Journal of Korea Multimedia Society Vol. 15, No. 6, June 2012(pp. 814-826)

VBR 비디오 스트림 전송을 위한 모바일 클라이언트 버퍼 수준 기반 스케쥴링 알고리즘

김 진 환†

요 약

모바일 통신망에서 클라이언트들의 재생 버퍼 수준을 이용하여 VBR 비디오 스트림을 전송하는 스케쥴링 알고리즘들이 본 논문에서 제시된다. 이 알고리즘들은 모바일 통신망에서 집중식 비디오 서버와 클라이언트들 간에 제한된 통신 대역폭을 최대한 활용할 수 있다. 비디오 서버는 동시에 다수의 비디오 요청을 서비스하기 때문에 통신망 대역폭을 공평하고 효율적으로 할당하여 활용하는 것이 중요하다. 각 비디오 재생시 서비스 품질과 실시간적 성능을 향상시키기 위하여 비디오 서버는 일시적으로 버퍼 수준이 낮은 비디오 요청 작업을 우선적으로 서비스하고자 더 많은 대역폭을 할당하게 된다. 버퍼 수준이 상이한 모바일 클라이언트들에게 공평한 서비스와 부하 균형이 제공되는 시뮬레이션 결과가 나타났으며 각 클라이언트의 재생 시간전까지 성공적으로 전송되는 프레임의 수가 최대화되었다.

Mobile Client Buffer Level-based Scheduling Algorithms for Variable-Bit-Rate Video Stream Transmission

Jinhwan Kim†

ABSTRACT

In this paper, we propose scheduling algorithms for transporting variable-bit-rate video stream using playback buffer level of the clients over wireless communication networks. The proposed algorithms attempt to maximize the utilization of limited bandwidth between the central video server and the clients over a mobile network. Since a video server may serve several video request at the same time, it is important to allocate and utilize network bandwidth to serve them fairly and efficiently. In order to improve the quality of service and real-time performance of individual video playback, the video server attempts to allocate temporarily more network bandwidth to serve a video request with the lower buffer level preferentially. The simulation results prove the fair service and load balancing among the mobile concurrent clients with different buffer levels and hence maximizing the number of frames that are transported successfully to the client prior to their playback times.

Key words: VBR video(VBR 비디오), stream transmission(스트림 전송), scheduling(스케쥴링), buffer level(버퍼 수준), real-time(실시간)

※ 교신저자(Corresponding Author) : 김진환, 주소 : 서울 성북구 삼선동2가 389(136-792), 전화 : 02)760-4340, FAX : 02)760-4488, E-mail : kimjh@hansung.ac.kr
접수일 : 2011년 6월 27일, 수정일 : 2011년 12월 20일
완료일 : 2012년 4월 5일
† 정회원, 한성대학교 멀티미디어공학과
※ 본 연구는 한성대학교 교내연구비 지원 과제임

1. 서 론

스마트폰이나 스마트패드를 이용하는 클라이언트들은 현재 3G 모바일 통신망과 WiFi 무선 근거리 통신망 또는 WiBro 무선 인터넷 통신망 등으로 비디오 서버에 연결된다. 차세대 모바일 통신망으로 부각되고 있는 LTE(Long Term Evolution) 시스템을 이용할 경우 클라이언트는 이동 중 100Mbps 그리고 정지 상태에서 1Gbps 이상의 전송 속도로 서버와 통신할 수 있다[1]. 이와 같이 모바일 환경에서 사용되는 통신망의 대역폭이 급격히 개선되고 있지만 HD나 3D 화면 등 고품질 비디오 화면의 해상도를 비롯하여 다양한 부하 특성을 갖는 비디오 재생 요청시 서비스 품질 및 실시간적 성능 향상을 위하여 클라이언트와 서버 간 통신망 대역폭을 효율적으로 할당하는 비디오 스트림의 전송 스케줄링 방법이 중요하다[2,3].

유선 통신망 기반의 클라이언트-서버 시스템의 성능 개선을 위하여 제시된 스케줄링 기법들을[4] 모바일 환경에 동일하게 적용하기에는 통신망 특성과 대역폭의 크기 등 여러 가지 제약 사항들이 있다. 특히 휴대폰 등 모바일 단말기의 버퍼는 아직 노트북이나 데스크탑 컴퓨터에 비하면 버퍼 용량이 크지 않다. 이러한 모바일 단말기에 재생될 비디오 스트림을 위하여 모바일 통신망의 대역폭을 정적으로 할당하는 기존의 스케줄링 기법들은[2,3] 비디오 트래픽 크기의 변동성과 일시적인 과부하 현상을 효과적으로 해결하기 어려운 면이 있으며 또한 반드시 여분으로 사용할 수 있는 대역폭을 확보해야 한다는 문제점이 있다.

본 논문에서는 모바일 통신망 기반의 클라이언트-서버 시스템[5-7]에서 비디오 서버가 비디오, 음성, 데이터 등 멀티미디어 트래픽에 대한 서비스 품질을 효율적으로 제공할 수 있도록 집중식 스케줄러의 역할을 수행하게 된다. cell 단위의 모바일 통신망에서 집중식 스케줄러는 트래픽의 양과 채널 조건에 따라 통신망의 대역폭을 동적으로 할당할 수 있으며 스케줄링 서비스를 공평하게 수행할 수 있는 장점이 있다[8].

동영상 압축 기법의 VBR(Variable Bit Rate) 특성상[9] 각 프레임은 시간과 종류에 따라 상이한 크기를 갖게 되므로 비디오 트래픽의 변동성과 일시적인 과부하 현상이 유발될 수 있다[10]. 본 논문에서 서버는 VBR 특성으로 인한 비디오 트래픽의 변동성과 일시적 과부하를 고려하여 각 클라이언트의 버퍼 수준에 따라 통신망 대역폭을 동적으로 할당하게 된다. 즉 서버는 클라이언트들이 요청한 비디오 스트림 전송을 위한 통신망 대역폭이 부족할 경우 버퍼 수준이 낮은 클라이언트들에게 더 많은 대역폭을 일시적으로 할당하여 해당 프레임들이 정해진 재생시간 전까지 전송될 수 있도록 한다. 이 경우 상대적으로 버퍼 수준이 높은 클라이언트는 서버로부터 새로운 프레임들을 수신하는 대신 단말기 버퍼에 있는 현재 프레임들을 재생함으로써 비디오 재생의 연속성을 유지할 수 있게 된다. 제시된 전송 스케줄링 기법으로 재생 시간 전에 전송되는 프레임 수를 최대화함으로써 비디오 재생에 관한 실시간적 성능과 서비스 품질을 향상시키고자 한다.

본 논문은 2장에서 모바일 환경에서 비디오 스트림을 요청하고 전송하기 위한 클라이언트-서버 시스템 특성이 기술되며 3장에서 클라이언트의 단말기 버퍼 수준에 따라 대역폭을 동적으로 할당하며 비디오 스트림 전송을 스케줄링하는 기법이 기술된다. 그리고 4장에서 시뮬레이션을 통하여 제시된 기법들과 다른 스케줄링 기법과의 성능이 비교 분석되며 5장에서는 결론을 기술한다.

