# 패킷 손실에 강인한 중복 비디오 패킷 전송 기법

정회원 서 만 근*, 정 요 원*, 서 광 덕**, 김 재 균*

# Duplicate Video Packet Transmission for Packet Loss-resilience

Man-keon Seo*, Yo-won Jeong*, Kwang-deok Seo**, Jae-kyoon Kim* ***Regular Members***

요 약

패킷 손실망을 통한 비디오 전송시 중복 패킷 전송에 의하여 복구에 소요되는 불필요한 시간지연을 방지하고 손실된 패킷에 대한 강인한 복구 성능을 얻을 수 있다. 그러나 이 방법의 단점은 중복적인 데이터 전송으로 전송 데이터량이 증가하여 망에서의 트래픽량을 증가시키는 것이다. 본 논문에서는 중복 전송 기법을 위한 효과적인 중복 데이터 생성, 패킷화 및 전송원리를 제안한다. 제안된 방법은 영상 재생을 위해 필요한 중요한 부호화 정보만을 중복 패킷화하여 전송함으로써 중복 데이터 생성을 위해 필요한 추가 비트량을 감소시킨다. 또한 중복 데이터 전송을 위해 필요한 패킷화 과정에서 이전 영상의 중복 데이터를 현재 영상의 패킷에 포함시켜 패킷화 하는 Piggyback 패킷화 개념을 도입하여 패킷 오버헤드를 급격히 감소시킨다. 실험결과를 통해 제안된 중복 패킷 전송 방법은 단일 패킷 전송 방법에 비해 적은 량의 추가적인 비트량으로 패킷 손실 환경에서 매우 우수한 복구 특성을 보임을 확인 한다.

**Key Words** : Video Transmission, Error Resilient Transmission, Video Packetization

ABSTRACT

The transmission of duplicate packets provides a great loss-resilience without undue time-delay in the video transmission over packet loss networks. But this method generally deteriorates the problem of traffic congestion because of the increased bit-rate required for duplicate transmission.

In this paper, we propose an efficient packetization and duplicate transmission of video packets. The proposed method transmits only the video signal with high priority for each video macroblock that is quite small in volume but very important for the reconstruction of the video. The proposed method significantly reduces the required bit-rate for duplicate transmission. An efficient packetization method is also proposed to reduce additional packet overhead which is required for transmitting the duplicate data. The duplicated high priority data of the previous video slice is transmitted as a piggyback to the data packet of the current video slice. It is shown by simulations that the proposed method remarkably improves the packet loss-resilience for video transmission only with small increase of redundant duplicated data for each slice.

## Ⅰ. 서 론

가입자 엑세스 네트워크의 광대역화로 인터넷 접속 속도가 비약적으로 증가함에 따라 멀티미디어 서비스에 대한 사용자들의 수요도 증가하고 있다. 인터넷을 통한 멀티미디어 스트리밍은 이제 기본적인 어플리케이션으로 자리를 잡아가고 있으며, 실시간 방송 서비스, 영상전화, 영상회의 등과 같은 대

* 한국과학기술원 전자전산학과(mkseo@core.kaist.ac.kr)
** 연세대학교 컴퓨터정보통신공학부(kdseo@dragon.yonsei.ac.kr, 교신저자)
논문번호 : KICS2005-01-017, 접수일자 : 2005년 1월 10일

화형 영상서비스에 대한 관심도 크게 높아지고 있다.

그에 따라 효과적인 영상 전송을 위해 RTP(Real-time Transmission Protocol)[7], RTCP(Real-time Transmission Control Protocol)[7], QoS(Quality of Service)[4] 제어와 같은 다양한 프로토콜들과 기법들이 제안되고 있지만, 영상 정보는 기본적으로 압축을 하여도 정보량이 크고, 이를 전달하는 네트워은 인터넷과 같이 사용자가 매우 많고 다양해서 가용 대역폭이 시간에 따라 심하게 변하고 실시간이라는 특성이 갖는 많은 제한요소로 인해 실시간 영상 전송 서비스의 구현은 여전히 어려움을 겪고 있다. 그 중 대표적인 것이 실시간 영상 전송은 지연과 에러(또는 손실)에 매우 민감하다는 것이다. 특히 인터넷과 같이 송신단과 수신단 사이에 패킷(packet)의 안정성을 보장해주지 못하는 유선네트워에서는 패킷의 비트 오류(bit-error)보다 패킷 손실(packet loss)이 수신측의 재생영상 화질에 큰 영향을 미치게 된다[1,4,13]. 따라서 영상을 보내는 송신측에서 영상을 전송하기 전에 채널에서의 패킷 손실을 극복할 수 있는 보호장치를 마련해 둘 필요가 있다.

통신을 위한 영상정보는 기본적으로 그림 1과 같은 과정으로 송수신된다. 송신단에서는 부호화(encoding)를 통해 영상의 정보를 압축하고, 압축된 영상정보를 네트웍을 통해 전송할 수 있도록 패킷화(packetization)를 하게 된다. 부호화 과정에서의 오류 제어(error control) 장치에는 오류 내성 부호화(error-resilient coding)가 있고, 패킷화 과정에서의 오류 제어 장치에는FEC(forward error correction), 재전송, 중복 전송(packet duplication)이 있다. 오류 내성 부호화는 패킷 손실에 의해 발생하는 비디오의 오류 전파(error propagation)와 군집성(burst) 오류의 발생을 최대한 억제하기 위해 압축 계층(compression layer)에서 행해지는 기법을 말한다. 표준 압축 기법에서 사용되는 오류 내성 부호화 기법에는 재동기 표시자(resynchronization marker), 데이터 분할(data partitioning), 역방향 가변길이 부호화(reversible variable length coding)와 같은데이터 복원 방식 등이 있다[2,10,11,15]. 이러한 기법들은 무선 환경에서처럼 비트 오류가 주로 발생하는 환경을 주된 적용 대상으로 삼기때문에 패킷 손실이 주된 오류인 인터넷 환경에는 적합하지 않다. FEC 방식은 비디오의 정보를 담고 있는 데이터 패킷에 중복적인(redundant) 데이터 패킷을 추가해서 보내고 일정 수 이상의 패킷을 수신하면 손실된 패킷을 완전히 복원할 수 있게 하는 장점을 가진다[12]. 그러나, 오류 보정을 위해 추가하는 패킷으로 인해 전송 비트율이 크게 증가한다는 단점이 있고, 오류 보정을 위한 지연시간을 줄이려면 오버헤더(overhead)가 크게 증가한다. TCP(Transmission control protocol)와 같은 재전송을 통한 손실 보정을 할 경우에는 필요 이상의 중복(redundant) 패킷을 전송하지 않아도 되므로 전체적인 전송률을 줄일 수 있지만, ACK 패킷 처리에 의해 왕복지연시간(round trip time)이 길어지기 때문에 네트웍상에서 실시간성을 보장하기 힘들다. TCP의 오류 제어 방식에서는 재전송된 영상데이터가 쓸모 없는 데이터가 되지 않게 하기 위해 수신측에서는 재전송이 완료되는 동안 재생을 멈추어야 하므로 그 만큼 단대단(end-to- end) 지연이 증가하게 되고 수신측 사용자는 정지된 영상과 증가된 지연으로 인해 불쾌감을 느끼게 된다[4,12].

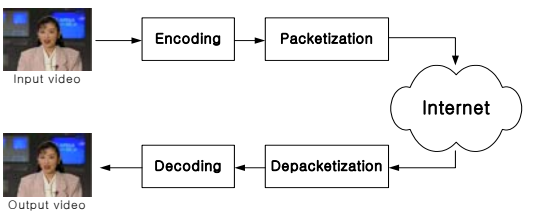



그림 1. 영상 통신의 기본 구조.

