韓國컴퓨터情報學會 論文誌
第15卷 第11號, 2010. 11.

2010-15-11-3-3

프록시 시스템에서 multi-level 스트리밍 서비스를 위한 세그먼트 기반의 버퍼관리

이 종 득*

Segment-based Buffer Management for Multi-level Streaming Service in the Proxy System

Chong Deuk Lee*

요 약

프록시 시스템에서의 QoS는 혼잡 (congestion), 지연 (delay), 재전송 (retransmission) 등과 같은 간섭에 의해 많은 영향을 받는다. 또한 멀티-레벨 스트리밍 서비스는 시간 동기화에 의해 영향을 받으며, 이로 인하여 서비스 성능이 저하된다. 본 논문에서는 프록시 시스템에서 발생하는 스트리밍 서비스의 성능 저하를 개선하고 스트리밍 처리율을 향상시키기 위한 세그먼트 기반의 버퍼 관리 메커니즘을 제안한다. 제안된 논문의 목적은 다음과 같다. 1) 세그먼트 기반의 버퍼관리 메커니즘을 이용하여 다중 스트리밍 서비스를 최적화한다. 2) 혼잡, 간섭 등으로 인해 발생되는 오버헤드를 줄인다. 3) 끊김 현상, 지연 등으로 인해 발생하는 재전송의 문제를 최소화한다. 이러한 목적을 수행하기 위해 우리는 퍼지 값 μ와 비용 가중치 ω를 이용한다. 시뮬레이션 결과 제안된 메커니즘은 버퍼 캐시 제어율, 평균 패킷 손실률, 그리고 스트림 적합성 척도에 따른 지연 절약율에 있어서 기존의 고정길이 세그먼트 기법, 피라미드 (pyramid) 세그먼트 기법, 그리고 스카이스크랩퍼 (skyscraper) 세그먼트 기법보다 성능이 효율적임을 보였다.

Abstract

QoS in the proxy system are under heavy influence from interferences such as congestion, latency, and retransmission. Also, multi-level streaming services affects from temporal synchronization, which lead to degrade the service quality. This paper proposes a new segment-based buffer management mechanism which reduces performance degradation of streaming services and enhances throughput of streaming due to drawbacks of the proxy system. The proposed paper optimizes streaming services by: 1) Use of segment-based buffer management mechanism, 2) Minimization of overhead due to congestion and interference, and 3) Minimization of retransmission due to disconnection and delay. This paper utilizes fuzzy value μ and cost weight ω to process the result. The simulation result shows that the

• 제1저자 : 이종득
• 투고일 : 2010. 06. 23, 심사일 : 2010. 09. 09, 게재확정일 : 2010. 09. 11.
* 전북대학교 전자공학부 교수

proposed mechanism has better performance in buffer cache control rate, average packet loss rate, and delay saving rate with stream relevance metric than the other existing methods of fixed segmentation method, pyramid segmentation method, and skyscraper segmentation method.

▸ Keyword : 혼잡, 멀티-레벨 스트리밍, 프록시 시스템, 세그먼트-기반 버퍼관리(congestion, multi- level streaming, proxy system, segment-based buffer management)

Ⅰ. 서 론

최근에 대부분의 프록시 캐시 구조에서 스트리밍 서비스는 네트워크 링크와 디바이스 프로파일에 일치되는 세그먼트 기반의 스트리밍 서비스 구조로 빠르게 진화되고 있다. 프록시 구조에서 멀티-레벨 스트리밍을 위해서는 스트리밍이 수행될 미디어 객체의 크기를 고려해야 한다. 미디어 객체의 크기를 고려한 프록시 캐시는 보다 나은 전송 품질을 제공받을 수 있는데 비해서 미디어 객체의 크기를 고려하지 않는 경우에는 프록시 구조에서 캐시 용량 제한으로 인한 지터 지연, 혼잡 등의 오버헤드 문제가 발생하게 된다.

따라서 보다 나은 스트리밍 서비스 품질을 제공받기 위해서는 프록시 캐시의 크기, 비트율, 그리고 대역폭 등을 고려한 세그먼트 기반의 미디어 객체 분할이 수행되어야 한다. 그러나 기존의 미디어 캐싱 기법은 동일한 미디어 객체의 여러 버전들을 분할하여 스트리밍을 수행하기 보다는 미디어 객체의 각 버전들을 하나의 객체로 간주하여 스트리밍을 수행하고 있다[1][2][3]. 예를 들어 하나의 객체의 버전을 세그먼트 단위로 분할하지 않고 객체 전체를 캐싱한다면 프록시에는 데이터 크기로 인한 혼잡 및 지터 지연과 같은 오버헤드가 발생하게 된다. 이러한 이유로서 기존의 프록시 기법들은 크기에 따른 객체들의 캐시 상태들은 고려하지 않은 채 객체들의 단일 버전만을 고려하여 캐싱을 수행하고 있다. 이러한 문제를 해결하기 위해 트랜스 코딩 프록시 기법이 제안되고 있다[2][4]. 트랜스 코딩 프록시 기법은 캐시할 객체들의 분할된 세그먼트의 버전을 파악하여 분할된 객체에 대한 여러 버전 속성이 있을 경우 이들 여러 버전 속성을 고려하여 캐싱과 스트리밍을 효과적으로 수행하는 기법이다. Chang과 Chen은 캐싱이 수행될 때 객체의 각기 다른 버전들의 이득을 계산하기 위한 범용 이득함수(profit function)를 제안하였다[1]. 그러나 이 기법은 웹 객체들을 효율적으로 스트리밍하기 위한 기법으로서 크기가 큰 미디어 객체의 경우에는 크기속성 제약으로 인해 캐시용량에 따른 인코딩율과 디코딩율의 불균형 문제가 발생하고 있다. 이와 같은 불균형 문제가 발생하면, 지터 지연이 발생하게 되며, 캐시히트율이 떨어지게 된다. [4]는 미디어 객체들을 스트리밍하기 위한 기법으로서 트랜스 코딩 기법과 캐싱 기법을 결합한 하이브리드 기법을 제안하였다. 이 기법은 트랜스코딩 버전이 비교적 낮은 요청들을 스트리밍 서비스하는 기법으로서 FVO (Full Version Only)가 Full 일 때만 원본 (Original) 객체를 캐싱하는 기법이다. 그러나 이 기법은 이미 트랜스코드된 객체들에 대해서는 캐시하지 못하는 단점을 가지고 있다. 그리고 TVO (Transcoded Version Only)는 트랜스코드된 객체들만을 캐시하는 알고리즘으로서 사용자 요청이 정확하지 않으면 트랜스코드된 버전을 처음부터 다시 패치 해야 하는 단점을 가지고 있다.