2. 모바일 클라이언트-서버 시스템

모바일 환경에서 모바일 클라이언트와 비디오 서버는 cell 단위를 기반으로 하는 모바일 통신망이나 무선 근거리 통신망인 WiFi 등으로 연결된다(그림 1 참조). 모바일 통신망과 무선 근거리통신망의 대역폭이 점차 개선되고 있긴 하나 아직도 유선 통신망에 비하면 대역폭 크기가 작고 전송시 오류 발생 확률이 크며 통신망 내에서의 단절 현상도 빈번하게 발생한다. 모바일 클라이언트들은 이러한 무선 통신망을 이용하여 비디오 서버에 비디오 재생을 요청하게 된다. 휴대폰, 스마트폰 등을 사용하는 thin 클라이언트들은 노트북 컴퓨터나 스마트 패드를 사용하는 thick 클라이언트에 비하여 비디오 버퍼 크기가 작고 제한이 있게 된다[11]. 클라이언트들마다 작업 부하의 크기가 상이한 비디오를 요청하게 되며 예를 들어 고성능 노트북을 이용하는 thick 클라이언트의 비디오 작업 부하량은 휴대폰을 사용하는 thin 클라이언트에

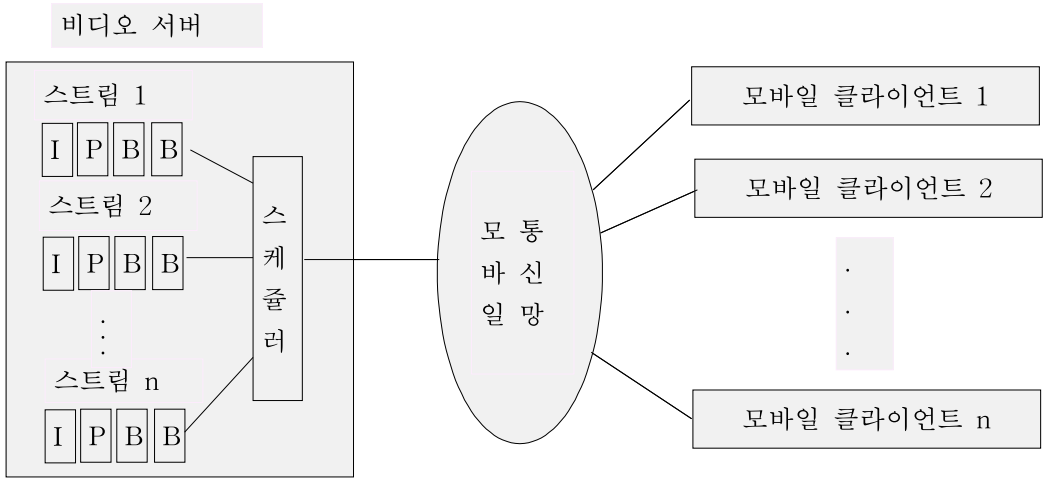

그림 1. 모바일 클라이언트와 비디오 서버 구성

비하여 훨씬 커지게 된다.

모바일 클라이언트는 서버로부터 수신한 프레임들을 자신의 버퍼에 저장하며 버퍼 수준이 사전에 정의된 수치에 도달하면 비디오의 재생을 시작하게 된다. 즉 복원 과정이 시작되는 것이다. 그러나 비디오 프레임의 재생 시간을 놓치게 되면 해당 프레임은 즉시 재생이 취소되며 폐기된다. 이러한 현상은 끊김 없는(seamless) 비디오 스트리밍(streaming) 유지에 심각한 영향을 미치게 되며 재생율(1초간 재생되는 프레임들의 수)에 관한 서비스 품질과 실시간적 성능도 저하시키게 된다[12]. 모바일 환경에서 화면이 단절되거나 손상되는 jitter 현상을 최소화하기 위해 클라이언트에 버퍼 기법이 필요하나 thin 클라이언트 경우 thick 클라이언트에 비하면 버퍼 용량이 충분하지 않다. 결국 버퍼 크기가 작은 클라이언트일수록 버퍼가 큰 클라이언트에 비하여 통신망 오류에 민감하게 되며 비디오 재생시 과부하에 따른 성능 저하가 발생하게 된다[12].

비디오 서버가 다수의 클라이언트로부터 비디오 요청을 수신할 때 수락 제어 과정을 먼저 실시한 이후 다수의 비디오 요청 작업을 동시에 서비스할 수 있다. 서버가 비디오 요청의 수락을 허용한 경우에는 해당 비디오 파일을 저장 장치에서 검색하며 비디오 프레임들을 요청별로 지정된 비디오 버퍼에 저장하게 된다. 본 논문에서 비디오 서버는 VBR 특성을 갖는 압축 표준인 MPEG-4 동영상 비디오를 가지고 있는 것으로 가정하며 각 비디오 스트림은 GOP (Group of Picture) 시퀀스로 구성되어 있다. 각 GOP는 I, P, B 프레임들로 구성되며 통상 I 프레임의 크기는 B 프레임이나 P 프레임보다 훨씬 크다. GOP마다 크기가 상이하며 비디오의 고정된 재생율에 따라 각 비디오 스트림의 대역폭 요건은 가변적 특성을 가지게 된다. 이 비디오 프레임들은 패킷으로 구성되며 스케줄러와 모바일 통신망을 통하여 해당 모바일 클라이언트에게 전송된다.

3. 비디오 스트림 전송 스케줄링 기법

3.1 버퍼 수준과 비트율

각 클라이언트는 스마트폰 또는 일반 휴대폰 등(이하 단말기로 기술함)에 설치된 버퍼에 재생율을 고려한 일정 시간 동안 프레임들이 서버로부터 도달하면 이때부터 버퍼에 있는 프레임을 디코딩하여 재생하게 된다. 따라서 버퍼 수준이란 클라이언트의 단말기가 서버로부터 수신한 프레임들을 저장할 수 있는 버퍼의 크기를 시간 단위로 표현하게 되며 클라이언트의 버퍼 수준이 높을수록 실제 버퍼의 용량이 큰 것을 의미한다. 모바일 클라이언트는 단말기의 특성상 기억장치의 크기가 상이하며 실제 버퍼의 수준도 상이하게 된다. 버퍼 수준이 높을수록 오랜 시간 동안 재생할 수 있는 많은 수의 프레임들을 버퍼에 유지하게 되며 통신망의 일시적 장애나 오류 발생 시에도 덜 민감하게 반응할 수 있는 장점이 있다[6]. 그러나 버퍼 수준이 높을 경우 비디오 재생이 시작되는 시간이 늦어지고 기억장치 활용도가 저하되는 문제가 있으며 모바일 단말기에 대용량의 버퍼를 유지하기 어려운 면이 있다.

본 논문에서 비디오 서버의 스케쥴러는 각 클라이언트의 버퍼 수준을 고려하여 매주기마다 통신망 대역폭을 동적으로 할당하게 된다(그림 1 참조). n(>0) 개의 클라이언트들 중 임의의 i($1 \le i \le n$) 클라이언트가 요청한 비디오 스트림을 수신하기 위한 단말기의 버퍼 수준을 L_i라 하며 시간 단위인 초를 이용하여 크기를 설정한다. 예를 들어 L_i가 5인 경우 해당 단말기의 버퍼에 5초간 재생할 수 있는 프레임들이 저장되어 있음을 의미한다. 실제로 클라이언트의 버퍼 수준이 1이라 하더라도 비디오 스트림에 따라 1초간 저장되는 프레임들의 평균 크기가 다를 수 있으므로 클라이언트 버퍼마다 버퍼의 실제 크기는 상이할 수 있다.