이처럼 여러 가지 오류 제어 방식이 있지만, 전송률과 지연을 최소화하면서 동시에 오류에 대한 강인성을 높이는 데에는 많은 어려움이 있으며 응용분야에 따라 적합한 방식을 선택하고 고안해야 할 필요가 있다. 본 논문에서는 저비트율, 저복잡도, 실시간성을 동시에 만족하는 오류 제어 기법과 패킷화에 초점을 맞추고 있다. 중복 전송 방식은 오류 복원을 위해서 추가적인 연산을 해주지 않아도 되므로 복잡도가 낮고, 지연 발생이 거의 없는 방법이므로 저복잡도, 실시간성이라는 요구 조건에도 적합하다. 그러나, 중복 전송 방식은 동일한 정보의 데이터를 이중으로 전송해야 하므로 전송해야 할 데이터량이 많아지므로 네트웍 혼잡을 가중시키고, 네트웍의 혼잡은 패킷 손실률을 높이므로 악순환의 반복을 겪을 수 있다. 그러므로 모든 정보를 중복해서 보내는 것보다 중요도가 높은 정보만을 선별하여 중복 전송하는 것은 좋은 방법이 될 수 있다.

본 논문에서는 중복 전송될 데이터 량을 최소로 조절하기 위해 데이터 분할 방식의 기본 개념을 채택한다. 데이터 분할 방식에서는 high priority data와 low priority data로 데이터를 분할하는데, high

priority data는 전체 데이터량에서 작은 비율을 차지하지만, 영상의 복원에 있어서는 매우 중요한 정보이다[2,5]. 모든 데이터를 중복 전송하지 않고 high priority data만을 중복 전송하게 되면 전송 데이터량을 크게 줄일 수 있다.

중복 전송 방식을 사용하게 되면 중복되는 데이터 외에도 데이터를 전송하기 위한 패킷 헤더(RTP, UDP, IP 헤더)의 량도 두 배 이상 많아지므로 중복 데이터만을 전송하더라도 저비트율(low bit-rate) 통신일 경우는 여전히 전송해야 할 데이터량이 부담스러울 정도로 많다. 패킷에 포함되는 헤더의 크기가 전체 데이터의 크기와 비교할 때 무시할 수 없을 정도로 크기 때문이다. 패킷의 크기가 작고, 저비트율 통신일수록 중복되는 패킷의 헤더를 줄이는 것또한 매우 중요하기 때문에 패킷 헤더를 효과적으로 줄이기 위한 새로운 패킷화 방식에 대해서도 제안한다.

데이터 분할을 할 경우와 이중 데이터(duplicate data)를 전송할 경우 다양한 패킷화 방식이 가능한데, 본 논문에서는 각 패킷화 방식에 따른 오류에 대한 화면 재생 확률 성능이 어떻게 차이가 나는지에 관한 확률에 근거한 분석을 통해 가장 최적의 패킷화 방식을 고찰한다.

## Ⅱ. 중복패킷 전송 기법

### 2.1 중복 패킷 전송 개요

대부분의 오류 내성 부호화는 비트 오류를 고려한 오류 제어 방법이기 때문에 패킷 손실이 빈번하게 발생하는 인터넷 환경에는 적용하기 힘들고, 재전송 기법은 지연으로 인해 실시간 통신 시스템에 적용하기 어려운 점이 있다.

그러나, 중복패킷 전송 방식은 실시간성을 보장해 주면서, 패킷 손실로 인한 오류를 막을 수 있는 방법이다. 중복패킷 전송의 기본 개념은 송신단에서 동일한 데이터를 담고 있는 패킷을 여러 번 전송함으로써 패킷 손실 확률을 낮추는 것이다. 송신단은 수신단으로부터 피드백 정보를 기다릴 필요가 없고, 수신단도 송신단으로부터 패킷을 받는 즉시 재생할 수 있으므로 오류 정정을 위한 추가적인 지연이 생기지 않는다. 그러나, 동일한 정보를 담고 있는 패킷을 중복 전송하게 되면 전송률이 증가하며, 네트웍의 부하가 가중된다. 네트웍 부하의 가중은 채널 상황을 더욱 악화시키므로, 중복 패킷에 할당되는 비트률을 조절할 필요가 있다. 중요도가 높은 정보

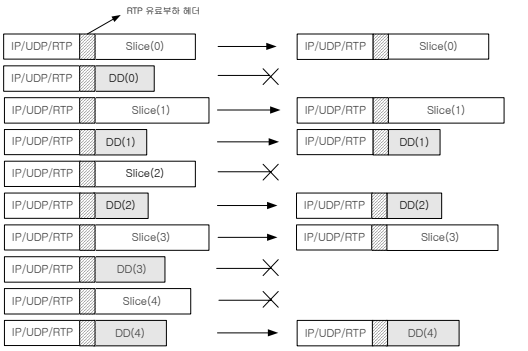



그림 2. 중요정보 중복 전송 방법의 예

만을 추출하여 중복 전송하게 되면 비교적 적은 비트율을 유지하면서 오류 복원률을 높일 수 있다. 영상 압축 정보의 경우에는 재생 화질에 영향을 많이 주는 데이터일수록 중요한 데이터로 볼 수 있고, 이러한 데이터만을 선별하여 중복 전송하면 비교적 적은 비트율을 유지하면서도 오류에 대한 강인성 효과를 얻을 수 있다[3].

그림 2는 압축된 영상 데이터를 패킷 손실 채널을 통해 전송할 때 중복패킷 전송 기법의 기본적 개념을 보이고 있다. 보통 압축 영상 데이터는 slice 단위로 패킷에 실어서 보내게 되는데, 본 논문에서는 주로 1개의 GOB로 구성되는 slice를 패킷화할 것이다. 특정 slice는 다른 slice와 독립적으로 부호화되기 때문에 특정 slice가 전송중에 손실되더라도 다른 slice의 재생에 영향을 미치지 않는다. 또한 인터넷에서 영상을 전송하기 위해서는 비디오 스트림을 패킷화해야 하고 이를 위해 각 slice별로 RTP/UDP/IP및 RTP유료부하 헤더 정보를 생성하게 된다. DD (Duplicate high-priority Data)는 slice의 중요 정보를 의미하며 Slice(k)는 k번째 전송되는 slice를 의미하며 DD(k)는 Slice(k)의 중요 정보를 의미한다. 그림은 Slice(0)에서부터 Slice(4)까지를 패킷화해서 손실성 채널을 통해 수신단에 전송하는 것을 나타내고 있다. 전송 과정에서 Slice(2), Slice(4), DD(0), DD(3)에 해당하는 패킷이 손실 되었고, 수신단에서는 Slice(0), Slice(1), Slice(3)와 DD(1), DD(2), DD(4)에 해당하는 패킷을 수신하였다. 손실된 Slice(2), Slice(4)에 해당하는 위치의 영상을 중복 데이터 정보 DD(2)와, DD(4)를 이용하여 복원해 낼 수 있다.

### 2.2 비디오 패킷 및 중복 패킷의 구조

비디오 패킷은 그림 3과 같이 크게 패킷헤더와

RTP 유료부하(payload)로 구성된다[3,4]. RTP/UDP/IP 프로토콜을 사용할 경우 패킷헤더는 IP, UDP, RTP 헤더로 구성된다.

