분산 트랜스코딩 환경에서 트랜스코딩 시간 예측 기반 부하 분산 기법

(Load Distribution Method based on Transcoding Time Estimation on Distributed Transcoding Environments)

김 종 우 † 서 동 만 †† 정 인 범 †††
(Jongwoo Kim) (Dongmahn Seo) (Inbum Jung)

요 약 무선 통신 기술의 발달로 인해 PDA나 휴대폰 같은 이동 단말 장치를 통해 멀티미디어 스트리밍 서비스가 가능해졌다. 이동 단말 장치는 낮은 무선 대역폭에서 서비스 되고, 하드웨어의 성능제약이 있다. 이런 조건으로 인해서 주어진 이동 단말 환경에 적합하게 미디어 스트리밍 서비스를 해주는 트랜스코딩 기술이 필요하다. 트랜스코딩 서버는 트랜스코딩작업을 트랜스코딩 서버들의 자원을 활용하여 원본 미디어를 요청된 등급에 맞게 트랜스코딩해야 한다. 목적 트랜스코딩의 요구에 따른 다양한 트랜스코딩 부하 때문에 트랜스코딩 서버 사이에서의 효율적인 부하 분산 정책이 필요하다. 트랜스코딩 과정이외에 서버에서는 이동 단말이 요구하는 전체 서비스 시간에 대한 QoS도 만족시켜야 한다. 본 논문에서는 트랜스코딩 서버들 사이의 공평한 부하 분산을 가능하게 하는 새로운 트랜스코딩 부하 분산 기법을 제안한다. 제안된 기법은 예상 트랜스코딩 시간과 영화정보, 목적 트랜스코딩 비트율을 바탕으로 부하 분산을 수행하며, 부하 분산을 통해 새로운 트랜스코딩 요청에 대한 진입제어를 수행한다.

키워드 : 트랜스코딩, 부하 균형, 시간 예측, 진입 제어

Abstract Due to improved wireless communication technologies, it is possible to provide multimedia streaming service for mobile device clients like PDAs and cellphones. Wireless networks are serviced on low bandwidth channels and mobile devices work on limited hardware specifications. In these conditions, transcoding technologies are needed to adapt the media for streaming services to given mobile environments. To transcode from the source media to the target media for corresponding grades, transcoding servers perform transcoding jobs as exhausting their resources. Since various transcoding loads occur according to the target transconding grades, an effective transcoding load balancing policy is required among transcoding servers. In addition to transcoding process, servers should maintain QoS streams for mobile clients for total serviced times. It requires real-time requirements to support QoS for various mobile clients. In this paper, a new transcoding load distribution method is proposed. The proposed method can be driven for fair load balance between distributed transcoding servers. Based on estimated transcoding time, movie information and target transcoding bit-rate, it provides fair transcoding load distribution and also performs admission control to support QoS streams for mobile clients.

Key words : transcoding, load balance, time estimation, admission control

· 본 연구는 지식경제부 및 정보통신산업진흥원의 "IT융합 스마트조명 고급인력 양성사업"의 연구결과로 수행되었음(NIPA-2010-C6150-1001-0016)

† 학생회원 : 강원대학교 컴퓨터정보통신공학과
jw_kim@snslab.kangwon.ac.kr
†† 정 회 원 : KIST 영상미디어센터 Research Resident
sarum@imrc.kist.re.kr
††† 종신회원 : 강원대학교 컴퓨터정보통신공학과 교수
ibjung@kangwon.ac.kr

논문접수 : 2009년 7월 21일
심사완료 : 2010년 5월 10일

정보과학회논문지: 시스템 및 이론 제37권 제4호(2010.8)

1. 서 론

최근 멀티미디어와 정보통신망의 발전에 따라 영상 정보 서비스에 대한 요구가 날로 다양해지고 있다. 멀티미디어 서비스의 급속한 발전으로 사용자는 유선망 이외에 무선망을 통하여 무선 이동 단말로 스트리밍 미디어를 전송하고 재생하는 서비스를 받을 수 있게 되었다. 그러나 스트리밍 미디어 서비스를 위해서는 영상 정보의 양이 텍스트 기반의 데이터 정보량에 비하여 매우 크기 때문에 광대역 네트워크 대역폭 및 고성능의 컴퓨터를 필요로 하고 있다[1-5].

무선망에서는 네트워크 대역폭이 유선망보다 상대적으로 열악한 환경을 가지고 있으며, 이동 단말의 낮은 컴퓨팅 파워와 시스템 자원은 서버로부터 전송되는 높은 품질의 스트리밍 미디어를 실시간으로 처리하기 어렵다. 이러한 문제점을 해결하기 위하여 최근에 스트리밍 미디어를 이동 단말에 적합한 품질로 바꾸는 트랜스코딩 기술이 연구되고 있다[4-7]. 트랜스코딩은 멀티미디어 컨텐츠를 최초 인코딩한 형태에서 목적하는 단말에 적합하게 변환하는 기술이다. 변환 종류로는 단말 사양에 맞게 스트리밍 미디어의 프레임율, 해상도, 디스플레이 크기, 비트율의 조절을 포함하여 MPEG-1, 2 미디어를 MPEG-4 미디어로 변환하는 요구도 포함한다[4,5].

트랜스코딩 시스템은 인코딩한 영상 데이터를 가지고 있는 멀티미디어 서버와 영상 데이터를 단말 환경에 맞게 변환하는 작업을 수행하는 트랜스코딩 서버로 구성된다. 이동 단말에서 스트리밍 미디어 서비스 요청을 하게 되면 멀티미디어 서버에 저장중인 영상 미디어를 사용자 단말 환경에 적합한 형태로 변경하기 위하여 트랜스코팅 서버로 전송한다. 트랜스코딩 서버는 전송한 영상 미디어를 이동 단말 환경에 적합한 형태의 스트리밍 미디어로 바꾸어 스트리밍 서비스를 한다. 트랜스코딩은 많은 CPU 자원을 요구하는 작업으로, 일반적인 트랜스코딩 시스템에서 트랜스코딩 서버가 단일로 구성되는 경우 동시 접속 사용자 수가 제한된다. 또한 짧은 시간에 대규모 사용자가 트랜스코딩 작업을 요청하면 사용자의 QoSrk 보장되는 기간 안에 작업을 처리하기 어렵다. 대규모 트랜스코딩 서비스에서는 단일 서버로 부하가 집중되는 문제를 해결하기 위해 여러 대의 트랜스코딩 서버로 시스템을 구성하여 트랜스코딩 작업 요청을 분배 할 수 있다[4-7].

