http://dx.doi.org/10.7840/kics.2015.40.10.1976

성공적인 확인응답이 필요한 비디오 정보 파일에 의한 토큰버킷 자동 파라메타 설정 기법을 가진 비디오 스트리밍 수신기

이 현 노◆, 김 동 회*, 남 부 희*, 박 승 영°

Video Streaming Receiver with Token Bucket Automatic Parameter Setting Scheme by Video Information File needing Successful Acknowledge Character

Hyun-no Lee◆, Dong-hoi Kim*, Boo-hee Nam*, Seung-young Park°

요 약

비디오 스트리밍 수신기의 재생 버퍼에 있는 패킷량은 네트워크 상태에 따라 변화되며 지연 및 지터의 영향으로 인해 포화 및 고갈 현상이 일어날 수 있다. 특히, 유입되는 비디오 트래픽의 양이 재생 버퍼의 최대 양을 넘으면 버퍼 오버플로우(buffer overflow)문제가 발생된다. 버퍼 오버플로우는 화질의 열화를 발생시키고 스킵(skip) 현상으로 인해 재생의 불연속성을 발생시킨다. 또한 네트워크 혼잡으로 인하여 패킷의 유입이 늦어지면 버퍼 언더플로우(buffer underflow) 문제에 의한 버퍼링에 의해 영상의 멈춤 현상이 일어날 수 있다. 상기 문제들을 해결하기 위하여 본 논문에서는 토큰버킷(Token Bucket)의 주요 파라미터인 토큰 발생률 파라메타와 버킷의 최대 용량 파라미터를 각각의 비디오 패킷들의 패턴에 따라서 자동적으로 설정하는 토큰버킷 기법을 장착한 비디오 스트리밍 수신기를 제안한다. NS-2(Network Simulator-2)와 JSVM(Joint Scalable Video Model)을 이용하는 시뮬레이션 결과는 제안하는 토큰버킷 파라메타 자동설정 기법이 기존의 수동설정 기법보다 3개의 시험 비디오 시퀀스들에 대해 오버플로우/언더플로우 횟수와 패킷 손실 비율 및 PSNR(Peak Signal to Noise Ratio)측면에서 우수한 성능을 제공함을 확인 할 수 있었다.

Key Words : Video streaming, Token bucket, Buffer overflow, Token generation rate, Bucket maximum capacity, Peak signal to noise ratio

ABSTRACT

The amount of packets in palyout buffer of video streaming receiver can be changed by network condition, and saturated and exhausted by the delay and jitter. Especially, if the amount of incoming video traffic exceeds the maximum allowed playout buffer, buffer overflow problem can be generated. It makes the deterioration of video image and the discontinuity of playout by skip phenomenon. Also, if the incoming packets are delayed by

※ This study was supported by 2015 Research Grant from Kangwon National University(No. 520150453)
◆ First Author : Dept. of Electrical and Electronic Engineering, College of Information Technology, Kangwon National University, leehn900115@naver.com, 학생회원
° Corresponding Author : Dept. of Electrical and Electronic Engineering, College of Information Technology, Kangwon National University, s.young.park@kangwon.ac.kr, 종신회원
* Dept. of Electrical and Electronic Engineering, College of Information Technology, Kangwon National University, donghk@kangwon.ac.kr, 종신회원, boonam@kangwon.ac.kr
논문번호 : KICS2015-06-170, Received June 4, 2015; Revised July 28, 2015; Accepted September 7, 2015

network confusion, the stop phenomenon of video image is made by buffering due to buffer underflow problem. To solve these problems, this paper proposes the video streaming receiver with token bucket scheme which automatically establishes the important parameters like token generation rate r and bucket maximum capacity c adapting to the pattern of video packets. The simulation results using network simulator-2 (NS-2) and joint scalable video model (JSVM) show that the proposed token bucket scheme with automatic establishment parameter provides better performance than the existing token bucket scheme with manual establishment parameter in terms of the generation number of overflow and underflow, packer loss rate, and peak signal to noise ratio (PSNR) in three test video sequences.

Ⅰ. 서 론

비디오 스트리밍 서비스는 사용자에게 제공함에 있어서 만족스러운 품질의 서비스가 이루어져야 한다. 종단 간 품질에 있어 중요한 고려사항은 재생의 연속성이다. 네트워크의 혼잡으로 인해 패킷의 지연이 생기거나 군집적인 형태의 패킷이 재생 버퍼로 유입된다면 버퍼의 고갈 및 포화현상을 일으키게 된다. 이러한 현상들은 버퍼링으로 인한 영상의 멈춤 또는 패킷의 손실로 인한 프레임의 손실 및 화질열화, 스킵현상으로 이어지게 된다[1,2]. 지금까지 비디오 스트리밍 서비스에 있어 이러한 서비스 품질 저하를 막기 위한 여러 가지 연구들이 진행되어 왔다. 무선 채널 상태를 인지하여 비디오의 전송률과 품질을 조절하는 네트워크 기반의 품질 적응 스트리밍 기법이 있었으며[3] 사용자 체감 품질을 만족시키기 위하여 최소 재생 버퍼 레벨을 보존하여 가변적인 무선 네트워크 환경에서도 재생의 끊김 현상, 재생 중 비디오 품질의 급격한 변화 등을 효율적으로 방지하는 비트율 적응 알고리즘도 제시되었다[4]. 또한 비디오 서비스 품질을 높이기 위한 기법으로 토큰버킷을 이용한 연구들로 낮은 대역폭을 가지는 네트워크를 통해 비디오를 전송하기 위해 토큰버킷을 이용한 실시간 비트율 제어 기법이 있었으며[5], 스트리밍 서비스와 같은 실시간 데이터 패킷의 요구 전송 지연과 요구 전송률의 보장을 지원하기 위한 TBS(Token Bucket-based Scheduling) 기법이 있었다. 이 기법은 데이터 전송 큐의 HOL(Head of Line) 지연을 이용한 가중치 값과, 토큰버킷의 HOL 지연을 이용한 가중치 값을 이용해 실시간 패킷의 전송 우선순위를 계산하여 성능을 개선시키는 방법이다[6]. 이 외에도 토큰버킷의 초당 토큰 재생률과 버킷의 최대 용량들을 조절하여 패킷 손실률을 감소시키기 위한 연구와[7], 멀티미디어 서비스에서 서비스 품질을 보장하기 위해 토큰버킷을 이용한 대역폭 할당 방식을 사용한 연구가 있었다[8]. 제안하는 토큰버킷 파라메타의 자동 설정 기법에서는 실질적인 비디오 패킷을 전송하기 전에 비디오 정보 파일을 하위 계층에서의 오류 제어 기법과 연동하여 반드시 성공적인 확인응답을 수신할 때까지 전송하게 된다. 이와 비슷한 방식으로는 기존의 저장형 A/V 스트리밍 서비스를 사용하기 위한 metafile 전송 방식이 있다. 웹 브라우저에서 미디어 플레이어를 통하여 요청한 오디오/비디오를 스트리밍하기 위해 웹 서버에서 비디오의 정보가 담긴 metafile을 전송받고 미디어 플레이어는 metafile안에 있는 URL 정보를 이용하여 스트리밍 서버에 접속하고 스트리밍을 진행하게 된다[9]. 본 논문에서 제안하는 방식에 사용되는 비디오 정보 파일의 경우 metafile과 유사한 형태의 파일이지만 보내려고 하는 비디오에 대한 정보들 중 프레임의 크기 정보만이 기록되어있으며 프레임의 크기를 전송 할 수 있는 최대 패킷 사이즈로 분할하여 기록하게 된다. 이 정보들은 수신단에서 토큰에 의해 유입되는 패킷의 양이 해당 비디오 시퀀스의 단위 시간당 평균 재생량을 넘지 않는 적절한 토큰버킷의 파라메타를 자동으로 설정해주는데 사용된다. 따라서 기존의 수동설정 기법과는 달리 적절한 파라미터 설정 값을 찾기 위해 시간을 소모할 필요가 없다. 제안하는 자동 설정 기법과 기존의 수동 설정 기법의 성능 차이는 시뮬레이션을 통해 비교분석 하였다. 본 논문의 II장에서는 스트리밍 서비스에서 수신단에 토큰버킷 기법을 적용했을 때 수동적으로 파라메타를 설정하여 성능을 분석했던 이전의 연구에 대해 설명하고 오버플로우를 해결하기 위한 시스템 모델을 설명한다. III장에서는 비디오에 따라 적응적인 파라메타를 자동으로 설정 할 수 있는 자동적인 파라메타 설정 토큰버킷 기법에 대해 설명한다. IV장에서는 시뮬레이션 기법에 대해 소개하며 V장에서는 시뮬레이션에 대한 결과를 분석하고 VI장에서는 결론을 제시하였다.

