한국지능시스템학회 논문지 2009, Vol. 19, No. 1, pp. 40-47

분산 멀티미디어 스트리밍 서비스를 위한 분할과 사상에 의한 프록시 캐싱 그룹화

Proxy Caching Grouping by Partition and Mapping for Distributed Multimedia Streaming Service

이종득

Chong Deuk Lee

전북대학교 응용시스템공학부 전자통신공학과

요 약

최근에 분산 환경에서 서버를 거치지 않고 사용자가 요구하는 미디어 객체를 프록시로부터 직접 서비스하기 위한 동적 프록시 캐싱 기법이 제안되고 있다. 그러나 이러한 기법들은 멀티미디어 데이터 크기, 지연시간, 연속적인 미디어 객체 서비스 요구 등으로 인하여 여러 문제점들이 제기되고 있다. 본 논문에서는 이러한 문제를 해결하기 위하여 분할과 사상기반의 퍼지필터링에 의한 캐싱 그룹화 기법을 제안하였다. 분할과 사상을 위해 미디어 블록세그먼트를 고정분할 참조블록(RfP)와 가변분할 참조블록(RvP)로 구분하였으며, 시멘틱 관계성을 위해 고정분할 시간동기화(Tf)와 가변분할 시간동기화(Tv)에 의해 퍼지 관계성이 수행되도록 하였다. 시뮬레이션 결과 제안된 기법이 평균 요청응답률과 캐시 히트율을 증가시키고 시간지연율을 줄이며 스트리밍 서비스를 효율적으로 수행하게 됨을 알 수 있다.

키워드 : 미디어객체 서비스, 퍼지 필터링, 블록세그먼트, 고정분할참조블록, 가변분할참조블록

Abstract

Recently, dynamic proxy caching has been proposed on the distributed environment so that media objects by user's requests can be served directly from the proxy without contacting the server. However, it makes caching challenging due to multimedia large sizes, low latency and continuous streaming demands of media objects. To solve the problems caused by streaming demands of media objects, this paper has been proposed the grouping scheme with fuzzy filtering based on partition and mapping. For partition and mapping, this paper divides media block segments into fixed partition reference block(R_fP) and variable partition reference block(R_vP). For semantic relationship, it makes fuzzy relationship to performs according to the fixed partition temporal synchronization(T_f) and variable partition temporal synchronization(T_v). Simulation results show that the proposed scheme makes streaming service efficiently with a high average request response time rate and cache hit rate and with a low delayed startup ratio compared with other schemes.

Key Words : media object service, fuzzy filtering, block segment, fixed partition reference block, variable partition reference block

1. 서 론

최근에 무선 서비스 및 분산 네트워크 기반의 멀티미디어 응용 서비스 요구가 증가됨에 따라 멀티미디어 서비스를 사용자 위주로 스트리밍하기 위한 여러 가지 기법들이 제안되고 있다. 멀티미디어 스트리밍 서비스를 위해서는 끊김현상을 방지하고 미디어 데이터의 연속적인 전송 서비스를 보장하는 스트리밍 자원 관리, 서비스가 시작된 시작시간부터 평균대기시간을 최소화하는 채널관리, 한정된 저장 공간을 관리하고 사용하기 위한 효율적인 저장 공간, 그리고 접근 지연시간과 네트워크 대역폭을 최소화하기 위한 기능 등이 보장되어야 한다. 이와 같은 기능을 위해 최근에는 분산 환경에서 사용자 서비스 위주의 정적 미디어 객체들을 스트리밍하기 위한 기법으로 프록시 캐시 기법이 이용되고 있으며[1, 2, 3], 이 기법은 서버에 접속하지 않고서도 서비스 데이터 블록을 프록시 서버에 유지하여 미디어 객체들을 연속적으로 스트리밍을 수행하는 기법이다. 그러나 분산 환경에서 응용 멀티미디어 스트리밍을 위해서는 서비스 객체들에 대한 지터(jitter) 최소화, 시간지연 감소, 그리고 전송 히트율 감소 등과 같은 여러 가지 문제점들이 고려되어야 한다. 이러한 문제를 개선하기 위해 세그먼트 기반의 프록시 캐싱 기법이 제안되고 있지만[4-7] 이 기법은 미디어 객체들을 부분적으로 캐싱하여 스트리밍을 수행하기 때문에 분류시간에 따른 스트리밍지연 시간이 발생되는 문제점을 가지고 있다. 일반적으로 분산 멀티미디어 서비스를 위한 기법으로는 CMD(Centralized Multimedia Distribution), 클라이언트-서버 구조화 기법(Client-Server Architecture), CDN(Content Distribution Network), P2P(Peer to Peer) 기반의 서비스 기법 등이 주로 이용되고 있다[8]. CMD 기

접수일자 : 2008년 4월 24일
완료일자 : 2008년 8월 20일

법은 웹 기반 분산 서비스 환경에서 서버 미러링 을 이용하여 저장 공간과 I/O 용량을 효율화하기 위한 기법으로서 중앙 서버의 서비스 기능을 개선하기 위한 기법이다. 그러나 이 기법은 분산 서비스의 성능에 중요한 영향을 미치는 병목 현상을 줄일 수 없다는 문제점이 제기되고 있다. 클라이언트 구조화 기법은 배칭(batching), 패칭(patching), 그리고 주기적인 방송 기법(periodic broadcasting)에 기반을 둔 기법으로서 이 기법은 IP 네트워크상의 멀티캐스트 기능 부족으로 인하여 구현이 어려우며, 대규모의 스트리밍 서비스에는 적합하지 않는 문제점을 가지고 있다. CDN 기법은 인터넷상의 CDN 서버의 수에 기반을 둔 기법으로서 스트리밍 서비스를 수행할 멀티미디어 콘텐트가 먼저 이들 서버에 분산되어 있어야 하고 이들 각 서버를 통하여 이웃한 클라이언트에 콘텐트가 전송될 수 있어야 한다. 이 기법은 광역의 전송 구조에는 많은 비용이 투자되어야 하며 멀티미디어 서비스를 수행하기 위해서는 많은 자원이 요구된다는 문제점을 가지고 있다. P2P기반의 네트워크 서비스 기법은 Napster, Gnutella, FreeNet, CenterSpan, Vtrails와 같은 시스템에서 사용되는 기법[8]로서 피어는(peer)는 다른 피어 그룹의 데이터를 공유하며 이웃 피어나 디렉토리 서버에 질의를 통하여 원하는 데이터를 탐색하여 서비스를 수행하는 기법이다. 그러나 이 기법은 적은 비용으로 많은 커뮤니티들의 데이터를 공유할 수 있는 기능은 제공하지만 QoS, 데이터 대역폭, 느린 이동성 등과 같은 문제점이 제기되고 있다[9, 10]. 분산 멀티미디어 응용 환경에서 클라이언트가 요청한 QoS 및 평균 데이터 전송률은 전송 효율에 중요한 영향을 미치게 되며, 프록시 지터에 영향을 주어 클라이언트들에게 사용자 위주의 스트리밍 데이터 전송 서비스를 보장하게 된다. 그러나 QoS를 위해서는 데이터 전송률을 위한 캐시 히트율과 평균 대기시간이 고려되어야 하며 스트리밍 서비스 성능을 고려할 때 일반적으로 캐시 히트율이 증가되면 시간 지연률 즉 평균 대기시간이 떨어지게 되고, 시간 지연률이 줄어들면 캐시 히트율이 떨어지는 문제가 발생되고 있다[11-12].