따라서 본 논문에서는 프록시에서 캐시 용량을 고려하여 인코딩율과 디코딩율에 따른 지터 지연 혼잡, 재전송 등을 최적화하기 위한 세그먼트 기반의 버퍼 관리 메커니즘을 제안한다. 제안된 메커니즘은 패킷 크기가 보다 큰 멀티미디어 스트림들을 적응적으로 캐싱하기 위해 미디어 객체를 세그먼트 단위로 분할하여 세그먼트화하며, 세그먼트들의 캐싱 오버헤드를 줄이기 위해 퍼지 값 μ를 적용한다[5]. 퍼지 값 μ는 세그먼트들의 스트리밍 우선순위와 사용빈도에 따른 중요도를 결정하게 된다. 제안된 논문의 목적은 다음과 같다. 1) 퍼지 값 μ를 이용하여 다중 스트리밍 서비스를 최적화한다. 2) 혼잡, 지터 지연 등으로 인해 발생되는 패킷 손실을 최소화한다. 3) 끊김, 재전송 등으로 인해 발생하는 re-streaming의 문제를 최소화한다. 우리는 시뮬레이션 결과를 통하여 제안된 기법의 성능이 기존의 고정길이 세그먼트 기법, 피라미드 세그먼트 기법, 그리고 스카이 스크랩퍼 세그먼트 기법보다 성능이 효율적임을 보인다. 본 논문의 구성은 다음과 같다. 2장에서는 관련연구에 대해서 알아보고, 3장에서는 제안된 논문의 멀티-레벨 스트리밍 서비스를 위한 세그먼트 기반의 버퍼 관리 메커니즘에 대해서 알아본다. 4장에서는 제안된 기법의 시뮬레이션 결과에 대해서 알아보며, 끝으로 결론에 대해서 살펴본다.

Ⅱ. 관련연구

무선 모바일 네트워크에서 멀티미디어 스트리밍 서비스는 주로 RTP (Real-Time Transport Protocol), RTCP (Real-Time Control Protocol)을 이용한다. RTCP는 5개의

패킷으로 구성되어 있으며, 이 중에서 RTCP SR (Sender Report), RR (Receiver Report)을 이용하여 RTT (Round Trip Time), 지연, 패킷 손실 등을 모니터링 한다[6][7][8]. 그러나 이들은 패킷 손실, 지터 지연 등의 이슈들을 적응적으로 파악하기 어려우며, 클라이언트로부터 수신된 패킷 정보를 주기적으로 모니터링 해야 하는 단점이 있다. 이러한 문제를 해결하기 위하여 세그먼트 기반의 스트리밍 서비스 기법이 제안되고 있으며, 이 기법은 분산 모바일 응용에서 서비스의 성능을 향상시키고 보다 나은 스트림의 품질을 제공해 주기 위해 여러 분야에서 적용되고 있다[9][10][11][12]. 세그먼트 기반의 스트림 품질 향상을 위한 기법으로는 고정 길이 세그먼트 (fixed segmentation) 기법, 피라미드 세그먼트 (pyramid segmentation) 기법, 그리고 스카이스크랩퍼 세그먼트 (sky scraper segmentation) 기법이 주로 사용되고 있다[13][14].

고정 길이 세그먼트 기법은 각각의 세그먼트 크기를 고정된 크기로 분할하는 기법이다. 이 기법의 단점은 세그먼트 크기를 너무 작게 분할하거나 또는 너무 크게 분할하면 캐시 버퍼 관리의 어려움으로 인해 오버헤드가 심각하게 발생한다. 고정 길이 세그먼트 기법에서 세그먼트 크기를 너무 작게 분할하면 지터 지연 문제가 발생하게 되며, 또한 너무 크게 분할하게 되면 미디어 객체 크기로 인한 혼잡문제가 발생하게 된다. 피라미드 세그먼트 기법은 세그먼트 관리를 피라미드 형태로 관리하는 기법으로서 세그먼트 크기는 지수적으로 증가하게 된다. 예를 들어 그림1에서 보는바와 같이 세그먼트 i+1의 크기는 세그먼트 i의 크기보다 두 배로 증가하게 된다. 피라미드 세그먼트 기법은 고정길이 세그먼트 기법과 비교할 때 시작부분에 배치된 세그먼트들은 미디어 객체들의 뒷부분에 배치된 세그먼트들에 비해서 중요도가 높은 세그먼트들이 된다. 일반적으로 중요도가 높은 세그먼트들은 우선순위가 높은 세그먼트들로서 이러한 세그먼트들은 스트리밍 서비스를 우선적으로 수행하게 된다. 그러나 이 기법은 중요도가 높은 세그먼트들을 우선적으로 처리함으로서 상대적으로 중요도가 떨어지는 세그먼트들에 대해서는 스트리밍이 수행될 기회가 제공되지 못하는 문제점이 발생하고 있다. 또한 중요도가 떨어지는 세그먼트가 캐시될 때 캐시 용량 제약으로 인해 버퍼 오버플로우가 발생되는 단점을 가지게 된다.