Ⅱ. 기존의 토큰버킷 파라메타 수동 설정 방법

기존의 토큰버킷 파라메타 수동 설정 기법에서는 그림 1과 같이 수신 버퍼의 포화현상과 재생 품질 저하를 유발하는 버퍼 오버플로우를 해결하기 위해 주 파라메타인 초당 생성되는 토큰의 개수(*r*)와 토큰이 저장될 수 있는 최대 버킷 용량(*c*)를 수동적으로 선택한다[10]. 그림 1에서 보는바와 같이 수신단으로 유입되는 패킷들은 네트워크의 영향을 받아 불규칙적인 패턴으로 임시버퍼에 저장되고 하나의 패킷에 한 개의 토큰을 할당하는 토큰버킷 기법에 의해서 군집적인 패킷들을 규칙적인 패턴으로 변환하여 재생 버퍼의 포화현상을 방지 할 수 있게 된다.

토큰버킷 기법을 이용하여 포화현상을 방지하기 위해서는 적절한 파라메타 값을 설정해 주어야 한다. 여기서 적절한 파라메타란 토큰버킷 기법으로 인해 재생 버퍼로 유입되는 패킷량이 오버플로우를 발생시키지 않는 설정값을 말한다. 그러기 위해 수신단에서는 재생버퍼의 최대값($B_{\max}$), 단위 시간당 재생량($L_{playout}$), 패킷 최대 크기($Ptxmax$)를 고려하여 토큰버킷의 파라미터를 정해줘야 한다. 임시버퍼에 충분한 패킷이 있고 재생 버퍼가 비어있다고 가정할 경우, 토큰버킷에 의해 유입될 수 있는 최대 패킷량은 토큰 생성에 따라 유입될 수 있는 최대 패킷량($r\times Ptxmax$)과 버킷 안에 있는 토큰에 의해 유입되는 최대 패킷량($c\times Ptxmax$)의 합으로 볼 수 있다. 또한 토큰버킷에 의해 재생버퍼로 비디오 패킷이 들어온 후에 재생버퍼에서는 일정 프레임이 재생된다. 즉, 버퍼의 포화상태를 방지하기 위해서는 최소한 식 (1)과 같이 토큰버킷으로 인해 유입될 수 있는 패킷량은 재생버퍼의 최대값과 단위 시간당 재생량을 합한 값과 같거나 작아야 할 것이다.

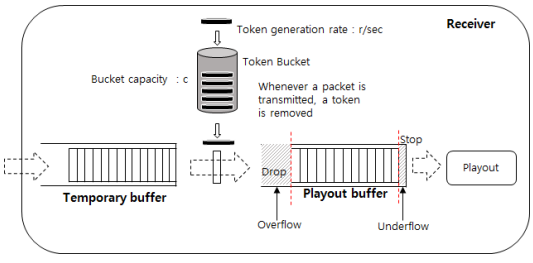


그림 1. 비디오 스트리밍 수신기에 토큰버킷 기법이 적용된 구조
Fig. 1. The structure of video streaming receiver with token-bucket scheme

$$(r\times Ptxmax)+(c\times Ptxmax) \le B_{\max}+L_{playout} \quad (1)$$

식 (1)에서 *r*과 *c*를 더한 값을 구하기 위해 식을 변경하면 다음 식 (2)와 같이 나타낼 수 있다.

$$r+c\le \frac{B_{\max}+L_{playout}}{Ptxmax} \quad (2)$$

하지만 토큰버킷으로 인한 패킷 유입량이 과도하게 적을 경우에는 반대로 버퍼의 언더플로우 현상이 발생 할 수 있다. 언더플로우는 재생 버퍼 내에 재생하기에 충분한 양의 데이터가 없기 때문에 버퍼링으로 인하여 영상의 멈춤현상이 발생할 수 있으며 체감품질을 저하시킨다는 문제점이 있다. 따라서 *r*과 *c*를 더한 값인 *Tsum*을 정할 경우 식 (3)을 사용하여 결정한다.

$$Tsum=r+c=\lfloor \frac{B_{\max}+L_{playout}}{Ptxmax} \rfloor \quad (3)$$

이 과정에서 구해진 *Tsum*은 처음에 한번만 구하게 된다. 이후 토큰버킷 파라미터를 수동으로 설정하여 시뮬레이션을 진행하게 되며 이 경우 설정할 수 있는 값의 경우는 식 (4)와 같다.