따라서 본 논문에서는 캐시 히트율과 지연 지연률 등을 고려하여 스트리밍의 QoS를 보장하기 위한 퍼지 필터링 기반의 프록시 캐시 그룹화 기법을 제안하고자 한다. 제안된 기법은 퍼지 필터링에 의해 캐시 될 객체 프로파일과 캐시되지 않을 객체 프로파일을 자동으로 식별하여 식별된 객체 프로파일에 대해서 스트림을 수행하게 된다. 제안된 그룹화 기능은 프록시 서버로 하여금 스트리밍을 수행할 캐시 데이터를 반복적으로 참조하는 참조기능을 줄여 QoS 성능을 보장하게 되며, 이렇게 보장된 QoS에 의해 네트워크 대역폭 및 지터, 평균대기 시간 등의 개선효과를 가져오게 된다. 본 논문은 다음과 같다. 2장에서는 관련연구에 대해서 기술하고, 3장에서는 제안된 기법인 퍼지 필터링 기반의 프록시 캐시 그룹화 기법에 대해서 기술한다. 그리고 4장에서는 제안된 기법의 성능을 비교분석하며, 끝으로 결론 및 향후 연구방향에 대해 기술한다.

2. 관련연구

분산 멀티미디어 응용환경에서 네트워크 대역폭, 지터 개선, 적시 스트리밍(Timely Streaming), 신뢰성 있는 정보 전송 등의 보장을 위해 프록시 캐싱 기법에 대한 많은 연구가 수행되고 있다. 프록시 캐싱을 위한 기법으로는 미디어 객체들을 스트리밍하기 위해 사용되는 세그먼트 기반의 프록시 캐싱 기법[2, 7], 초기에 미디어 객체들을 세그먼트 하기 위해 사용한 전치 캐싱 기법(Prefix Caching)[3, 7], 세그먼트들이 접근되기 전에 캐시 되지 않은 세그먼트들을 패치(fetch)함으로서 프록시 지터를 줄이기 위한 프리패칭(pre-fetching) 기법[6, 12], 그리고 같은 네트워크 구조에 있는 이웃한 호스트와 피어들의 응용 그룹을 클러스터하기 위한 네트워크 오버레이 기법[13] 등이 있다. 세그먼트 기반의 프록시 캐싱 기법은 스트리밍을 위한 세그먼트들을 전체적으로 캐싱하는 기법이 아니라 부분적으로 캐싱하는 기법이다. 전치캐싱(Prefix Caching)기법은 세그먼트를 전치와 후치(suffix)로 구분하여 캐싱을 하는 기법으로서 클라이언트의 접근 패턴에 따라 세그먼트 길이를 일정한 크기로 분할하여 캐싱을 수행하는 기법이다. 그러나 이들 기법은 일정한 크기로 인해 히트율은 증가시킬 수 있으나 동적 세그먼트들을 고려하지 않음으로 인하여 캐싱 효율성이 떨어지는 문제점이 제기되고 있다. 프리패칭 기법은 프록시 지터를 줄이기 위한 기법으로서 캐시 되지 않은 세그먼트를 프리패치 하는 기법이다. 그러나 이 기법은 캐시 되지 않은 세그먼트들을 프리패치 하여 수집하는 일은 많은 네트워크 트래픽을 가져오게 되며 기억공간의 낭비를 가져오게 된다. 최악의 경우에는 프리패치 된 세그먼트들이 접근되기 전에 클라이언트 세션이 종료되는 문제를 가져오게 한다. 그리고 네트워크 오버레이 기법은 하나의 그룹 안에 있는 피어들의 관련성을 이용하여 QoS를 향상시키기 위한 기법으로서 다른 피어들과의 서로 협력적 관계를 통하여 서버 부담을 줄이기 위한 기법이다. 그러나 이 기법은 네트워크상에 분포된 많은 피어들로 인하여 기억 공간 및 네트워크 대역폭과 같은 문제가 발생되고 있으며, 이로 인하여 클라이언트-서버 구조화 기법[14] 및 CDN 시스템[13]에 비해서 신뢰성이 떨어지는 문제점이 발생되고 있다. 이 외에도 [9]는 비디오 스테이징을 이용한 캐싱기법을 제안하였으며 이 기법은 임계값을 이용한 기법으로서 임계값보다 큰 프레임을 추출하여 캐싱을 수행하는 기법이다. 이 기법은 대역폭을 줄여 프록시에 대한 캐싱을 향상시키기 위한 기법이지만 프레임의 수가 증가되거나 프레임에 대한 프로파일 정보가 제공되지 않을 경우 히트율이 떨어지는 문제점이 제기되고 있다. 그리고 [1, 5, 7]에서는 혼잡 제어 메커니즘(Congestion Control Mechanism)을 이용한 캐싱 기법을 제안하였으며 이 기법은 캐시 되지 않은 레이어 미디어 프레임을 프리 패치하는 기법으로서 캐시 된 프레임에 대해서는 프리패칭 윈도우를 적용하고 캐싱이 수행되기 전 캐싱을 수행할 프레임들이 존재하는지를 분석하여 캐싱을 수행하는 기법이다.

