온라인 대역폭 평활화를 위한 태스크 스케줄링 기법*

김 재 욱* 하 란** 차 호 정***
*홍익대학교 정보공학과
**홍익대학교 컴퓨터공학과
***광운대학교 컴퓨터과학과
{jwkim, rhanha}@cs.hongik.ac.kr hojung@cs.kwangwoon.ac.kr

Task Scheduling Technique for Online Bandwidth Smoothing

Jae-Wook Kim* Rhan Ha** Hojung Cha***
*Dept. of Information Engineering, Hongik University
**Dept. of Computer Engineering, Hongik University
***Dept. of Computer Science, Kwangwoon University

요 약

자원 예약을 보장하지 못하는 네트워크 상에서 실시간 멀티미디어 응용의 만족스러운 서비스를 위해서는 효과적인 종단 시스템의 운영이 필요하다. 이러한 운영 기술 중 버퍼를 이용한 응용 프로그램 수준의 대역폭 평활화 기술과 태스크 스케줄링 기법은 핵심적인 부분을 차지한다. 기존에 제안된 온라인 대역폭 평활화 기법은 동적 슬라이딩 윈도우와 동적 버퍼를 이용하여 네트워크에 적응할 뿐만 아니라 요구 대역폭 평활화를 통해 지터를 흡수, 완충시킴으로써 더 나은 서비스 품질을 보장한다. 그러나 응용 프로그램 수준에서 동작하기 때문에 프로그램의 수행 품질 보장을 위한 시간 대역을 보장하지 못할 뿐만 아니라 지터의 발생 가능성이 높다. 본 연구에서는 효과적인 버퍼 관리를 통해 서비스 품질의 손실을 최소화하는 적응성 있는 온라인 대역폭 평활화 기법이 안정적으로 동작하기 위한 최적화된 태스크 스케줄링 기법을 제안한다. 먼저 태스크 스케줄링 기법은 멀티미디어 데이터의 생산과 소비를 적시에 보장할 수 있도록 하여 지터의 발생을 최소화하고, 만일 네트워크 지터가 발생하더라도 응용 프로그램 수준의 버퍼 관리를 통해 완충시킴으로서 서비스 품질의 손실을 최소화한다. 모의 실험에서는 제안된 기법이 온라인 대역폭 평활화 기법을 효과적으로 지원함을 보인다.

1. 서론

컴퓨터와 네트워크의 빠른 발전은 원격지 사용자를 위한 다양한 멀티미디어 응용 서비스를 가능하게 하였다. 멀티미디어 서비스는 주기적 전송을 위한 큰 대역폭을 요구하지만 기존의 TCP/IP를 기반으로 하는 일반적인 네트워크는 이러한 요구를 보장하지 못한다. 이러한 문제점 해결을 위해 RSVP나 IPv6 등이 제안되었으나 스위치를 교체하는 네트워크의 전반적 구조 재편이 필요하다. 그러나 충분한 구조 재편을 위해서는 적지 않은 시간이 필요하므로 현실적으로 적용 가능한 새로운 기술이 필요하다. 이러한 기술에는 여러 가지가 있으나, 종단 시스템에서 동작하는 태스크 스케줄링 기술과 버퍼 관리를 이용한 응용 프로그램 수준의 대역폭 평활화 기술이 핵심적인 부분을 차지한다.

제안된 온라인 대역폭 평활화 기법은 동적 네트워크 상에서 멀티미디어 서비스 품질을 최대화하기 위해서 정적인 평활화 방법의 이점을 적용하여 동적 네트워크에 잘 적응하며 요구 대역의 평활화를 통해 더 좋은 서비스 품질을 보장하는 기법이다. 제안된 온라인 대역폭 평활화 기법은 기존의 버퍼 관리 기법들과 다른 몇 가지 특징을 가지고 있다[1, 2, 3, 4]. 기존의 윈도우 기반의 평활화 기법은 네트워크의 상태를 고려하지 않고 고정 크기의 윈도우를 사용하기 때문에 네트워크 부하의 변화에 적응하기 못한다. 반면 제안된 온라인 대역폭 평활화 기법은 네트워크의 상태에 따라 동적 윈도우의 크기를 결정하는 동적 슬라이딩 윈도우를 사용한다[5]. 또한 클라이언트 버퍼를 두 영역으로 나누어 운영함으로서 네트워크 지터의 적응성과 요구 대역폭 평활화를 동시에 지원한다. 그러나 응용 프로그램 수준에서 동작하기 때문에 프로그램의 수행을 위한 시간 대역을 보장하지 못할 뿐만 아니라 지터의 발생 가능성이 있다.