대규모 분산 트랜스코딩 시스템에서는 단일 서버로 부하가 집중되는 문제를 해결하기 위하여 서버 간에 트랜스코딩 작업을 균형적으로 분배하는 것이 중요하다. 미디어 데이터의 경우 각 데이터의 크기가 일정하지 않으며 비트율, 프레임율 변경 등 트랜스코딩하는 방식에 따라 작업 시간의 차이가 발생하기 때문에 미디어 데이터의 정보와 각 트랜스코딩 서버의 자원 정보를 이용하여 트랜스코딩 작업 부하를 정확하게 예측하고, 예측한 정보를 통하여 트랜스코딩 서버를 동적으로 선택한다면 특정 트랜스코딩 서버로 작업 부하가 집중되는 현상을 방지 할 수 있다.

본 논문에서는 분산 트랜스코딩 환경에서의 트랜스코딩 작업 시간 예측 기반의 부하 분산 기법을 제안한다. 제안하는 기법은 미디어 데이터의 원본 비트율과 트랜스코딩 목적 비트율, 서버의 자원 정보를 이용하여 트랜스코딩 작업 시간을 예측하고, 이를 기반으로 트랜스코딩 작업을 분산 트랜스코딩 서버에 분배한다. 제안된 기법은 실험을 통해 기존의 부하 분산 기법들보다 높은 서버 확장성을 지원함을 보인다.

본 논문의 구성은 다음과 같다. 2장에서는 본 논문의 관련 연구에 대하여 설명한다. 3장에서는 MPEG-2 미디어 데이터의 트랜스코딩 과정을 분석하여 트랜스코딩 작업 시간 예측 기법을 제안하고, 이를 실험을 통해 실제 트랜스코딩 작업 시간과 유사함을 보인다. 4장에서는 트랜스코딩 작업 시간 예측을 기반으로 한 부하 분산 기법을 제안한다. 5장에서는 실험을 통해 제안된 부하 분산 기법을 다른 기법들과 비교 분석하여, 높은 성능 확장성이 지원됨을 보인다. 마지막으로 6장에서는 본 논문의 결론을 맺고 향후 연구 계획을 설명한다.

2. 관련 연구

2.1 트랜스코딩 시스템

일반적인 트랜스코딩 시스템의 구조는 그림 1과 같다. 사용자는 트랜스코딩에 필요한 정보를 트랜스코딩 서버에 전송 한다. 트랜스코딩 서버에서는 요구한 스트리밍 미디어의 원본을 미디어 서버에서 읽어 사용자가 요구한 해상도, 비트율, 프레임율에 따라서 트랜스코딩한 후 사용자에게 전송한다. 예를 들면 트랜스코딩 서버로부터 CIF(352X288)등급의 25프레임/초, 비트율 100Kbps의 비디오 스트리밍을 QCIF(176X144)등급의 15프레임/초, 비트율 50Kbps의 스트리밍으로 사용자에게 전송할 수 있다. 트랜스코딩 서버에서는 무선망의 특성에 적합하도록 비트율, 해상도, 프레임율을 변경할 수 있다.

기존의 트랜스코딩 시스템을 구축하는 방법들로 소스기반 정적 인코딩 시스템과 트랜스코딩 서버 시스템이 있다[4,8]. 소스기반 정적 인코딩 시스템 방식은 사용자의 요구에 대하여 미디어 파일들을 미리 사용자 등급별로 트랜스코딩한 후 서버에 저장하여 사용하는 방식이다. 이러한 시스템은 스트리밍 미디어를 실시간으로 트랜스코딩하여 전송하는 방식보다 서버의 부하가 심하지 않다는 장점이 있으나 무선 랜 환경에서 사용자의 이동

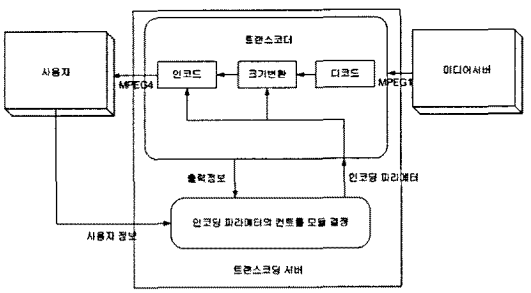

그림 1 트랜스코딩 시스템 구조

에 의한 네트워크 변화에 적응할 수 없다는 단점이 있다. 또한 모든 미디어 파일들을 각 등급별로 트랜스코딩하여 각각의 다른 미디어 파일들로 저장해야 된다는 부담이 있다.

정적 트랜스코딩 서버 시스템은 이동 단말을 사용하는 사용자와 연결된 무선기지국에서 가장 가까운 트랜스코딩 서버를 선택하고 스트리밍 서비스를 받는다. 그러나 트랜스코딩 서버와 이동 단말 사이의 네트워크 상태 변화에 따라 적절하게 서비스 하지 못하는 단점이 있다. 정적 트랜스코딩 서버 방식의 문제점인 특정 서버에 부하가 집중되는 부하 불균형을 피하기 위하여 분배 서버를 두고 트랜스코딩 서버들의 부하 상태를 파악하여 트랜스코딩 요구들을 처리하는 부하 분산 트랜스코딩 시스템 방식이 있다. 이 방식에서는 분배서버는 트랜스코딩 서버들과 연동하여 정해진 부하분배 정책에 따라서 트랜스코딩 서버를 선택하게 된다. 전통적으로 클러스터 시스템에서는 부하분배 정책으로 라운드 로빈, 가중치기반 라운드로빈, 동적 가중치기반 라운드로빈, 가중치기반 최소 접속, 자원 가중치기반 부하분산 알고리즘 등이 사용되고 있다[4,6,7].

2.2 부하분배 정책

라운드로빈(RR : Round Robin) 방식은 사용자 요청이 들어오면 순서대로 서버에 할당하는 방식으로, 부하간 우선순위의 개념이 없고 서버의 연결 개수나 응답시간 등은 고려하지 않는다. 스케줄링의 기초는 사용자 기반이며 실제 서버들의 상태들을 고려하지 않기 때문에 효과적인 부하 분배를 기대하기 힘들고 서버 사이에 부하 불균형이 심각해지는 문제가 발생할 수 있다[4,6,7].

최소 접속(LC : Least Connection) 방식은 가장 접속이 적은 서버로 요청을 연결하는 방식이다. 각 서버에서 동적으로 실제 접속한 숫자를 세어야 하므로 동적인 스케줄링 알고리즘 중 하나이다. 비슷한 성능의 서버로 구성된 시스템에서는 부하가 아주 큰 요청이 한 서버로만 집중되지 않기 때문에, 데이터의 크기가 작고 사용자 접속 부하가 매우 큰 경우에 효과적으로 부하 분배할 수 있다. 그러나 미디어 스트리밍과 같이 긴 접속을 요구하는 사용자 요청과 비교적 짧은 접속을 요구하는 사용자 요청이 섞여있는 동작 환경에서는 서버들 간의 부하 불균형을 초래할 수 있다. 특히 성능이 서로 다른 이종 노드들로 구성된 클러스터 서버 환경에서는 LC같이 사용자 접속 숫자로 부하 배분을 수행하는 것은 서버들의 비효율적인 사용 결과가 발생하기 쉽다[4,6,7].