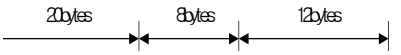



그림 3. RTP 패킷의 구조

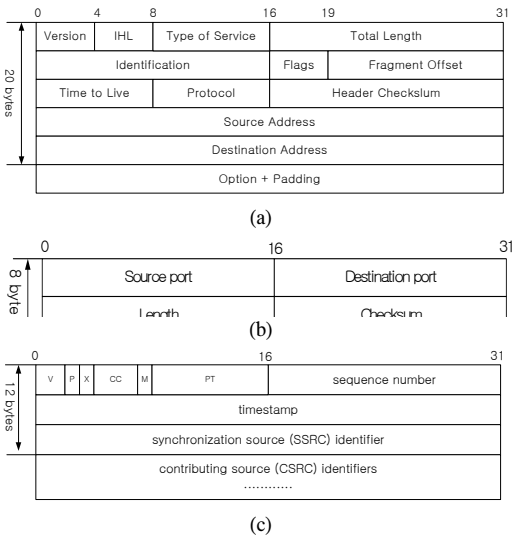



그림 4. (a) IP 헤더, (b) UDP 헤더, (c) RTP 헤더

인터넷망을 통한 실시간 영상통신에서는 RTP/UDP/IP의 프로토콜 조합을 이용한다. IP는 네트워크계층의 프로토콜로서 인터넷 주소를 이용하여 패킷을 목적지까지 보내는 프로토콜이고, UDP는 실시간 멀티미디어 데이터의 전송시에 사용되는 전송계층(transport layer)의 프로토콜이다. RTP(real-time transport protocol) 프로토콜은 응용계층의 프로토콜로서 실시간 응용에서 필요한 시간 정보와 정보매체의 동기화 기능을 제공한다. 이 프로토콜들을 이용하여 패킷화할 때는 각 프로토콜의 헤더 정보를 패킷에 포함시켜야 한다. 그림 4는 전송 패킷에 붙는 헤더의 종류와 각각의 구조를 나타내고 있다. IP 헤더는 20바이트(byte) 이상, UDP에 헤더는 8바이트, RTP헤더는 12바이트 이상의 크기를 갖는다. 따라서, 하나의 패킷은 총 40바이트 이상의 크기를 갖게 된다.

H.263+에는 선택적 기능으로 매크로블록 단위로 크기가 조절되는 slice가 있다. 또한, H.263+에는 GOB 시작부호와 GOB헤더로 시작하는 GOB계층이 있고, 이는 slice와 거의 동일한 기능을 한다. 본

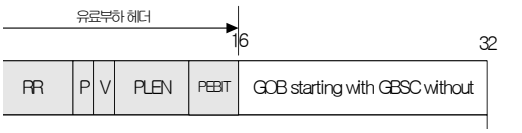



그림 5. 263+의 RTP 유료부하

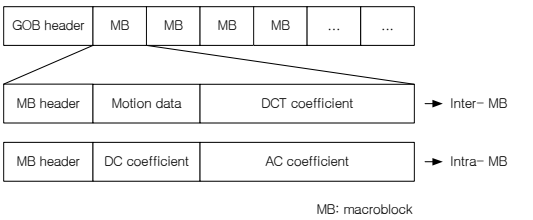



그림 6. H.263+의 GOB 구조

논문에서 GOB를 패킷화 단위로 이용할 것이며, 편의상 하나의 GOB를 하나의 slice로 간주한다.

그림 5는 H.263+의 RTP 유료부하 포맷의 구조를 나타내고 있다[8]. 이 그림은 GOB시작부호(GBSC)로 시작되는 패킷을 나타낸 것이다. 유료부하 헤더가 2바이트 첨가되었지만, GOB의 시작부호(GOB start code)의 앞부분에 위치해 있는 2바이트(all zero bit)가 생략되었으므로 RTP유료부하의 전체 데이터량은 부호기에서 생성된 원본 GOB의 데이터량과 같다.

유료부하 헤더 이후에는 GOB단위의 영상 데이터가 붙게 되며, GOB는 그림 6과 같이 GOB 헤더와 여러 개의 매크로블록(macroblock)들로 이루어진다. 매크로블록은 움직임 추정(motion estimation)과 보상(compensation)의 사용 여부에 따라 크게 화면내 부호화 블록(intra macroblock)과 화면간 부호화 블록(inter macroblock)으로 나누어 진다. 화면내 부호화 블록은 매크로블록 헤더와 DCT계수들로 이루어지고, 화면간 부호화 블록은 매크로블록 헤더 및 움직임 정보 그리고 DCT계수들로 이루어진다. 화면간 부호화 블록에서는 매크로블록 헤더와 움직임 정보가 영상의 재생에 큰 영향을 미치고, 화면내 부호화 블록에서는 INTRA_DC가 영상의 재생에 큰 영향을 준다. 그러므로 표준 압축 방식의 데이터 분할(data parititioning)과 마찬가지로 화면내 부호화 블록의 경우는 매크로블록 헤더와 INTRA_DC를 high priority data로, 화면간 부호화 블록에서는 매크로블록 헤더와 움직임 정보를 high priority data로 구성할 수 있다[5].

그러므로, 중복패킷에서의 GOB는 high priority data만을 이용하여 그림 7과 같이 구성할 수 있다.

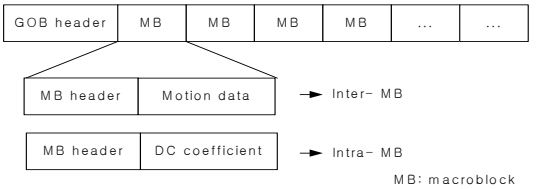



그림 7. 매크로블록 종류에 따른 중복패킷의 GOB 구조

비디오 스트림의 대부분을 차지하고 있는 DCT 계수의 상당부분이 제거되므로, 중복패킷의 크기를 크게 줄일 수 있다.

### 2.3 중복 데이터 생성

본 절에서는 H.263+를 중심으로 중복 데이터DD (Duplicate high-priority Data)를 어떻게 생성할 것인지 설명한다. DD는 데이터량은 작지만, 영상의 복원에 있어서 중요도가 높은 정보들로 이루어진다. DD를 생성할 때는 표준 압축에서 사용하는 데이터 분할 기법과 유사하게 중요도가 높은 데이터만을 추출해서 스트림을 형성하지만, 스트림의 세부 구조는 데이터 분할 방식과는 다르다.

H.263+에서 Advanced Prediction mode나 Deblocking Filter mode, PB-frames mode등의 옵션을 사용하지 않는다면, 매크로블록의 구조는 그림 8과 같이 구성된다[11].


그림 8. H.263+의 매크로블록의 구조

그림 8의 각 신텍스가 의미하는 바는 다음과 같다.

- COD(Coded macroblock indication): "0"은 매크로블록이 부호화 되었음을 표시해준다. 만일 "1"로 설정되어 있다면, 해당 매크로블록 내에는 부호화할 데이터가 없음을 의미한다.
- MCBPC(Macroblock type & coded block pattern for chrominance): 매크로블록의 부호화 방식과 색도신호 성분의 블록에 대한 부호화 패턴을 나타내는 부호이다.
- CBPY(Coded block pattern for luminance): 휘도성분 블록 중에서 INTRA_DC를 제외한 DCT변환계수 중 부호화할 계수가 있는 블록을 나타내는 부호이다.
- DQUANT(Quantizer information): QUANT의 변화를 나타내는 부호이다. 따라서 현재 매크로블록에 적용될 QUANT값은 이전 QUANT값에 DQUANT를 더한 값이 된다.
- MVD(Motion vector data): 현재 매크로블록의 움직임벡터와 움직임벡터의 예측값의 차이를 나타내는 부호로서, 화면간 부호화를 수행하는 모든 매크로블록은 MVD를 갖는다.