* 본 연구는 정보통신연구진흥원(과제번호 : C1-1999-1234-00), 한국과학재단(과제번호 : KOSEF97-0102-05-01-3), 교육부 BK(과제번호 : 991031001)의 후원으로 연구되었습니다.

본 연구에서는 제안된 온라인 대역폭 평활화 기법이 안정적으로 동작할 수 있도록 최적화된 태스크 스케줄링 기법을 제안한다. 제안된 태스크 스케줄링 기법은 온라인 대역폭 평활화 기법이 멀티미디어 데이터를 적시에 생산 및 소비할 수 있도록 지원하고 지터가 최소화 될 수 있도록 한다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 온라인 대역폭 평활화의 기법과 특징에 대해 기술하고 3장에서는 온라인 대역폭 평활화를 위한 최적화 된 태스크 스케줄링 기법을 제안한다. 4장에서는 실험을 통해 제안한 기법의 성능을 분석하고 5장에서는 결론과 향후 과제에 대해 논한다.

2. 온라인 대역폭 평활화 기법

온라인 대역폭 평활화를 위한 시스템은 그림 1과 같이 서버 측의 동적 윈도우와 클라이언트 쪽의 동적 버퍼로 구성되어 있다. 서버의 평활화 윈도우는 $\alpha < W$인 α씩 슬라이딩하며 중첩되게 전송 계획을 수립한다. 매 α마다 전송 계획을 수립하기 때문에 슬라이딩하지 않는 일반 윈도우에 비해 평균 W/α배 더 계산을 해야하는 부담을 가지고 있지만 네트워크의 변화를 반영하여 재 계산할 수 있는 기회를 가지기 때문에 적응성 있는 전송 계획을 작성할 수 있다. 슬라이딩 폭 α는 실험에 의해 가장 효율적으로 밝혀진 $W/2$의 값을 사용한다[1]. 그러나 제안한 동적 슬라이딩 윈도우는 기존의 접근 방법과 차이가 있다. 기존의 접근 방법이 고정된 크기의 슬라이딩 윈도우를 이용하였지만 제안하는 방법은 네트워크의 상태에 따라 윈도우의 크기가 변하도록 하였다. 제안한 시스템은 클라이언트의 버퍼로 전송되어 적재된 데이터의 적재 상태에 의해 네트워크의 상태를 예측한다. 네트워크의 상태는 initial state, congestion state, slow-start state, steady state로 정의하고 각 상태는 그림 2에서와 같이 전이된다. 처음 initial state에서 윈도우의 크기는 초기 입력 값을 가지게 된다.

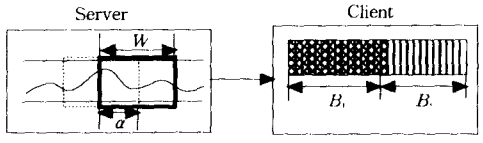

그림 1. 온라인 대역폭 평활화 기법을 위한 시스템 구성

그러나 네트워크의 상태 변화나 지터로 인하여 적재한 데이터의 양이 전송 계획과 다르게 진행되어 클라이언트 평활화 영역의 언더플로우가 초래되면 congestion state로 들어가고 윈도우의 크기는 반으로 줄인다. congestion state에서는 윈도우의 크기를 변화시킨 후, slow-start state로 들어간다. slow-start state에서는 초기 상태의 윈도우 크기로 돌아가기 위해 윈도우 크기를 서서히 증가시킨다. 윈도우의 크기가 초기 상태와 같아지면 slow-start state에서 벗어난다. 반대로 네트워크의 상태가 안정적이라면 steady state로 들어가게 되고 initial state의 윈도우 크기 이상 증가하여 평활화 영역의 범위를 넓힌다. 제시한 알고리즘에 의해 윈도우의 크기가 결정되면 서버는 현재 가용 대역폭으로 동적 윈도우 내의 모든 데이터가 전송될 수 있는지 검사한다. 만일 전송이 불가능하다면 서버는 네트워크의 대역폭에 맞도록 데이터의 품질을 제어한다. 전송 계획은 [2]에 제시된 $O(n)$의 복잡도를 가진 *optimal off-line smoothing algorithm*을 사용한다.