가중치기반 라운드로빈(WRR : Weighted Round Robin)은 실제 서버에 서로 다른 가중치를 설정하여 부하 분배에 사용하는 방식이다. 각 서버에 가중치를 임의로 부여할 수 있으며, 여기서 지정한 가중치 값을 통해 처리 순서를 정한다. 예를 들어 기본 가중치가 1이고 서버 A, B, C가 각각 4, 3, 2의 가중치를 갖는 경우 스케줄링 순서는 ABCABCABA가 된다. 가중치기반 라운드로빈을 사용할 경우 서버의 연결 개수를 셀 필요가 없고 동적 스케줄링 알고리즘보다 스케줄링 부담이 적다. 그러나 사용자 요청이 폭주할 경우 라운드로빈과 마찬가지로 서버 사이에 심각한 부하 불균형이 발생할 수 있다[4,6,7].

가중치기반 최소 접속 방식(WLC : Weighted Least Connection)은 최소 접속 방식의 한 부분으로서 서버 간의 성능 가중치를 부여하여, 가중치가 높은 서버에서 더 많은 사용자 요청을 받아들일 수 있도록 한다. 가중치기반 최소 접속 방식은 최소 접속 방식에 비해 가중치에 따라 부하를 배분하는데 부가적인 비용이 발생한다[4,6,7].

자원 가중치 기반 부하분배(RWLD : Resource Weighted Load Distribution) 정책은 트랜스코딩 서버의 실제 자원 소모량을 측정하고, 이를 기준으로 등급별 자원 소모량 가중치 및 각각의 트랜스코딩 서버별로 처리할 수 있는 등급별 총 자원가중치 값을 바탕으로 트랜스코딩 서버들의 부하분배를 수행한다[4].

2.3 트랜스코딩 작업 시간 예측 기법

트랜스코딩 작업 시간 예측 기법은 원본 미디어 데이터의 정보와 목적 트랜스코딩 비트율을 이용하여 작업 시간을 예측한다. 이 기법에서는 원본 미디어 데이터의 재생시간과 비트율, 프레임율, 목적 트랜스코딩 비트율을 기반으로 하여 트랜스코딩 작업 시간을 예측한다[9]. 본 기법은 트랜스코딩 작업 시간을 예측하여 데이터를 프리패칭 하기 위하여 연구되었으며, 단일 서버에서 트랜스코딩 작업 시간을 측정하여 이를 기반으로 앞서 언급한 데이터를 대입하여 공식을 유도하였다. 따라서 이 기법은 다양한 환경의 트랜스코딩 서버를 가지는 분산 트랜스코딩 환경에 적용하기 어렵다. 본 논문에서는 트랜스코딩 서버의 프로세서의 클럭과 원본 미디어 데이터의 정보, 목적 트랜스코딩 비트율 등을 이용하여 분산 트랜스코딩 환경에서의 부하분배 정책에 적합한 트랜스코딩 작업 시간 예측 기법을 제안한다.

3. 트랜스코딩 작업 시간 예측

트랜스코딩 작업 시간은 트랜스코딩 각 과정의 수행 시간을 합한 것으로, 표 1의 기호를 이용하여 수식으로 정리하면 식 (1)과 같이 표현할 수 있다. VLD와 IQ, IDCT 과정을 거쳐 미디어 데이터를 매크로블록 단위까지 완전하게 디코딩한 후 ME/MC (Motion Estimation / Motion Compensation)와 샘플링(Sampling) 등으로 MV와 매크로블록 정보를 재조정하고 DCT와 양자화, VLC 과정을 통해 다시 인코딩하는 일반적인 트랜스코딩 과정의 작업시간이다.

$$Tr_T = VLD + iQ + iDCT + MC + Sampling + ME/MC + DCT + Q + VLC \quad (1)$$

그러나 본 논문에서는 비트율을 변경하는 트랜스코딩의 작업 시간만을 고려한다. 프레임율과 해상도 등의 변경이 없기 때문에 움직임 차이값 등의 매크로블록의 정보를 재조정하는 과정이 생략되므로 ME/MC와 샘플링 등의 과정에 소요되는 시간은 0으로 처리할 수 있다. 또한 DCT와 양자화는 8×8 크기의 블록 단위로 수행되는 과정으로 비트율의 높고 낮음에 관계없이 동일한 양의 연산을 수행하므로 비트율 변경에 따른 트랜스코딩 시간에 영향을 주지 않는다. 따라서 비트율만 변경하는 트랜스코딩의 작업 시간은 비트율 변경에 직접적으로 영향을 주는 VLC와 그 외 과정의 작업시간의 합으로 식 (2)와 같이 표현할 수 있다.

$$Tr_T = VLC + \alpha \quad (2)$$
$$(\because \alpha = VLD + iQ + iDCT + DCT + Q)$$
$$(\because MC + Sampling + ME/MC = 0)$$

```
Algorithm VLC {

    /* DC 계수 */
    putDClum(dc_value);

    /* AC 계수 */
    run = 0;
    for(n=1; n<64; n++) {
        if(ac_value != 0) {
            putAC(run, dc_value)
            run = 0;
        }
        else run++;
    }
}
```

그림 2 MPEG-2 인코더의 가변길이 부호화 알고리즘

가변길이부호화(VLC)는 양자화 결과 값을 읽어 들여 엔트로피 부호화함으로써 압축을 수행하는 과정으로 MPEG-2 인코더에서는 런 랭스 코딩(Run-Length Coding)과 허프만 코딩을 이용한다. 양자화 된 결과 값을 지그재그 또는 대체 스캔 방법을 통해 읽어들이면 비슷한 수준의 양자화 결과 값들이 서로 모이게 되는데 이때 비트율이 낮을수록 0인 양자화 결과 값이 연속적으로 나타나는 빈도가 높아진다. 런 랭스 코딩을 이용하여 0이 아닌 양자화 결과 값이 나타났을 때 이전까지 읽어들인 양자화 결과 값을 파일에 쓰기 위하여 허프만 코딩을 이용한다. 그림 2는 MPEG-2 인코더의 가변길이 부호화 과정 중 런 랭스 코딩에 대한 유사코드이다 [10,11]. 8×8 크기의 블록 단위로 양자화가 수행되기 때문에 한 양자화 결과 값은 블록 당 저주파에 해당하는

