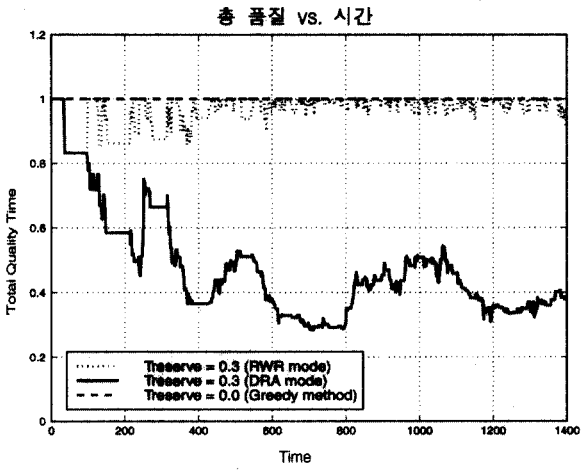
(그림 3) 총 시스템 이용률 vs. 시간

4.4 실험 3 : 시스템 이용률

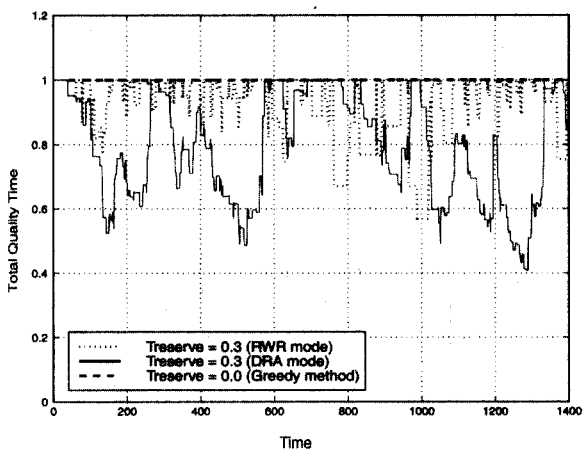
이 장에서는 주로 총 시스템 이용률과 디스플레이 재생 품질(quality)과 관련해서 RAC 알고리즘이 미치는 영향을 시각적으로 표현하는데 초점을 맞춘다. 4.2절에서 RWR과 DRA 알고리즘의 총 품질이 개선되었음을 살펴보았다. (그림 3)은 응용프로그램 각각의 done_ratio 값의 합을 이용해 총 시스템 이용률을 보여준다. 응용 프로그램의 도착 분포는 결과곡선에 영향을 미치는데 RWR에 의해 얻어진 총 시스템 이용률은 탐욕 알고리즘에 비해 대부분의 시간동안 더 높게 나타난다. 혼잡한 트래픽 상태에서 실시한 시뮬레이션 결과를 보면 서비스 품질이 상당히 증가했다는 것을 알 수 있다. 따라서, RWR 알고리즘이 탐욕 알고리즘에 비해 더 효과적으로 시스템 자원들을 이용할 수 있다는 결론을 내릴 수 있다.

4.5 실험 4 : 재생 총 품질

RAC 알고리즘의 디스플레이 재생 품질을 살펴보기 위해서 다른 관점에서 살펴보았다. (그림 4)에서 총 품질은 현재로 나누어 계산하였다. (그림 4)를 살펴보면 총 품질은 RWR 알고리즘과 탐욕 알고리즘이 거의 같다는 사실을 알 수 있다. 단지 시뮬레이션의 초기 부분에서 총 품질이 때때로 1.0 이하로 떨어지긴 하나 대체적으로 0.85 이상을 유지해 좋은 양상을 보인다. 그러나 그 이후에는 총 품질이 거의 1.0 값을 유지한다. 이와는 대조적으로 DRA 알고리즘의 총 품질은 전반적으로 상당히 낮게 나타난다. 이것은 DRA 알고리즘이 미래의 스트림들에 대해 예약을 하지 않았기 때문인데, 현재 실행되고 있는 스트림들 중의 일부가 종료되기도 전에 스트림들이 도착해 서비스의 질을 떨어뜨린다. 중간 트래픽에서 DAR 알고리즘은 대체적으로 양호한 성능을 보이긴 하나 전체적으로는 낮은 성능을 보인다. 이 사실로부터 자원을 재요청하기 위해 예약을 할 경우 더 신중해야 한다는 사실을 알 수 있다. RWR 모드에서는 반환된 자원들에 대해 재요청을 예약하지 않았다.