$$r=[1,2,3,\cdots\cdots,Tsum]$$
$$c=[Tsum-1,Tsum-2,Tsum-3,\cdots\cdots,0] \quad (4)$$

*r*값을 1로 설정한 경우에는 *c*값을 *Tsum* - 1로 설정해 준다. 식 (4)에서 보듯이 설정할 수 있는 파라메타 값의 경우의 수는 *Tsum*이 된다. 물론 *r*값을 너무 작은 값인 1이나 너무 큰 값인 *Tsum*까지 설정해준다면 *r*=1인 경우에는 토큰에 의한 지나친 패킷제한으로 언더플로우가 심하게 일어나 영상의 멈춤 현상이 심해질 것이며 *r*=*Tsum*으로 설정했을 경우에는 지나친 패킷의 유입 때문에 오버플로우로 인한 프레임의 손실 및 화질의 열화가 일어날 것이라고 예상할 수 있다. 하지만 최저 설정값과 최대 설정값을 제외하고도 많은 경우의 수가 존재할 수 있기 때문에 가장 적절한 파라미터를 찾는 것에는 많은 시간을 요구하게 된다. 게다가 비디오마다 다른 특성을 가지며 그에 따라 전송되는 패킷의 개수나 크기는 다를 수밖에 없기 때문에 특정 비디오에서 가장 적합한 *r*값과 *c*값을 찾았다고 해도 다른 종류의 비디오가 전송될 때는 그 비디오에 가장

적합한 토큰버킷 파라미터를 식 (1)부터 식 (4)까지의 과정을 통해 또 다시 구해야 한다는 문제가 있다. 결과적으로 토큰버킷 기법을 적용할 때 고정적인 파라메타를 사용하여 버퍼오버플로우를 해결하기에는 많은 문제점들이 존재하며 주어진 적은 시간 내에서 비디오의 종류에 따라 가장 적합한 파라메타를 자동적으로 설정해 줄 수 있는 기법의 연구가 절실히 요구된다. 따라서 본 연구에서는 이러한 비디오의 특성을 고려하여 자동적으로 토큰버킷 파라메타를 설정해 줄 수 있는 방법을 제안한다.

Ⅲ. 제안하는 토큰버킷 파라메타의 자동 설정 방법

본 논문에서 제안하는 방법은 그림 2와 같이 동작하며 토큰버킷 파라메타인 *r*값과 *c*값을 정할 때, 전송되는 비디오의 패킷 사이즈 정보를 전송받고 그 정보를 토대로 *r*값과 *c*값을 적응적으로 자동 설정하게 된다.

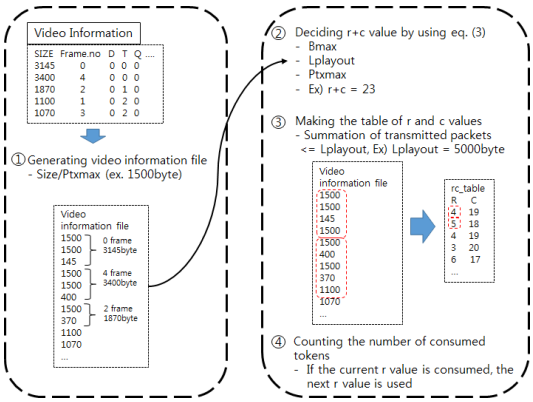


그림 2. 토큰버킷 파라메타의 자동 설정을 위한 4 단계 동작
Fig. 2. The 4-step operation for automatic establishment of token-bucket parameter

3.1 비디오 정보 파일 생성

그림 2에서의 ①과 같이 먼저 첫 번째로 비디오 서버에서는 전송하려는 비디오의 각 프레임 크기 정보만을 바탕으로 보낼 수 있는 최대 패킷의 크기로 나누어 비디오 정보 파일에 기록하게 된다. 예를 들어, 전송 되는 비디오의 프레임 사이즈 frame_size={3145, 3400, 1870, 1100, 1070, ……}일 경우, 비디오 정보 파일에 기록하는 프레임의 각 사이즈는 frame_size/ $Ptxmax$ (전송되는 패킷의 최대 크기)로 나누어 기록한다. 만일 전송되는 패킷의 최대 크기가 1500 byte라고 할 때, 3145 byte의 사이즈를 가지는 첫 번째 프레임의 경우 3145byte/1500byte($Ptxmax$) 로 계산하여 비디오 정보 파일에는 {1500, 1500, 145}로 기록하게 되고, 그 이후 비디오 프레임에 대해서도 마찬가지로 기록하게 된다. 비디오 정보 파일은 단순한 텍스트 파일이다. 이러한 비디오 정보 파일은 작은 크기를 가지며 전송 오버헤드는 매우 적다[11]. 송신단에서는 먼저 비디오 정보 파일을 전송하고 그 정보파일이 하위 계층에서의 오류 제어 기법과 연동하여 성공적인 확인응답을 수신한 후에 비디오 데이터를 전송한다고 가정한다. 수신단에서는 이 파일의 정보를 바탕으로 전송될 비디오 패킷들의 최대 패킷 크기 및 전송되어지는 각 패킷들의 크기를 알 수 있다.

3.2 r+c값 초기 설정 방법

두 번째로 그림 2에서의 ②와 같이 수신단에서는 재생버퍼의 최대값($B_{\max}$), 단위 시간당 재생량($L_{playout}$), 패킷 최대 크기($Ptxmax$)정보를 바탕으로 식 (3)을 통해 처음에 한번 구한 *r*값과 *c*값을 더한 값인 *Tsum*을 구해주게 된다. 이후 *r*값은 시간에 따라 적응적으로 선택되어지며 그에 따른 *c*값은 처음에 구한 *Tsum*값에서 해당 시점의 *r*값을 뺀 나머지 값을 사용하게 된다. 새로운 비디오 트래픽을 전송할 경우에는 그에 맞는 *Tsum*을 다시 구하게 된다.

3.3 r값과 c값의 테이블 생성

이 후, 그림 2에서의 ③과 같이 수신단에서는 비디오 정보 파일을 바탕으로 연속적으로 전송되는 패킷들에 대해 패킷 크기의 합을 구하고, 그 합이 재생량을 넘지 않는 최대 구간만큼 분할하여 *r*값 테이블을 생성한다. 먼저 비디오 정보 파일은 최대 패킷 크기로 나누어진 각 프레임의 크기가 기록되어 있으며 식 (5)와 같다. *n*은 전송되어 지는 패킷의 최대 개수 이다. 또한 이 과정을 거쳐 만들어 지는 *r_table*의 형태는 식 (6)과 같다.

$$Video\ information\ file = \{S_{p1}, S_{p2}, S_{p3}, \cdots\cdots, S_{pn}\} \quad (5)$$

$$r\_table = \{r_1, r_2, r_3, \cdots\cdots, r_m\} \quad (6)$$

전송되는 비디오의 *r_table*의 값은 그림 3의 과정을 거쳐서 기록이 되며 완성된 *r_table*을 가지고 시간이 흐름에 따라 파라메타 값을 설정하게 된다.