3. 퍼지 필터링 기반의 프록시 캐시 그룹화 구조

프록시 캐시 그룹화 구조는 사용자 위주의 스트리밍 서비스가 요청될 때 응답 시간을 향상시키고 네트워크 대역폭 및 트래픽을 줄이기 위한 기법으로서 블록 데이터의 전체를 스트리밍 하는 것이 아니라 사용자의 서비스 요구가 발생되는 블록 데이터에 대해 캐시 그룹화 인덱스를 검사하여 캐싱 상태에 따라 퍼지 필터링을 수행하여 스트리밍 서비스를 수행하는 구조이다. 그룹화 과정은 클라이언트에서 스트리밍 서비스를 수행할 미디어 블록 사상을 통하여 캐시 서비스 계층 구조를 생성하게 되며, 서비스 계층 구조에서 참조

빈도가 높은 블록 및 퍼지 필터링에 의해 필터링 된 블록 데이터들에 대해서 스트리밍 서비스를 수행하게 된다. 이 장에서는 스트리밍을 수행할 블록 데이터에 대해서 분할 사상 기법에 의한 캐시 그룹화 구조를 제안하며, 이때 서버의 부담과 클라이언트의 네트워크 부하 및 트래픽를 줄이고, 대역폭을 활성화하기 위해 분할과 사상을 참조하여 스트리밍 전략을 수행하게 된다. 스트리밍 전략을 위한 캐싱 그룹화는 블록 분할, 사상, 통합 과정을 거쳐 수행되며, 퍼지 필터링 된 참조 블록에 대한 동적인 캐싱 전략을 수행하게 된다.

3.1 블록분할

블록분할은 전체 블록을 스트림할 때 캐시 되지 않은 블록들을 언제 분할하여 패치 할 것인가를 결정하는 기법으로서 프록시 지터가 자원 사용량을 최소화하여 네트워크 대역폭과 트래픽, 시간 지연과 같은 문제를 줄이기 위한 기법이다. 프록시 지터가 자원 할당량을 최소화기 위해서는 캐싱을 위한 세그먼트 블록이 준비되어야 하고 블록 접근은 순차적이어야 한다. 그리고 프록시 캐싱 그룹화 구조-클라이언트 링크 대역폭은 충분히 확보되어야 하고 프록시에서의 클라이언트 미디어 스트리밍 서비스는 충돌이 발생되지 않아야 한다. 이때 블록분할을 위한 각각의 블록 세그먼트들은 서버로부터 패치 되며, 패치된 세그먼트들에 대해 프록시 지터를 최소화하기 위해 인코딩율과 네트워크 대역폭을 고려하여 블록 분할을 결정하게 된다. 따라서 프록시 지터를 최소화하기 위한 블록 분할 결정은 식(1)과 같고 인코딩율과 대역폭의 관계는 식(2)와 같다.

$$BP(x)=\sum_{i=1}^{n}\mathrm{BSL}_i-\frac{BSL(n+1)\times(Er-Bw)}{Bw} \quad (1)$$

$$\frac{\sum_{i=1}^{n}BSLi-BP(x)+BSL(n+1)}{Er}\geq\frac{BSL(n+1)}{Bw} \quad (2)$$

그리고 결정된 전체 블록 분할에 대한 버퍼링 결정은 식(3)과 같다.

$$\frac{\sum_{i=1}^{n}BSLi-BP(x)}{Er}\times Bw \quad (3)$$

식(1),(2),(3)에서 $BP(x)$는 임의 블록 세그먼트에 대한 블록분할, BSL_i는 i 번째의 블록 세그먼트 크기, E_r은 블록 세그먼트에 대한 평균 인코딩 율, B_w는 평균 네트워크 대역폭이다.

각 블록 분할이 결정되면 결정된 블록분할에 대해서 고정 분할 참조블록(Fixed Partition Reference Block) (R_fP)과 가변분할 참조블록(Variable Partition Reference Block)(R_vP)을 수행하여 지터를 최소화 한다.

3.1.1 고정 분할 참조 블록(R_fP)

고정 분할 참조블록은 블록 세그먼트 분할을 같은 크기로 분할하는 블록구조로서 분할을 수행할 BP(x)에 대해서 인코딩율과 대역폭을 고려해야 하며 고정 분할을 위한 참조블록 $R_fP(x)$는 다음과 같이 정의된다.

(정의1) $R_fP(x)=BSL_i(n+1)\times T_f-\frac{Er}{Bw}\times BSLi$이다.

여기서 T_f는 분산 환경에서 고정 분할 멀티미디어 스트리밍을 위한 시간 동기화이다.

그리고 고정 분할 참조블록 R_fP가 결정되면 이에 대한 버퍼 크기 $R_fP(x)(B_{uffer})$는 다음과 같이 정의 된다.

(정의2) $R_fP(x)(\mathrm{Buffer})=(T_f\times BSL_1)\times\frac{Er-Bw}{Er}$ 이다.

3.1.2 가변 분할 참조 블록(R_vP)

가변 분할 참조블록은 블록 세그먼트 분할을 서로 다른 크기로 분할하는 블록구조로서 분할을 수행할 BP(x)에 R_vP를 수행하게 되며 가변 분할을 위한 참조 블록 $R_vP(x)$는 다음과 같이 정의된다.

(정의3) $R_vP(x)=BSL_i(n+1)\times T_v-\frac{Er}{Bw}\times BSLi$ 이다.

여기서 T_v는 분산 환경에서 가변 분할 멀티미디어 스트리밍을 위한 시간 동기화이다.

그리고 고정 분할 참조블록 RvP(x)가 결정되면 이에 대한 버퍼 크기 $R_vP(x)(B_{uffer})$는 다음과 같이 정의 된다.

(정의4)$R_vP(x)(\mathrm{Buffer})=(T_v\times BSL_1)\times\frac{Er-Bw}{Er}$ 이다.

$R_fP(x)$, $R_vP(x)$에서 만일 $R_fP(x),R_vP(x)=0$이면 이 경우 세그먼트는 분할되지 않게 되며, 이때에는 미분할로 인하여 시작 지연 시간이 발생되어 프록시 지터가 발생된다. 이러한 프록시 지터를 피하기 위해 $R_fP(x),R_vP(x)=0$이 아닌 세그먼트 즉 $\lceil (E_r/B_w)\rceil^{th}$ 번째 세그먼트 즉 $R_fP(x),R_vP(x)>0$에 대해서 분할을 수행하여 지연 시간 및 프록시 지터를 최소화해야 한다.