표 1 논문에서 사용된 기호 및 의미

기 호	의 미	단 위
TrT	트랜스코딩 작업 수행에 소요되는 시간	sec
TrR	변환 비트율에 따라 상대적으로 소요되는 트랜스코딩 작업 수행 시간	sec
SourceBitrate	미디어 데이터의 원본 비트율	bps
TargetBitrate	트랜스코딩 작업을 통해 변환할 비트율	bps
DiffBitrate	원본 비트율과 트랜스코딩 목적 비트율의 차이 값	bps
VLC, VLD	가변길이부호화, 가변길이복호화에 소요되는 시간	sec
Q, iQ	양자화와 역양자화에 소요되는 시간	sec
DCT, iDCT	DCT와 역DCT에 소요되는 시간	sec
MC	Motion Compensation에 소요되는 시간	sec
ME	Motion Estimation에 소요되는 시간	sec
Sampling	Sampling에 소요되는 시간	sec
WriteTime	1Byte의 데이터를 메모리에 쓰는데 소요되는 시간	sec
CPUClock	트랜스코딩 서버의 CPU Clock	Hz
playtime	미디어 데이터의 재생시간	sec
inst	putAC 함수를 호출할 경우 실행되는 명령어의 개수	개
UnitOfWrite	putAC 함수에서 데이터를 파일에 쓰는 최소 단위	bit
STT	원본 비트율과 동일하게 트랜스코딩 수행할 때 소요되는 시간	sec

1개의 DC 계수와 고주파에 해당하는 63개의 AC 계수로 발생한다. 이때 DC 계수는 이전 블록의 DC 계수와 차분부호화를 통해 별도로 처리되고 나머지 63개의 AC 계수가 런 랭스 코딩을 통해 처리한다. run 변수를 이용하여 연속적으로 0값을 가지는 AC 계수가 나타나면 putAC 함수를 호출하여 이전까지의 데이터를 부호화한다. 변환할 비트율이 높아질수록 연속적으로 0값을 가지는 AC 계수의 출현 빈도가 낮아지고, putAC 함수를 호출하는 횟수가 많아지기 때문에 변환할 비트율이 낮을 때 연속적으로 0 값을 가지는 AC 계수의 출현 빈도만 계산하는 것보다 상대적으로 더 많은 작업 시간이 소모된다.

$$VLC = n \times WriteTime + \beta \quad (3)$$
$$(\because n = \text{바이트저장함수의호출횟수})$$

$$WriteTime = \frac{inst}{CPUClock} \quad (4)$$

가변길이부호화 과정에 소요되는 작업 시간은 식 (3)과 같이 표현할 수 있다. n은 각 블록에 발생하는 0이 아닌 값을 가지는 AC 계수가 연속적으로 발생하는 횟수를 나타내며 WriteTime은 putAC 함수로 1 바이트의 데이터를 파일에 쓰는데 걸리는 시간을 의미한다. β는 연속적으로 0 값을 가지는 AC 계수의 출현 빈도를 세는 등의 기타 작업에 소요되는 시간이다. 이 때 WriteTime에 해당하는 작업 시간을 구하기 위해 putAC 함수의 어셈블리어 코드를 분석하였다. 프로그램을 구성하는 어셈블리어 명령의 개수는 프로세서의 종류에 따라 다를 수 있으며 본 연구에서는 IA-32의 명령어 체계를 따르는 인텔과 AMD 프로세서를 고려하였다. 읽어 들인 AC 계수의 수가 정해진 범위를 벗어나지 않고 기타 오류 없이 동작한다고 가정했을 때 putAC 함수를 호출할 경우 총 55개의 명령이 실행되는 것을 확인하였다. CPU의 한 클럭 당 하나의 명령을 수행한다면 putAC 함수를 수행하는데 걸리는 시간은 식 (4)와 같다. 또한 원본 비트율과 변환할 비트율의 차를 통해 변환한 비트율의 높고 낮음에 따라 0이 아닌 값을 가지는 AC 계수가 연속적으로 발생하는 빈도 n의 상대적인 값을 구할 수 있다. 이와 같은 내용으로 식 (3)과 식 (4)를 정리하여 식 (5)로 표현할 수 있다. 식 (5)의 계산 결과는 비트율에 따라 상대적인 것으로 변환할 비트율에 따라 식 (5)의 계산 결과만큼 적은 트랜스코딩 작업 시간이 소모된다. 식 (6)은 식 (5)와 동일한 내용으로 상수항과 변수항에 따라 정리한 것이다. 식 (6)에 의하면 상대적인 트랜스코딩 시간 TrR은 원본 비트율과 변환한 비트율의 차에 비례하며, 변환할 비트율이 클수록 그 값이 작아져 보다 높은 비트율로 변환할수록 더 많은 트랜스코딩 작업 시간을 소요함을 알 수 있다.

$$Tr_R = \frac{(SourceBitrate - TargetBitrate) \times playtime}{UnitOfWrite} \times \frac{inst}{CPUClock} \quad (5)$$

$$Tr_R = \left(\frac{inst \times playtime}{UnitOfWrite \times CPUClock} \right) \times (SourceBitrate - TargetBitrate) \quad (6)$$

식 (1)에서 부터 유도한 식 (6)에 의하여 임의의 트랜스코딩 서버에서 임의의 영화를 트랜스코딩하는데 소요되는 상대적인 시간을 예측할 수 있었다. 그러나 이러한 상대적인 시간을 이용해서는 실제 분산 트랜스코딩 서비스 환경에서의 트랜스코딩 서버 간 부하 균형을 맞추기 어렵다. 따라서 상대적인 트랜스코딩 시간을 절대적인 트랜스코딩 시간으로 변환하는 과정이 필요하다. 식 (7)에서 식 (11)까지의 식은 상대적인 트랜스코딩 시간을 이용하여 절대적인 트랜스코딩 시간을 예측하기 위한 식을 보여준다. 앞서 언급한 바와 같이 트랜스코딩 목적 비트율에 따라 트랜스코딩 시간이 비례하기 때문에, 상대적인 트랜스코딩 시간과 원본 비트율로 트랜스코딩했을 때의 시간의 비와 트랜스코딩 목적 비트율 값들의 비를 이용하여 식 (7)을 유도할 수 있다. 식 (7)을 전개하여 풀어보면 최종적으로 식 (11)을 이용하여 절대적인 트랜스코딩 시간을 예측할 수 있다.

$$SourceBitrate : TargetBitrate = STT : (x \times Tr_R) \quad (7)$$

$$TargetBitrate \times STT = x \times Tr_R \times SourceBitrate \quad (8)$$

$$x = \frac{TargetBitrate \times STT}{Tr_R \times SourceBitrate} \quad (9)$$

$$x' = x \times \frac{SourceBitrate}{DiffBitrate} \quad (10)$$

$$Tr_T = x' \times Tr_R \quad (11)$$

4. 트랜스코딩 시간 예측 기반 부하 분산 기법

4.1 트랜스코딩 부하 예측 기반 부하 분배

사용자가 요구한 QoS를 만족하는 스트리밍 미디어 서비스를 제공하기 위해서는 미디어 데이터가 유효한 기간 안에 트랜스코딩 작업을 완료하여 사용자에게 전달해야한다. 때문에 부하 분산 서버는 사용자로부터 요청을 받아들였을 경우 적절한 트랜스코딩 서버를 선택하여 트랜스코딩 작업을 분배하는 것이 중요한 문제이다. 본 논문에서는 미디어 데이터의 트랜스코딩 작업 시간을 예측하여, 해당 미디어 데이터를 가장 적합한 시간 내에 처리하여 사용자에게 제공할 수 있는 트랜스코딩 서버를 선택하는 트랜스코딩 시간 예측 기반 부하 분산 정책인 TELD(Transcoding Time Estimation based Load Distribution)를 제안한다.