예를 들어, 비디오 정보 파일에 기록되어 있는 전송되는 패킷의 크기가 {1500, 1500, 145, 1500, 400, 1500, 370, 1100}이며, 단위시간 0.1초당 재생되는 평

```
Process making r_table
j = 0, k = 1
for(i = 1; i ≤ n; i = i + 1) // n  is the maximum number of transmirred packets
  Pktsum = Pktsum + S_pi // accumulated size of thransmitted packets
  j = j + 1
    if(Pktsum ≥ L_playout) // if Pktsum is larger than average video playout size
      r_k = j − 1 // r_k is the k− th value recorded r_table, j−1 is
                  the number of transmitted packets which is less than L_playout
      k = k + 1
      Pktsum = 0
      j = 0, i = i − 1
    else if(i ≡ n) // when the value of i is equal to the maximum
                   number of transmitted packets
      r_k = j // the last r value recorded in r_table is written
              as the number of remaiing packets
    end if
end for
```

그림 3. r_table을 만드는 과정
Fig. 3. The process making r_table

균 재생량이 5000 byte일 경우, 재생량을 넘지 않는 최대 구간, 즉, {1500, 1500, 145, 1500}, {1500, 400, 1500, 370, 1100}으로 분할한다. 그리고 분할 된 각각의 구간에 있는 패킷의 개수를 *r*값 테이블에 기록하게 된다. 즉, 이러한 *r*값을 시간에 따라 적용 했을 경우, 단위 시간동안 생성되는 토큰에 의해 유입되는 패킷의 양이 단위 시간 동안의 재생량과 비슷하거나 보다 적은 현상을 보이기 때문에 버퍼의 오버플로우를 방지 할 수 있다. 이런 과정을 거쳐 생성된 최종 테이블에는 구간 별 사용되어야 하는 *r*값과 그에 따른 *c*값이 기록되어 있다.

3.4 시간에 따른 적응적 r값 설정 방법

마지막으로 그림 2에서의 ④와 같이 토큰 소비에 따른 *r*값의 변경이 이루어지며 동작은 다음과 같다. 설정 된 *r*값만큼 패킷들이 토큰에 의해 임시버퍼에서 재생 버퍼로 유입된 경우 *r*값 테이블에서 그 다음 *r*값으로 변경하여 설정한다. 예를 들면, *r_table* = {4, 5, 4, 3, 6 ……} 이라고 하면, 첫 번째 패킷부터 네 번째 패킷이 임시버퍼에서 재생버퍼로 유입되는 동안은 토큰의 단위 시간당 생성량인 *r*값을 4로 설정하고, 그 이후 다섯 번째 패킷이 들어올 때부터 *r*값을 그 다음 값인 5로 설정한다. 마찬가지로 열 번째 패킷부터는 *r*값을 4로 설정하여 사용하게 된다. 이러한 기법을 통해 각기 다른 비디오 및 시간의 흐름에 따라 적응적으로 다른 *r*값을 사용할 수 있다. 또한 해당 *r*값에 의해 결정되는 *c*값이 각 구간별로 설정된다. 이로 인해 단위 시간당 재생되는 양에 근접한 비디오 패킷의 유입을 제어 할 수 있으며 언더플로우의 발생횟수를 최소화 시키면서 버퍼의 포화현상을 방지 할 수 있다.

Ⅳ. 시뮬레이션 기법

본 논문에서는 NS-2와 JSVM을 사용하여 시뮬레이션을 진행하게 되며 성능 분석으로는 오버플로우의 발생 빈도, 패킷 손실률, 시간에 따른 토큰버킷 파라메타의 변화 및 수신 영상의 *PSNR*등을 비교해 본다. 여기서 *PSNR*이란 동영상의 화질 정보를 수치적으로 표현한 값으로 두 영상에 대해 평균 수치를 확인함으로써 대략적인 화질의 차이를 알 수 있다. *PSNR*을 계산하는 식은 식 (7)과 같다.

$$PSNR = 10\log_{10}\left(\frac{MAX_I^2}{MSE}\right) = 20\log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right) \quad (7)$$

여기서 *MAXI*는 해당 영상의 최대 신호값이다. *MSE*(Mean Squared Error)는 오차제곱의 평균이다. 원영상과 수신된 영상에 대해 *PSNR*을 구할 경우 두 영상에 대한 오차가 크면 클수록 *PSNR*값은 낮아진다. 일반적으로 *PSNR*값이 30dB이 넘으면 두 영상의 차이를 눈으로 구분하기 어렵다. 그림 4는 본문에서 사용한 시뮬레이션 절차를 보여주고 있다. NS-2에서는 trace 기반의 시뮬레이션으로 진행한다. 여기서 비디오 trace는 실제 비디오 시퀀스로부터 중요한 정보들을 추출하여 텍스트 파일의 형식으로 저장한 후 그 데이터를 바탕으로 NS-2를 통해 전송하여 시뮬레이션 한다[1]. 이 비디오 trace 파일은 패킷의 전송간격, 프레임의 크기 및 프레임의 부호화 유형 등 실제 비디오 시퀀스로부터 추출되어 저장된 형태이다.

이러한 일련의 과정들은 MyEvalSVC를 참조하여 진행하였다[12,13]. 이 시뮬레이션 방법은 SVEF (Scalable Video-Streaming Evaluation Framework)에 기반을 두어 NS-2 시뮬레이션 환경으로 프레임워크를 확장한 것이며 제안하는 네트워크 구조 또는 프로토콜을 SVC, H.264에 있어 평가할 수 있는 방법이다. 그림 4와 같이 시뮬레이션의 대상이 될 원시 YUV 비디오 시퀀스를 JSVM을 이용하여 파라메타를 조정

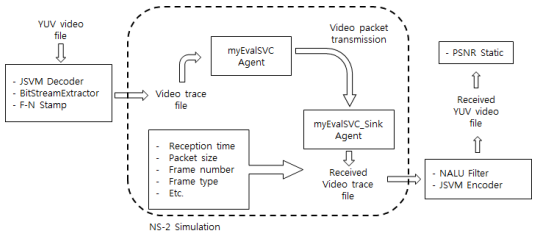


그림 4. trace 파일 기반의 시뮬레이션을 위한 비디오 파일 송수신 과정
Fig. 4. The transmitter-receiver process of video for the simulation based on trace file