중복 데이터를 효율적으로 생성하기 위해서 필요없는 신텍스는 중복 전송에서 제외하거나 데이터 구조를 효율적으로 변경할 필요가 있다. 이를 위해 그림 8의 매크로블록 신텍스에 대한 수정 및삭제가 필요하고 본 논문에서 제안하는 방식은 아래과 같다.

#### 2.3.1 MCBPC의 수정

색도신호의 부호화 패턴 CBPC 는 "00"으로 정한다. 즉, 색도 신호에 대해서는 부호화하지 않음을 나타낸다. 원본 데이터의 부호화 방식이 INTER+Q, INTER일 때는 모두 INTER로 정하고, INTRA+Q, INTRA일 때는 INTRA로 정한다. INTER+Q와 INTRA+Q는 해당 매크로블록에 DQUANT 필드(field)가 있음을 의미하고, INTER와 INTRA는 해당 매크로블록에서 DQUANT 필드를 사용하지 않는다.

#### 2.3.2 CBPY의 수정

CBPY은 "0000"으로 정한다. 휘도성분 블록 중에서 INTRA_DC를 제외한 DCT변환계수 중 부호화할 계수가 있는 블록이 없음을 표시한다.

#### 2.3.3 DQUANT 의 제거

DD에서는 INTRA_DC를 제외한 DCT계수를 사용하지 않으므로 DQUANT의 값은 사용하지 않는다. 이를 위해 MCBPC의 수정이 선행되었다.

#### 2.3.4 INTRA_DC를 제외한 Block data의 제거

화면내 부호화 블록의 DD인 경우 INTRA_DC외의 모든 DCT계수를 제거하고, 화면간 부호화 블록의 DD에서는 모든 DCT계수에 해당하는 부분을 제거한다.

그림 9는 이와 같은 과정으로 생성된 DD의 매크로블록에 대한 간략화된 데이터 구조를 나타낸다.

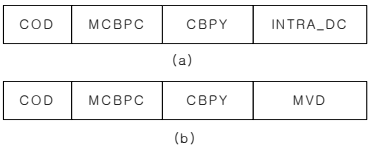



그림 9. DD의 매크로블록 데이터구조: (a)화면내 부호화 블록, (b)화면간 부호화 블록.

| 1프레임당 slice 개수 | 압축 영상 스트림의 데이터량 | DD의 데이터량 | DD의 상대적 데이터 크기 |
|---|---|---|---|
| 1 | 128kbps | 24.42kbps | 19.07% |
| 3 | 128kbps | 20.85kbps | 16.28% |
| 9 | 128kbps | 19.68kbps | 15.37% |
| 1 | 256kbps | 24.24kbps | 9.46% |
| 3 | 256kbps | 20.74kbps | 8.10% |
| 9 | 256kbps | 19.41kbps | 7.58% |

(a)

| 1프레임당 slice 개수 | 압축 영상 스트림의 데이터량 | DD의 데이터량 | DD의 상대적 데이터 크기 |
|---|---|---|---|
| 1 | 128kbps | 11.87kbps | 9.27% |
| 3 | 128kbps | 9.06kbps | 7.07% |
| 9 | 128kbps | 8.11kbps | 6.33% |
| 1 | 256kbps | 15.57kbps | 6.08% |
| 3 | 256kbps | 12.73kbps | 4.97% |
| 9 | 256kbps | 11.63kbps | 4.54% |

(b)

그림 10. 압축 영상 스트림과 DD의 데이터량 비교: (a) QCIF Foreman 시험 영상, (b) QCIF Akiyo 시험 영상.

그림 10은 압축된 영상 스트림과 지금까지 설명한 중복 데이터 생성 방법에 의해 발생된 DD의 데이터량을 비교한다. (a)와 (b)는 모두 초당 15 프레임의 QCIF 규격이며 (a)는 “Foreman” 영상에 대한 비교이고, (b)는 “Akiyo” 영상에 대한 비교이다. 프레임당 slice의 개수와 압축 영상 스트림의 데이터량을 변화시키면서 DD의 비율을 측정하고, 압축 영상 스트림의 데이터량에 대한 DD의 상대적 데이터 크기를 백분율로 나타내었다. DD의 데이터량은 영상의 특성에 의해 크게 좌우된다. “Forman” 영상과 같이 움직임이 비교적 큰 영상에 대해서는 데이터량이 많아지지만, “Akiyo” 영상과 같이 움직임이 비교적 작은 영상에 대해서는 데이터량이 작음을 알 수 있다. 그러나, DD에 할당된 데이터량은 압축 영상 스트림의 데이터량에 관계없이 크게 변하지 않는 특성을 가진다. 따라서, 압축 영상 스트림의 데이터량이 커질수록 DD의 상대적인 데이터 크기는 작아지는 것이다.

한편, 표준에서 제공하는 데이터 분할 방식을 이용하면 데이터 영역 표시자 등으로 인해 비트량이 증가하게 되며, 부호기 및 복호기의 복잡도도 증가하게 된다. 그러나, 제안된 방식으로 중복 데이터를 생성하게 되면 복호기 구조의 변화 없이도 high priority data만으로 손실 패킷에 대한 복구가 가능하다.

그림 11은 패킷 손실률 변화에 따른 단일 패킷 전송 방식과 중복 패킷 전송 방식에 대한 수신단에서의 재생 영상의 화질을 비교한 실험결과이다. 송신단에서의 전송률은 128kbps, 15fps로 일정하게 유지했다. 단일 패킷을 사용했을 때의 전송률은 순수한 영상 압축 데이터와 RTP/UDP/IP 헤더와 RTP 유료부하 헤더를 포함하여 128kbps가 되며, 중복패킷 방식의 전송률은 원본 영상 데이터, 중복 데이터, 원본 영상 데이터 및 중복 데이터를 패킷화할 때 붙이는 RTP/ UDP/ IP 헤더 및 RTP유료부하 헤더를 포함하여 128kbps가 된다.

그림 11에 의하면 패킷 손실 확률이 낮을 때는 단일패킷 전송방법이 중복패킷 전송방법 보다 더 높은 PSNR을 보인다. 그러나, 패킷 손실확률이 커짐에 따라 단일패킷 방법의 PSNR이중복 패킷 방법보다 급격하게 감소함을 볼 수 있고, 패킷의 손실확률이 20% 이상이 될 때부터 중복 패킷 전송방식의 PSNR이 단일 패킷 전송방식보다 높아짐을 알 수 있다. 이처럼 중복 패킷 전송방식은 패킷 손실에 대해 강인한 특성을 가진다. 그러나, 전송 데이터량의 많은 부분을 중복 데이터와 중복패킷의 RTP/UDP/IP헤더 정보에 할당해야 하므로 원본 영상 데이터에 적은 비트량을 할당할 수 밖에 없다. 이와 같은 이유로 일정한 전송률로 데이터를 전송한다고 할 때 중복 패킷 전송방식은 패킷 손실에 대한 강인성을 제공함에도 불구하고 낮은 패킷 손실률에서는 전체적인 화질의 열화를 가져온다.

결과적으로 중복 패킷 전송방식을 효과적으로 사용하기 위해서는 전체 화질을 일정수준 이상으로

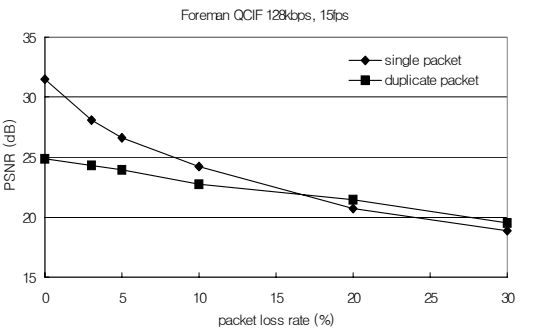



그림 11. 단일 패킷 전송방식과 중복 패킷 전송방식의 재생 화질 비교.