TELD는 기본적으로 각 트랜스코딩 서버들에서 트랜

```
struct mts_info mts[node_no] // 트랜스코딩 서버의 정보를 저장하고 있는 구조체 변수
double AccLoad[node_no] // 현재 트랜스코딩 서버의 트랜스코딩 작업 총 시간
double load[node_no] // 새로 요청된 트랜스코딩 작업의 예측 시간
double tmp[node_no] // 트랜스코딩 부하 예측 시 사용되는 임시 변수

int TELD(struct new_job)
{
  for(i=0; i<node_no; i++)
    // 각 트랜스코딩 서버별 비트율에 따른 트랜스코딩 시간 예측
    load[i] = Time_Estimation(new_job.s_bitrate, new_job.t_bitrate,
                                                    new_job.playtime, mts[i].cpu_hz);
    return select_mts(load, new_job.playtime);
}

double Time_Estimation(double s_br, double t_br, double p_time, double cpu)
{
  tr_r = ((INST * p_time) / (8 * cpu * 1024)) * (s_b - t_b) * 1024;
      // 변환할 비트율에 따라 상대적인 트랜스코딩 부하량 계산
  tr_t = p_time - tr_r; // 트랜스코딩 시간 계산
}

int select_mts(double load[], int playtime)
{
  time_gap = current_time - prev_time;  // 이전 시각과 현재 시각의 차이값 계산
  for(i=0; i<node_no; i++)
    // 현재 각 트랜스코딩 서버의 부하량과 새로 요청된 작업의 부하량 합산
    tmp[i] = AccLoad[i] + load[i] - time_gap;

  // 새로운 작업이 추가될 경우 가장 적은 부하를 받는 트랜스코딩 서버 선택
  for(i=0; i<node_no; i++) {
    if(tmp[i] < min) {
      min = tmp[i];
      idx = i;
    }
  }

  if (tmp[idx] <= playtime) {
    AccLoad[idx] = tmp[idx];
    return idx;
  }
  else return -1;
}
```

그림 3 TELD 알고리즘

스코딩에 걸리는 시간을 예측하고 이를 바탕으로 새로운 트랜스코딩 작업을 할당한다. 각 트랜스코딩 서버마다 수행하고 있는 모든 트랜스코딩 작업 예측 시간의 합을 저장한다. 이 예측 시간의 합은 현재 모든 작업을 완료하는데 걸리는 시간을 의미하므로, 가장 작은 값을 가지는 서버에 기본적으로 새로운 작업을 할당한다. 이 때 기존의 누적 예측 시간 값에서 현재까지 지난 시간을 빼 주어야 현재 남은 작업 시간을 구할 수 있다. 또한 각 트랜스코딩 서버에서 작업이 완료되는 시간을 이 누적 값을 통해 알 수 있으므로, 새로운 작업이 미디어 재생 시간 이내에 완료되는지 여부를 바로 확인할 수 있다. 이를 이용하여 각 서버에서의 새로운 작업 수행 가능 여부를 판단할 수 있으며, 이를 통해 새로운 작업의 진입 제어가 가능하다.

그림 3은 TELD의 유사코드이다. 비트율을 변경하는 트랜스코딩 작업을 수행할 경우의 트랜스코딩 부하를 예측하고 해당 작업을 처리할 최적의 트랜스코딩 서버를 선택하는 과정을 보여준다. *Time_Estimation()* 함수는 요청된 비트율 변경 트랜스코딩 작업 시간을 예측하여 반환한다. 예측된 시간은 *load[]* 에 저장하고, 각 트랜스코딩 서버에서 트랜스코딩 작업의 총 시간을 *AccLoad[]* 에 저장한다. *select_mts()* 함수는 *AccLoad[]* 와 *load[]* 의 값을 합산하여, 새로운 트랜스코딩 작업에 따른 각 트랜스코딩 서버의 총 작업 시간을 계산한다. 그 후 각 트랜스코딩 서버의 총 작업시간 중에서 가장 빠른 시간 안에 트랜스코딩을 완료할 수 있는 서버의 노드 번호를 결정하고, 해당 노드의 총 작업 시간이 요청 미디어의 상영 시간보다 작은지 확인한 후 노드 번호를 반환한다. 만약 총 작업시간이 요청 미디어의 상영 시간보다 크다면, 해당 요청에 대한 서비스가 불가능하다고 판단하고 서비스를 거부한다. 부하 분산 서버는 함수의 결과를 반영하여 새로운 트랜스코딩 작업 요청을 트랜스코딩 서버에 할당한다.

4.2 TELD의 동작

그림 4는 분산 트랜스코딩 서버들에게 부하를 할당하는 부하 분산 서버에서 동작하는 TELD 기법을 제어 흐름도를 통해 보여준다. 시스템이 시작하면 부하 분산 서버는 각 트랜스코딩 서버의 현재 작업에 대한 정보를 초기화 하고 사용자로부터의 트랜스코딩 요청을 기다린다. 사용자로부터 새로운 트랜스코딩 작업 요청이 발생

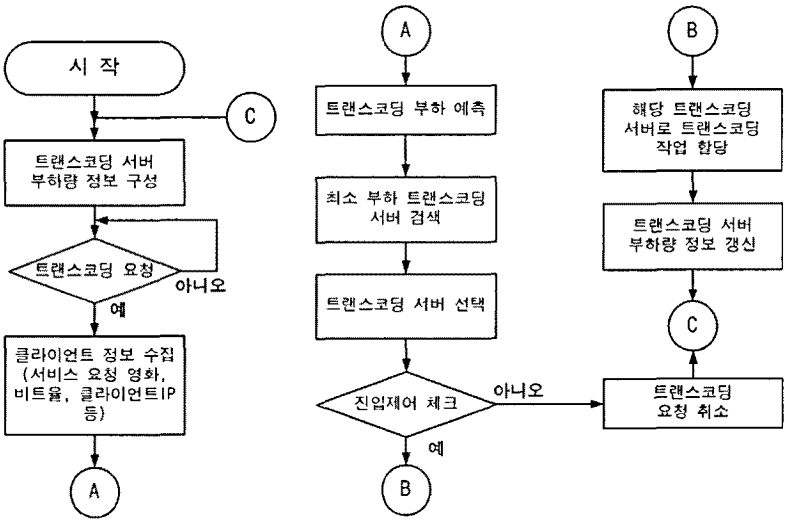

그림 4 TELD의 제어흐름도

하면 각 트랜스코딩 서버에서의 새로운 트랜스코딩 작업 시간을 예측하고, 가장 빠른 시간 내에 모든 작업이 완료될 것으로 예상되는 트랜스코딩 서버를 검색한다. 트랜스코딩 시간 예측을 통해 최적의 트랜스코딩 서버를 선발한 후 새로운 트랜스코딩 작업 요청의 진입 제어 여부를 판별하고, 이상이 없을 경우에 작업을 해당 트랜스코딩 서버에 할당한다.