비디오 서버는 일정 주기(본 논문에서는 1초로 가정함)마다 클라이언트들이 요구한 비디오 스트림의 프레임들을 저장장치에서 검색 후 서버의 버퍼에 저장하게 되며 이 경우 통신망 대역폭의 한계로 인해 서버가 전송을 중단시켜야 하는 프레임들이 발생하게 된다. 전송이 중단되는 프레임들의 수가 증가할수록 해당 클라이언트는 끊김없는 비디오 스트리밍 유지가 어렵게 되고 재생율 및 서비스 품질(QoS)이 저하되는 현상이 발생한다. 본 논문에서는 이러한 현상을 최소화시키고자 서버가 각 클라이언트의 버퍼 수준을 고려하여 매주기마다 대역폭을 동적으로 할당한다. 각 비디오 스트림의 재생율은 동일하지만 실제 대역폭은 VBR 특성에 따라 매초마다 크기가 가변적으로 구성된다. 매초마다 i 클라이언트가 요청한 비디오 스트림의 프레임들을 비트로 표시한 실제 비트율 B_i와 이들의 합인 B_{sum}은 수식 (1)의 관계로 성립된다.

$$B_{sum} = \sum_{i=1}^{n} B_i \quad (1)$$

본 논문에서 서버와 클라이언트들 간에 사용가능한 통신망의 최대 대역폭을 B_{total}이라 하며 실제 사용가능한 통신망 대역폭을 B_{avail}로 정의한다. 즉 B_{avail}은 B_{total}에서 오류 발생시 재전송을 위한 대역폭과 비디오 스트림 전송 이외의 목적으로 사용되는 대역폭을 제외한 대역폭을 의미한다. B_{sum}, B_{avail}, B_{total}들에 대해서는 $B_{sum} \le B_{avail} \le B_{total}$ 관계가 적용된다. 모바일 통신망의 비디오 서버는 클라이언트가 요청한 비디오 스트림을 전송하기 전 수락 제어(admission control) 절차를 수행한다. n 개의 클라이언트들이 요청한 비디오 스트림의 평균 비트율(비트로 표현되는 프레임의 평균 크기와 재생율을 곱한 값)의 합이 B_{avail}보다 작거나 같아야만 한다. 예를 들어 n+1번째 클라이언트가 새로운 비디오 스트림을 요청할 경우 서버는 기존 n 클라이언트들의 평균 비트율의 합과 새로 요청된 스트림의 평균 비트율의 합이 B_{avail}보다 작거나 같은 관계를 계속 만족하면 서버가 이를 수용하지만 그렇지 않은 경우에는 이를 거부할 수 있다. 그리고 거부된 n+1번째 클라이언트의 비디오 스트림은 나중에 서버가 재수용할 수 있으나 이에 관한 구체적인 과정은 본 논문에서 기술하지 않는다.

비디오 서버의 스케쥴러는 승인 제어 과정 이후 클라이언트들이 요청한 스트림들의 대역폭 합 B_{sum}과 이용가능한 대역폭 B_{avail}을 비교하여 각 스트림의 대역폭을 조정하여 전송하는 전체적인 알고리즘은 그림 2와 같이 기술된다. 그림 2의 2행과 같이 이 보다 작거나 같을 경우에는 버퍼 수준에 상관없이 모든 스트림들이 전송되나 이 보다 클 경우(3행) 본 논문에서 제시하는 버퍼수준 정책에 따라 각 스트림의 대역폭이 조정된 후 전송된다. 버퍼수준 정책은 3.2절과 3.3절에서 구체적인 내용이 기술된다.

```
1: if (Bsum ≤ Bavail)
2:    버퍼 수준에 상관없이 모든 스트림을 전송;
3: else
4:    begin
5:       버퍼수준 정책 적용;
6:        case 1: 최소값 버퍼수준 우선 전송;
7:        case 2: 최대값 버퍼수준 우선 보류;
8:       조정된 대역폭에 따라 각 스트림 전송;
9:    end
```

그림 2. 비디오 서버 스케줄러의 대역폭 조정 알고리즘

3.2 최소값 버퍼수준 우선 전송

서버는 전송하고자 하는 실제 비트율의 합 B_{sum}이 B_{avail}보다 클 경우 버퍼 수준이 가장 낮은 클라이언트의 비디오 스트림부터 먼저 전송하도록 통신망 대역폭을 할당하는 최소값 버퍼수준 우선 기법이 적용되며 이 알고리즘은 그림 3에 의사 언어로 기술되어 있다. 따라서 버퍼 수준이 상대적으로 높은 클라이언트는 새로운 프레임들을 수신하지 못한 상태에서 버퍼에 저장하고 있는 현재 프레임들을 재생하게 되며

버퍼 수준이 감소되는 결과가 발생하게 된다. 그러나 버퍼 수준이 낮았던 클라이언트는 서버로부터 새로운 프레임들을 수신함으로써 수신한 만큼 버퍼 수준이 증가하게 된다.

그림 3의 1행에서 B_t는 버퍼수준이 최소값인 클라이언트부터 전송되는 비트율의 합을 의미하며 매초마다 서버는 B_t의 초기치를 0으로 설정한다. 서버는 n개의 클라이언트들 중 버퍼수준 L_i가 최소인 클라이언트 i($1 \le i \le n$)를 파악하고 이 i를 min으로 설정한다(4행). 최소값인 버퍼수준을 L_{min}으로 설정하며 해당 클라이언트의 비트율을 B_{min}으로 설정한다(5행). L_{min}이 0보다 클 경우(6행) 서버는 B_{min}과 B_t의 합을 B_{avail}과 비교한다. B_t와 B_{min}의 합이 B_{avail}보다 클 경우(8행) 결과적으로 비트율 B_{min}중 B_{avail}에서 B_t를 차감한 만큼만 전송될 수 있으므로 최소값 L_{min}을 가진 클라이언트의 전송가능한 비트율 B'_{min}은 $B_{avail}-B_t$가 된다(9행). B_{min}에서 B'_{min}을 차감한 $B_{min}-B'_{min}$ 비트율은 다음 주기에 전송될 수 있다.

B_t와 B_{min}의 합이 B_{avail} 이하일 경우에는(10행) B_{min}이 모두 전송될 수 있으므로 B'_{min}은 B_{min}과 같다(11행). B'_{min}이 결정될 때마다 B_t값이 증가된다(12행). 그리고 해당 클라이언트의 버퍼수준 L_{min}은 B'_{min}/B_{min} 비율만큼 증가된다(13행). 즉 B'_{min}이 B_{min}과 동일할 경우 버퍼수준 L_{min}은 1.0이 증가되며 B'_{min}이 B_{min}의 절반일 경우 버퍼수준은 0.5만큼 증가된다.

5행에서 설정된 L_{min}이 0일 경우(15행) 버퍼수준의 최소값이 0인 모든 클라이언트들에 대하여 공평하게 동일한 비율로 스트림을 전송하게 된다. 버퍼수준이 0인 클라이언트들의 수를 n′라(17행) 하고 해당 비트율의 합을 B'_{sum}으로 정의한다(18행). B'_{sum}이 B_{avail}보다 큰 경우(21행) 버퍼수준이 0인 n′개의 클라이언트들 중 임의의 i($1 \le i \le n'$) 클라이언트로 전송될 비트율 $B_{min,i}$는(21행) B_{avail}/B'_{sum}에 비례한 만큼만 전송될 수 있으며 이는 $B'_{min,i}$로 표기된다(23행). 즉 버퍼수준의 최소값이 0인 클라이언트들은 자신의 비트율과 B_{avail}/B'_{sum}을 곱한 결과만 전송될 수 있는 것이다. $B_{min,i}$에서 $B'_{min,i}$를 차감한 비트율은 일단 전송이 보류되며 다음 주기인 1초 후에 전송될 수 있다. B'_{sum}이 B_{avail} 이하일 경우에는(24행) $B_{min,i}$가 모두 전송될 수 있으므로 $B'_{min,i}$는 $B_{min,i}$가 된다(25행). 23행이나 25행에서 결정된 $B'_{min,i}$만큼 B_t는 증가되며(26행) 현재 버퍼수준이 0인 클라이언트의 버퍼수준 $L_{min,i}$는 $B'_{min,i}/B_{min,i}$ 비율만큼 증가된다(27행). 버퍼수준의 최소값이 0보다 크고 전송될 비트율의 합 B_t가 B_{avail}보다 작을 경우 4행부터 13행까지의 과정들이 반복된다(2행). 버퍼수준의 최소값이 0이 되면 17행부터 28행까지의 과정이 수행된 후 2행의 조건이 거짓이 되므로 그림 3의 과정들은 모두 종료된다.