유지할수 있는 방법이 필요하고, 이를 위해서는 중복 패킷에 할당되는 비트량을 효과적으로 줄일 필요가 있다.

## Ⅲ. 개선된 중복 패킷전송 기법

본 장에서는 앞서 분석에 의해 제시된 중복 패킷 전송 기법의 문제점을 해결하기 위한 효과적인 방법들을 제시한다. 제안된 방법은 크게 두가지로 구성된다. 첫째는 중복 패킷 전송 기법에 요구되는 비트량을 효과적으로 줄이기 위한 Piggyback 패킷화에 관한 것이고, 둘째는 화면내 부호화 블록에 대한 중복 데이터 생성시 블록의 움직임의 크기(motion activity)를 고려하여 중복 데이터를 생성함으로써 비트량 감축 및 화질향상 효과를 동시에 얻는 것이다.

### 3.1 중복 데이터의 Piggyback 패킷화

그림 12(a)에서 보듯이 중복 패킷을 만들기 위해서 추가적으로 RTP/UDP/IP 헤더 및 유료부하 헤더를 삽입해야 한다. 그러나, 이러한 패킷의 헤더들은 중복 데이터 DD와 비교할 때 적지 않은 데이터량을 차지한다. 보통 RTP/UDP/IP 헤더를 모두 합하면 최소 40바이트의 크기가 되는데, 이러한 헤더들을 줄인다면 중복 데이터를 전송하기 위해 추가해 주어야 하는 비트량을 크게 감소시킬 수 있다. 그림 12(b)는 중복 패킷의 패킷 헤더 크기를 줄이기 위해 제안된 Piggyback 패킷화 방법이다. k번째 slice에 해당되는 중복 데이터 DD(k) 를 k+1번째 slice를 포함하는 패킷에 같이 실어 보냄으로써, DD를

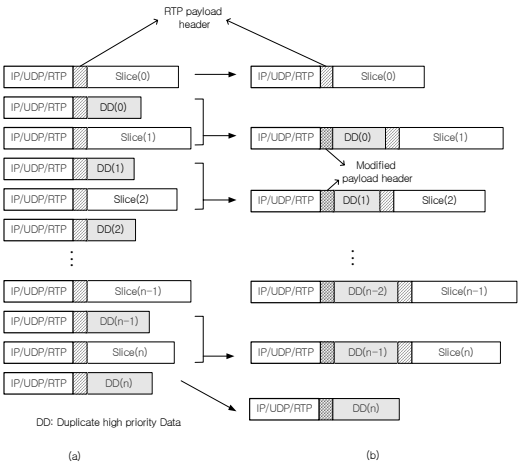



그림 12. (a) 중복 패킷 전송, (b) 제안된 방식(piggyback packetization)

전송하기 위해 필요한 추가적인 RTP/UDP/IP 헤더 및 유료부하 헤더를 사용하지 않아도 된다. 그러나, 수신단에서 한 패킷에 존재하는 DD와 원본 slice 정보를 분리하여 처리하게 하기 위해서, 제안된 패킷의 구조에서는 중복 데이터 정보의 크기를 나타내는 추가적인 헤더필드의 도입이 필요하다. 문제는 이러한 용도의 필드는 RTP헤더나 그림 5의 H.263+ RTP 유료부하 헤더에는 존재하지 않는다. 제안된 방법에서는 기존의 RTP 유료부하 헤더 구조를 그림 13과 같이 변형한다.

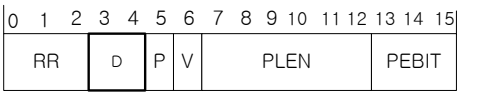



그림 13. 제안된 방법의 적용을 위해 변형한 RTP 유료부하 헤더 구조.

기존의 H.263+ RTP 유료부하 헤더에는 5비트의 예약(reserved) 비트가 있지만, 변형된 H.263+ RTP 유료부하 헤더는 이들 중에서 상위 3비트만 예약비트로 할당하고 하위 2비트를 D필드에 할당한다. 여기서 D필드는 중복 데이터 정보의 크기를 나타내기 위해 추가되는 확장헤더(header extension)의 크기로 정의되고 추가되는 확장헤더를 DLEN 필드로표시한다. D가 0이면 중복 데이터 정보와 DLEN필드는 존재하지 않으며, D > 0일 경우 D의 크기(바이트)를 갖는 DLEN 필드가 추가되고 DLEN필드의 값은 중복 데이터의 전체 크기를 지정한다.

그림 14는 중복 데이터를 효과적으로 패킷에 포함하기 위해 제안된 유료부하형식을 적용한 예를 보이고 있다. 이 예에서는 생성한 중복 데이터가 255바이트 이하의 크기인 252바이트임을 가정하여 D필드에는 "01"의 값을 넣는다. "01"은 DLEN의 크기가 1바이트(8비트) 임을 의미하며 중복 데이터의 크기가 252바이트임을 "11111100" 부호를 통해 표시한다. DLEN필드의 값은 중복 데이터 정보의 전체 유료부하의 크기를 나타낸다. 즉, 이 값은실제적인 영상에 관련된 중복 데이터 정보와 RTP 유료부하 헤더 크기까지를 모두 포함하는 값이다.

제안된 Piggyback 패킷화를 적용하게 되면, 유료

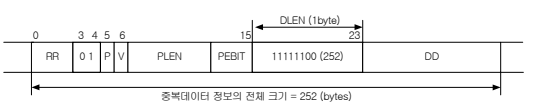


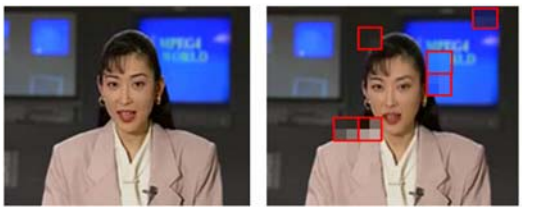

그림 14. 중복 데이터를 싣기 위한 RTP 유료부하 형식(payload format)의 적용 예.

부하 헤더에 확장헤더가 1~3바이트 정도 붙게 되지만, 중복 데이터 정보의 패킷화를 위해서 추가해 주어야 했던 IP/UDP/RTP헤더를 사용하지 않아도 되므로 하나의 slice를 전송할때마다 37~39 바이트 정도가 절감되는 것이다. 이 값은 DD의 비트량과 비교하면 결코 작지 않은 값이다. 또한 패킷화 단위가 작을수록 IP/ UDP/RTP헤더와 같은 패킷 헤더에 할당되는 비트량은 전체 전송률에 큰 영향을 미치게 되고, 제안된 방법의 효과는 더욱 크게 나타난다.

그러므로, 제안된 Piggyback 패킷화 방법에 의해 기존의 중복패킷 전송 방식에서의 패킷 손실에 대한 강인성 효과를 유지함과 동시에 패킷 헤더에 의한 오버헤더로 소모되는 비트량을 효과적으로 줄일 수 있다.