5. 실험 결과 및 분석

5.1 실험 환경

3 장에서는 분산 트랜스코딩 서비스 환경에서 트랜스코딩 서버간의 부하 균형을 위해 트랜스코딩 서버에서의 트랜스코딩 시간을 예측하였다. 예측된 트랜스코딩 시간이 타당함을 보이기 위해 분산 트랜스코딩 시스템을 구현하였다. 실험을 위한 분산 트랜스코딩 시스템의 서버는 사용자의 서비스 요청을 받아들이고, 이를 각 트랜스코딩 서버로 서비스 요청을 전달하는 하나의 부하 분배 서버와 트랜스코딩 작업을 수행하는 다수의 트랜스코딩 서버로 구성하였다. 실험에 사용한 트랜스코딩 서버의 사양은 표 2와 같다. 두 종류의 트랜스코딩 서버는 각각 2.2GHz과 1.8GHz의 클럭으로 동작하는 AMD 계열의 CPU를 장착한 서버를 사용하였다.

미디어 데이터의 비트율을 변환하는 트랜스코딩 작업을 수행하는 트랜스코더는 오픈 소스 프로젝트로 개발되고 있는 ffmpeg 0.4.9[12] 버전을 사용하였다. 표 3은 실험에 사용한 MPEG-2 미디어 데이터를 보여준다.

5.2 트랜스코딩 시간 예측 결과 및 분석

표 4는 표 2의 각 서버에서 표 3의 세 가지 미디어 데이터를 트랜스코딩하였을 경우의 상대적인 트랜스코딩 시간을 식 (6)을 이용하여 예측한 값을 표로 나타낸 것이다. 각 미디어 데이터의 해상도에 따라 최소한으로 가질 수 있는 비트율이 한계가 있기 때문에 일정 수준 이하의 목적 트랜스코딩 비트율로 트랜스코딩을 하여도 실제 트랜스코딩된 데이터는 그 일정 수준 이하로 변경되지 않는다. 따라서 각 수준별 데이터를 이용하여 각 목적 트랜스코딩 비트율로 트랜스코딩을 하였다. 표 4에

표 2 트랜스코딩 서버 환경

구분	CPU	메모리	운영체제	수량
트랜스코딩서버1	AMD Athlon MP Throughbred 2200 + 1.80GHz	1GB	RedHat Linux 7.4 (Kernel 2.4.18)	4
트랜스코딩서버2	AMD Opteron 248 2.20GHz	1GB	Asianux 2.0 (Kernel 2.6.9)	5

표 3 실험에 사용한 MPEG-2 미디어 데이터

구분		비트율	재생시간	프레임율	해상도
반지의 제왕 - 두개의 탑	CIF급	1362Kbps	1448sec	24fps	656×320
	QCIF급	769Kbps	1448sec	24fps	328×160
	SQCIF급	468Kbps	1448sec	24fps	164×80

표 4 상대 트랜스코딩 작업 시간 예측 결과

QCIF 급	비트율	100Kbps	200Kbps	300Kbps	400Kbps	500Kbps
	서버 1	3.783679초	3.218107초	2.652535초	2.086962초	1.52139초
	서버 2	3.096453초	2.633605초	2.170757초	1.707909초	1.245061초
CIF 급	비트율	600Kbps	700Kbps	800Kbps	900Kbps	1000Kbps
	서버 1	4.309662초	3.744089초	3.178517초	2.612944초	2.047372초
	서버 2	3.526902초	3.064054초	2.601206초	2.138358초	1.67551초

따르면 서버 1에서 미디어 데이터의 비트율을 600Kbps로 트랜스코딩할 경우, 원본 비트율인 1362Kbps로 변환할 때 소요되는 트랜스코딩 작업 시간보다 상대적으로 4.309662초 짧은 시간이 소요된다는 것을 알 수 있다.

이렇게 예측한 상대 트랜스코딩 시간 값을 이용하여 식 (11)에 대입하면 각 서버에서 미디어 데이터를 트랜스코딩할 때 소요되는 시간을 예측할 수 있다. 각 서버에서 예측한 시간과 실제 트랜스코딩에 소요된 시간을 측정하여 그림 5, 6, 7에서 보여 준다. 예측한 값은 해당 서버에서 순수하게 트랜스코딩 작업에 소요되는 시간만을 예측한 것이지만, 실제 트랜스코딩 작업 시간을 측정할 경우 운영체제에서 소요되는 시간과 다른 프로세스와의 문맥 교환에 필요한 시간 등 추가적인 시간이 측정될 수 있다. 그러한 특성을 감안하지 않더라도 그림 5, 6, 7에서의 예측 시간과 실측 시간은 상당히 유사함을 확인할 수 있다. 실험을 통해 측정한 시간은 매 측정마다 오차를 가질 수 있음을 감안한다면, 예측된 트랜스코딩 작업 시간은 실제 트랜스코딩 시간과 유사하다고 할 수 있다. 이러한 오차를 줄이기 위해 20회 이상 트랜스코딩 작업 시간을 측정하고 평균을 내어 그림 5, 6, 7