3.2 사상

블록분할에서 $R_fP(x)$, $R_vP(x)$가 결정되면 관계성에 의해 사상을 결정하게 된다. 블록사상은 캐싱 그룹화 과정에서 참조될 블록 세그먼트와 참조되지 않을 블록 사상을 구분하기 위한 과정으로서 시멘틱 관계성에 의한 사상, $R_fP(x)$에 의한 사상, $R_vP(x)$에 의한 사상으로 구분하여 수행된다.

3.2.1 시멘틱 관계성에 의한 사상

시멘틱 관계성에 의한 사상은 $R_fP(x)$, $R_vP(x)$에 대해서 의미적 관계성이 높은 BP(x)를 집성화(aggregation)하거나 그룹화 하기 위한 사상이다. 시멘틱에 의한 사상은 (R, BP(x), FR)의 3개의 튜플로 구성되며, R은 BP(x)에 대해 의미적 관련성을 위한 관계성이고, FR은 BP(x)사이의 관련 정도를 나타내는 퍼지 관련성이다. 이때 사상을 위한 관계 R은 스트리밍을 위한 고정분할 시간동기화 T_f와 가변분할 시간동기화 T_v에 의해 수행된다.

i) T_f

T_f는 사상할 BP(x)들이 ($R_fP(x_i)$, $R_fP(x_i).T_f$)의 시간동기화 관계를 유지할 때 "temporally consists of "와 "member-of" 관계성을 가진 $R_fP(x)$로서 상위레벨과 하위레벨의 시멘틱 관계성은 다음과 같이 정의된다.

(정의5) $BP(x)=T_f(R_fP(x_1), R_fP(x_2), \ldots, R_fP(x_n)) \Leftrightarrow \forall R_fP(x_i) \in A$이고, $member\text{-}of((R_fP(x_1), R_fP(x_1).T_f) \wedge (R_fP(x_2), R_fP(x_2).T_f) \wedge, \ldots, (R_fP(x_n), R_fP(x_n).T_f))$이면 $BP(x)=\{(R_fP(x_i), R_fP(x_i).T_f), \bigcup_{i=1}^{n}(R_fP(x_i), R_fP(x_i).T_f)\}$이다.

예를 들어 그림2와 같이 VideoFrme 블록이 T_f와 T_v에 의해 동기화 되어 있다고 하면 T_f에 의한 사상은 다음과 같이 표현된다.

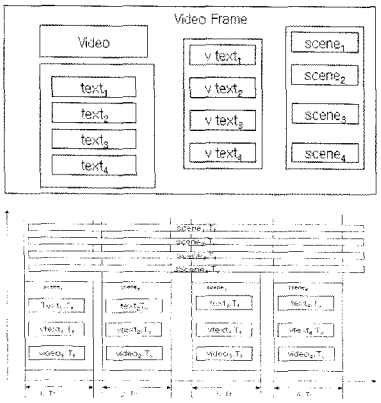

그림 1. T_f와 T_v에 의한 동기화
Fig. 1. Synchronization by T_f and T_v

$SynchroVideoFrame_i=T_f(videoFrame_i.\tau, vtext_i.\tau, text_i.\tau, scene_i.\tau) \Leftrightarrow$
$member\text{-}of((Synchroscene_1, vtext_1.T_f, trtext_1.T_f, scene_1.T_f)) \wedge$
$member\text{-}of((Synchroscene_3, vtext_3.T_f, trtext_3.T_f, scene_3.T_f)) =$
$1.T_f \cup 3.T_f$ 가 된다.

ii) Tv

T_v는 사상할 BP(x)들이 ($R_vP(x_i), R_vP(x_i).T_v$)의 시간 동기화 관계를 유지하는 $R_vP(x)$로서 상위레벨과 하위레벨의 시맨틱 관계성은 다음과 같이 정의된다.

(정의6) $BP(x)=T_v(R_vP(x_1), R_vP(x_2), \ldots, R_vP(x_n)) \Leftrightarrow \forall R_vP(x_i) \in A$이고, $member\text{-}of((R_vP(x_1), R_vP(x_1).T_v) \wedge (R_vP(x_2), R_vP(x_2).T_v) \wedge, \ldots, (R_vP(x_n), R_vP(x_n).T_v))$이면 $BP(x)=\{(R_vP(x_i), R_vP(x_i).T_v), \bigcup_{i=1}^{n}(R_vP(x_i), R_vP(x_i).T_v)\}$이다.

T_f에처럼 T_v 또한 다음과 같이 표현된다.
$SynchroVideoFrame_i=T_v(videoFrame_i.\tau, vtext_i.\tau, text_i.\tau, scene_i.\tau)$
$member\text{-}of((Synchroscene_2, vtext_2.T_v, trtext_2.T_v, scene_2.T_v)) \wedge$
$member\text{-}of((Synchroscene_4, vtext_4.T_v, trtext_4.T_v, scene_4.T_v)) = 2.T_v \cup 4.T_v$ 가 된다.

3.2.2 RfP(x)에 의한 사상

$R_fP(x)$에 의한 사상(mapping)은 분할된 BP(x)들에 대해서 의미적으로 연관관계를 가진 $R_fP(x)$들에 대해서 사상을 수행하는 기법으로서 스트리밍을 수행할 캐싱 $R_fP(x)$를 필터링하기 위한 사상이다. 이렇게 필터링 된 $R_fP(x)$는 스트리밍 수행 시 시간지연 및 지터를 줄이고 대역폭을 효율화 할 수 있으며, 사상을 위한 블록 세그먼트 도메인을 D_{omain}이라 하고, 필터링을 위한 사상은 $f_i \in F$, 사상에 의해서 필터링 된 $R_fP(x)$ 그룹을 $g_i \in G$ 라 할 때, $R_fP(x)$에 의한 사상 $M_{RfP(x)}^{filtering}$은 다음과 같이 정의된다.

(정의7) $\partial\text{-}cut_{(Rfp(x)}=\{ M_{RfP(x)}^{filtering}(BP(x)) \mid (R_fP(x_i), R_fP(x_i).T_f), \bigcup_{i=1}^{n} R_fP(x_i), R_fP(x_i).T_f) \geq \partial \}$ 이다.

여기서 ∂는 사상에 의해 퍼지 필터링으로서 $0 \leq \partial \leq 1$의 퍼지값이다.

본 논문에서는 블록 세그먼트에 대한 사상을 위해 퍼지 필터링 ∂-cut[2]를 이용하며, $M_{RfP(x)}^{filtering}$에 대한 의미적 관련성을 위해서는 다음과 같은 조건을 만족해야 한다.