이후 서버는 변경된 각 클라이언트의 버퍼수준을 다시 파악한다. 임의의 클라이언트 i의 버퍼수준 L_i가 0.5라면 해당 클라이언트는 0.5초 동안만 자신의 버퍼에 있는 비디오 스트림을 재생할 수 있으며 나머지 0.5초는 새로운 프레임들을 재생할 수 없게 된다. 따

```
1: Bt=0;
 2: while (Bt < Bavail)
 3:   begin
 4:    버퍼수준이 가장 낮은 클라이언트 i를 min으로 설정;
 5:    이 버퍼수준을 Lmin으로 비트율은 Bmin으로 설정;
 6:    if (Lmin > 0)
 7:      begin
 8:        if (Bmin+Bt >Bavail)
 9:          B′min=Bavail-Bt;
10:        else
11:          B′min=Bmin;
12:       Bt=Bt+B′min;
13:       Lmin=Lmin+B′min/Bmin;
14:      end
15:    else // Lmin==0
16:      begin
17:      버퍼수준이 0인 클라이언트의 수 n′로 설정;
18:      n′개 클라이언트의 비트율의 합 B′sum으로 설정;
19:        for each i from 1 to n′
20:          begin
21:          클라이언트 i의 비트율 Bmin,i로 설정;
22:           if (B′sum > Bavail)
23:             B′min,i=Bmin,i*Bavail/B′sum;
24:           else
25:             B′min,i=Bmin,i;
26:           Bt=Bt+B′min,i;
27:           Lmin,i= Li+B′min,i/Bmin,i;
28:          end
29:      end
30: end
```

그림 3. 최소값 버퍼수준 우선 전송 알고리즘($B_{sum} > B_{avail}$)

라서 서버는 정해진 시간 내에 재생할 수 없는 나머지 0.5초 분량에 해당하는 프레임들의 전송을 중단하게 된다. 이와 같이 서버에 의해 전송 중단되는 스트림 i의 비트율을 수식 (2)에서 D_i로 표기한다.

$$D_i=(1.0-L_i)*B_i \quad (2)$$

전송이 중단되는 프레임의 수가 많아질수록 서비스 품질과 실시간적 성능이 저하되는 것을 의미한다. 각 클라이언트에서 프레임들이 재생된 시간만큼 버퍼수준은 감소되며 본 논문에서는 초 단위로 감소되는 것을 가정한다. 그러나 버퍼수준 L_i가 1.0미만인 경우에는 버퍼에 있는 모든 프레임들이 재생되므로 L_i는 0으로 재설정된다. 전송이 보류된 대역폭에 해당하는 프레임들은 다음 주기에 전송될 수 있으나 전송이 중단된 대역폭에 속한 프레임들은 서버에 의해 폐기되는 것을 의미한다. 본 논문에서는 전송이 중단되는 프레임들에 대하여 우선순위 정책이 적용된다. 즉 중요도가 가장 낮은 B 프레임들이[13] 먼저 중단되며 이후 P 프레임, I 프레임 순으로 결정됨으로써 중요도가 큰 프레임일수록 전송이 중단되는 경우가 최소화되도록 하였다.

한편 비트율의 합 B_{sum}이 B_{avail}보다 작거나 같을 경우 서버는 모든 클라이언트의 비트율을 전송할 수 있기 때문에 버퍼수준 L_i는 모두 1.0만큼 증가된다. 이후 B_{avail}에서 B_{sum}을 차감한 여유분의 비트율을 활용하기 위해 서버는 버퍼수준이 가장 낮은 클라이언트부터 추가로 스트림을 전송하며 이에 대한 알고리즘은 그림 4에서 기술된다. 즉 서버는 다음 주기에 전송할 클라이언트의 프레임들을 이번 주기에 미리 전송할 수 있도록 하는 것이다. 모두 전송된 B_{sum}은 그림 4의 31행에서 B_t의 초기치로 설정된다. 이미 1.0씩 증가된 버퍼수준 L_i 중 최소값인 $L_{min,i}$를(34행과 35행) 갖는 스트림의 1초 후의 비트율 $B_{min,i}$를 B_{sum}과 동일한 B_t에 더한 값이 B_{avail}을 초과하면(36행) $B_{min,i}$ 비트율 중 $B_{avail}-B_t$만큼만 전송되며 이는 $B'_{min,i}$로 표기된다(38행). 해당 단말기의 버퍼 수준 $L_{min,i}$는 $B'_{min,i}/B_{min,i}$ 비율만큼 추가로 증가된다(39행). $B_{min,i}$에서 $B'_{min,i}$를 차감한 비트율은 서버가 다음 주기에 전송하게 된다. B_t와 $B_{min,i}$의 합이 B_{avail}보다 작은 경우에는(41행) $B_{min,i}$가 모두 전송되므로 $B'_{min,i}$는 $B_{min,i}$가 되고(43행) $L_{min,i}$는 다시 1.0만큼 증가된다(44행). 현재의 B_t에 $B'_{min,i}$가 더해지며(46행) 증가된 B_t가 B_{avail} 이하일 경우에는 최소 버퍼수준을 가진 클라이언트의 비트율을 추가로 전송하는 과정이 반복된다(32행).

```
31: Bt=Bsum;
32: while (Bt ≤ Bavail)
33:   begin
34:   버퍼수준이 가장 낮은 클라이언트 i를 min으로 설정;
35:   이 버퍼수준을 Lmin,i로 비트율은 Bmin,i로 설정;
36:     if (Bt+Bmin,i > Bavail)
37:       begin
38:         B'min,i=Bavail−Bt;
39:         Lmin,i=Lmin,i+B'min,i/Bmin,i;
40:       end
41:     else
42:       begin
43:         B'min,i=Bmin,i;
44:         Lmin,i=Lmin,i+1.0;
45:       end
46:     Bt=Bt+B'min,i;
47:   end
```

그림 4. 최소값 버퍼수준 우선 전송 알고리즘($B_{sum} \le B_{avail}$)

모든 클라이언트는 자신의 버퍼에 있는 프레임들을 1초 동안 재생하게 되면 버퍼수준이 1.0씩 감소되며 서버는 모든 클라이언트의 버퍼 수준을 매초마다 파악할 수 있음을 본 논문에서 가정하고 있다.