### 3.2 움직임 크기에 적응적인 중복 데이터 생성

H.263+ 압축 부호기에서는 화면간 오류의 전파(error propagation)를 막기 위해 강제 화면내 부호화 (Forced Intra MB coding)를 적용한다[10,11]. 기본적으로 매크로블록의 타입을 결정할 때 움직임이 적은 매크로블록인 경우 화면간 부호화 블록으로 결정된다. 그러나, 매크로블록 형태를 결정해주는 단계에서 화면간 부호화가 적합하더라도 강제 화면내 부호화를 적용해야 할 시점이라면 강제적으로 화면내 부호화 블록으로 부호화해야 한다. 이때 생성되는 중복 데이터 DD는 매크로블록 헤더와 INTRA_DC로 구성된다. 그러나 패킷이 손실되어 중복패킷에 포함된 DD에 의해 복구를 할 경우 움직임의 정도가 매우 적은 영상에서는 INTRA_DC값만으로 전체 블록값을 재생할 경우 만족스럽지 못한 화질을 얻을 수 있다. 그림 15는 움직임이 적은 테스트 영상인 Akiyo 영상의 29번째 프레임에 대한 실험결과이다. Akiyo영상의 특징은 인접 화면간 움직임 변화가 매우 작다는 것이다. 특히 화면의 배경 부분은 거의 고정되어 있다.



(a) (b)

그림 15. INTRA_DC 기반의 복원시 화질 열화:
(a) 패킷 손실없이 복원, (b) INTRAC_DC 기반의 패킷 손실 복원.

그림 15(a)는 패킷 손실없이 재생된 영상이고, 그림 15(b)는 손실된 slice 부분이 DD에 포함된 INTRA_DC 정보에 의해 복원된 영상을 나타낸다. 그림 15(b)에서 사각형으로 표시된 부분은 블록화 현상이 심하고 영상이 크게 손상되어 있다. 손실된 slice에서 사각형으로 표시된 부분의 화질열화가 두드러지는 이유는 표시된 블록들은 DD의 INTRA_DC값으로만 복원이 되었기 때문이다. Akiyo 영상은 움직임이 적은 영상이므로 DD의 움직임 정보를 이용하면 손실된 블록을 효과적으로 복원할 수 있다는 것이다.

따라서, 화면의 움직임이 적은 영상의 경우에는 INTRA_DC정보를 이용하는 것보다 재생된 이전 화면의 동일 위치의 블록을 그대로 가져와서 현재 화면에서 재생하면 더 좋은화질을 얻을 수 있다. 그러나, 움직임이 많은 영상의 경우에는 화면간 변화가 심하므로 이전 화면의 블록을 그대로 가져와서 재생할 때 보다 INTRA_DC 정보를 이용하여 재생하는 영상의 화질이 더 좋다. 따라서, 손실된 화면내 부호화 블록은 블록의 움직임 정도에 따라 적응적으로 복원 방식을 달리 할 수 있다. 블록이 이전 화면과 비교하여 움직임 정도가 크면 INTRA_DC를 이용하고, 블록의 움직임 정도가 작으면 이전 화면의 블록을 그대로 이용하는 개념이다. 이것은 송신단에서 DD를 적응적으로 만들어 줌으로써 가능하다. 제안된 방법의 적응적 중복 데이터 생성을 위한 구현 순서도는 그림 16과 같다.

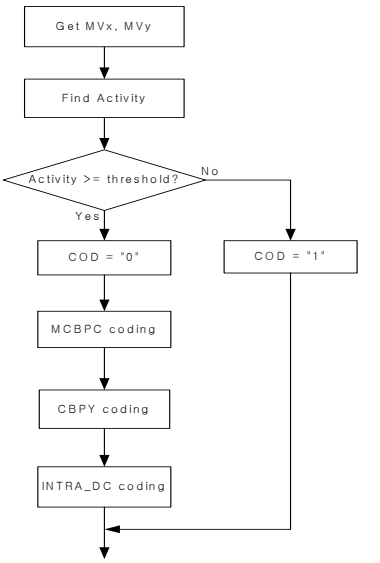



그림 16. 화면내 부호화 블록의 중복 데이터 생성 순서도.

그림 16의 순서도에서의 주요 단계는 다음과 같다.

1) 1단계: 매크로블록 당 움직임벡터 $MV_x, MV_y$ 을 구한다.
2) 2단계: 움직임 크기(Activity)를 구한다.
   $Activity = |MV_x| + |MV_y|$
3) 3단계:
   i) $Activity \geq Threshold$ 이면, DD에 INTRA_DC 계수를 포함시킨다.
   ii) $Activity < Threshold$ 이면, 이전 화면에서의 동일 위치의 블록으로 손실된 블록을 복원하기 위해서 현재 매크로블록 내에 부호할 데이터가 없음을 나타내는 필드인COD (coded macroblock indication) 값을 1로 세팅한다. COD 1비트 외에 다른 부호화 정보는 전송할 필요가 없으므로 DD 생성 시 약간의 비트량 감축 효과를 가져온다.

그림 17은 이와 같은 방식에 의해 생성되는 DD의 매크로블록 데이터 구조를 나타낸다.

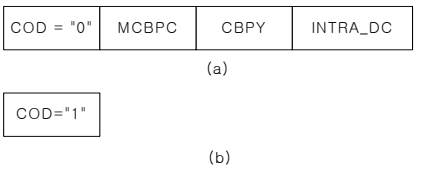



그림 17. DD의 매크로블록 데이터 구조:
(a) $Activity \geq Threshold$, (b) $Activity < Threshold$.

## Ⅳ. 중복패킷 전송 방식에 따른 패킷복원 확률 분석

본 장에서는 중복 데이터를 사용하여 중복패킷 전송 방식을 적용할 때 패킷화 방식에 따라 이론적으로 얻을 수 있는 화면의 복원 확률에 대해 비교 분석한다.

수학적 검증의 편리를 위해서 몇 가지 가정을 하기로 한다. 첫째, 영상 스트림을 high priority data와 low priority data 두 부분으로 나눌 경우 분할된 부분들의 데이터 크기의 총합은 분할되기 전의 데이터 크기와 동일하다고 가정한다. 둘째, 모든 패킷은 영상 압축 데이터 외에 패킷 헤더를 가지지 않는다고 가정한다. 패킷은 순수한 영상 압축 데이터로만 이루어진다고 가정하는 것이다. 세째, 패킷의 크기가 MTU(maximum transfer unit) 크기 이하로만 생성되면, 패킷의 크기에 따라 패킷 손실 확률이 변하지 않는다고 가정한다. 이러한 가정이 유효함은 실제적인 유선 인터넷 망에서 실험적으로 밝혀진 바 있다[6]. 이 가정에 의하여 패킷이 손실될 확률은 p로 가정한다. 넷째, 모든 패킷은 독립적으로 부호화된 slice 단위로 이루어진다고 가정한다. 즉, 다른 패킷의 손실과는 상관없이 수신된 패킷의 영상 데이터는 복원이 가능하다.

패킷 손실로 인해 해당 slice에 관한 재생 정보를 전혀 얻지 못했을 경우를 복원 불가능, slice의 정보 중에서 high priority data 정보만을 받아서 화면을 재생할 경우를 불완전 복원, 부호화된 slice의 모든 정보를 손실 없이 수신하여 화면을 재생할 경우를 완전 복원이라고 정의한다. $L_k$는 k번째 slice에 해당하는 영상이 복원 불가능한 경우를 나타내며, $I_k$는 k번째 slice가 불완전 복원되는 경우, $C_k$는 k번째 slice에 해당하는 영상이 완전 복원되는 경우이다.

### 4.1 분할된 정보를 각각 패킷화해서 전송하는 경우

모든 slice정보가 그림 18과 같이 high priority data와 low priority data로 분할되어 별도로 패킷화 되고, high priority data는 중복전송이 되는 경우이다.