(조건1) $R_fP(x)$에 대한 조상 및 이웃에 대해서 $M_{RfP(x)}^{filtering}(BP(x), R_fP(x)) \geq \partial\text{-}cut$

이와 같은 조건이 만족될 때 사상을 위한 퍼지 필터링은 0.5이상으로 설정하며, 퍼지 필터링을 만족하는 G_i는 다음과 같이 정의된다.

(정의8) $G_i = \sum_{조상및이웃한RfP(x)} M_{RfP(x)}^{filtering}(BP(x), R_fP(x)) \geq \partial\text{-}cut$이다.

3.2.3 RvP(x)에 의한 사상

$R_vP(x)$에 의한 사상(mapping)은 분할된 BP(x)들에 대해서 의미적으로 연관관계를 가진 $R_vP(x)$들에 대해서 사상을 수행하는 기법으로서 스트리밍을 수행할 캐싱 $R_vP(x)$를 필터링하기 위한 사상이다. $R_vP(x)$에 의한 사상은 $R_fP(x)$와 같으며, 다음과 같이 정의된다.

(정의9) $\partial\text{-}cut_{(Rvp(x)}=\{ M_{RvP(x)}^{filtering}(BP(x)) \mid (R_vP(x_i), R_vP(x_i).T_v), \bigcup_{i=1}^{n} R_vP(x_i), R_vP(x_i).T_v) \geq \partial \}$ 이다.

여기서 ∂는 사상에 의해 퍼지필터링으로서 $0 \leq \partial \leq 1$의 퍼지값이다.

$R_vP(x)$에 대한 의미적 관련성 조건은 조건2와 같으며, 조건2를 만족할 때 G_i는 다음과 같이 정의된다.

(조건2) $R_vP(x)$에 대한 조상 및 이웃에 대해서 $M_{RvP(x)}^{filtering}(BP(x), R_vP(x)) \geq \partial\text{-}cut$

(정의10) $G_i = \sum_{조상및이웃한RvP(x)} M_{RvP(x)}^{filtering}(BP(x), R_vP(x)) \geq \partial\text{-}cut$이다.

예를 들어 응용 도메인 NewsVideo에서 BP(x)가 $R_fP(x)$와 $R_vP(x)$로 분할되었다고 가정할 때 하고 $R_fP(x)$와 $R_vP(x)$ 사상에 의한 퍼지필터링 관계가 <표1>과 같다고 가정하자.

이 경우 $R_fP(x)$에 의한 사상 즉 $M_{RfP(x)}^{0.6} \geq 0.6-cut$은 $0.6\text{-}cut_{Rfp(x)}=\{x_1, x_2, x_4, x_6, x_8\}$이 되고 $R_vP(x)$에 의한 사상은 $M_{RvP(x)}^{0.6} \geq 0.6-cut$ 즉 $0.6\text{-}cutR_vP(x)=\{x_2, x_3, x_4, x_8\}$이 된다.

표1. $R_fP(x)$와 $R_vP(x)$ 사상에 의한 퍼지필터링
Table 1. Fuzzy filtering by $R_fP(x)$ and $R_vP(x)$ mapping

	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8
$R_fP(x)$	0.72	0.93	0.48	0.86	0.53	0.78	0.25	0.82
$R_vP(x)$	0.18	0.70	0.83	0.90	0.37	0.03	0.21	0.65

3.3 통합

통합과정은 사상에 의해서 참조된 $R_fP(x)$, $R_vP(x)$들에 대해서 퍼지필터링을 수행하여 관련된 BP(x)들을 그룹화하게 된다. 이와 같은 방법은 서로 관련 있는 $R_fP(x)$, $R_vP(x)$들을 생성하여 스트리밍으로 인한 반복적인 캐싱 과정을 줄이기 위한 과정으로서 분할된 각 BP(x)들에 대해서 *∂-cut*을 적용하여 통합을 수행하게 된다.

따라서 분산 멀티미디어 응용 도메인에서 BP(x)가 주어지고 시멘틱 관계성에 의한 T_f와 T_v가 만족될 때 분할된 $R_fP(x)$, $R_vP(x)$들을 통합하기 위한 과정은 다음과 같이 진행된다.

단계1 VideoScene들에 대한 BP(x) 및 사상수행
procedure BP(x)
{
// VideoScene들에 대한 $R_fP(x)$, $R_vP(x)$블록분할 수행
procedure mapping($M^{filtering}_{RfP(x)}(BP(x)$, $M^{filtering}_{RvP(x)}(BP(x))$
// 분할된 블록들에 대한 사상 수행
procedure fuzzy_filtering relation()
{
procedure fuzzy_relation(x_i, x_j)
// $R_fP(x)$, $R_vP(x)$블록분할 정보들에 대해서 퍼지 필터링을 수행한다.
procedure extend_search(x_k)
// $R_fP(x)$, $R_vP(x)$블록분할 정보들의 탐색을 확장한다.
for(i=0; i++)
fuzzy filtering(x_i, x_{i+1})
extract ∂_filtering()}
}

단계2 사상을 만족하는 $R_fP(x)$, $R_vP(x)$블록분할 생성
procedure extend_search()
{
extract_∂(membership function)()
// 임의의 $R_fP(x)$, $R_vP(x)$블록분할정보들에 대해서 퍼지 관련성 값을 생성한다.
for(i=0; i++){
if(($M^{filtering}_{RfP(x)}(BP(x)$, $R_fP(x)) \geq$ ∂-cut) or $M^{filtering}_{RvP(x)}$ $(BP(x)$, $R_vP(x)) \geq$ ∂-cut))
create member-of Group(x_i)
// 분할된 $R_fP(x)$, $R_vP(x)$들에 대해서 member-of를 만족하는 그룹을 생성한다.
}

4. 성능 평가

이 장에서는 본 논문에서 제안된 기법의 성능을 알아보기 위하여 기존의 연구방법과 비교 분석하며, 성능 평가 모델을 분석하고 분석된 모델에 대한 시뮬레이션 수행한다.

4.1 성능평가 모델분석

퍼지 필터링이 수행된 캐싱 그룹화 구조에서 블록 세그먼트들에 대한 모델분석은 스트리밍 서비스 요청 응답률, 시간 지연률 및 서비스 히트율에 따라 수행되며, 모델을 정형화하기 위한 성능평가 모델 분석 연산자를 표2와 같이 가정한다.