3.3 최대값 버퍼수준 우선 보류

서버는 전송하고자 하는 실제 대역폭의 합 B_{sum}이 사용가능한 통신망 대역폭의 합 B_{avail}보다 클 경우 단말기 버퍼 수준을 고려하여 버퍼 수준이 높은 클라이언트일수록 비디오 스트림을 다음 주기에 전송하도록 대역폭을 조정하며 이 알고리즘은 그림 5에 기술되어 있다. 버퍼 수준이 가장 높은 클라이언트부터 프레임들의 전송이 보류되며 보류된 만큼의 비트율은 버퍼 수준이 상대적으로 낮은 다른 클라이언트들을 위해 할당된다. 그리고 이번 주기에 전송이 보류된 해당 클라이언트의 비트율은 다음 주기에 전송될 수 있다. 따라서 버퍼 수준이 높았던 클라이언트는 새로운 프레임들을 수신하지 못한 상태에서 버퍼에 있는 현재 프레임들을 재생하게 되며 버퍼 수준이 감소되는 결과가 발생한다. 버퍼 수준이 낮았던 클라이언트는 서버로부터 새로운 프레임을 수신할 경우 수신한 만큼 버퍼 수준이 증가하게 된다.

```
1: Bp=Bsum;
2: while (Bp > Bavail)
3:   begin
4:    버퍼수준이 가장 높은 클라이언트 i를 max로 설정;
5:    이 버퍼수준을 Lmax로 비트율은 Bmax로 설정;
6:     if (Bmax < Bp-Bavail)
7:       begin
8:         Pmax=Bmax;
9:         Lmax=Lmax-1.0;
10:        Bp=Bp-Pmax;
11:      end
12:     else
13:      begin
14:        Pmax=Bmax-(Bp-Bavail);
15:        Lmax=Lmax-Pmax/Bmax;
16:        Bp=Bp-(Bp-Bavail);
17:      end
18:   end
19:  B'sum=Σ_{i=1}^{n}(Bi-Pi)
20: if (B'sum> Bavail)
21:  begin
22:    for each i from 1 to n
23:      Di=(B'sum-Bavail)*(Bi-Pi)/B'sum;
24:  end
```

그림 5. 최대값 버퍼수준 우선 보류 알고리즘($B_{sum} > B_{avail}$)

그림 5의 1행에서 전송이 보류되는 비트율의 합은 B_p로 표기되며 초기치는 B_{sum}으로 설정한다. B_p와 B_{avail}의 차는 결국 B_p를 전송할 때 부족한 대역폭을 의미하며 B_p가 B_{avail}보다 클 경우(2행) 서버는 n개의 클라이언트들 중 버퍼수준 L_i가 최대인 클라이언트 $i(1 \le i \le n)$를 파악하고 이 i를 max로 설정한다(4행). 최대값인 버퍼수준을 L_{max}로 설정하며 해당 클라이언트의 비트율을 B_{max}로 설정한다(5행). B_{max}가 B_p-B_{avail}의 차보다 작을 경우(6행) B_{max}는 모두 전송 보류된다(8행). 이때 전송 보류된 클라이언트의 비트율은 P_{max}로 정의된다. 해당 클라이언트는 1초간 서버로부터 새로운 프레임의 전송이 보류되며 대신 현재 단말기 버퍼에 있는 프레임들을 1초간 재생함으로써 버퍼 수준 L_{max}가 1.0만큼 감소되는 것이다(9행). 이와 같이 서버는 버퍼 수준이 낮은 다른 클라이언트들의 프레임들이 더 많이 전송될 수 있도록 버퍼 수준이 높은 클라이언트부터 프레임들의 전송을 보류하게 된다. B_p에서 전송보류된 P_{max}를 차감한 새로운 B_p가 설정된다(10행).

한편 B_{max}가 B_p-B_{avail}보다 크거나 같을 경우(12행) B_{max}중 B_p-B_{avail} 비트율은 전송가능한 반면 나머지인 B_{max}-(B_p-B_{avail})는 전송이 보류되며 이는 P_{max}로 표기된다(14행). 이 경우 서버가 해당 클라이언트에게 전송하는 프레임들의 실제 재생 시간은 (B_{max}-P_{max})/B_{max} 초 즉 1.0-P_{max}/B_{max} 초가 되므로 이 클라이언트는 P_{max}/B_{max} 초 동안은 현재 단말기 버퍼에 있는 프레임들을 재생하게 된다. 따라서 버퍼 수준 L_{max}도 동일한 시간인 P_{max}/B_{max} 초만큼 차감된다(15행). 현재의 B_p에서 B_p-B_{avail}을 차감하여 새로 설정되는 B_p(16행)는 결국 B_{avail}과 동일하게 되므로 2행의 조건은 거짓이 되고 3행부터 18행까지의 반복 과정을 수행하지 않게 된다.

임의의 스트림 i의 실제 비트율 B_i에서 전송 보류된 비트율 P_i(8행과 14행에서 P_{max}로 표기됨)를 차감한 비트율은 이번 주기에 전송가능한 비트율을 의미하며 B_i-P_i의 합을 B'_{sum}이라 한다(19행). 그러나 B'_{sum}이 B_{avail}보다 여전히 클 경우(20행) B'_{sum}-B_{avail}은 전송 불가능한 비트율이 된다. 각 클라이언트의 전송가능한 비트율 B_i-P_i를 B'_{sum}으로 나눈 비율과 B'_{sum}-B_{avail}을 곱하여 전송을 중단해야 하는 비트율을 다시 공평하게 산정하며 이 비트율은 D_i로 정의된다(23행). 따라서 실제 전송되는 각 스트림의 비트율은 B_i에서 전송 보류된 비트율 P_i를 차감한 후 전송 중단된 비트율 D_i를 차감한 B_i-P_i-D_i가 된다. 전송이 보류된 비트율에 해당하는 프레임들은 다음 주기에 전송될 수 있으나 전송이 중단된 비트율에 속한 프레임들은 서버에 의해 폐기되는 것을 의미한다. 이 기법에서도 전송이 중단되는 프레임들에 대하여 우선순위 정책을 적용한다. 즉 중요도가 가장 낮은 B 프레임들이 먼저 중단되며 이후 P 프레임, I 프레임 순으로 전송이 중단되도록 하였다.

그리고 B_{sum}이 B_{avail}보다 작은 경우에는 서버가 모든 클라이언트의 프레임들을 전송한 후 여분의 대역폭을 버퍼 수준이 가장 낮은 클라이언트부터 추가로 할당하여 다음 주기에 전송할 프레임들을 이번 주기에 미리 전송함으로써 해당 클라이언트의 버퍼 수준이 추가로 증가된다. 실제 대역폭의 합 B_{sum}이 B_{avail}

보다 작거나 같을 경우의 전송 스케줄링 알고리즘은 그림 4와 동일하게 적용되므로 본 절에서는 기술을 생략한다.

3.4 할당된 대역폭내 전송

본 논문에서는 3.2절과 3.3절에서 기술된 버퍼수준을 활용하여 통신망 대역폭을 조정하는 스케줄링 알고리즘과 비교하기 위하여 클라이언트의 버퍼수준을 직접 활용하여 통신망 대역폭을 조정하지 않는 알고리즘을 그림 6에서 기술한다. 서버는 임의의 클라이언트에게 비디오 스트림을 전송할 경우 다른 클라이언트의 버퍼수준이 높거나 낮은 값에 상관없이 해당 클라이언트에게 할당된 대역폭만을 활용하게 된다. 서버는 각 클라이언트 i의 평균 비트율 M_i(비트로 표현된 프레임의 평균 크기와 재생율의 곱)를 실제 비트율 B_i와 매 주기마다 비교한다. 3.2절과 3.3절에 기술된 알고리즘과는 달리 B_i가 M_i보다 클 경우(1행) 다른 클라이언트에 할당된 대역폭을 활용할 수 없기 때문에 M_i만큼의 비트율만 전송할 수 있다. 이때 클라이언트 i의 버퍼 수준 L_i가 0보다 크면(2행) 전송할 수 없는 나머지 B_i-M_i 비트율은 전송이 보류되며 이는 P_i로 설정된다(4행). P_i 비트율은 다음 주기에 전송될 수 있다. 즉 해당 클라이언트는 자신의 버퍼에 저장한 프레임들과 이번 주기에 수신한 프레임들을 포함하여 최소 1초 이하의 시간 동안 재생할 수 있는 여유가 있다. 그러나 L_i가 0인 경우(7행) 버퍼에 프레임들이 전혀 없고 이번 주기에 수신한 프레임들만 재생할 수 있으므로 재생시간은 1초 미만이 된다. 따라서 이 경우 B_i-M_i 비트율은 전송 보류대신 서버가 전송 중단으로 처리하게 되므로 전송 보류할 비트율 P_i는 0이 되고(9행) 전송 중단을 의미하는 비트율 D_i는 B_i-M_i가 된다(10행). 서버는 현재 버퍼 수준 L_i에 M_i/B_i 값을 증가시키게 된다(5행과 11행).