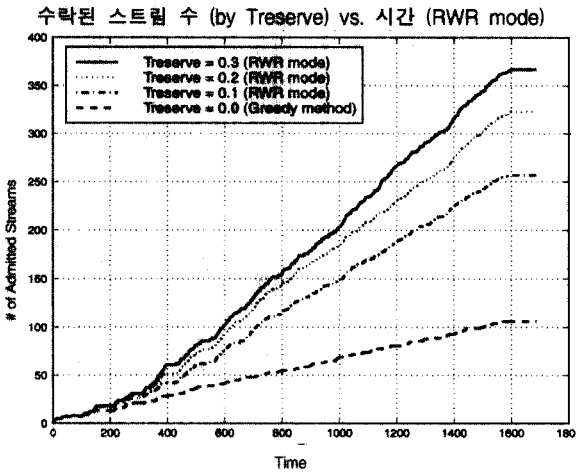


(a) 혼잡한 트래픽

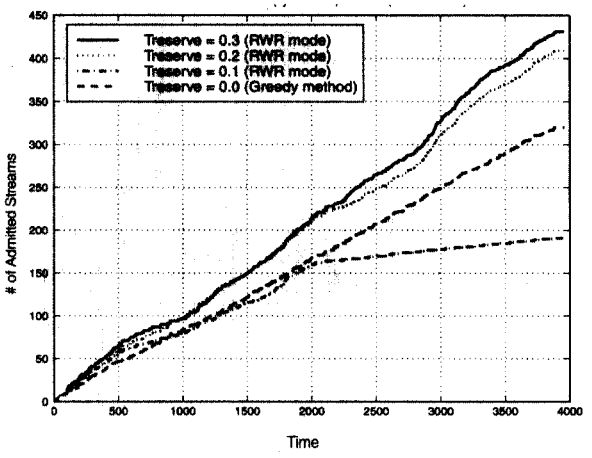


(b) 중간 트래픽

(그림 4) 총 품질 vs. 시간



(a) 혼잡한 트래픽



(b) 중간 트래픽

(그림 5) 수락된 스트림 수 vs. 시간

4.6 실험 5 : 스트림 수 대 임계치

본 논문에서는 예약($T_{reserve}$)을 위해 준비된 자원들이 시스템 성능에 미치는 영향에 대해 관심을 가졌다. 시뮬레이션 실험을 통해 자원 할당(자원을 예약 한 후 응용 프로그램에게 1/k 만큼 자원을 할당한다)은 일관된 정책으로 검증할 수 있으며 그 성능은 $T_{reserve}$에 의존한다는 사실을 알게 되었다. 이러한 결과는 시스템과의 관계에서 이해될 수 있는데 예약 상태에서 더 많은 자원들이 유지된다는 것은 시스템이 질적인 손실 없이 응용 프로그램을 지원하고 있다는 것이다. 만약 응용 프로그램의 통계치가 일정하다면, 이것은 $T_{reserve}$가 최적의 값이며 총 품질이 최대로 증가했다는 것을 의미한다. (그림 5)는 수락된 스트림들의 수와 시간에 따른 네 종류의 $T_{reserve}$ 값과의 관계를 보여준다. 실험 결과에 따르면 시스템은 예약 값(reserve value)이 0.2와 0.3 사이일 때 최대의 성능을 얻을 수 있다.

5. 관련 연구

수락 제어(admission control)는 자원을 요구하는 많은 응용 프로그램에서 사용되는데, 예를 들어, 비디오 스트림과 같은 응용 프로그램이나 네트워크 상에서 사용되거나 연속적인 미디어 서버에서 사용되는 자원 할당을 위한 수락 등을 들 수 있다[15, 14]. 재생 비율 요구에 따라 한 라운드 동안 검색되는 미디어 스트림에는 많은 방해 요소가 존재한다. 따라서, 여러 수락 제어 알고리즘들이 생겨났다. 각 스트림 마다 확실한 연속 검색의 서비스 시간은 최소한의 재생시간을 넘겨서는 안 된다. 이것이 전형적인 탐욕 수락 제어 전략이다. 항상 기본적인 수락 제어 결정은 최악의 경우를 기반으로 하는데, 가장 최악의 경우는 멀티미디어 트래픽이 버스트(burst) 상태에 있을 때이다. 이때 이용할 수 있는 자원이 충분하지 않으면 응용 프로그램은 실행되지 않을 수 있다[17].