표2. 성능평가 연산자
Table 2. Performance Metric Operators

연산자	의미
n	블록 분할 된 참조 블록 세그먼트들의 수
m	그룹화된 참조블록 세그먼트
p_i	i 번째 블록 세그먼트들의 확률
θ	Zipf 분포 skew factor
ARS(**λ**)	스트리밍 서비스 요청 응답률
BSL_i	동기화에 따른 i번째 블록 세그먼트 크기
BSL_{ave}	블록 세그먼트들의 평균 크기
T_{Cache}	전체 캐시
$F_{cache-1}$	처음 블록 세그먼트를 캐시 하기 위한 예약 공간
$R_{rest-cache}$	나머지 블록 세그먼트를 캐시하기 위한 예약공간
α	$R_fP(x)$, $R_vP(x)$들을 캐시하기 위한 시작길이
β	$R_fP(x)$, $R_vP(x)$들을 캐시하기 위한 예약공간

4.1.1 평균서비스 요청응답률

평균서비스 요청응답률을 위한 블록 분할 된 참조 블록 세그먼트들의 분포는 Zipf 분포 $p_i=(BP(x_i)/\sum_{i=1}^{n}BP(x_i))$에 의해 수행되며, p_i (i=1,2,...,n, n은 블록 분할 된 전체 세그먼트들의 수)는 분할 된 세그먼트들에 대한 확률 집합이다. p_i에서 $BP(x_i)=(1/i^{\theta})$이고, **θ**는 **θ**>0인 skew factor이다. 이와 같은 확률 집합이 제공될 때 스트리밍 서비스 요청 응답률은 평균 요청 응답률 ARS(**λ**)에 의해 수행되며, ARS(**λ**)는 포아송 분포를 따른다. 각 객체에 대한 스트리밍 요청 응답률은 p_i와는 독립적으로 수행되며, $\sum_{i=1}^{n}p_i=1$이다.

따라서 클라이언트들은 요청된 $R_fP(x)$, $R_vP(x)$들의 참조 블록 세그먼트들에 대해서 최대 응답률 ARS(**λ**)는 1로 한 각 세그먼트들에 대한 평균 서비스 요청 응답률은 $ARS(\boldsymbol{\lambda}_i)=ARS(\boldsymbol{\lambda})\times p_i=ARS(\boldsymbol{\lambda})\times\frac{1/i^{\theta}}{\sum_{i=1}^{n}(1/i^{\theta})}$에 의해 수행된다.

4.1.2 시간 지연율

시간 지연율이란 스트리밍을 요구한 서비스 요청응답이

클라이언트에 제때에 프리패치 되지 않아 발생되는 지연시간으로서 프리패칭이 지연될 경우 클라이언트 측에서는 잠재적으로 지터를 발생하는 원인이 된다. 이때 시간 지연율은 스트리밍을 수행하는 캐시에 따라 발생될 수 있으며, 블록 세그먼트를 인코딩하는 E_r과 $R_fP(x)$, $R_vP(x)$들의 참조 블록 세그먼트들에 대한 평균 네트워크 대역폭 B_w에 의해 시간 지터가 결정된다. 본 논문에서는 시간지연율을 고정분할 참조블록$R_fP(x)$와 가변분할 참조블록$R_vP(x)$에 따라 다르게 분석하며 $\sum_{i=1}^{m} BSL_i \le E_r$이고 $\sum_{i=1}^{m} BSL_i \le B_w$일 때 $\frac{\beta}{\alpha}$를 수행한다. 만일 $\frac{\beta}{\alpha}>1$이면 고정분할 참조블록 $R_fP(x)$를 수행하고, $\frac{\beta}{\alpha}<1$이면 가변분할 참조블록 $R_vP(x)$를 수행하게 된다. 따라서 $R_fP(x)$, $R_vP(x)$들을 제안된 캐싱 그룹화에 따라 블록 세그먼트들을 스트리밍하기 위한 조건과 시간지연율은 다음과 같다.

(조건1) 블록 세그먼트들을 스트리밍 하기위한 조건

$$BPS(x)=\frac{\sum_{i=M+1}^{n} ARS(\lambda_i)}{\sum_{i=1}^{n} ARS(\lambda_i)} \text{ 이다.}$$

고정분할 참조블록 시간지연률

$$F(\delta)=1-\frac{\sum_{i=M+1}^{n} ARS(\lambda_i)\times BP(x_i)\times R_fP(x_i)\times\alpha\times\beta}{\sum_{i=1}^{n} ARS(\lambda_i)\times BP(x_i)\times E_r\times B_w} \text{ 이다.} \quad (4)$$

가변분할 참조블록 시간지연율

$$V(\delta)=1-\frac{\sum_{i=M+1}^{n} ARS(\lambda_i)\times BP(x_i)\times R_vP(x_i)\times\alpha\times\beta}{\sum_{i=1}^{n} ARS(\lambda_i)\times BP(x_i)\times E_r\times B_w} \text{ 이다.} \quad (5)$$

4.1.3 캐시 히트율

캐시히트는 클라이언트가 프록시 캐싱 그룹화에서 스트리밍 서비스를 요청할 때 그룹화 구조에서 적합한 캐싱 데이터를 스트림 함으로서 ARS(λ)를 줄이기 위한 과정이다. 이때 그룹화 구조는 스트림을 수행할 블록 세그먼트들의 평균크기 BSL_{avc}를 고려하면서 처음 블록 세그먼트의 캐시 요약 공간 $F_{cache-1}$을 트랜스코딩하여 적합한 스트림 서비스를 수행하게 된다. 그리고 난 후 나머지 블록 세그먼트들을 캐시하기 위한 예약공간 $R_{rest-cache}$를 파악하여 이후 적합한 캐시 데이터를 스트림하게 되며, 고정분할 참조블록 $R_fP(x)$와 가변분할 참조블록 $R_vP(x)$에 의해 적합한 캐시 데이터를 스트림하기 위한 캐시 히트율은 다음과 같다.