하여 인코딩을 하게 된다. 본 시뮬레이션에서 대상이 된 비디오 시퀀스는 해상도나 화질에 확장성을 가지는 SVC가 아닌 단일 계층으로 인코딩을 하여 진행한다. 인코딩 후 JSVM에서 제공하는 BitStreamExtractor와 SVEF에서 제공하는 F-N Stamp를 사용하여 비디오 trace파일을 생성하게 되며, NS-2에서는 시뮬레이션 환경을 만들어 주게 된다. 이 환경에서 송신측 단말(myEvalSvc Agent)은 비디오 trace파일을 기반으로 한 패킷들을 생성해 주고 수신단측(myEvalSVC_Sink Agent)에서는 전송된 패킷들을 바탕으로 수신 trace파일(Received Video trace file)을 만들게 된다. 수신 trace파일에는 패킷의 전송 시간 및 수신시간, 패킷 사이즈, 프레임유형 등이 기록되어 있다. 이 수신 trace 파일 정보를 바탕으로 패킷의 손실 및 재생의 불연속성을 알 수 있으며 이후 NALUfilter 및 JSVM에서 디코딩 과정을 거치게 된다. NALU filter에서는 프레임 종속성으로 인하여 디코딩 할 수 없는 프레임과 너무 늦게 도착한 프레임들을 폐기하는 역할을 하게 되며 이 필터링 된 데이터를 디코딩 하였을 때 실질적으로 네트워크를 거쳐 최종적으로 수신된 비디오 시퀀스라고 할 수 있다. 이 비디오 시퀀스를 가지고 PSNR static툴을 이용하면 *PSNR*측정을 할 수 있으며 플레이어를 이용한 영상의 재생도 가능하다. NS-2를 이용한 시뮬레이션 시 송신측에서는 위에서 설명한 기법들을 이용하여 비디오 trace파일을 바탕으로 한 패킷을 생성하고 네트워크의 혼잡과 불규칙한 지터의 영향을 가정하여 비디오 패킷을 불규칙하게 전송한다. 또한 비디오 패킷 전송 전 토큰버킷의 적응적 파라메타 자동설정을 위한 비디오 정보 파일을 최초 1회 전송한다. 수신단에서는 자동으로 파라메타를 설정해주는 토큰버킷 기법을 수행하는 시스템을 구현하고 적용하게 된다.

V. 시뮬레이션 결과분석

사용된 비디오 시퀀스는 YUV 형식으로 JSVM 인코더를 사용하였으며 표 1과 같은 시뮬레이션 파라메타를 사용하였다[14].

사용된 세 개의 영상은 모두 단일 계층 비디오로 인코딩 하였다. 즉 공간적 확장성과 화질적 확장성은 가지지 않으며, GOP사이즈는 모두 8로 설정하였다. foreman_cif 비디오 시퀀스와 foreman_2cif 비디오 시퀀스는 각기 다른 해상도를 가지는 영상이며 하나는 cif 해상도를 가지고 다른 하나는 2cif 해상도를 가진다. 또한 foreman 비디오 시퀀스와는 영상 내 움직임 정보가 다른 soccer_cif 비디오 시퀀스를 가지고 시뮬레이션을 진행한다. 전송단에서 생성되는 각 비디오 시퀀스들에 대한 비디오 정보 파일은 비디오 시퀀스의 크기에 비례하는 크기를 가진다. 또한 수신측에서 비디오 정보 파일은 재생버퍼에 저장되는 것이 아닌 별도의 버퍼에 저장이 되며 토큰버킷 파라메타를 설정하는 용도로 쓰인다. 따라서 이후 성능평가에서 비디오 정보파일의 크기는 오버플로우/언더플로우, 패킷 손실 비율, PSNR 수치에 영향을 주지 않는다. 모든 비디오 시퀀스의 프레임 수는 300이며 30 fps의 재생 속도를 가진다. 따라서 재생 시 0.1 초당 3 frame에 해당하는 정보를 버퍼에서 꺼내 기록하도록 되어 있다. 토큰버킷의 파라메타는 r과 c로 이루어져 있는데 적응적 토큰버킷 기법 사용 시 이 값들의 초기 설정을 위해 수신단에서는 식 (3)을 사용하여 각 비디오에 대한 $r+c$값을 자동으로 설정해준다. 그 이후 시간에 따른 r값 또한 자동으로 설정하며 c값은 처음에 구

표 1. 사용된 시뮬레이션 파라메타들
Table 1. The used simulation parameters

Used simulation tool	NS-2, JSVM		
Simulation video sequences	foreman _cif	soccer _cif	foreman _2cif
Frame number	300		
GOP size	8		
Frame rate	30 frame/Sec		
Video resolution	CIF (352x288)		2CIF (704x288)
Total video packet size	518,146 byte	643,181 byte	670,675 byte
Video information file size	2,705 byte	2,840 bye	3,110 byte
Node number	Video sender 1, Receiver 1		
Replayed rate per unit time of video sequences (3 frame/0.1 sec)	5,181 byte	6,432 byte	6,707 byte
Initial r+c values(Tsum) of video sequences	23	24	24
Token generation rate (r/0.1 sec) in manual establishment	4 / 5 / 6 / 7		
Maximum packet size (Ptxmax)	1,500 byte		
Buffer size (Bmax)	30,000 byte		

한 r+c값에 적응적으로 설정된 r값을 뺀 값을 사용한다. 또한 수동 설정 시뮬레이션을 위한 r값은 4/0.1 sec부터 7/0.1 sec까지 임의의 값들을 직접 설정하며 그에 따른 c값을 사용하여 성능을 분석한다. 직접적으로 추가적인 노드를 생성하여 혼잡의 영향을 주지 않고 단지 무선 네트워크의 혼잡이나 지터의 영향을 가정하여 송신단에서는 트래픽 발생을 불규칙적으로 스케줄링 하도록 설정하였다. 한 패킷의 최대 사이즈는 1500 byte이며 수신측 재생버퍼의 최대 용량은 30000 byte이다. 정해진 r값에 의해 들어오는 패킷의 양이 초당 재생되는 양보다 많이 들어오게 되면 오버플로우의 위험성이 커진다. 즉, 수신측 재생 버퍼 내에 있는 비디오 정보량이 30000 byte가 넘게 되면 그 이후 유입되는 패킷에 대해서는 손실 처리를 하게 된다.