그리고 B_i가 M_i보다 작거나 같을 경우(13행) B_i는 모두 전송되므로 P_i는 0이 되고(15행) L_i는 1.0만큼 증가된다(16행). 이후 M_i-B_i 만큼의 여분 대역폭을 활용하기 위하여 다음 주기에 전송할 비트율 $B_{next,i}$의 일부 또는 전부를 전송하게 된다. 이때 $B_{next,i}$가 M_i-B_i보다 작거나 같을 경우(17행) $B_{next,i}$가 모두 전송되며 L_i는 1.0만큼 추가로 증가된다(18행). 그러나 $B_{next,i}$가 M_i-B_i보다 클 경우(19행)에는 $B_{next,i}$ 중 M_i-B_i 크기만 전송될 수 있으므로 버퍼 수준 L_i는 $(M_i-B_i)/B_{next,i}$만큼 추가로 증가된다(20행).

```
1: if (Bi > Mi)
2:   if (Li > 0)
3:     begin
4:       Pi=Bi-Mi;
5:       Li=Li+Mi/Bi;
6:     end
7:   else
8:     begin
9:       Pi=0;
10:      Di=Bi-Mi;
11:      Li=Li+Mi/Bi;
12:    end
13: else
14:   begin
15:     Pi=0;
16:     Li=Li+1.0;
17:     if (Bnext,i ≤ Mi-Bi)
18:       Li=Li+1.0;
19:     else
20:       Li=Li+(Mi-Bi)/Bnext,i;
21:   end
```

그림 6. 할당된 대역폭내 전송 알고리즘

3.5 ABS(Adaptive Buffer Sensitive) 기법

본 논문에서는 클라이언트 버퍼수준을 사전에 파악하여 비디오 스트림별로 통신망 대역폭을 정적으로 할당하는 ABS 부하 조정 기법을 제시된 스케줄링 알고리즘들과 성능을 비교 분석하고자 한다. 이 ABS 기법은 이용가능한 통신망 대역폭을 요청된 스트림들의 수로 나눈 후 이 값을 각 스트림의 평균 대역폭과 비교하여 더 작은 값을 해당 스트림의 실제 대역폭으로 결정하는 방식이다. ABS 기법은 1 단계에서 이용가능한 통신망 대역폭 $BW_Available$을 클라이언트들의 수 n으로 나눈 후 이 값을 임의의 스트림 i의 평균 대역폭 $BW_Consume_i$와 비교하여 더 작은 값을 해당 스트림의 실제 대역폭 BW_Ad_i로 결정한다.

$$BW\_Ad_i = \min\left(\frac{BW\_Available}{n}, BW\_Consume_i\right) \quad (3)$$

이후 오류가 발생하지 않은 클라이언트들의 수를 N이라 할때 1 단계에서 할당된 대역폭 BW_Ad_{FR}은 수식 (4)와 같이 결정된다.

$$BW\_Ad_{FR}=\sum_{i\in N}\min\left(\frac{BW\_Available}{n}, BW\_Consume_i\right) \quad (4)$$

모든 스트림의 실제 대역폭의 합이 이용가능한 통신망 대역폭보다 작을 경우에만 여분의 통신망 대역폭이 발생하며 이 여분의 대역폭 $1-BW_Ad_{FR}$을 2단계에서 스트림 i는 단말기 버퍼 수준 $Buffer_Playback_Duration$을 고려하여 다음과 같이 추가 할당받게 되며 수식 (5)에서 BW_AS_i로 정의된다.

$$BW\_AS_i=\frac{(Buffer\_Playback\_Duration)^{-1}}{\left(\sum_{i=1}^{n} Buffer\_Playback\_Duration^{-1}\right)}\times(1-BW\_Ad_{FR}) \quad (5)$$

즉 버퍼 수준이 낮을수록 버퍼 수준이 높은 스트림에 비하여 여유 대역폭을 더 많이 할당받은 후 프레임들을 전송하게 된다. 그러나 ABS 기법은 스트림별로 통신망 대역폭을 정적으로 할당한 후 전송 중 발생하는 단말기 버퍼 수준을 별도로 고려하지 않기 때문에 이러한 사항을 고려하여 동적으로 대역폭을 조절하는 본 논문에서 제시된 알고리즘들과 성능을 직접 비교하기는 사실상 어렵다.

4. 성능 분석

4.1 실험 환경

본 논문에서는 압축률이 우수한 VBR(Variable Bit-Rate) 방식으로 부호화된 MPEG-4 비디오 스트림[13,14] “Silence of the Lambs", "Star Wars IV", "Terminator One" 들을[15] 대상으로 실험을 수행하였다. 이 비디오 스트림들은 모두 QCIF 형식이며 비율 제어(rate control)없이 I, P, B 프레임에 대한 양자화 계수가 30으로 동일하게 설정되었으며 시간적 scalable 방식으로 압축되었다. 한 GOP내 프레임 수는 12이고 GOP 패턴은 IBBPBBPBBPBB이다. 세 비디오 스트림들의 총 프레임 수는 각각 108000개이며 "Star Wars IV", "Jurassic Park One", "Terminator One"의 프레임의 평균 크기는 각각 1980.016비트, 2215비트, 2605비트로 분석되었다. 비디오 스트림의 재생율을 24로 동일하게 적용할 경우 "Star Wars IV", "Jurassic Park One", "Terminator One"의 평균 비트율(=프레임의 평균크기*재생율)은 각각 47760.384 비트/초, 53160비트/초, 62520비트/초가 되며 20 개의 클라이언트를 대상으로 실험이 수행되었다. 각 스트림의 평균비트율의 합이 통신망의 이용가능한 전송 대역폭 B_{avail}과 각각 같아서 여분의 대역폭이 없는 것으로 가정한다. 각 스트림의 108000개 프레임을 재생율 24로 재생할 경우 4500초 즉 75분이 필요하며 이 시간 동안 실험이 수행되었으나 실험 결과에 오류를 최소화하고 신뢰성을 높이기 위하여 처음 10분과 마지막 15분을 제외한 10분 간격으로 20분부터 60분까지 클라이언트들의 평균 실제 재생율이 측정되었다(그림 7, 8, 9 참조). 실제 재생율이란 매초마다 서버가 각 클라이언트에게 실제로 전송하여 클라이언트가 재생한 프레임 수를 의미하며 대역폭의 부족으로 인하여 전송 중단된 프레임들이 많을수록 실제 재생율의 값은 작아지게 된다. 그림 7, 8, 9에서 통신망의 대역폭 부족 현상이 없을 경우 이론적으로 재생율은 초기치 24와 동일하게 나타나겠지만 비디오 스트림의 가변적인 비트율로 인하여 통신망 대역폭 부족 현상이 발생하고 이를 해결하는 방법에 따라 실제 재생율은 달라지게 된다.