고정분할 참조블록 캐시히트율

$$FH(\delta)=1-\frac{\sum_{i=M+1}^{n} R_fP(x_i)\times BSL_i\times F_{cache-1}\times\alpha\times\beta}{\sum_{i=1}^{n} BP(x_i)\times T_{cache}\times E_r\times B_w} \text{ 이다.} \quad (6)$$

가변분할 참조블록 캐시히트율

$$VH(\delta)=1-\frac{\sum_{i=M+i}^{n} R_vP(x_i)\times BSL_i\times F_{cache-1}\times\alpha\times\beta}{\sum_{i=1}^{n} BP(x_i)\times T_{cache}\times E_r\times B_w} \text{ 이다.} \quad (7)$$

캐시 데이터 스트림 과정에서 캐시 히트율은 퍼지 필터링 $\partial_filtering()$에 의해 결정되며 본 논문에서는 히트율을 위해 $\partial_filtering()$을 위한 ∂-cut을 0.5이상으로 설정하였다.

4.2 시뮬레이션

제안된 프록시 캐싱 그룹화 구조 시스템의 성능을 시뮬레이션하기 위하여 본 논문에서는 <표2>의 성능평가 모델 분석 연산자를 사용하여 평가를 수행한다. 블록세그먼트의 접근 스트리밍 시뮬레이션을 위해 프록시와 서버 사이의 객체 선택은 임의로 선정하여 수행하였다. 시뮬레이션을 위한 전체 블록 세그먼트 스트리밍 요청수는 28,000개로 설정하고, ARS(λ)는 0에서5($0\le\lambda\le5$), $\boldsymbol{\theta}$는 0에서0.35($0\le\boldsymbol{\theta}\le0.35$)이내로 설정하였다. 그리고 B_w는 10/100Mbps의 대역폭을 가지고 ∂-cut을 0.5에서 0.9($0.5\le\partial$-cut≤0.9)로 설정하여 0.5-cut$\le R_fP(x)\le$0.9-cut 일 때와 0.5-cut$\le R_vP(x)\le$0.9-cut일 때로 구분하여 평균 시간지연율과 평균 캐시 히트율을 시뮬레이션 하였다. 스트리밍을 위한 프록시 캐싱 성능평가는 AS1040-C 시리즈의 듀얼 프로세서를 장착한 서버의 분산 환경에서 수행하였으며, 그룹화 구조를 위해 퍼지 필터링된 블록 세그먼트를 고정분할 참조블록 $R_fP(x)$와 가변분할 참조블록 $R_vP(x)$를 각 그룹에 3,500개로 하여 m_1, m_2, m_3, m_4의 4개 그룹으로 분류하였다. 분류된 각 그룹에 대하여 5회씩 반복 수행하여 평균 서비스 요청 응답률, 시간지연율, 캐시 히트율을 평가하였다. 여기서 서비스 요청응답률이란 그룹화 구조에서 스트리밍을 요구한 전체 서비스 데이터가 클라이언트에 제때에 응답하지 않아 발생되는 요청응답 비율을 말하고, 시간지연율이란 서비스 요청 응답률이 지연되는 시간을 말한다. 그리고 캐시 히트율이란 프록시 캐싱 그룹화 구조에서 클라이언트에 전송되는 데이터의 적합비율을 말한다.

이 절에서는 제안된 기법의 성능을 알아보기 위하여 세그먼트 기반의 프록시 캐싱(proxy caching), 전치 캐싱(prefix caching), 프리 패칭(pre-fetching), 오버레이(overlay)기법과 비교 평가를 수행한다.

스트리밍 서비스의 작업부하를 위해 블록 세그먼트의 인코딩율은 128Kbps에서 256Kbps 사이로 하며, 캐시 되지 않은 블록 세그먼트의 네트워크 대역폭은 인코딩율에 따라 0.5-2배 사이에서 임의로 선택하여 수행하였다.

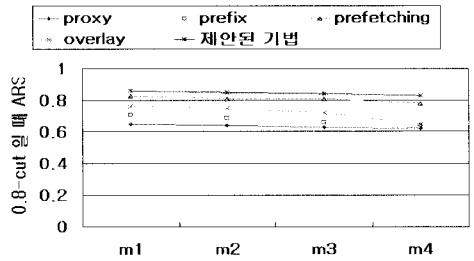

그림 2. 평균 서비스 요청 응답률 ARS(λ)
Fig. 2. Average Service Response Rate ARS(λ)

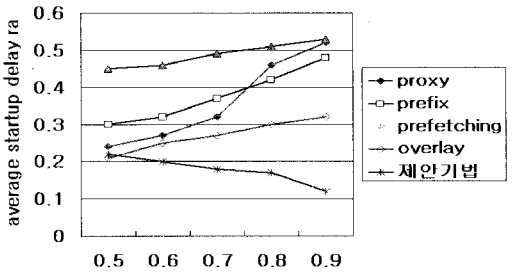

그림 3. 0.5-cut≤$R_vP(x)$≤0.9-cut일 때 평균 시간지연율F(δ)
Fig. 3. Average Latency Time Rate with 0.5-cut≤$R_vP(x)$≤0.9-cut

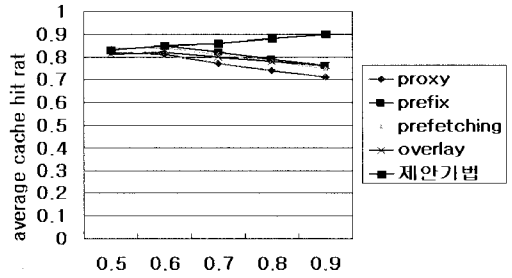

그림 4. 0.5-cut≤$R_vP(x)$≤0.9-cut일 때 평균 캐시 히트율 VH(δ)
Fig. 4. Average Cache Hit Rate with 0.5-cut≤$R_vP(x)$≤0.9-cut

그림2는 제안된 기법이 클라이언트에게 비교적 효율적인 스트리밍 서비스를 제공하고 있음을 보여주고 있다. 그림에서 보듯이 제안된 기법은 블록 세그먼트의 수가 m_1, m_2, m_3, m_4로 증가 될 때 ARS가 크게 변화하지 않고 성능이 유지됨을 알 수 있다.

그림 3은 제안된 기법이 가장 적은 시간 지연율을 보이고 있음을 알 수 있다. 그림에서 다른 기법들은 그룹화 개념으로 시간 지연율을 평가하지 않고 있는 데 반해서 제안된 기법은 퍼지 필터링을 적용한 그룹화 개념에 따라 평가를 수행하기 때문에 다른 평가에 비해 효율성이 증가됨을 알 수 있다.