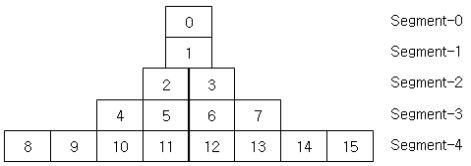

그림1. 피라미드 세그먼트화
Fig. 1. Pyramid Segmentation

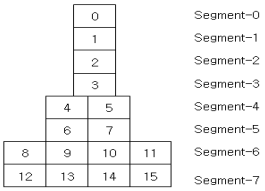

그림2. 스카이 스크렙퍼 세그먼트화
Fig. 2. Skyscraper Segmentation

그리고 스카이 스크렙퍼 세그먼트기법은 피라미드 기법을 변형한 세그먼트 기법이다. 이 기법은 피라미드 기법에 비해서 세그먼트 크기가 서서히 변한다. 스카이 스크렙퍼 세그먼트 기법은 그림2에서 보는바와 같이 세그먼트 i+1의 크기는 세그먼트 i의 크기와 같거나 또는 두 배로 증가하게 된다. 이 때 세그먼트들은 피라미드 구조와 달리 스크랩퍼 형태로 구성된다. 스카이 스크렙퍼 세그먼트 기법은 피라미드 기법에 비해서 많은 세그먼트들을 관리할 수 있는 장점은 있지만 고정길이 세그먼트 기법처럼 세그먼트 크기가 작게 분할될 경우 관리 비용이 증가하게 되며, 이로 인해 오버헤드가 증가되는 단점을 가지고 있다.

III. 세그먼트 기반의 버퍼관리 메커니즘

분산 모바일 네트워크 응용에서 RTP (Real-Time Transport Protocol), RTCP (Real-Time Control Protocol) 프로토콜은 스트리밍 서비스 품질을 적응적으로 파악하기 어렵다[15]. 이 장에서는 세그먼트 기반의 버퍼 관리를 위한 멀티-레벨 스트림 구조, 멀티-레벨 구조에서의 스트림 적합성, 그리고 버퍼 관리 기법에 대해서 살펴본다.

3.1 멀티-레벨 스트림 구조

프록시 서버에서 세그먼트 단위로 스트리밍을 수행하기 위해서는 스트림 패킷들은 연속적인 스트림 버퍼 캐시에 세그먼트화 되어야 한다[11][12][13]. 실제로 버퍼 캐시에 스트림 패킷이 세그먼트화 되어 있지 않으면 세그먼트화 과정으로 인한 스트림 지연이 발생하게 된다. 이러한 스트림 지연은 혼잡과 재전송의 문제를 야기하며, 이로 인해 스트리밍 서비스 품질이 떨어지게 된다. 이러한 문제를 해결하기 위해 임계값을 이용한 거리 기반 기법 [6]이 제안되었으나 이 기법은 임계값에 따른 스트림 적합성을 파악하기가 어려운 문제점이 있다. 본 논문에서는 이러한 문제점을 해결하기 위하여 퍼지 값 μ를 이용한다. 퍼지 값 μ는 스트림 패킷 상에서 발생되는 모호성과 스트리밍의 부정확성을 해결하기 위해 사용되며, 또한 스트

리밍의 적합성과 우선순위를 정하는데 사용된다. 본 논문에서는 퍼지 값 μ를 적용하기 위하여 초기 스트리밍 단계에서 적응적 멀티-레벨 스트림이 가능하도록 스트림 패킷을 버퍼 캐시에 버켓화하여 세그먼트를 레벨 단위로 구조화한다. 이렇게 구조화된 세그먼트는 스트림 패킷의 수가 증가될 때 스트리밍을 최적화하게 된다. 이때 멀티-레벨의 최적화 여부는 스트림 패킷의 버스트 크기와 타임-스탬프 (time-stamp)에 의해 결정된다. 실제 버스트 크기와 타임-스탬프는 스트림의 적합성과 우선순위를 결정하는 중요한 요소가 된다. 이제 멀티-레벨 스트림 구조에서 스트림을 위한 타임스탬프 구간 ts를 [t_s, t_e]라 하자. 여기서 t_s는 타임-스탬프가 시작되는 시간구간이며, t_e는 타임-스탬프가 끝나는 시간구간이다. 각 레벨을 구성하는 버스트 크기는 먼저 (L-0)레벨을 구성한 후 스트리밍을 수행할 세그먼트 크기 S에 의해 결정된다. 이 후에 한 단계 상위 레벨 L-1의 구성은 세그먼트 스트림의 평균 적합성 비용과 서브 스트림의 우선순위를 측정하여 비용이 적고, 유사도가 높은 세그먼트 스트림을 상위 레벨의 스트림으로 결정한다. 그리고 난 후 세그먼트 스트림 결정은 퍼지 값 μ를 적용하여 퍼지 값 μ가 큰 순서로 각 레벨을 결정하게 된다. 이 작업은 세그먼트 스트림들에 대해서 타임스탬프 레벨 ts_e^L이 끝날 때까지 상위 레벨에까지 반복 수행된다. 그림3은 L=4, S_n=8, ts=8인 멀티-레벨 스트림 구조를 나타낸 예이다. 여기서 L은 레벨, S_n은 세그먼트 스트림의 수, 그리고 ts는 타임 스탬프이다.