본 논문에서 제시한 최소값 버퍼수준 우선 전송 알고리즘과 최대값 버퍼수준 우선 보류 알고리즘은 Min_First와 Max_First로 각각 표기되고 할당된 대역폭내 전송 알고리즘은 Limit_Only로 각 그림에서 표기된다. 실험에서 각 클라이언트의 최대 버퍼수준은 5.0이 되며 이는 클라이언트가 처음에 서버로부터 프레임들을 5초 동안 수신한 이후 재생할 수 있음을 의미한다. 그림 7부터 9까지 클라이언트의 버퍼 수준을 사전에 파악하여 비디오 스트림별로 통신망 대역

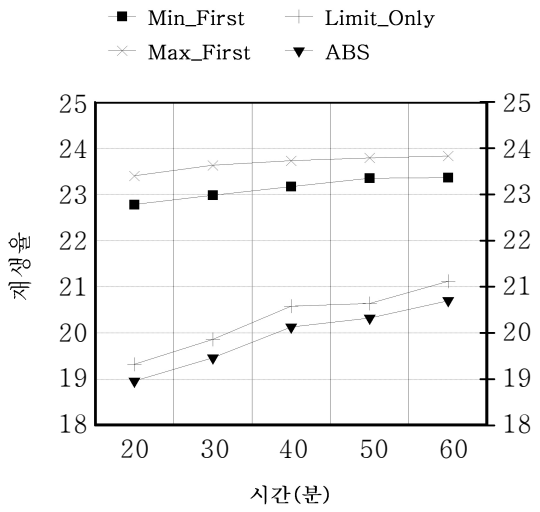

그림 7. “Star Wars IV”의 실제 재생율

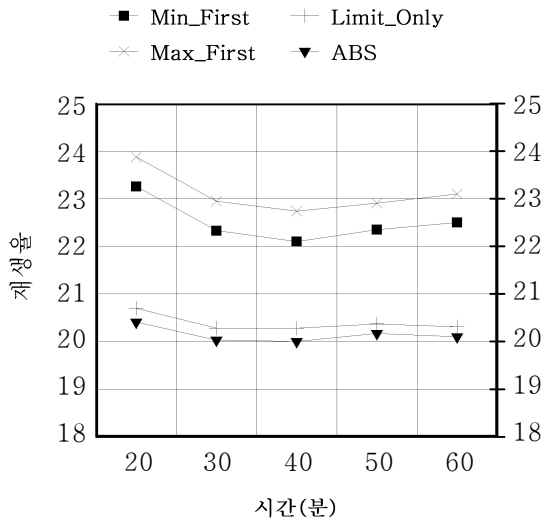

그림 8. "Jurassic Park One"의 실제 재생율

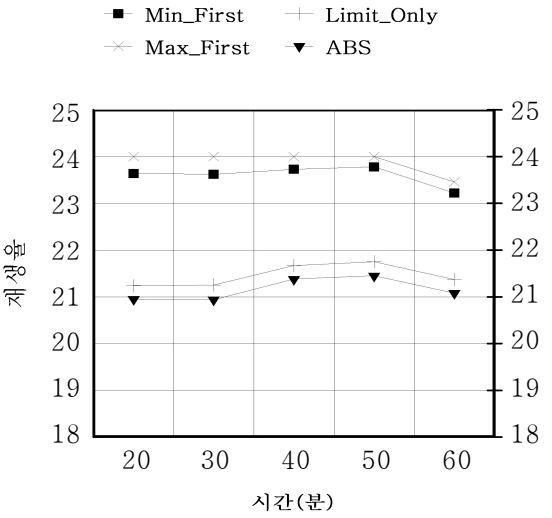

그림 9. "Terminator One"의 실제 재생율

폭을 정적으로 할당하는 ABS(Adaptive Buffer Sensitive) 기법[2]과 제시된 스케줄링 알고리즘들의 성능이 비교되고 있다. 특히 본 실험에서는 통신망의 여유 대역폭은 전혀 없는 것으로 가정하고 ABS 기법과 본 논문에서 제시된 세가지 알고리즘들과의 성능이 비교 분석되었다.

4.2 실제 재생율

그림 7은 "Star Wars IV" 스트림의 실제 재생율이 측정된 결과를 나타내고 있다. Max_First 기법의 실제 재생율은 이 23.4에서 23.83의 범위로 나타났고 Min_First 기법은 22.78에서 23.36의 범위로 나타났다. Max_First 기법은 Min_First 기법에 비하여 0.47-0.62 정도의 차이로 재생율이 더 높게 나타났으며 이론적 재생율인 24와 비교할 경우 97.5%에서 99.3%로 근접하고 있다. 클라이언트별로 할당된 대역폭만을 활용하는 Limit_Only 기법은 Min_First 기법에 비하여 실제 재생율이 2.24에서 3.66 정도 낮게 나타났다. 버퍼수준을 활용하여 대역폭을 조정하는 스케줄링 기법들인 Min_First와 Max_First 기법의 결과가 대역폭을 별도로 조정하지 않는 Limit_Only 기법보다 재생율이 높게 나타난 결과가 확인되었다.

Limit_Only 기법은 ABS 기법보다는 재생율이 0.36에서 0.51 정도로 더 높게 나타났으며 이는 대역폭을 정적으로 할당한 후 버퍼수준에 상관없이 스트림을 전송하는 결과보다 버퍼수준에 따라 전송량을 조정하는 기법의 결과가 향상될 수 있음을 의미하는 것이다. "Jurassic Park One"에 대한 재생율의 차이를 나타내는 그림 8과 "Terminator One"에 대한 재생율 차이를 나타내는 그림 9에서도 다소 정도의 차이가 있으나 전체적으로 그림 7과 유사한 결과가 나타났다.

그림 7, 8, 9에서 Max_First 기법의 재생율이 Min_First 기법의 재생율보다 더 높게 나타난 것은 Max_First 기법에서 서버가 각 클라이언트의 버퍼 수준이 높을수록 전송 보류한 이후 다음 주기에 보류된 프레임들을 전송함으로써 결과적으로 Min_First 기법보다 더 많은 프레임들이 해당 클라이언트 단말기에서 재생될 수 있었던 것으로 분석된다. 이 두 기법의 평균 버퍼 수준을 20분부터 60분까지 10분 간격으로 분석한 결과는 표 1에 기술되어 있다. 실험에서 버퍼수준은 최소값이 0이며 최대값이 5.0으로 설정되었다. 표 1에서 세 비디오 스트림에 대하여 Max_First 기법이 Min_First 기법보다 항상 평균 버퍼 수준이 높게 유지되고 있다. 이는 통신망 대역폭이 부족할 경우 Max_First 기법에서 서버가 전송 중단시켜야 하는 프레임들의 수를 최소화하고 전송 보류하는 프레임들의 수를 증가시킴으로써 Min_First 기법보다 실제 재생율을 더 향상시킬 수 있음을 의미한다.