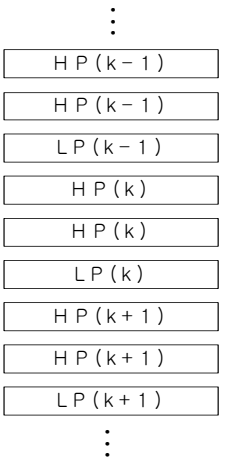



그림 18. 중복된 각각의 high priority data와 low priority data를 별도로 패킷화 및 전송

1) 복원 불가능할 확률:

$$P(L_k) = p \cdot p = p^2$$

$$P(I_k) = (1-p)^2 \cdot p + p \cdot (1-p) \cdot p + (1-p) \cdot p^2$$

2) 불완전 복원될 확률:

$$= (1+p)(1-p)p$$

3) 완전 복원될 확률:

$$P(C_k) = (1-p)\cdot p\cdot(1-p) + p\cdot(1-p)^2 + (1-p)^3$$
$$= (1+p)(1-p)^2$$

### 4.2 High priority data와 slice를 각각 패킷화하여 전송하는 경우

그림 19와 같이 하나의 slice를 하나의 패킷으로 전송하고, 해당 slice의 high priority data를 별도의 패킷으로 전송하는 경우이다.

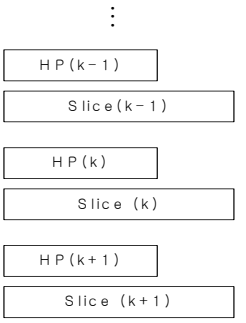



그림 19. High priority data와 slice를 별도로 패킷화 및 전송.

1) 복원 불가능할 확률:

$$P(L_k) = p\cdot p = p^2$$

2) 불완전 복원될 확률:

$$P(I_k) = (1-p)\cdot p$$

3) 완전 복원될 확률:

$$P(C_k) = 1-p$$

### 4.3 High priority data를 slice에 piggyback 하여 전송하는 경우

그림 20과 같이 slice 정보와 이전 패킷 slice의 high priority data를 하나의 패킷에 묶어서 전송하는 경우이다. 그림 20의 패킷화 방식은 헤더 정보를 갖지않는 다는 점을 제외하고는 그림 12에서 제안된 패킷화 방식과 동일하다.

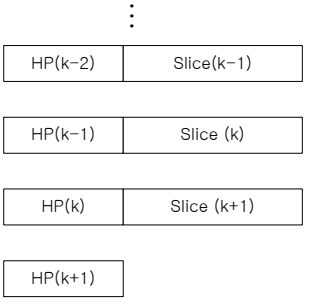



그림 20. Slice와 high priority data의 piggyback 패킷화 및 전송.

$k+1$번째 slice를 가장 마지막으로 수신된 slice로 가정할 경우, $k+1$번째 slice에 대해서는 다음과 같은 복원확률을 얻을 수 있다.

$$P(L_{k+1}) = p^2$$
$$P(I_{k+1}) = (1-p)\cdot p$$
$$P(C_{k+1}) = 1-p$$

$k$번째 slice에 대해서는 다음과 같은 복원확률을 얻을 수 있다.

1) 복원 불가능할 확률:

$$P(L_k) = P(L_k \mid L_{k+1})P(L_{k+1}) + P(L_k \mid C_{k+1})P(C_{k+1})$$
$$+ P(L_k \mid I_{k+1})P(I_{k+1}) = p\cdot p^2 + 0\cdot(1-p)$$
$$+ p\cdot(1-p)p = p^2$$

2) 불완전 복원될 확률:

$$P(I_k) = P(I_k \mid L_{k+1})P(L_{k+1}) + P(I_k \mid C_{k+1})P(C_{k+1})$$
$$+ P(I_k \mid I_{k+1})P(I_{k+1}) = 0\cdot p^2 + p\cdot(1-p)$$
$$+ 0\cdot(1-p)p = (1-p)p$$

3) 완전 복원될 확률:

$$P(C_k) = P(C_k \mid L_{k+1})P(L_{k+1}) + P(C_k \mid C_{k+1})P(C_{k+1})$$
$$+ P(C_k \mid I_{k+1})P(I_{k+1}) = (1-p)\cdot p^2 + (1-p)\cdot(1-p)$$
$$+ (1-p)\cdot(1-p)p = (1-p)$$

위의 식들로부터,

$$P(L_{k+1}) = P(L_k) = p^2$$
$$P(I_{k+1}) = P(I_k) = (1-p)\cdot p$$
$$P(C_{k+1}) = P(C_k) = 1-p$$

이며, $k-1$, $k-2$, $k-3$, …, 1번째 slice에 대해서도 동일한 복원확률을 얻을 수 있다.

표 1은 위의 4.1~4.3 절의 복원확률 결과를 비교하고 있다. 그림 18의 방식은 그림 19, 20의 방식에 비해 완전 복원 확률이 낮음을 알 수 있는데, 패킷 손실 환경에서는 slice 데이터를 분할해서 따로 패킷화하는 것보다 하나의 패킷으로 보내는 방법이

표 1. 중복 전송 방식에 따른 패킷 복원확률 비교

| | 그림 18의 방식 | 그림 19의 방식 | 그림 20의 방식 |
|---|---|---|---|
| 복원 불가 | $p^2$ | $p^2$ | $p^2$ |
| 불완전 복원 | $(1+p)(1-p)p$ | $(1-p)p$ | $(1-p)p$ |
| 완전 복원 | $(1+p)(1-p)^2$ | $1-p$ | $1-p$ |

손실에 더 강인한 방법임을 알 수 있다.

또한 그림 19처럼 하나의 slice와 그 slice에 해당하는 high priority data를 따로 패킷화해서 전송하는 방식과 그림 20처럼 하나의 slice와 그 이전 slice의 high priority data를 하나의 패킷으로 piggyback 해서 전송하는 방식이 동일한 복원확률을 가지고 있음을 알 수 있다. 그러나, 그림 19와 20의 패킷화 방식들은 패킷 헤더를 고려하지 않고 있는데, 실제적인 데이터 전송에서는 패킷 헤더가 필요하다. 패킷을 생성할 때마다 패킷 헤더가 추가되므로 패킷의 크기를 크게 하고 패킷의 수를 줄이는 것이 바람직하다. 그러므로 4.1절에서의 가정과 같이 패킷의 길이에 따라 패킷 손실 확률이 변하지 않는 채널 환경에서는 high priority data와 slice 정보를 piggyback하여 하나의 패킷으로 전송하는 것이 오류에 대한 강인성을 높이면서 전송률을 낮출 수 있는좋은 방법이다.

## Ⅴ. 모의실험 및 성능 평가

영상 압축은 H.263+ TMN8[9] S/W를 이용하였다. 모든 영상의 화면율(frame rate)은 15fps로 고정하였다. 오류전파를 막기 위하여 각 매크로블록이 적어도 30프레임에 한번씩 화면내 부호화 블록으로 부호화되도록 설정하였다. 비교 대상이 되는 단일 패킷 전송 방식을 위한 복호기단의 오류은닉 방법은 H.263+ S/W에서 기본적으로 제공하는TCON(temporal concealment) 방식을 사용하였다.