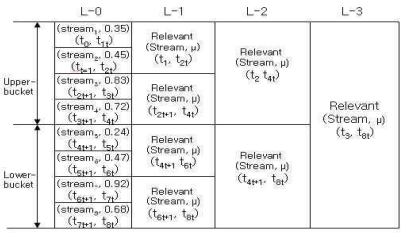

그림3. 멀티-레벨 스트림 구조
Fig. 3. Multi-level Stream Structure

그림3에서 L-0은 시작 레벨이며, L-1은 레벨-1, L-2는 레벨-2, L-3은 레벨-3를 의미한다. 레벨-0의 입력 값은 먼저 가상 버퍼 캐시에 저장된다. 가상 버퍼 캐시에 저장된 세그먼트 스트림은 적합성과 우선순위를 결정하기 위해 퍼지 값 μ가 적용된다. 일반적으로 L-0은 스트림의 순서가 정해져 있지 않은 스트림 패킷들로서 버스트의 크기가 랜덤하게 구성된 스트림들이다. 즉 L-0은 스트림 적합도가 수행되기 전의 세그먼트 스트림들이다. 이후 스트림 적합도가 완료되었을 때 L-3은 세그먼트 스트림들이 가장 최적화 된 스트림들로 구성된다. 그리고 L-3는 세그먼트의 버스트 연산이 가장 작은 세그먼트 스트림들로서 우선순위가 가장 높은 스트림들이다. L-3 스트림들은 L-0에서부터 이동된 값들이며, 각 레벨은 스트림들 중 가장 최근의 레벨 스트림 퍼지 값μ를 유지한다. 새로운 스트림이 레벨 L-0에 입력되면 레벨 L에서의 퍼지 값 μ는 최근의 스트림과 병합하여 새로운 레벨 (L+1)을 생성한다. 그리고 난 후 레벨 (L+1)의 끝 시간 t_e^{L+1}을 갱신한다. 각 레벨에서 이러한 과정을 반복하여 적합도가 가장 큰 최근의 레벨 순으로 스트리밍을 수행하게 된다.

3.2 스트림 적합성 척도

최적의 스트림 적합성 척도 수행은 가상 버퍼 캐시에 저장된 세그먼트 스트림들로부터 시작되며, 각 스트림에 할당된 퍼지 값 μ를 이용한다. 본 논문에서 제안된 적합성은 프록시 서버의 버퍼 캐싱에서 발생하는 혼잡, 지연 등으로 인한 패킷 손실을 최소화하고, 끊김, 재전송 등으로 인한 스트림 간섭을 최소화하기 위한 것이다.

적합성을 수행하기 위해 시스템은 멀티-레벨 스트림 구조에서 질의를 통하여 상위 레벨과 하위 레벨로의 탐색을 수행한다. 상위 레벨의 조건을 만족하는 스트림들은 퍼지 적합성이 완료된 스트림들로서 버스트 연산이 짧은 스트림들이며, 이 스트림들은 스트리밍의 요구조건을 만족하기 때문에 혼잡 및 지연으로 인한 패킷 손실이 적게 발생한다. 이에 반해서 하위 계층의 스트림들 특히 L-0의 스트림들은 적합도가 수행되기 이전의 스트림들로서 이 스트림들은 버스트 연산이 긴 스트림들이기 때문에 대역폭 제약과 단말 지연 등의 문제가 발생할 수 있다. 이러한 문제를 줄이기 위해서 하위 레벨의 스트림은 상위 레벨의 스트림들에 비해 명세적이어야 한다. 그러나 기존의 스트리밍 기법들 [3][13][14]는 하위 레벨의 스트림들이 명세적이지 못하기 때문에 혼잡 및 단말 지연으로 인한 패킷 손실과 재전송의 문제가 발생된다. 우리는 먼저 각 레벨 L에서 패킷 손실과 재전송의 문제를 최소화하기 위해 스트림의 적합성을 검토한다. 적합성 검토는 스트리밍을 보다 안정적으로 유지해 주며 또한 성능을 향상시키기 위한 것이다. 이에 따른 적합성 검토는 다음과 같다.

(정의1) 각 레벨 L에서 스트림 SB_i의 스트림 s_j의 적합성을 수행하기 위해 j번째의 스트림 세그먼트의 적합성은 다음과 같이 표현한다.

가상 버퍼 캐시에서의 스트림 적합성은 $\{(s_a(L_1),|\mu_1|), (s_a(L_2),|\mu_2|), \ldots, ((s_a(L_j),|\mu_j|)\}$이다. 여기서 L_j는 레벨 L의 SB_i에서 j번째의 스트림이다.

그리고 각 레벨 L의 스트림들에 대한 적합성을 검토한 후에는 적합성에 따른 비용평가를 수행한다. 각 레벨 L에서 스트림들에 대한 전체 비용은 개별적인 스트림들의 비용에 의해 측정되며, 레벨 L에서 탐색된 스트림들에 대한 적합성 비용은

다음과 같다.

(정의2) 레벨 L의 SBi에서 탐색된 스트림의 적합성 비용은 $SR_{cost}SB_i(s_a)=\sum_{i=1}^{N}\omega_i CostSB_i(s_a)/N$이다. 여기서 N는 SB_i에서 탐색된 스트림의 수이고, ω_i는 스트림 s_i에 대한 비용 가중치이다.

스트림 적합성 비용은 스트림 세그먼트의 성능 향상을 위한 파라미터이며, 비용이 적을수록 스트림의 성능은 향상된다. 또한 적합성 비용을 측정한 후에는 또 다른 적합성 척도로서 스트림 쌍들에 대한 관련성척도를 수행한다. 우리는 각 레벨 L의 스트림 버켓에서 두 서브스트림 $stream_1$과 $stream_2$ 쌍 사이의 관련성 척도 SR(x)를 다음과 같이 정의한다.