그림 5-1, 5-2, 5-3은 사용된 3개의 비디오에 대해 토큰버킷 파라메타를 수동 설정한 경우와 자동 설정한 경우에 오버플로우 발생횟수 및 패킷 손실률을 나타낸 그림이다. 그림 5-1에서 foreman_cif 영상의 경우, r값을 각 4, 5로 수동 설정하였을 때 오버플로우가 0회로 나타나며 r값을 6이상 올렸을 경우에는 단위 시간당 재생량보다 패킷의 유입량이 더 많아 오버플로우가 발생한다는 것을 알 수 있다. r값이 4인 경우와 5인 경우를 보면 두 경우 모두 오버플로우는 0회이지만 언더플로우를 보면 4인 경우 31회, 5인 경우 5회를 보이고 있다. 즉, 단위 시간당 생성되는 토큰의 개수를 4로 한 경우 지나친 패킷 유입의 제한으로 인해 오히려 버퍼가 고갈되는 현상이 심하게 나타날 수 있다는 것을 이 결과로 알 수 있다. 또한 r값이 6과 7인 경우 중에 7로 설정하였을 때 6으로 설정했을 때 보다 더 많은 언더플로우 현상이 일어나는 것을 볼 수 있다. 패킷 손실율 그래프의 경우, 버퍼의 포화로 인해 매우 많은 패킷들이 손실되었기 때문에 전체적인 비디오 패킷 정보량이 감소하여 발생하는 언더플로우 현상이라고 볼 수 있다. foreman_cif 영상의 경우 평균적인 r값을 사용 했을 경우엔 5값을 설정하는 것이 가장 최선의 파라메타 설정이라고 할 수 있다. 또한, 초기 전송된 비디오 정보 파일을 기반으로 토큰버킷 파라메타를 자동으로 설정할 경우 오버플로우는 0회, 언더플로우는 5회를 나타내고 있다.

수동으로 설정했을 경우 가장 좋은 결과를 보였던 r=5와 동일한 결과를 보인다. 즉, 이런 자동 설정 기법을 사용하였을 경우에는 파일 정보를 바탕으로 적절한 값을 시간에 따라 적절히 설정해주기 때문에 모든 해당 r값들을 모두 적용한 후에 가장 적절한 r값을 찾아주는 수동 설정 기법보다 적은 시간이 소모된다.

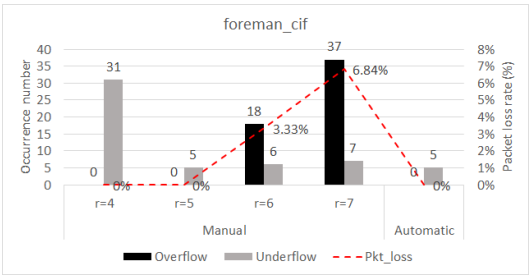


그림 5-1. foreman_cif 비디오 시퀀스에서 토큰버킷 파라메타의 수동 및 자동 설정방식에 따른 패킷 손실율과 언더플로우/오버플로우 발생 회수 비교
Fig. 5-1. The comparisons of packet loss rate and underflow/overflow occurrence numbers between manual establishment and automatic establishment of token-bucket parameter in foreman_cif video sequence

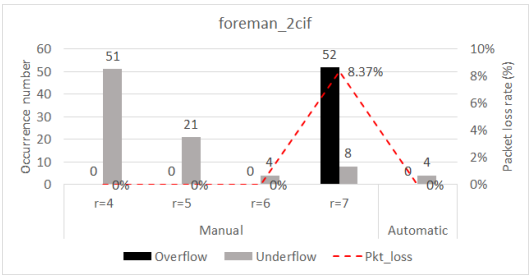


그림 5-2. foreman_2cif 비디오 시퀀스에서 토큰버킷 파라메타의 수동 및 자동 설정방식에 따른 패킷 손실율과 언더플로우/오버플로우 발생 회수 비교
Fig. 5-2. The comparisons of packet loss rate and underflow/overflow occurrence numbers between manual establishment and automatic establishment of token-bucket parameter in foreman_2cif video sequence

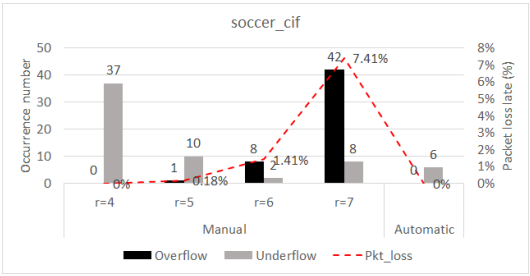


그림 5-3. soccer_cif 비디오 시퀀스에서 토큰버킷 파라메타의 수동 및 자동 설정방식에 따른 패킷 손실율과 언더플로우/오버플로우 발생 회수 비교
Fig. 5-3. The comparisons of packet loss rate and underflow/overflow occurrence numbers between manual establishment and automatic establishment of token-bucket parameter in soccer_cif video sequence

그림 5-2는 foreman_2cif 비디오 시퀀스에서 토큰버킷 파라메타의 수동 및 자동 설정방식에 따른 패킷 손실율과 언더플로우/오버플로우 발생 횟수를 비교한 그

림이다. foreman_2cif 비디오 시퀀스와 같은 경우 4, 5, 6으로 수동 설정한 경우에 오버플로우가 모두 0회로 나타났지만 *r*값이 4, 5인 경우 언더플로우가 많이 발생한다. 이 비디오의 경우는 *r*값이 6으로 설정되었을 때 가장 토큰버킷 기법의 효과가 나타난다. 물론 이 경우에도 자동으로 설정해주는 제안하는 기법을 사용했을 경우에 6으로 수동 설정한 경우와 동일하게 가장 좋은 결과를 얻을 수 있다.

그림 5-3는 soccer_cif 비디오 시퀀스에서 토큰버킷 파라메타의 수동 및 자동 설정방식에 따른 패킷 손실율과 언더플로우/오버플로우 발생 횟수를 비교한 그림이다. soccer_cif에서 *r*값을 5로 했을 경우에는 오버플로우가 1회이고 언더플로우는 10회가 발생되며, *r*값을 6으로 했을 경우 오버플로우가 8회 발생되고 언더플로우는 2회 발생된다. 토큰버킷은 군집적인 패킷을 제어하기 위해 사용하기 때문에 이 목적을 기준으로 하기 때문에 오버플로우가 적게 발생하는 *r*=5로 설정하는 것이 가장 유리하다. 반면에 자동설정 기법을 사용하였을 때에는 오버플로우가 0회, 언더플로우가 6회로 모든 오버플로우를 해결하면서 언더플로우의 발생횟수를 최소화하고 있다. soccer_cif 영상에서 수동설정 파라메타의 경우 토큰버킷을 사용해도 오버플로우를 완전히 제어하지 못하는 이유는 *r*값을 전체 시간 동안 고정된 값을 사용한 결과이다. 비디오마다 패킷들의 크기는 모두 일정하지 않으며 패턴 또한 불규칙하다. 따라서 *r*값을 하나의 비디오 스트리밍 서비스 동안 한 가지만 사용하기 보다는 상황에 맞게 유동적으로 바꿔준다면 더 좋은 효과를 낼 수 있다.