표 2는 각 스트림에 대하여 Min_First, Max_First, Limit_Only, ABS 기법에서 전송 중단된 프레임들을 분석한 결과이다. 평균적으로 전송 중단된 프레임 수 자체는 Max_First 기법이 가장 작고 Min_First 기법과는 근소한 차이가 나타났다. 이후 Limit_Only 기법과 ABS 기법 순서로 전송 중단된 프레임 수가 증가

표 1. 평균 버퍼 수준(초)

비디오 스트림	스케쥴링 기법	20분	30분	40분	50분	60분
Star Wars IV	Min_First	1.44	0.37	1.92	4.90	1.35
	Max_First	2.87	2.09	4.08	4.10	3.39
Jurassic Park One	Min_First	0.59	0.09	0.05	1.55	0.40
	Max_First	2.89	1.44	0.25	3.31	3.74
Terminator One	Min_First	1.29	1.70	4.59	3.82	1.66
	Max_First	4.17	4.12	4.10	4.10	2.02

표 2. 프레임별 전송 중단 비율(%)

	스케쥴링 기법	I 프레임	P 프레임	B 프레임
Star Wars IV	Min_First	0	0.56	99.44
	Max_First	0	0	100.0
	Limit_Only	0	1.20	98.80
	ABS	0	2.19	97.81
Jurassic Park One	Min_First	0.03	1.61	98.36
	Max_First	0	0	100
	Limit_Only	0	1.31	98.69
	ABS	0	2.32	97.68
Terminator One	Min_First	0	0.01	99.99
	Max_First	0	0	100.0
	Limit_Only	0	0	100.0
	ABS	0	0.83	99.17

했으며 이러한 결과는 그림 7, 8, 9에서 실제 재생율로 확인한 바 있다. 표 2는 전송 중단된 프레임들에 대한 I 프레임, P 프레임, B 프레임의 비율을 분석한 것이다. 네가지 기법 모두 공통적으로 I 프레임의 비율이 최대 0.03% 수준으로 가장 낮고 이후 P 프레임이 최대 2.32% 정도로 나타났으며 B 프레임이 최소 97.81%에서 100.0% 정도로 가장 높게 나타났다. 본 논문의 Min_First와 Max_First 기법에서 서버는 B 프레임의 중요도가 가장 낮으므로 전송 중단해야 하는 프레임들에 대해서는 우선순위가 가장 낮은 B 프레임이 먼저 중단되도록 하였으며 이후 P 프레임, I 프레임 순서로 중단되도록 하였다. 이러한 우선순위 정책은 Limit_Only 기법과 ABS 기법에서도 실험 결과를 Min_First 기법 그리고 Max_First 기법과 공정하게 비교하기 위하여 전송 과정에서 우선순위가 높은 프레임들이 먼저 전송되도록 실험이 수행되었다.

5. 결 론

모바일 통신망에서 클라이언트가 요청한 비디오 스트림을 서버가 전송할 때 통신망 대역폭이 부족할 경우 단말기 버퍼 수준을 활용하여 전송을 중단해야 하는 프레임을 최소화하고 다음 주기에 전송할 수 있도록 전송을 보류하는 프레임을 증가시킴으로써 실제 재생율이 향상될 수 있는 스케쥴링 기법이 본 논문에서 제시되었다. 특히 최대값 버퍼수준 우선(Max_First) 전송 보류 스케쥴링 기법은 클라이언트의 버퍼수준을 최소값 버퍼수준 우선(Min_First) 전송 스케쥴링 기법보다 더 효과적으로 활용함으로써 결과적으로 서버가 전송을 중단해야 하는 프레임 수를 더 감소시킴은 물론 전송을 보류할 수 있는 프레임 수를 증가시켜 클라이언트의 실제 재생율을 더욱 향상시킬 수 있다. 제시된 두 기법의 평균 버퍼수준도 실험 결과에서 최대값 버퍼수준 우선 전송 보류 스케쥴링 기법이 최소값 버퍼수준 우선 전송 스케쥴

링 기법보다 항상 높게 유지되는 것으로 나타났다.

본 논문에서 제시된 스케줄링 기법들은 통신망 대역폭을 정적으로 할당하는 다른 기법과의 성능 비교 결과 비디오 스트림의 서비스 품질과 seamless video streaming 효과를 향상시킬 수 있는 것으로 분석되었다. 향후 통신망 대역폭이 한정된 모바일 통신망 기반의 멀티미디어 시스템에서 실시간적 성능과 서비스 품질을 더욱 향상시킬 수 있는 것으로 기대된다.

참 고 문 헌

[1] M. Rumney, *LTE and Evolution to 4G Wireless*, Agilent Technologies, 2009.

[2] J. Yuen, K.Y. Lam, and E. Chan, "A Fair and Adaptive Scheduling Protocol for Video Stream Transmission in Mobile Environment," *Proc. IEEE Int'l Conf. on Multimedia and Expo,* pp. 409-412, 2002.

[3] K.Y. Lam, J. Yuen, S.H. Son, and E. Chan, "Scheduling Video Stream Transmissions for Distributed Playback over Mobile Cellular Networks," *Proc. Int'l Conf. on Parallel and Distributed Systems*, pp. 363-368, 2002.

[4] S. Rao and A.M.K. Cheng, "Scheduling and Routing of Real-Time Multimedia Traffic in Packet-Switched Networks," *Proc. IEEE Int'l Conf. on Multimedia*, pp. 545-548, 2000.

[5] M. Burza, J. Kang, and P. van der Stok, "Adaptive Streaming of MPEG-based Audio/Video Content over Wireless Networks," *Journal of Multimedia,* Vol.2, No.2, pp. 17-27, 2007.

[6] X. Hei, Y. Liu, and K.W. Ross, "Inferring Network-Wide Quality in P2P Live Streaming Systems," *IEEE Journal on Selected Areas in Communications,* Vol.25, Issue 9, pp. 1640-1654, 2007.

[7] H. Espeland, C.H. Lunde, H.K. Stensland, C. Griwodz, and P. Halvorsen, "Transparent Protocol Translation and Load Balancing on a Network Processor in a Media Streaming Scenario," *Proc. Network and Operating Systems Support for Digital Audio and Video*, 2008.

[8] L. Xu, X. Shen, and J. W. Mark, "Dynamic Fair Scheduling With QoS Constraints in Multimedia Wideband CDMA Cellular Networks," *IEEE Trans. on Wireless Communications,* Vol.3, No.1, pp. 60-73, 2004.

[9] M. Krunz, "Bandwidth Allocation Strategies for Transporting Variable-Bit-Rate Video Traffic," *IEEE Communications Magazine*, Vol.37, Issue 1, pp. 40-46, 1999.

[10] F. Yang, Q. Zhang, W. Zhu, and Y. Zhang,"Bit Allocation for Scalable Video Streaming over Mobile Wireless Internet," *Proc. IEEE INFOCOM*, pp. 2142-2151, 2004.

[11] S. Mohapatra and N. Venkatasubramanian, "Proactive Energy-Aware Video Streaming to Mobile Handheld Devices," *Proc. IEEE Conf. on Mobile and Wireless Communications Networks*, pp. 187-190, 2003.

[12] X. Cheng, P. Mohapatra, S. Lee, and S. Banerjee, "Performance Evaluation of Video Streaming in Multihop Wireless Mesh Networks," *Proc. Network and Operating Systems Support for Digital Audio and Video*, pp. 57-62, 2008.

[13] International Organization for Standardization, *Information Technology-Coding of Audio-Visual Objects(MPEG-4) Part 2: Video, international standard*, ISO/IEC JTC 1/SC 29 14496-2, 2002.

[14] 임현정, 임순범, "MPEG-4 BIFS 기반 모바일 방송 환경에서 3D 객체 및 GUI 표현 기술 연구," 멀티미디어학회논문지, 제12권, 제5호, pp. 677-687, 2009.

[15] Trace Files and Statistics: Video Library, http://trace.eas.asu.edu/cgi-bin/main.cgi, 2011.

김 진 환

1982년 3월～1986년 2월 서울대학교 컴퓨터공학과 학사
1986년 3월～1988년 2월 서울대학교 컴퓨터공학과 석사
1988년 3월～1994년 2월 서울대학교 컴퓨터공학과 박사
1994년 3월～1996년 2월 서울대학교 컴퓨터신기술공동연구소 특별연구원
1995년 3월～현재 한성대학교 멀티미디어공학과 교수
관심분야 : 멀티미디어시스템, 분산 실시간 시스템