(정의3) 레벨 L의 스트림 버켓의 두 서브스트림 $stream_1$과 $stream_2$ 사이의 관련성척도 $SR(x)=\{\sum_k(|stream_{2k}-stream_{2k-1}|)2\times\alpha\times\beta\times|L_d|)\}$이다.

여기서 α는 스트림들 간의 유사도 이며, β는 유사도에 의한 가중치이다. 그라고 L_d는 거리 척도를 수행하는 스트림 쌍들의 수이다.

각 레벨 L에서 적합성 비용측정과 관련성 측정을 완료한 후에는 이들 조건을 만족한 스트림 쌍들에 대한 적합도 판정을 수행한다. 이때 스트림 쌍들에서 퍼지 값 μ을 비교한 후 μ가 최대인 스트림을 최상의 세그먼트 스트림으로 선정한다. 이와 같이 각 레벨에서 최상의 세그먼트 스트림을 선정하기 위한 적합성 척도 $HR_L(x)$는 다음과 같다.

(정의4) 각 레벨 L에서 최상의 세그먼트 스트림을 선정하기 위한 적합성 척도는 $HR_L(x)=max\{pairs((stream_1, \mu), (stream_2, \mu))\}$이다.

3.3 버퍼 관리

스트림 적합성이 수행된 세그먼트는 스트리밍을 최적화하기 위해 전송율에 따른 버퍼를 제어한다. 프록시에서 버퍼 캐시 제어는 적합성이 측정된 세그먼트 스트림을 명세화하기 위한 것이며, 지연시간과 패킷 손실을 줄이기 위한 것이다. 스트림 적합성이 결정된 스트림 데이터를 전송율에 따라 관리하기 위한 버퍼 캐시 구조는 그림4와 같다.

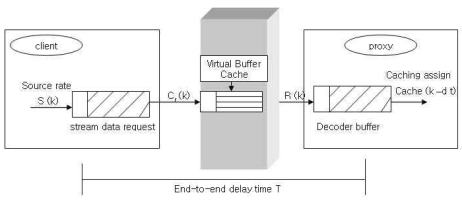

그림4. 프록시 버퍼 캐시 구조
Fig. 4. Proxy Buffer Cache Structure

프록시 버퍼 캐시 구조에서 적합성 척도가 수행된 스트림 세그먼트 k가 가상 버퍼 캐시에 입력될 때 $E_b(k)$는 인코더 버퍼, $D_b(k)$는 디코더 버퍼, $V_b(k)$는 가상 버퍼라 하자. S(n)는 n 번째 세그먼트 크기, $C_r(k)$는 클라이언트 측에서 요청된 데이터라 하고, R(k)는 수신측에서 실제 수신된 데이터라 하자. 프록시에서 스트리밍이 수행될 때 지연 및 재전송이 발생하지 않으면 패킷 손실은 발생하지 않는다. 그러나 스트리밍이 수행될 때 지연 및 재전송이 발생하면 버퍼 캐시에서는 스트림 패킷 손실이 발생되며, 패킷 손실 제어를 위한 인코딩 버퍼 제어, 디코딩 버퍼 제어, 그리고 가상 버퍼 제어는 식(1), 식(2), 식(3)과 같다.

$$E_b(k)=E_b(k-1)+S(k)-C_r(k) \quad \cdots\cdots (1)$$

$$D_b(k)=D_b(k-1)+R(k)-Cache(k-dt) \quad \cdots\cdots (2)$$

$$V_b(k)=V_b(k-1)+C_r(k)-R(k) \quad \cdots\cdots (3)$$

여기서 dt는 캐시에서 세그먼트 크기에 따른 지연시간이다.

그리고 임의의 시간 t에서 스트림 세그먼트 k를 디코딩 할 때 디코더 버퍼에서의 버퍼 제어를 위한 지연 측정은 식(4), 식(5)와 같다.

$$D_b(k+dt)=\sum_{i=k+1}^{k+dt}Cache(i)-E_b(k) \quad \cdots\cdots (4)$$

여기서 i는 캐시용량이다.

이때 디코더 버퍼에서 패킷 손실을 줄이기 위해 퍼지 값과 스트림 적합성을 고려하여 인코더 버퍼를 제어하면 디코더 버퍼에서 발생하는 오버플로우로 인한 패킷 손실과 언더플로우로 인한 지터 지연을 줄일 수 있다. 그러나 퍼지 값과 스트림 적합성이 고려되지 않으면 멀티-레벨 스트리밍은 오버플로우와 언더플로우로 인한 패킷 손실과 지터 지연이 발생하게 된다. 이것은 세그먼트 크기에 따른 버퍼에 제약을 받기 때문이다. 특히 멀티-레벨 스트리밍에서 세그먼트 크기는 인코더 버퍼, 디코더 버퍼, 그리고 전송율에 영향을 받으며 이에 따라 오버헤드 또한 인코더 버퍼, 디코더 버퍼, 그리고 전송율에 제약을 받게 된다. 따라서 인코더 버퍼, 디코더 버퍼, 전송율 제약에 따른 세그먼트 스트림 제약 조건은 식(5)와 같이 결정된다.

$$\min\Big(\sum_{i=k+1}^{k+\delta}Cache(i)-V_b(k)-E_b(\delta)\Big)\le SS\le \max\Big(\sum_{i=k+1}^{k+\delta}Cache(i)-V_b(k)-D_b(\delta)\Big) \quad \cdots\cdots (5)$$

여기서 $E_b(\delta)$은 인코더 버퍼 크기, SS는 세그먼트 스트리밍, $D_b(\delta)$은 디코더 버퍼의 크기이다.