그림 6은 토큰버킷 파라메타의 수동 설정과 자동설정사이의 *r*값의 변화를 나타낸 그림이다. 수동으로 *r*값을 설정 할 경우 모든 시간동안 같은 *r*값을 사용하는데 비해 적응적인 토큰버킷 파라메타를 사용할 때는 전체 비디오 스트리밍 시뮬레이션 시간 동안 들어오는 패킷에 따라 적응적인 *r*값을 설정하는 것을 볼 수 있다. 즉, 토큰버킷 파라메타의 수동 설정 사용으로 발생하는 문제들을 토큰버킷 파라메타의 자동 설정을 통해 해결할 수 있다는 것을 알 수 있다. 전체적인 결과에서 보듯이 비디오 마다 적절한 *r*값이 다르게 나타난다. 또한, foreman_cif에서는 *r*=5를, foreman_2cif에서는 *r*=6을 설정해줘야 가장 좋은 효과를 볼 수 있었으며 soccer_cif 영상의 경우에는 고정적인 값을 사용하여 오버플로우를 방지하면서 언더플로우를 최소화하기에는 부족함이 있었다. 즉, 자동설정 파라메타를 가지는 토큰버킷 기법과 같이 비디오의 특성을 고려하여 각 비디오마다, 시간에 따라 적절한 파라메타를 사용해야 오버플로우를 방지하고 그에 따른 패킷 손실률을 줄일 수 있다는 것을 알 수 있다.

그림 7-1, 7-2, 7-3은 각 영상들에 대해 *PSNR*을 나타낸 그래프들이다. *PSNR* 계산 시 손실된 프레임 위치에는 이전 프레임 정보를 복사하여 모든 경우에 대해 300frame에 대한 *PSNR*을 측정하였다. 즉, 손실된 프레임에 대해서는 낮은 *PSNR*값을 보이게 된다. 그림 7-1에 있는 foreman_cif의 *PSNR*그래프의 경우 *r*값의 수동설정에서 *r*값을 6으로 설정했을 때와 7로 설정했을 때에는 오버플로우가 발생하는 경우이므로 패킷의 손실이 일어난다. 특히 *r*값을 7로 설정했을 때에는 6으로 설정했을 때보다 더 많은 패킷의 손실이 일어나게 된다. 그림 7-1에서 보듯이 전체적으로 *r*값을 7로 했을 때 낮은 *PSNR*을 보이는 프레임이 더 많이 존재하는 것을 알 수 있다.

반대로 *r*값의 수동설정에서 4, 5로 설정한 경우와 *r*값의 자동설정의 경우에는 오버플로우로 인한 손실이 일어나지 않아 모든 300frame에 대해서 높은 *PSNR*값을 나타내는 것을 볼 수 있다. 이와 같이 손실이 일어나지 않는 경우는 자동설정에 따른 *PSNR* 그림으로 대표하여 나타낸다. 물론 *r*=4의 경우에는 언더플로우 현상으로 인한 영상의 멈춤 현상이 일어나는 문제점이 있다. 만일 토큰버킷 기법 사용 시 임의의 값 *r*=6을 사용하여 모든 비디오 시퀀스를 수신하였다고 가정해보자. 이럴 경우 foreman_2cif에서는 오버플로우로 인한 패킷 손실을 방지하여 좋은 *PSNR*을 보이겠지만, foreman_cif와 soccer_cif 비디오 시퀀스의 경우에는 프레임의 손실이 일어나고 특정 프레임에서 낮은 *PSNR*을 나타낼 수 있다는 것을 알 수 있다. 따라서 수동적으로 하나의 *r*값을 설정해주는 경우보다는 제안한 적응적 토큰버킷 기법 사용 시 모든 비디오에 대해 손실을 방지하여 좋은 *PSNR* 수치를 보이는 것을 알 수 있다.

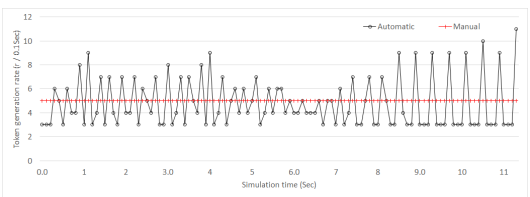


그림 6. 토큰버킷 파라메타의 수동 설정과 자동설정사이의 r 값 변화
Fig. 6. The variation of r value between manual establishment and automatic establishment of token-bucket parameter

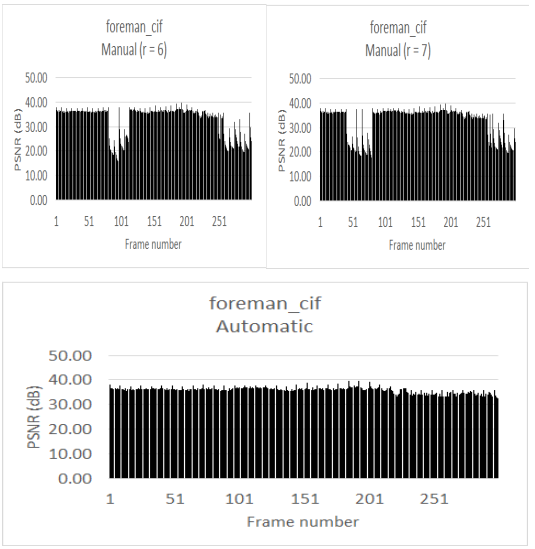


그림 7-1. foreman_cif 비디오 시퀀스에서 토큰버킷 파라메타의 수동 및 자동설정에 따른 PSNR 비교
Fig. 7-1. The comparisons of PSNRs between manual establishment and automatic establishment of token-bucket parameter in foreman_cif video sequence

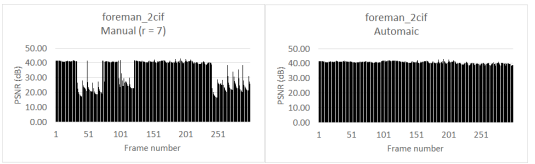


그림 7-2. foreman_2cif 비디오 시퀀스에서 토큰버킷 파라메타의 수동 및 자동설정에 따른 PSNR 비교
Fig. 7-2. The comparisons of PSNRs between manual establishment and automatic establishment of token-bucket parameter in foreman_2cif video sequence