평가에 있어서 전체 캐시 크기가 80% 미만 일 때는 프록시 지터가 약 30% 정도 개선됨을 알 수 있었으며, 캐시 히트율은 최대 20% 정도 개선됨을 알 수 있다. 따라서 제안된 기법은 퍼지필터링에 의한 우선순위 기법을 적용함으로서 평균요청 응답률 및 시간 지연율 그리고 캐시 히트율의 문제를 개선하여 시스템의 성능이 향상됨을 알 수 있다.

5. 결론 및 향후 연구

최근에 분산 환경에서 다양한 형태의 멀티미디어 서비스를 위한 여러 가지 기법이 제안되고 있으나 네트워크 대역폭 부하, 지터, 낮은 응답 서비스율, 실시간성 등의 여러 문제가 발생되고 있다. 본 논문에서는 이러한 문제를 개선하기 위하여 분할과 사상기반의 퍼지 필터링을 이용한 캐싱 그룹화 방법을 제안하였다. 제안된 기법은 전체 미디어를 프리패칭하여 스트리밍하는 기법이 아니라 블록 세그먼트로 분할하여 사용자가 요구하는 선호도 기반의 서비스가 수행되도록 하였다. 스트리밍을 위한 블록분할은 캐시의 크기에 따라 수행되도록 고정분할 참조블록$R_fP(x)$와 가변분할 참조블록$R_vP(x)$로 구분하였으며, 분할된 블록세그먼트들에 대하여 스트림 캐시서비스를 결정하기 위하여 T_f와 T_v에 따라 퍼지 필터링이 수행하도록 제안하였다. 제안된 기법의 성능평가를 알아보기 위하여 28,000개의 블록 세그먼트를 m_1, m_2, m_3, m_4의 4개의 그룹으로 그룹화 하였으며, 그룹화된 블록 세그먼트에 대하여 0.5-cut≤$R_vP(x)$≤0.9-cut일 때 평균서비스 요청응답률, 시간지연율, 그리고 캐시 히트율을 시뮬레이션 하였다. 그 결과 제안된 기법의 성능이 효율적임을 알 수 있었으며, 향후 연구로는 다양한 분산 네트워크 트래픽 환경에서 시스템 오버헤드를 줄이기 위한 기법이 연구되어야 할 것이다.

참 고 문 헌

[1] Chi-Feng Kao and Chung-Nan Lee, " Aggregate Profit-Based Caching Replacement Algorithm for Streaming Media Transcoding Proxy Systems," *IEEE TRANSACTION ON MULTIMEDIA*, VOL. 9, NO. 2, pp. 221-230, 2007.

[2] K. Wu, P.S. Yu, and J. Wolf, " Segment-based Proxy Caching of Multimedia Streams, " *in Proc. WWW*, Hong Kong, 2001.

[3] R. Rejaie, M. H. H. Yu, and D. Estrin, " Multimedia Proxy Caching Mechanism for Quality Adaptive Streaming Application in the Internet, " *in Proc, IEEE INFOCOM*, vol.2 Israel, pp. 980-989, 2000.

[4] S.Chen, B.Shen, S.Wee, and X. Zhang, " Adaptive and Lazy Segmentation based Proxy Caching for Streaming Media Delivery, " *in Proc. ACM NOSSDAV*, Monterey, CA, pp. 429-441, 2003.

[5] S. Chen, B. Shen, S. Wee, and X. Zhang, " Designs of High Quality Streaming Proxy Systems, " *in Proc, IEEE INFOCOM*, vol. 3, Hong Kong, pp. 1512-1521, 2004.

[6] S. Chen, B. Shen, S. Wee, and X. Zhang, " Investigating Performance Insights of Segment-based Proxy Caching of Streaming Media Strategies," *in Proc. ACM/SPIE Conf. Multimedia Computing and Networking*, San

Jose, Ca, 2004.
[7] Songqing Chen, Bo Shen, Susie Wee, and Xiaodong Zhang, " Segment-Based Streaming Media Proxy: Modeling and Optimization," *IEEE TRANSACTION on MULTIMEDIA*, VOL. 8, NO. 2, pp. 243-256, 2006.
[8] 이종득, 안정용, " 사용자 위주의 멀티미디어 서비스를 위한 시멘틱 기반의 사서함 구조," *퍼지 및 지능 시스템학회 논문지* 제 16 권 제 4 호, pp. 402-409, 2006.
[9] Peng Zhu, Wenjun Zeng and Chunwen Li, " Joint Design of Source Control and QoS-Aware Congestion Control for Video Streaming Over the Internet," *IEEE TRANSACTION ON MULTIMEDIA*, VOL. 9, NO. 2, pp. 366-376, 2007.
[10] Sudipto Guha, Adam Meyerson, Nina Mishra, Rajeev Motwani, and Liadan O'Callaghan, " Clustering Data Streams : Theory and Practice, " *IEEE TRANSACTION ON KNOWLEDGE AND DATA ENGINEERING*, VOL. 15, NO. 3, pp. 515-519, 2003.
[11] J. Kangasharju, F. Hartanto, M. Reisslein, and K. W. Ross, " Distributed Layered Encoded Video Through Caches," *IEEE TRANS. COMP.*, VOL 51, NO. 6, pp. 622-636, 2002.
[12] Bi-Ru Dai, Jen-Wei Huang, Mi-Yen Yeh, and Ming-Syan Chen, " Adaptive Clustering for Multiple Evolving Streams, " *IEEE TRANSACTION on KNOWLEDGE AND DATA ENGINEERING*, VOL. 18, NO. 9, pp. 1166-1180, 2006.
[13] Ewa Kusmierek, Yingfei Dong, and David H. C. Du, " Loopback : Exploiting Collaborative Caches for Large-Scale Streaming, " *IEEE TRANSACTION ON MULTIMEDIA*, VOL. 8, NO. 2, pp. 233-242, 2006.
[14] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, " On Demand Classification of Data Streams," *Proc. ACM SIGKDD*, pp. 503-508, 2004.

저 자 소 개

이종득(Lee Chong Deuk)
1983년 2월 : 전북대학교 전산통계학과 졸업(이학사)
1989년 2월 : 전북대학교 대학원 전산통계학과 졸업(이학석사)
1998년 2월 : 전북대학교 대학원 전산통계학과졸업(이학박사)

2009년 현재 : 전북대학교 공과대학 응용시스템공학부 전자통신공학과 교수

관심분야 : 멀티미디어 통신, 이동통신, 모바일 성능평가 등
E-Mail : cdlee1008@chonbuk.ac.kr