그리고 인코더의 버퍼크기와 디코더의 버퍼 크기에 따른 전송율 제어는 식 (6)과 같이 수행된다.

$$\sum_{i=k+1}^{k+\delta} Cache(i) \le E_b(\delta) + D_b(\delta) + V_b(k) \quad \cdots\cdots (6)$$

식 (6)에서 버퍼 캐시에서 처리할 수 있는 용량보다 큰 세그먼트 스트림이 전송되면 디코더 버퍼는 오버플로우가 발생되고, 너무 작은 세그먼트가 전송되면 언더플로우가 발생된다. 이와 같은 오버플로우와 언더플로우 문제를 해결하기 위하여 가상 버퍼 $V_b(k)$를 이용하였다. 따라서 본 논문에서 버퍼 캐시 제어는 버퍼 큐에 저장된 세그먼트들을 연속적으로 피드백하여 수행하며, 버퍼 캐시로부터 피드백이 수행될 때 가상 버퍼 $V_b(k)$는 식 (7)과 같이 갱신된다.

$$V_b(K) = D_{data\_s}(K-1) - E_{data\_s}(K-1) - t_{uplink} \times \frac{TR_i(K-1)}{T} \quad \cdots\cdots (7)$$

여기서 $D_{data_s}(K-1)$는 디코딩 버퍼에서의 데이터 처리량, $E_{data_s}(K-1)$는 인코딩 버퍼에서 디코딩 버퍼로의 데이터 수신량, 그리고 t_{uplink}는 업링크 지연이다. K는 피드백이 수행되는 세그먼트, TR_i는 전송율, 그리고 T는 스트리밍이 수행되는 전체 시간이다.

버퍼 캐시 제어 과정에서 만일 $TR_i(K)$가 $TR_i(K) \times N >$ Eb(N)+Db(N)+Vb(k)이면 결과적으로 전송 지연이 발생하며, 이로 인해 스트리밍 재전송 문제가 발생한다. 이러한 문제를 해결하기 위하여 스트리밍을 수행할 이전 세그먼트와 새로운 세그먼트를 비교하여 버퍼 캐시를 제어하며, 히트율을 고려한 버퍼 캐시 제어는 식 (8)과 같다.

$$B_{cache}(i) = (TR_i(K)_{\neq w} - TR_i(K)_{before}) \times L(K) \quad \cdots\cdots (8)$$

여기서 $L(K)$는 버퍼 캐시에서 제어할 전체 세그먼트의 길이이다.

Ⅳ. 시뮬레이션 평가

4.1 시뮬레이션 환경

제안된 기법의 성능을 시뮬레이션 하기 위하여 본 논문에서는 이벤트 기반 기법을 이용하여 성능을 시뮬레이션 하였다. 비트율은 1.32Mbps, 스트리밍을 위한 세그먼트는 2,500개의 이미지 데이터를 이용하였다. 시뮬레이션에서 스트리밍을 위한 패킷크기는 512Kbyte, 링크 대역폭은 10/100Mbps, 평균 링크 대역폭은 약 1.4Mbps로 설정하였다. 시뮬레이션을 위해 스트림의 ts는 [1, 20s], 그리고 비용 가중치와 퍼지 값은 각각 $\omega \le 0.5$, $\mu \ge 0.5$로 설정하였다. ω는 적합성 율에 의해서 분석된 가중치이다. 본 논문에서는 제안된 기법의 성능을 알아보기 위하여 고정길이 기법, 피라미드 기법, 스카이 스크랩퍼 기법과의 성능을 평가 하였다. 제안된 기법에서 초기 L-0은 멀티-레벨 스트림 구조의 성능에 중요한 영향을 미치기 때문에 최상의 멀티-레벨 스트림 구조가 구축되어 있다고 가정하였다. 시뮬레이션을 위해 스트림 데이터의 수를 300, 700, 1000, 1200, 1400, 1600, 1900, 2200, 2500으로 점차 증가시켰으며, 실험을 5회 반복하였다. 그리고 난 후 우리는 스트림 레벨, 블록캐싱 크기, 패킷 데이터의 크기를 고려하여 적합성을 적용하였다.

4.2 시뮬레이션 결과

성능 평가를 위해서 본 논문에서는 캐시용량, 스트림 데이터 수, 세그먼트 크기 그리고 ω와 μ를 변화시켜 가면서 시뮬레이션을 수행하였다. 시뮬레이션에서 사용된 주요 성능 척도는 버퍼 캐시 제어율, 평균 패킷 손실율, 그리고 지연 절약율이다. 캐시 제어율이란 적합도를 만족하는 스트림 데이터들이 프록시 버퍼 캐시에 얼마나 만족되는지를 나타내는 비율이며, 평균 패킷 손실율이란 스트림 데이터들이 버퍼 캐시에서 히트되지 않고 손실이 발생하는 비율을 말한다. 그리고 지연 절약율은 버퍼 캐시에서 스트림 데이터들이 히트되어 대기 지연이 적게 발생하는 비율을 말한다. 본 논문에서는 이들 성능 척도에 기반을 두고서 시뮬레이션을 수행하였으며, 성능 비교는 고정길이 세그먼트 기법, 피라미드 기법, 스카이 스크랩퍼 기법, 그리고 제안된 기법으로 구분하여 평가하였다.