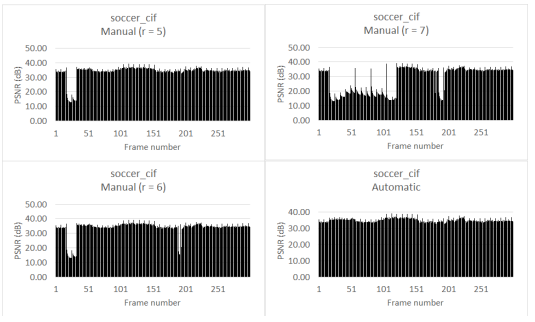


그림 7-3. soccer_cif 비디오 시퀀스에서 토큰버킷 파라메타의 수동 및 자동설정에 따른 PSNR 비교
Fig. 7-3. The comparisons of PSNRs between manual establishment and automatic establishment of token-bucket parameter in soccer_cif video sequence

Ⅵ. 결 론

비디오 스트리밍의 불연속성을 야기하는 버퍼 오버플로우를 해결하기 위한 적절한 토큰버킷 파라메타 설정은 비디오 특성마다 다르며 이를 적응적으로 설정하기 위해 송신단에서 제공하는 비디오 정보 파일을 이용한 적응적 토큰버킷 기법을 제시하였다. 이 기법을 적용했을 경우 스트리밍 서비스에 있어서 모든 비디오 데이터에 대해 오버플로우로 인한 패킷의 손실을 방지 할 수 있다는 것을 알 수 있었다. 기존의 토큰버킷 파라메타 수동 설정 방법을 사용하면 가장 적합한 값을 찾을 때에는 많은 시간이 소비되지만 제안하는 자동 설정 방법을 사용하면 적은 시간으로도 가장 적합한 값을 찾는 장점이 있다. 하지만 토큰버킷은 군집적인 트래픽에 대해 제어는 할 수 있어도 지연으로 인해 늦게 들어오는 패킷들은 제어를 할 수 없기 때문에 언더플로우 현상은 해결되지 않고 남아있는 것을 확인 할 수 있었다. 이러한 언더플로우 현상 또한 영상의 멈춤 현상을 발생시키며 재생의 연속성을 방해하는 요인이 된다. 따라서 향후 연구로는 적응적인 토큰 기법의 적용뿐만 아니라 버퍼의 언더플로우 문제까지 함께 해소하여 사용자 체감 품질을 높일 수 있는 기법과 제안 방법을 SVC(Scalable Video Coding) 기반의 라이브 비디오 시퀀스에 적용하는 연구를 진행할 계획이다.

References

[1] Y.-M. Jang, *The application of NS-2 Network Simulation*, Hongrung Publishing Company, 2008.

[2] J. Jo and J. Kim, "Synchronized one-to-many media streaming employing server-client coordinated adaptive playout control," *J. KICS,* vol. 28, no. 5c, pp. 493-505, May 2003.

[3] J. Koo and K. Chung, "Adaptive rate control of improving the QoE of streaming service in broadband wireless network," *J. KICS,* vol. 35, no. 2, pp. 334-344, Feb. 2010.

[4] D. Suh, I. Jang, and S. Pack, "A video bitrate adaptation algorithm for DASH-based multimedia streaming services to enhance user QoE," *J. KICS,* vol. 39B, no. 6, pp. 341-349, Jun. 2014.

[5] S.-H. Park and W.-G. Oh, "Real-time rate

control with token bucket for low bit rate video," *KIICE*, vol. 10, no. 12, pp. 2315-2320, Dec. 2006.

[6] P. Shin and K. Chung, "Token bucket-based scheduling mechanism for multimedia services in 3GPP LTE system," *JKIIT*, vol. 11, no. 1, pp. 101-110, Jan. 2013.

[7] Y.-C. Chen and X. Xu, "An adaptive buffer allocation mechanism for token bucket flow control," *IEEE VTC Fall*, vol. 4 pp. 3020-3024, Sept. 2004.

[8] R. Kim and H. Ryu, "The multi-queue bandwidth regulation method for multimedia QoS," *KIISE Fall Conf.*, vol. 2, pp. 469-471, Korea, Nov. 2005.

[9] V. S. Bagad and I. A. Dhotre, *Computer networks*, Technical Publications Pune, pp. 98-101, 2010.

[10] H.-N. Lee and D.-H. Kim, "The token bucket scheme to solve buffer overflow of video streaming in wireless network," *J. Digital Contents Soc.*, vol. 3, Jun. 2015.

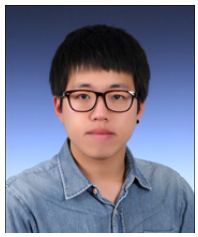
[11] N. Kim and H. Hwang, "Spatial and temporal resolution selection for bit stream extraction in H.264 scalable video coding," *J. Korea Multimedia Soc.*, vol. 13, no. 1, pp. 102-110, Jan. 2010.

[12] C.-H. Ke, "myEvalSVC : An integrated simulation framework for evaluation of H.264./SVC transmission," *KSII Trans. Internet and Inf. Syst.*, vol. 6, no. 1, pp. 378-393, Jan. 2012.

[13] C.-H. Ke, *How to do H.264 SVC transmission simulations*(2014), Retrieved Aug, 2014, from http://csie.nqu.edu.tw/smallko/ns2/svc.htm

[14] JSVM Software Manual, 2011.

이 현 노 (Hyun-no Lee)



2014년 2월 : 강원대학교 IT대학 전기전자공학부 공학사

2014년 3월~현재 : 강원대학교 IT대학 전기전자공학부 석사과정

<관심분야> 차세대 이동통신 및 차세대 무선 네트워크

김 동 회 (Dong-hoi Kim)

한국통신학회 논문지 (통신이론 및 시스템) 제38A권 제4호 참조

남 부 희 (Boo-hee Nam)

1983년 Iowa State University 공학박사, 현재: 강원대학교 IT대학 전기전자공학부 교수

2001년 8월~2002년 7월 : 미국 Virginia Tech 방문연구원

2006년 2월~2008년 1월 : 한국과학재단 전문위원

<관심분야> 통신 네트워크, 신경망, 영상처리

박 승 영 (Seung-young Park)

2002년 8월 : 고려대학교 전파공학과 박사

2003년 4월~2005년 12월 : 삼성전자종합기술원 책임연구원

2006년 1월~2007년 2월 : 미국 퍼듀대학교 박사후연구원

2012년 1월~2013년 1월 : 미국 퍼듀대학교 방문교수

2007년 3월~현재 : 강원대학교 전자통신공학과 부교수

<관심분야> 스몰셀 네트워크, 무선자원관리, 신경망