수락 제어 알고리즘의 공식화는 클라이언트가 요구하는 서비스의 특성에 달려 있다. 그러나 그것은 서비스 시간에 대한

최악의 경우를 고려해서 사용에 제한을 해야한다[1]. 기존의 많은 연구들이 이와 비슷한 응용 프로그램 스트림의 최악의 경우를 분석해 수락 제어를 공식화하였다[14, 15]. 수락 제어 전략을 사용한 시스템의 예로, Vosaic[3]을 들 수 있다. Vosaic은 웹 상에서 비디오나 오디오를 지원하는 Mosaic을 확장한 것이다. Vosaic은 매우 간단한 수락 제어 프로토콜을 사용하기 때문에 이것은 웹 서버에 연결된 많은 수의 스트림들을 동시에 제약한다. 최악의 경우를 가정하기 때문에 결정적인 수락 제어 알고리즘은 과도한 제약과 엄격한 성능 보장으로 인한 서버상 자원들의 낮은 이용률을 야기시킨다.

따라서 자원의 이용률을 개선하기 위해서는 클라이언트를 위한 관찰기반(observation-based) 수락 제어 알고리즘이 제안되어야 한다[12]. 이것은 최악의 경우를 가정하는 것 대신에 미디어 블록을 검색하는데 소요되는 평균 시간을 이용하는 방법이다. 이에 비해 Vin[19] 등은 통계적인 수락 제어 알고리즘을 제시하고 있다. 이것은 새로운 클라이언트들에게 가능하면 데이터 요구률이 최고조일 때 응답하기보다는 밀집된 데이터 요구율을 통계적으로 측정해서 서비스를 수락해주는 것을 말한다. 이 기법에서는 서버 자원의 이용률을 미디어 블록에 대한 디스크 엑세스 시간 편차를 이용하거나, 압축 기술의 가변율에 의해 요구되는 재생율의 편차를 이용해서 개선하고 있다. 그러나 이 방법은 통계상으로는 안정적일지 모르나 최악의 경우를 고려할 때 자원 혼잡으로 인해 응용 프로그램이 실행되지 않을 수 있다는 단점이 있다.

수락 제어의 다른 타입으로는 서비스 클래스로 구별하는 것이 있다. Symphony 시스템[24]은 QoS 인식(QoS-aware) 스케줄러를 제안하였는데, 한 라운드 안에서 클래스들이 서비스하는 데 소요되는 시간들을 분할해 주기적인 또는 비주기적인 실시간 요청이나 소위 best-effort 요청과 같은 서비스 클래스들을 인접하게 분류하는 방법이다. Fellini 멀티미디어 서버[13]는 Sympony와 비슷한 수락 제어 알고리즘을 제안하였다. 이것은 서비스 클래스(실시간과 비실시간)들을 기반으로 수락 제어 기능을 지원하는 방법이다. 응용 프로그램의 클래스들은 하나의 클래스 상에 있는 응용 프로그램들과 그것에 영향을 미치는 다른 응용 프로그램들로부터 분리시켜 구별한다. 그러나 하나의 클래스 안에는 여전히 자원 혼잡 문제가 존재한다.

위에 언급된 수락 제어 전략들은 모두 시스템으로부터 낮은 성능을 야기할 수 있는 QoS 협상 문제들을 고려하지 않았다.

6. 결론 및 향후 연구

본 논문에서는 비디오 서버 시스템을 위한 통합된 동적/적응성 수락 제어와 자원 스케줄링 알고리즘을 제시하였으며, 또한 수락 비율, 총 품질, 대역폭 이용률을 측정해서 RAC 알고리즘과 탐욕 알고리즘의 시뮬레이션 평가 결과를 제시하였다. 혼잡한 트래픽 상태에서 RAC 알고리즘은 탐욕 알고리즘과 비교해 볼 때 향상된 성능을 얻을 수 있었다. 본 논문에서 제안한 기법을 사용할 때 더 많은 스트림들이 주어진 시스템 자원을 이용하며 재생 서비스 품질을 유지할 수 있는 범위 하에서 실행 될 수 있다는 것을 확인할 수 있었다. 향후 연구 과제로는 RWR 알고리즘에 다중 차원의 것들(예 프레임 사이즈, 버퍼 요구, 압축 품질 요소, 네트웍 대역폭 등)을 고려한 확장된 QoS에 대한 연구가 필요할 것이다. 현재 사용자 요구 함수 및 자원 요청 함수를 이용한 시스템 모델링에 대한 연구가 진행 중이며 이것은 시스템과 사용자 요구를 더 세분화해서 공식화할 수 있다.

참 고 문 헌

[1] D. Anderson, Y. Osawa, and R. Govindan. A File System for Continuous Media. *ACM Trans. Computer Systems*, 10(4) : 311-337, 1992.