4.2.1 적합성 척도에 따른 버퍼 캐시 제어율

첫 번째 성능평가는 적합성 척도에 따른 버퍼 캐시 제어율이다. 스트림 적합성 척도 μ가 0.5 이하일 때는 버퍼 캐시 제어율에 영향을 미치게 된다. 이에 대한 시뮬레이션 결과는 그림5와 같다.

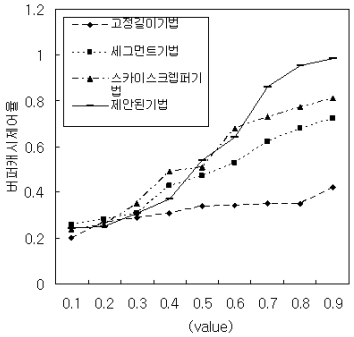

그림5. 버퍼캐시 제어율
Fig. 5. Buffer Cache Control Rate

그림5에서 보듯이 제안된 기법은 비교적 우수한 스카이스

크렙퍼 기법보다 버퍼 캐시 제어성능이 향상되었음을 알 수 가 있다. 이것은 제안된 기법에서는 스트림 데이터들에 대해서 적합도 척도를 수행했기 때문이다. 따라서 스트림 데이터의 크기를 버퍼 캐시에 적합하도록 분할 할 때 버퍼 캐시가 효율적으로 제어됨을 알 수 있다. 그러나 적합성 척도 관점에서 볼 때 μ값은 버퍼 캐시에 영향을 미치게 된다. 만일 μ값이 μ<0.8일 때는 콘텐츠 서버로부터의 패치 지연이 크게 발생하기 때문에 버퍼캐시의 성능에 영향을 미치게 된다. 본 논문에서는 스트림 데이터가 너무 작거나 캐시 용량을 초과하는 미디어 스트림 데이터들에 대해서는 μ를 적용하여 필터링이 수행되도록 하였다.

4.2.2 스트림 데이터 수에 따른 평균 패킷 손실율

두 번째 성능평가는 $\mu \geq 0.8$로 하여 스트림 데이터 수를 300, 700, 1000, 1200, 1400, 1600, 1900, 2200, 2500으로 점차 증가시켜 가면서 평균 패킷 손실율을 평가하였다. 스트림 데이터 수에 따른 시뮬레이션 결과는 그림6과 같다.

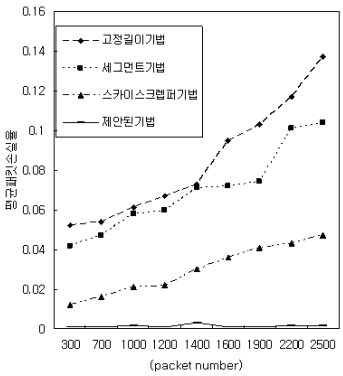

그림6. 평균패킷 손실율
Fig. 6. Average Packet Loss Rate

그림6에서 보듯이 스트림 데이터 수가 증가할 때 제안된 기법의 패킷 손실율은 거의 변화가 없음을 알 수 있다. 이것은 μ가 데이터 스트림에 영향을 미치기 때문이다. 결과적으로 스트림 데이터 수는 μ값에 영향을 받는 다는 것을 알 수 있다. 즉 μ값이 작으면 스트림 데이터들에 대한 참조가 분산되기 때문에 지연으로 인한 패킷 손실이 크게 발생한다. 따라서 제안된 기법은 각 스트림 데이터에 대한 적합성 척도가 반영되었기 때문에 데이터 스트림 수가 증가할 때도 다른 기법들에 비해서 성능이 향상되게 된다.

4.2.3 캐시 용량에 따른 지연절약율

세 번째 성능평가는 캐시 용량에 따른 지연 절약율이다. 지연 절약율 평가를 위해 μ값을 변화시켜가면서 제안된 캐시 교체 알고리즘의 성능을 분석하였다. 이에 대한 분석 결과는 그림7과 같다.

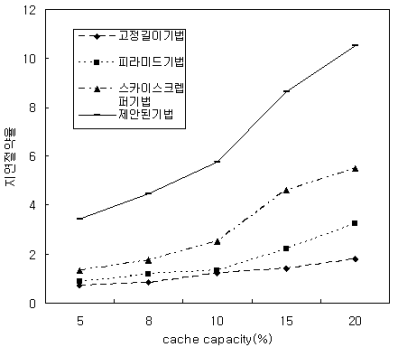

그림7. 캐시 용량에 따른 지연절약율
Fig. 7. Delay Saving Rate by Cache Capacity

그림7에서 보듯이 제안된 기법은 지연 절약율 관점에서 볼 때 다른 기법들에 비해서 성능이 우수함을 알 수 있다. 특히 비교적 성능이 우수한 스크이 스크렙퍼 기법과 비교해 볼 때 약 5%의 성능이 향상되었음을 알 수 있다. 고정길이 기법과 피라미드 기법의 성능이 상대적으로 낮은 이유는 스트림 데이터의 크기 및 캐시 용량에 따른 적합성 척도를 고려하지 않았기 때문이다. 특히 멀티미디어 객체는 웹 객체와는 달리 프록시 버퍼에서 지연 문제가 더 크게 발생된다. 그러나 제안된 기법은 적합성 척도를 반영하였기 때문에 프록시 버퍼 캐시에서 크기가 큰 스트림 데이터가 새로 입력되거나 버퍼 캐싱이 수행될 때 지연 절약을 효율적으로 수행하게 된다.

V. 결론

최근에 미디어 콘텐츠 서버를 거치지 않고 클라이언트 근처에서 스트리밍 서비스를 빠르게 지원하기 위한 프록시 서버 기법에 대한 많은 연구가 제안되고 있다. 그러나 프록시 서버는 스트림 데이터의 크기, 버퍼 캐시 용량 제약 등으로 인하여 멀티-레벨 스트리밍 서비스에는 많은 문제점이 나타나고 있다.