클라이언트 측의 버퍼는 지터의 제거뿐만 아니라 멀티미디어 스트림의 요구 대역폭 평활화를 위해 사용한다. 고정된 크기를 가진 클라이언트 버퍼를 B_t의 크기를 가진 지터 대비 영역과 B_s의 크기를 가진 평활화 영역의 두 부분으로 나눈다. 지터 대비 영역은 재생 시작 전에 일반 프레임 데이터로 채워두고 평활화 영역은 서버의 동적 슬라이딩 윈도우와 함께 전송 계획을 만들고 이에 따라 데이터를 적재한다. 만일 결정된 계획에 의해 전송하는 도중에 지터가 발생하여 전송에 차질이 생기면 지터 대비 영역에 선적재하여 둔 데이터를 소모하게 된다. 두 가상의 영역은 서버 측의 슬라이딩 윈도우와 유사하게 그림 2의 네트워크 상태에 따라 크기가 변한다.

제안된 온라인 대역폭 평활화를 위한 스케줄링 기법의 제안을 위해 각 태스크의 특징을 알아보면 다음과 같다. 서버나 클라이언트 시스템은 반드시 수행되어야 하는 경성 실시간 태스크, 수행되지 않아도 큰 성능 저하는 없는 연성 실시간 태스크, 시간적 제약이 없는 일반 태스크로 구성되어 있다. 경성 실시간 태스크의 예로는 서버의 전송 계획 수립이나 클라이언트 측의 버퍼 관리를 들 수 있다. 서버에서 전송 계획의 수립 없이는 클라이언트로 전송이 불가능하거나 클라이언트 측의 버퍼의 변화나 서버로의 피드백이 없으면 전송 계획의 오류로 버퍼의 언더플로우를 초래할 수 있다. 연성 실시간 태스크의 예는 클라이언트의 디코딩 작업을 들 수 있는데, 이 작업은 수행을 하지 않아도 응용 서비스에 큰 손실을 입지는 않는다.

3. 온라인 대역폭 평활화를 위한 태스크 스케줄링

이 상에서는 앞 절에 기술한 온라인 대역폭 평활화 기법이 안정적으로 동작될 수 있도록 시스템 특징에 맞게 최적화된 태스크 스케줄링 기법을 제안한다.

기존에 멀티미디어 응용을 위한 스케줄링 기법에 대한 많은 연구가 수행되었는데 대표적인 예로는 Fair Queueing 기법을 응용한 SMART[6]나 계층적 스케줄러[7]등이 있다. SMART의 경우 실시간 응용, 응답형 응용, 일반 응용으로 분류하여 중요도를 부여하고 각 태스크의 자원 할당을 결정하는 데 사용하였으며, biased virtual finish time이라는 간단한 메커니즘을 사용하여 우선 순위와 균등 분배를 동시에 고려하였다. 그러나 응용 프로그램 단위로 우선 순위를 부여하기 때문에 경성 실시간 태스크, 연성 실시간 태스크, 일반 태스크의 세 종류를 모두 가지고 있는 온라인 대역폭 평활화 기법에는 적용하지 못한다. 계층적 스케줄러의 경우, 여러 종류의 스케줄러를 계층적으로 동시에 지원한다. 각 스케줄러에 CPU 할당 비율을 정의하고 각 스케줄러 내의 태스크는 start-time fair queueing[8]을 이용하여 공정하게 CPU 자원을 나누어 갖는다. 그러나 이 기법은 각 스케줄러가 독립적으로 동작하기 때문에 각 태스크 사이의 선행 관계나 종속적인 관계를 만족시키지 못하여 응용 프로그램 단위로 스케줄링 하여야 한다. 따라서 온라인 대역폭 평활화에 적용시키지 못한다.