[2] J. T. C.-S. Chang. Effective Bandwidth in High-Speed Networks. *IEEE Journal on Selected Areas in Communication*, 13(6) : 1091-1100, Aug. 1995.

[3] Z. Chen, S. M. Tan, R. H. Campbell, and Y. Li. Real Time Video in the World Wide Web. In *Processing of 4th World Wide Web Conference*, Boston, MA, December, 1995.

[4] F. T. Ernst W. Biersack. Statistical Admission Control in Video Servers with Constan Data LENGTH retrieval of VBR Streams. In *Third International Conference on Multimedia Modeling*, Toulouse, France, Nov. 1996.

[5] J. W. G. de Veciana, G. Kesidis. Resource Management in Wide-Area ATM Networks Using Effective Bandwidths. *IEEE Journal on Selected Areas in Communication*, 13(6) : 1081-1090, Aug. 1995.

[6] R. Harinath, W. Lee, S. Parikh, D. Su, S. Wadhwa, D. Wijesekera, J. Srivastava, and D. Kenchammana-hosekote. A Multimedia Programming Toolkit/Environment. In *Proceedings of 1997 International Conference on Parallel and Distributed Systems (ICPADS97)*, Seoul, Korea, December, 1997.

[7] R. Haskin and F. Schmuck. The Tiger Shark File System. In *COMPCON 96*, 1996.

[8] D. Kenchammana-Hosekote and J. Srivastava. I/O Scheduling for Digital Continuous Media. *ACM-Springer Multimedia Systems Journal*, 5(4) : 213-237, 1997.

[9] S. W. Lau and J. C. S. Lui. A Novel Video-On-Demand Storage Architecture for Supporting Constant Frame Rate with Variable Bit Rate Retrieval. In *5th International Workshop on Network and Operating Systems Support for Di-*

gital Audio and Video, Durham, N. H., 1995.

[10] W. Lee and B. Sabata. Admission Control and QoS Negotiations for Soft-Real time Applications. In *Proceedings of the Workshop on Multimedia Networking Computing Systems (ICMCS99)*, Florence, Italy, June, 1999.

[11] D. Wijesekera and J. srivastava. Experimental Evaluation of Loss Perception in Continuous Media. *ACM Springer Multimedia Systems Journal*, 2000.

[12] D. Makaroff, G. Neufeld, and N. Hutchinson. An Evaluation of VBR Disk Admission Algorithm for Continuous media File Servers. In *Proceedings of the ACM Multimedia Conference*, Seattle, Wa, Dec. 1997.

[13] C. Martin, P. S. Narayanan, B. Ozden, R. Rastogi, and A. Silberschatz. The Fellini Multimedia Storage Server. In S. M. Chung, editor, *Multimedia Information Storage and Management*. Kluwer Academic Publishers, 1996.

[14] P. V. Rangan and H. M. Vin. Designing a Multiuser HDTV Storage Server. *IEEE Journal on Seleted Areas in Communications*, 11(1) : 153-164, January, 1993.

[15] A. Reddy and H. M. Vin. Designing a Multimedia System. *Computer*, 27(3) : 69-74, 1994.

[16] P. J. Shenoy, P. Goyal, S. S. Rao, and H. M. Vin. Symphony : An Integrated Multimedia File System. In *Proceedings of SPIE/ACM Conference on Multimedia Computing and Networking (MMCN98)*, San Jose, CA, January, 1998.

[17] S. K. T. S. V. Raghavan. *Networked Multimedia Systems : Concepts, Architecture, and Design*. Prentice Hall, 1998.

[18] H. M. Vin, P. Goyal, A. Goyal. An Observation-Based Approach for Designing Multimedia Servers. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, Boston, MA, May, 1994.

[19] H. M. Vin, P. Goyal, A. Goyal. A Statistical Admission Control Algorithm for Multimedia Servers. In *Proceedings of ACM Multimedia '94)*, San Francisco, October, 1994.

이 원 준

e-mail : wlee@korea.ac.kr

1989년 서울대학교 공과대학 컴퓨터공학과 졸업(학사)

1991년 서울대학교 공과대학 컴퓨터공학과 졸업(석사)

1996년 University of Maryland(M.S. in Computer Science)

1998년 Stanford Research Institute(SRI), Research Associate

1999년 University of Minnesota(Ph.D. in Computer Science & Engr.)

2000년 University of Missouri, Assistant Professor

2001년 이화여자대학교 컴퓨터공학과 전임강사

2002년~현재 고려대학교 컴퓨터학과 조교수

관심분야 : 멀티미디어 시스템 및 네트워크, 이동 컴퓨팅, 분산 시스템