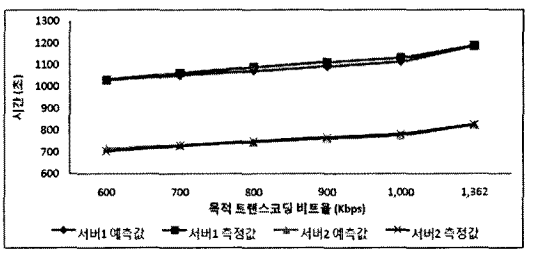

그림 5 CIF 급 영화를 사용한 트랜스코딩 시간

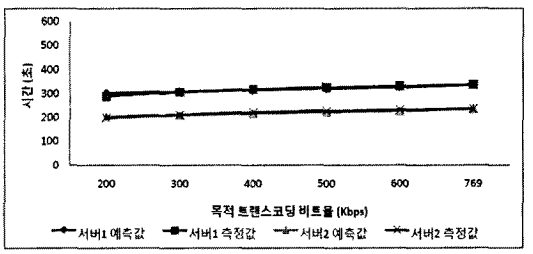

그림 6 QCIF 급 영화를 사용한 트랜스코딩 시간

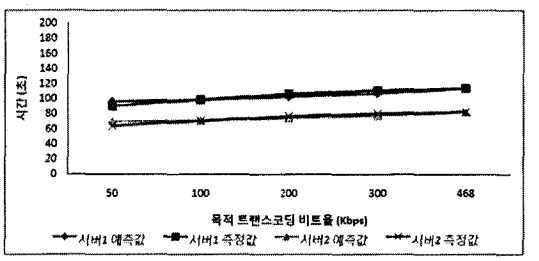

그림 7 SQCIF 급 영화를 사용한 트랜스코딩 시간

에 나타내었다. 따라서 식 (6)을 이용하여 상대적인 트랜스코딩 시간을 예측하고, 이를 식 (11)에 대입함으로써 트랜스코딩 시간을 예측할 수 있음을 확인하였다. 이렇게 예측된 트랜스코딩 시간은 분산 트랜스코딩 시스템 환경에서 각 서버별로 정확한 부하를 예측하여 서버간 부하 균형을 이룰 수 있는 부하 분배 알고리즘에 적용될 수 있다.

5.3 부하 분산 성능 측정 방법

성능 측정을 위하여 부하 발생 프로그램을 구현하였다. 구현한 부하 발생 프로그램은 λ가 0.25인 포아송 분포를 이용하여 사용자의 서비스 요청을 발생하며, 2008년 무선 인터넷 이용 실태 조사[13]의 무선 사용자의 단말기와 네트워크 대역폭 사용 현황을 기준으로 트랜스코딩 목적 비트율을 결정하였다. 80%의 사용자 요청을 56Kbps로 트랜스코딩 목적 비트율로 결정하고, 그 외는 100Kbps에서 1000Kbps 사이를 100Kbps 단위로 나누어 임의로 결정하였다.

실제 인터넷 환경과 유사한 실험을 위해 가상 서버와 가상 클라이언트로 구성된 계측 프로그램(Yardstick Program)[14]을 구현하였다. 가상 서버는 부하 분배 서버에서 동작하며 부하 발생기를 이용하여 성능 측정을 위한 사용자 수만큼 사용자 요청 정보를 발생시켜 가상 클라이언트에 전달한다. 가상 클라이언트는 가상 서버로부터 전송받은 사용자 요청 정보를 바탕으로 부하 분배 서버에 서비스를 요청하고, 부하 분배 서버는 정해진 부하 분배 정책을 바탕으로 적합한 트랜스코딩 서버를 선택하여 작업을 전달한다. 가상 클라이언트는 트랜스코딩된 미디어 데이터를 수신하여 소모함으로써 실제 클라이언트와 동일한 부하를 서버에 준다. 또한 가상 클라이

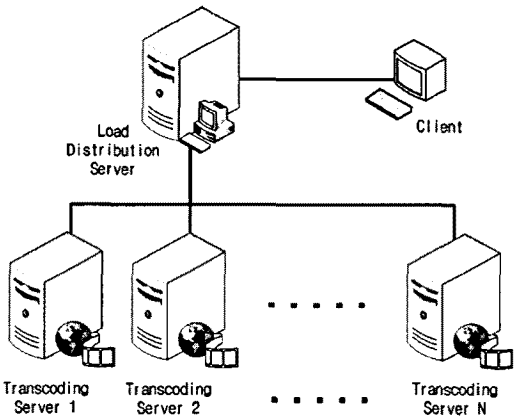

그림 8 분산 트랜스코딩 시스템의 구조도

언트는 수신하는 미디어 데이터의 비트율을 측정하여 서비스 요청한 트랜스코딩 목적 비트율보다 낮을 경우 트랜스코딩 서버의 과부하 상태임을 인식한다. 사용자의 요청 비트율 이하로 전송되는 스트리밍 미디어는 사용자의 QoS를 지원할 수 없으므로 서비스를 제공받는 사용자의 수에서 제외해야 한다.

그림 8은 실험을 위해 구현된 분산 트랜스코딩 시스템의 구조를 보여준다. 홀수 번째 트랜스코딩 서버는 표 2의 트랜스코딩 서버2이고, 짝수 번째 서버는 트랜스코딩 서버1이다. 총 사용한 트랜스코딩 서버는 9대로 표 2의 트랜스코딩 서버2 5대와 트랜스코딩 서버1 4대를 사용하였다.

5.4 부하 분산 결과 및 분석

구현한 분산 트랜스코딩 시스템을 이용하여 TELD와 RWLD, WRR, RR 네 가지 부하 분산 기법의 성능을 측정하였다. 분산 트랜스코딩 시스템에서 QoS가 보장되는 최대 사용자 수를 측정함으로써 특정 부하 분산 정책을 사용할 때 동일한 시스템 자원을 기반으로 얼마나 더 많은 사용자에게 서비스를 제공할 수 있는지를 확인하였다.

그림 9는 클러스터 서버의 수의 변화에 따른 QoS를 보장받는 최대 사용자 수를 보이는 그림이다. 그림에서 보는 바와 같이 전반적으로 TELD가 보다 많은 사용자에게 QoS를 보장할 수 있음을 보여준다. 클러스터 서버가 1개일 경우에는 부하 분산이 큰 의미를 가지지 않기 때문에 동일한 성능을 보여준다. 점차 클러스터 서버의 수가 증가함에 따라 총 사용자의 수도 증가하는 것을 볼 수 있다.

클러스터 서버의 개수가 8대에서 9대로 증가할 때 TELD의 총 사용자수가 급격하게 늘어나는 것을 확인할 수 있다. 이는 TELD가 시간을 기준으로 부하 분산을 수행하기 때문에 실험 시간이 늘어나면 현재 각 트

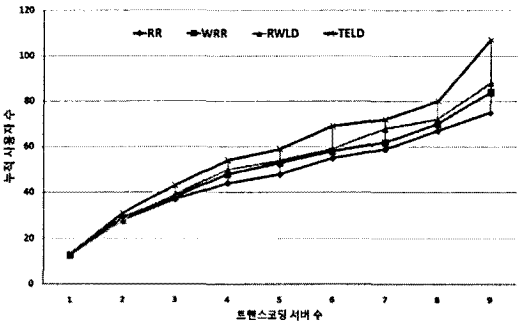

그림 9 클러스터 서버 수에 따른 총 사용자 수

랜스코딩 서버의 작업량이 감소한 것으로 간주하고 추가적인 서비스 요청을 수락하게 되기 때문이다. 따라서 보다 오랜 시간동안 서비스를 지속할 경우 TELD의 누적 사용자 수는 현저하게 증가할 것으로 예상된다. 각 서버의 작업량을 기준으로 부하를 분산하는 RWLD의 경우 서비스가 지속되는 동안 동일한 작업량으로 간주하기 때문에 서비스 시간이 지남에 따라 각 서버의 작업량에 변화가 없다. 이로 인해 서비스 시간이 길어지는 경우 새로운 요청을 지원할 자원이 있음에도 새로운 요청에 서비스를 제공할 수 없다.