따라서, 본 논문에서는 경성 실시간 태스크, 연성 실시간 태스크, 일반 태스크를 동시에 지원하며 각 태스크 사이의 선행 관계나 종속적인 관계를 유지할 수 있는 간략한 태스크 스케줄링 기법을 제안한다. 제안한 태스크 스케줄링 기법은 온라인 대역폭 평활화 시스템이 적시에 멀티미디어 데이터의 생산과 소비가 가능하도록 하며 지터를 최대한 줄인다. 온라인 대역폭 평활화를 위한 태스크를 먼저 아래와 같이 분류한다.

- 경성 실시간 태스크 : 서버의 전송 계획 수립, 클라이언트 버퍼 변화
- 연성 실시간 태스크 : 클라이언트의 디코딩 태스크
- 일반 태스크 : 데이터 상태 출력 태스크 등

분류된 세 가지 태스크는 각각 다음과 같은 특징을 갖는다. 경성 실시간 태스크는 지정된 데드라인 이전에 반드시 수행되어야 한다. 각 태스크의 선행 관계에 따라 경성 실시간 태스크의 수행이 없으면 연성 실시간 태스크를 수행할 수 없다. 예를 들면 서버의 전송 계획의 수립이 없이는 데이터의 전송이 불가능한 상황을 들 수 있다. 연성 실시간 태스크는 반드시 수행되어야 하지는 않지만 어느 한 태스크가 계속적으로 수행되지 않는 것을 막기 위하여 무시된 횟수를 유지하고 경쟁 관계에 있는 다른 연성 실시간 태스크의 무시 횟수와 비교하여 통과함으로써 어느 한 태스크의 기아현상을 막는다.

제시된 각 태스크 종류별로 별도의 큐를 둔다. 스케줄 되기를 원하는 각 태스크는 종류별로 각 큐에 대기한다. 대기하는 태스크는 데드라인이 빠른 순서로 정렬한다. 제일 먼저 수행될 경성 실시간 태스크가 있는지 찾아본다. 만일 수행될 태스크가 있으면 스케줄 한다. 스케줄 할 경성 실시간 태스크가 없으면 연성 실시간 태스크를 찾는다. 만일 인터럽트나 기타 지연에 의해 다음에 수행될 경성 실시간 태스크의 데드라인 위반을 초래하면 이 연성 실시간 태스크는 수행하지 않는다. 경성 실시간 태스크의 위반은 초래하지 않으나 다음에 스케줄 할 연성 실시간 태스크의 데드라인 위반을 초래하면 두 태스크의 무시된 횟수를 비교하여 더 많이 무시된 태스크를 수행한다. 즉 그림 3의 (a)와 같이 스케줄하고 다음 태스크를 큐에서 제거한다. 반대의 경우는 그림 3의 (b)와 같이 스케줄 할 태스크를 무시한다. 만일 무시된 횟수가 같으면 오래된 태스크 집합의 태스크를 우선적으로 무시한다. 횟수는 (무시 횟수, 수행 요청된 횟수)로 유지하는데, 그 비교는 (무시 횟수 / 수행 요청 횟수)로 한다. 마지막으로 CPU가 휴식 중에 있거나 무시에 의해 빈틈이 생기면 일반 태스크를 스케줄한다. 따라서 세 종류의 태스크를 모두 지원하면서 인터럽트의 발생에 효과적으로 대응할 수 있다. 알고리즘은 그림 4에 나타내었다.