본 논문에서는 이러한 문제를 개선하기 위하여 세그먼트 기반의 버퍼 관리 메카니즘을 제안하였다. 제안된 기법은 퍼지값μ와 비용 가중치 ω를 이용하여 적합성 척도를 수행하였으며, 적합성 척도에 따라 멀티-레벨 스트리밍서비스가 가능하도록 하였다. 멀티-레벨 스트리밍을 위해 가상 버퍼 캐시를 구성한 후 버퍼 캐시에 저장된 각 스트림들에 대해서 적합성 척도를 수행한 후 이들 스트림들을 레벨-업 시켰다. 그리고 난 후 비용측정, 적합성 측정 및 스트림 페어(pairs)를 선정하여 최적

의 스트림이 되도록 하였다. 이렇게 선정된 스트림들은 버퍼 캐시에 할당되어 버퍼 캐시, 패킷 손실, 그리고 지연 등을 효율적으로 제어함을 알 수가 있었다. 따라서 시뮬레이션 결과 제안된 기법이 기존의 고정길이 기법, 피라미드 기법, 그리고 스카이스크랩퍼 기법에 비해서 버퍼 캐시 제어율, 평균 패킷 손실율, 지연 절약율의 성능이 우수함을 알 수 있었다.

참고문헌

[1] C. Chang and M. S. Chen, "On Exploring Aggregate Effect for Efficient Cache Replacement in Transcoding Proxies," IEEE TRANSACTIONS ON PARELLEL DISTRIBUTED SYSTEM, VOL. 14, NO. 6, pp. 611-624, 2003.

[2] C. F. Kao and C. N. Lee, "Aggregate Profit-Based Caching Replacement Algorithms for Streaming Media Transcoding Proxy Systems," IEEE TRANSACTIONS ON MULTIMEDIA, VOL. 9, NO. 2, pp. 221-230, 2007.

[3] S. Chen, B. Shen, S. Wee, and X. Zhang, "Segment-Based Streaming Media Proxy: Modeling and Optimization," IEEE TRANSACTIONS ON MULTI MEDIA, VOL. 8, NO. 2, pp. 243-256, 2006.

[4] B. Shen, S. J. Lee, S. Basu, "Caching Strategies in Transcoding-Enabled Proxy Systems for Streaming Media Distribution Networks," IEEE TRANSACTIONS ON MULTIMEDIA, VOL. 6, NO. 2, pp. 375-386, 2004.

[5] K. T. Atanassov, "Intuitionistic Fuzzy Sets", Fuzzy Sets and Systems, vol. 20, pp. 87-96. 1986,

[6] D. Li, C. N. Chuah, G. Cheung and S. J. B. Yoo, "MUVIS : Multi-source Video Streaming Service over WLANs", Journal of Communication and Networks(JCN), vol.7, pp. 144-156, 2005.

[7] M. Qin and R. Zimmermann, "An Adaptive Strategy for Mobile Ad Hoc Media Streaming", IEEE Transactions on Multimedia, vol.12, no.4, pp.317-329, 2010.

[8] T. H. Hsu and Y. H. Li, "A Dynamic Cache Scheme for Multimedia Streams on Heterogeneous Networking Environments" 2009 3th International conference on Multimedia and Ubiquitous Engineering" pp. 91- 98, 2009.

[9] K. C. Chang, T. F. Chen, "Efficient Segment-based Video Transcoding Proxy for Mobile Multimedia Services," Journal of Systems Architecture 53, pp. 833-845, 2007.

[10] A. Maheshwari, A. Sharma, K. Ramamrithan, and P. Shenoy, "Transquid: Transcoding and Caching Proxy for Heterogeneous e-commerce Environments," in Proc. IEEE INFOCOM 2002, San Jose, CA, pp. 50-59, 2002.

[11] 이종득 "분산 멀티미디어 스트리밍 서비스를 위한 분할과 사상에 의한 프록시 캐싱 그룹화," 한국지능시스템학회논문지, 제 19권, 제 1호, 40-47쪽, 2009년 2월.

[12] 김진성,김동일,김은삼,배성일 "푸시-메시 구조 기반의 효율적인 피어투피어 스트리밍 기법," 한국컴퓨터정보학회논문지, 제 15권, 제 3호, 82-89쪽, 2010년 3월.

[13] K. L. Wu, S. Yu, and J. L. Wolf, "Segmentation of Multimedia Streams for Proxy Caching", IEEE Transactions on Multimedia, vol. 6, no. 5, pp. 770-780, 2004.

[14] J. Z. Wang and P. S. Yu, "Fragmental Proxy Caching for Streaming Multimedia Objects", IEEE Transactions on Multimedia, vol. 9, no. 1, pp. 147-156, 2007.

[15] M. Qin and R. Zimmermann, "Supporting Guaranteed Continuous Media Streaming Performance through Adaptive Layer Selection with Scalable Video Coding", in Proc. 15thAnnu. ACM International Conference on Multimedia, pp.23-29, 2007.

저 자 소 개

이 종 득

1983년 2월 : 전북대학교 컴퓨터과학과(이학사)

1989년 2월 : 전북대학교 컴퓨터과학과(이학석사)

1998년 2월 : 전북대학교 컴퓨터과학과(이학박사)

1992년 3월~2002년 2월 : 서남대학교 컴퓨터통신학과 교수

2002년 2월~2010 9월 현재 : 전북대학교 전자공학부 교수

관심분야 : 무선 모바일 네트워크, 무선 센서 네트워크, 무선 모바일 애드혹 네트워크, 유비쿼터스 통신 등