6. 결론 및 향후 연구

본 논문에서는 분산 트랜스코딩 서버 환경에서 균형적인 부하 분배를 위한 트랜스코딩 부하 예측 기법을 제안하였다. MPEG-2 미디어 데이터의 트랜스코딩 세부 과정을 분석하였고, 이를 바탕으로 비트율을 변경하는 트랜스코딩 과정에서는 VLC 부분이 트랜스코딩 작업 시간에 가장 큰 영향을 주는 것을 확인하였다. VLC 부분에서 소요되는 작업량을 분석하여 실제 트랜스코딩 작업에 미치는 영향을 수식을 통하여 보여주고, 이를 이용하여 비트율에 따라 소요되는 상대적인 트랜스코딩 시간을 예측하였다. 예측한 상대적인 트랜스코딩 시간과 원본 비트율로 트랜스코딩할 때의 시간을 이용하여 비례식을 만들어 실제 트랜스코딩에 소요되는 시간을 예측하는 식을 수립하였다. 이를 통해 각 서버의 CPU와 MPEG-2 미디어 데이터의 정보를 대입하고 트랜스코딩 시간에 소요되는 시간을 예측하였다.

제안한 기법을 이용하여 예측한 트랜스코딩 시간을 실제 분산 트랜스코딩 시스템을 구현하여 각 트랜스코딩 서버에서 측정한 트랜스코딩 시간과 비교 분석하였다. 그 결과 예측한 트랜스코딩 시간과 측정한 트랜스코딩 시간이 일치함을 확인하였다. 이를 바탕으로 트랜스코딩 시간 예측 기법을 기반으로 한 부하 분산 기법을 제안하였다. 제안된 기법은 실험을 통해 기존의 부하 분

산 기법과 비교 분석하여 높은 성능 확장성을 지원함을 확인하였다.

향후에는 본 논문에서 제안한 기법을 확장하여 MPEG-4와 H.264등의 다양한 영상 압축 기법에 대한 트랜스코딩 부하 예측 기법을 연구하고자 한다. 이를 통해 다양한 압축 기법을 지원하는 대규모 분산 트랜스코딩 시스템에서의 효과적인 부하 분배 알고리즘에 대해 연구하고자 한다. 또한 좀 더 다양한 무선 네트워크 상황과 모바일 단말기 환경을 고려하여 확장된 연구를 진행하고자 한다.

참 고 문 헌

[1] Dinkar Sitaram, Asit Dan, "Multimedia Servers: Applications, Environments, and Design," Morgan Kaufmann Publishers, 2000.

[2] Wu-chi Feng, Ming Liu, "Critical Bandwidth Allocation Techniques for Stored Video Delivery Across Best-Effort Networks," *The 20th International Conference on Distributed Computing Systems*, pp.201-207, April, 2000.

[3] David H.C. Du, Yen-Jen Lee, "Scalable Server and Storage Architectures for Video Streaming," *IEEE International Conference on Multimedia Computing and Systems*, pp.191-206, June, 1999.

[4] Dongmahn Seo, Joahyoung Lee, Yoon Kim, Changyeol Choi, Hwangkyu Choi, Inbum Jung, "Load Distribution Strategies in Cluster-based Transcoding Servers for Mobile Clients," *Lecture Notes in Computer Science*, vol.3983, pp.1156-1165, May, 2006.

[5] Dongmahn Seo, Joahyoung Lee, Yoon Kim, Changyeol Choi, Manbae Kim, Inbum Jung, "Resource Consumption-Aware QoS in Cluster-based VOD Servers," *Journal of Systems Architecture: the EUROMICRO Journal*, vol.53, Issue 1, pp.39-52, Jan, 2007.

[6] Harini Bhradvaj, Anupam Joshi, Sansanee Auephanwiriyakul, "An active transcoding proxy to support mobile web access," In *Proceedings of International Conference on Reliable Distrubuted System*, pp.118-123, Oct, 1998.

[7] Anthony Vetro, Huifang Sun, "Media Conversions to Support Mobile Users," *IEEE Canadian Conference on Electrical and Computer Engineering*, pp. 607-612, May, 2001.

[8] Sumit Roy, Michele Covell, John Ankcorn, Susie Wee, Takeshi Yoshimura, "A System Architecture for Managing Mobile Streaming Media Services," *Streaming Media Systems Group*, pp.408-413, May, 2003.

[9] Sungyong Lee, "Design and Implementation of Linux based Mobile Media Streaming System," *Master's thesis*, Aug, 2005.

[10] MPEG Study Site, http://www.mpeg.org

[11] MPEG Home Page, http://www.chiariglione.org/mpeg/

[12] ffmpeg develepment site, http://ffmpeg.sourceforge.net

[13] "2008 Survey on the Wireless Internet Use," National Internet Development Agency of Korea, Dec, 2008.

[14] Brian K. Schmidt, Monica S. Lam, J. Duane Northcutt, "The interactive performance of SLIM: a stateless, thin-client architecture," *ACM Symposium on Operating Systems Principles*, pp.32-47, Dec, 1999.

김 종 우

2009년 강원대학교 컴퓨터정보통신공학전공 학사. 2009년~현재 강원대학교 컴퓨터정보통신공학과 석사과정. 관심분야는 멀티미디어 시스템, 센서네트워크

서 동 만

2002년 강원대학교 컴퓨터학과 학사. 2004년 강원대학교 컴퓨터정보통신공학과 석사. 2010년 강원대학교 컴퓨터정보통신공학과 박사. 2010년 3월~KIST 영상미디어센터 Research Resident. 관심분야는 병렬처리, 멀티미디어 시스템, 운영체제, 센서네트워크

정 인 범

1985년 고려대학교 전자공학과 학사. 1985년~1995년 ㈜ 삼성전자 컴퓨터 시스템 사업부 선임 연구원. 1992년~1994년 한국과학기술원 정보통신공학과 석사. 1995년~2000년 8월 한국과학기술원 전산학과 박사. 2001년~현재 강원대학교 컴퓨터정보통신공학 전공 교수. 관심분야는 운영체제, 소프트웨어 공학, 멀티미디어 시스템, 센서네트워크