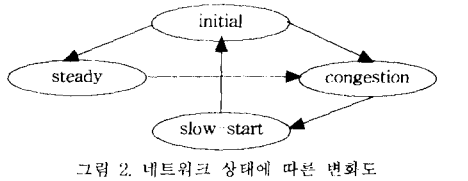

그림 2. 네트워크 상태에 따른 변화도

(a) 현재 수행될 태스크가 더 많이 무시된 경우

(b) 큐에 대기하고 있는 태스크가 더 많이 무시된 경우

그림 3. 연성 실시간 태스크의 무시 횟수 비교

step 1. 스케줄 되기를 원하는 태스크는 종류에 따라 경성 실시간, 연성 실시간, 일반 태스크로 분류하여 큐에 대기한다. (각 큐의 태스크는 데드라인 순으로 정렬)
step 2. 스케줄되기를 요청한 경성 실시간 태스크가 있으면 스케줄한다.
step 3. 스케줄 된 경성 실시간 태스크가 없으면 스케줄 되기를 요청한 연성 실시간 태스크를 찾는다.
step 4. 만일 스케줄 될 연성 실시간 태스크가 다음에 스케줄 될 경성 실시간 태스크의 데드라인 위반을 초래하면 무시 횟수를 증가시키고 큐에서 제거하고 무시한다.
step 5. 만일 스케줄 할 연성 실시간 태스크가 다음의 연성 실시 간 태스크의 데드라인 위반을 초래하면 다음 연성 실시간 태스크의 무시된 횟수와 비교하여 더 적게 무시된 태스크를 큐에서 제거하고 무시 횟수를 업데이트 한다.
step 6. 만일 CPU가 휴식 중이거나 연성 실시간 태스크의 무시로 여유가 생겼다면 일반 태스크를 수행한다.

그림 4. 태스크 스케줄링 알고리즘

4. 성능 분석

이 장에서는 제안한 태스크 스케줄링 기법을 사용하여 얻을 수 있는 성능을 평가하기 위해 실험 환경과 방법을 살펴보고, 그 결과를 분석 설명한다. 제안한 태스크 스케줄링 기법을 사용하였을 때 각 태스크의 데드라인 위반이 효과적으로 줄어들 수 있는지 모의 실험을 통하여 측정하였다. 앞에서 언급한 SMART와 계층적 스케줄러의 경우 응용 프로그램 단위로 태스크의 스케줄링을 지원하기 때문에 여기서는 따로 비교하지 않는다. 먼저 경성 실시간, 연성 실시간, 일반 태스크로 구성된 스트림 별 태스크 집합을 구성한다. 집합 내 각 태스크의 주기와 수행 시간은 다음과 같이 구성한다. 연성 실시간 태스크의 경우는 초당 30 프레임으로 구성된 비디오 파일에서 디코딩 작업이나 데이터의 전송 주기를 고려하여 33ms로 정한다. 수행 시간은 주기 내의 범위에서 난수 발생기에 의해 랜덤하게 주어진다. 경성 실시간 태스크는 30 프레임의 GOP 단위로 주기와 수행 시간이 랜덤하게 주어진다. 실험은 각 태스크 집합의 수를 증가시키며 경성 실시간 태스크의 데드라인 위반 정도와 연성 실시간 태스크의 스케줄 무시 정도를 측정하여 보고 각 스트림 내의 연성 실시간 태스크가 경쟁에 의해 골고루 무시되는지 살펴 보았다. 실험에 사용한 시뮬레이터는 C++로 구현하였으며 33000ms동안 측정하였다.

먼저 데이터 집합의 증가에 따른 경성 실시간 태스크의 데드라인 위반을 측정하여 보았다. 그림 5에서 볼 수 있듯이 태스크 집합의 증가에도 데드라인 위반이 전혀 일어나지 않음을 알 수 있다. 경성 실시간 태스크의 경우 데드라인을 위반하면 전체적인 서비스에 큰 문제를 일으키므로 경성 실시간 태스크에 가장 높은 우선 순위를 주어 스케줄 할뿐만 아니라 연성 실시간 태스크가 스케줄 되기 직전에도 다음에 스케줄 될 경성 실시간 태스크의 데드라인 위반을 초래하는지 측정하여 스케줄하기 하기 때문이다. 그림 6은 태스크 집합의 수가 늘어날 때 각 집합 내 연성 실시간 태스크의 스케줄 무시 횟수를 나타내었다. 태스크의 집합이 증가할수록 태스크들 사이의 경쟁이 치열해져 연성 실시간 태스크의 스케줄 무시 횟수가 늘어남을 볼 수 있다. 그러나 무시된 횟수를 계속적으로 유지 비교하여 다음 무시될 태스크를 선정하기 때문에 전체적으로 무시된 횟수는 태스크 별로 거의 동일하다.

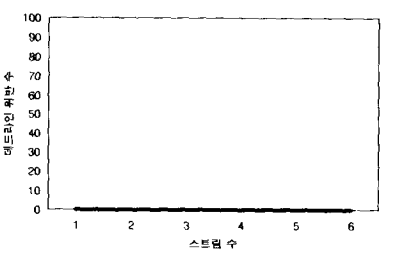

그림 5. 경성 실시간 태스크의 데드라인 위반 수

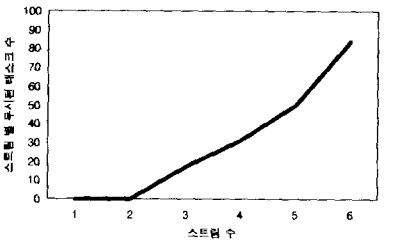

그림 6. 스트림 별 태스크 집합 내 연성 실시간 태스크의 스케줄 무시 횟수

5. 결론

버퍼 관리 기법과 태스크 스케줄링 기법은 네트워크 환경에 있는 종단 시스템에서 안정적으로 멀티미디어 서비스를 지원하기 위한 방편으로 사용되었다. 본 논문에서는 온라인 대역폭 평활화를 효과적으로 지원하는 태스크 스케줄링 기법을 제안하였다. 이 방식은 온라인 대역폭 평활화가 가지는 태스크를 경성 실시간 태스크, 연성 실시간 태스크, 일반 태스크로 분류하여 지원할 뿐만 아니라 각 태스크의 선행 관계를 지원한다. 경성 실시간 시스템의 데드라인 위반으로 인한 전체적인 시스템의 성능 저하를 막고 연성 실시간 태스크의 스케줄 무시 횟수를 유지하여 경쟁에 의한 어느 한 태스크의 기아 현상을 방지한다. 제안한 태스크 스케줄링 기법은 연성 실시간 태스크의 유연한 처리로 실제 시스템에서 예기치 않은 인터럽트를 효과적으로 처리할 뿐만 아니라 요청된 각 서비스의 품질을 균등하게 유지할 수 있을 것으로 예상된다. 현재 적응성 있는 대역폭 평활화 기법과 제안된 태스크 스케줄링 기법을 응용하여 시스템을 구현하고 있으며 전체적인 성능 평가를 계획하고 있다.

참고 문헌

[1] J. Rexford, S. Sen, J. Dey, W. Feng, J. Kurose, J. Stankovic and D. Towsley, "Online Smoothing of Live, Variable-Bit-Rate Video", University of Massachusetts Computer Science Technical Report, July 1998.

[2] J. Salehi, Z. Zhang, J. Kurose and D. Towsley, "Supporting Stored Video: Reducing Rate Variability and End-to-End Resource Requirements through Optimal Smoothing", Proceedings of ACM SIGMETRICS, May 1996.

[3] L. A. Rowe, K. Patel, B. C. Smith, and K. Liu, "MPEG Video in Software Representation, Transmission and Playback," In Proceedings of IS&T/SPIE International Symposium. on Electronic Imaging: Science and Technology, Feb. 1994.

[4] Wu-chi Feng and Stuart Sechrest, "Smoothing and buffering for delivery of prerecorded compressed video", In proceedings of IS&T/SPIE Multimedia Computing and Networking, Feb. 1995, San Jose, CA.

[5] 김 재 욱, 하 란, "멀티미디어 응용을 위한 동적 버퍼 관리 기법", 정보과학회 추계학술발표 논문집, 1999년

[6] J. Nieh, M. Lam, "The Design, Implementation and Evaluation of SMART: A Scheduler for Multimedia Application". Proceedings of the 16th ACM Symposium on Operating Systems Principles, Oct. 1997.

[7] P. Goyal, X. Guo and H. M. Vin, "A Hierachical CPU Scheduler for Multimedia Operating Systems", Proceedings of the 2nd USENIX Symposium on Operating System Design and Implementation, Oct. 1996.

[8] P. Goyal, H. M. Vin and H. Cheng, "Start-time Fair Queueing: A Scheduling Algorithm for Integrated Service Packet Switching Networks", Proceedings of ACM SIGCOMM '96, Aug. 1996.