채널 환경은 비트 오류가 발생하지 않는 독립 랜덤 패킷손실(independent random packet loss) 환경이라고 가정을 했으며, 0%, 1%, 3%, 5%, 10%, 15%, 20%의 패킷 손실률에 대해 실험을 하였다. 시뮬레이터는 설정된 패킷 손실률에 따라 패킷을 폐기하게 된다. 그러나, 영상 스트림의 첫 번째 프레임은 I-화면이며 이 프레임의 손실 여부에 따라 이후에 나오는 영상이 큰 영향을 받을 수 있다. 따라서, 첫 번째 I-화면의 정보는 손실되지 않는다고 가정하며, 시뮬레이터는 패킷 손실률에 관계없이 첫 번째 프레임에 대한 패킷은 손실 시키지 않는다.

그림 21은 QCIF 규격의 Foreman영상에 대한 패킷 손실율에 따른 복원된 영상의 화질 비교에 관한 실험결과이다. 이 실험에서의 "piggyback DD" 방식에는 3.2절의 적응적 중복 데이터생성 방법이 적용되지 않았다. II절에서 설명한 "separate DD" 방식과 III절에서 설명한 "piggyback DD" 방식은 단일패킷 전송 방식인 "single data" 에 비해 패킷 손실률이 증가함에 따라 PSNR 감소가 급격하지 않음을 볼 수 있다. 즉, "single data" 방식에 비해 손실에 강인한 특성을 가진다. 그러나, "separate DD" 방식은 전체적으로 낮은 PSNR을 보이고 있다. 제안된 "piggyback DD" 방식은 손실률 3% 이상에서는 세 가지 전송 방식 중에서 가장 높은 PSNR추이를 보여주고 있다. "Piggyback DD" 방식은 "single data" 방식으로 패킷화할 때보다 0.62dB~3.86dB의 PSNR의 향상이 있다. 그림 22는 QCIF Coast 영상에 대한 실험결과이고, "piggyback DD" 방식은 "single data" 방식에 비해 0.51~2.08dB의 PSNR의 향상을 보인다.

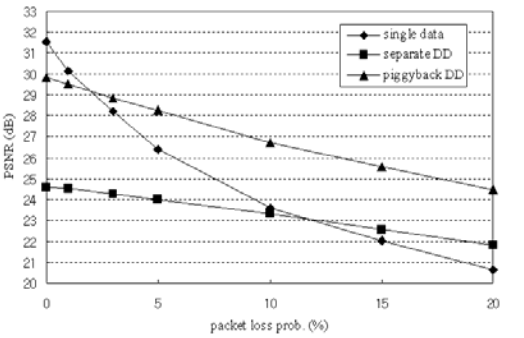



그림 21. 패킷 손실율에 대한 복원된 영상 화질 비교(Foreman 영상).

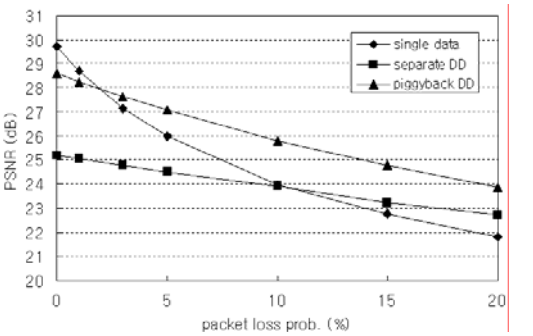



그림 22. 패킷 손실율에 대한 복원된 영상 화질 비교(Coast 영상).

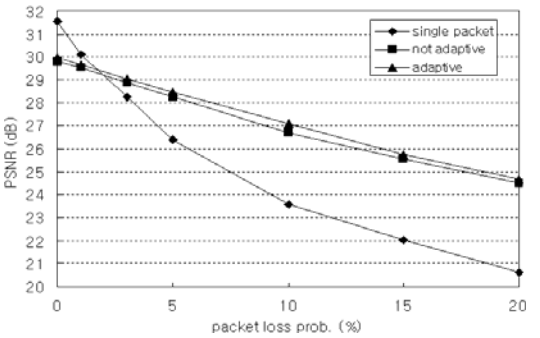



그림 23. 패킷 손실율에 대한 복원된 영상 화질 비교(Foreman 영상)

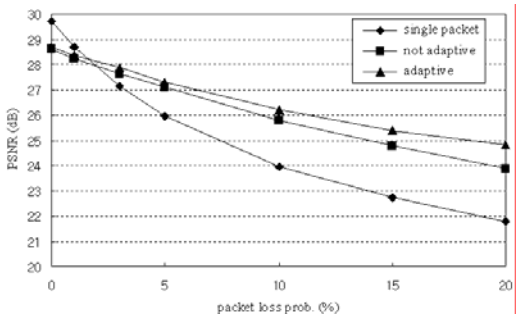



그림 24. 패킷 손실율에 대한 복원된 영상 화질 비교(Akiyo 영상).

그림 23와 24는 적응적 중복 데이터 생성 방법의 성능 비교에 관한 실험결과이다. DD를 3.1절에 제안된 “piggyback” 방법으로만 패킷화 했을 경우를 “not adaptive”로 표시하였고, 이에 더하여 DD 생성을 위하여 3.2절에 제안된 움직임 크기에 적응적으로 생성하는 방법을 추가적으로 적용했을 경우를 “adaptive”로 표시하였다. 이 실험에서 적응적 DD 생성 방법에 필요한 Threshold 값은 3으로 설정하였다. 실험결과로부터 움직임 정도에 따라 중복 데이터를 적응적으로 변화시켜주는 방법의 적용으로 약간의 화질의 향상 효과를 얻을 수 있음을 알 수 있다. 특히 그림 24는 그림 23의 Foreman 영상보다 움직임이 매우 적은 Akiyo 영상에 대한 실험 결과인데, 움직임이 적은 부분에 대해서는 적응적 중복 데이터 생성 기법을 적용함으로써 보다 더 큰 화질 향상 효과를 얻을 수 있음을 확인할 수 있다.

패킷 손실에 대응하기 위한 기존의 대표적인 채널부호화 방법이FEC(forward error correction) 기법이다. FEC에서는 오류 정정을 위해 더해주는 중복(redundant) 데이터를 임의로 조절할 수 있지만, 본 논문의 중복 전송 방식에서는 영상마다 high priority data의 크기가 다르므로 중복 데이터의 크기를 임의로 조절할 수 없다. 그러나, Foreman이나 Coast 같은 시험 영상을 128kbps의 저전송률로 전송할 때 high priority data가 전체 유료부하의 약 14%를 차지한다. 이에 상응하는 FEC 부호는 중복 데이터가 전체 유료부하의 약 14.28%를 차지하는 RS(7,6) 부호이다. 따라서, 공평한 비교를 위하여 모의 실험에서는 제안된 방식과 FEC의 중복 데이터의 크기를 비슷하게 만들기 위해 FEC는 RS(7,6)부호를 이용하였다.

실험에서는 네트웍에 의한 지연이나 영상 압축 부호화나 복호화 처리 과정에서 발생하는 지연(latency)을 무시하고, 오류 정정을 위해서 필요한 지연시간만을 고려하였고, FEC방식과 제안된 방식의 오류 정정을 위한 지연 시간을 1프레임 지연으로 고정하였다. 이를 위해서 중복 전송 방식에서는 한 패킷에 한 화면의 데이터 및 이전 화면의 DD를 싣고, QCIF의 경우 FEC에서는 패킷당 3 GOB 크기의 Slice를 싣게 된다. 이것은 FEC가 RS(7,6)부호화를 사용하기 때문인데 RS(7,6)부호화는 6개 이상의 패킷을 받아야만 오류를 복원할 수 있다. 6개의 slice는 두 프레임 해당하고, 결과적으로 오류 정정을 하기 위해서는 1 프레임의 지연을 허용해야 하는 것이다.

그림 25는 QCIF Foreman 영상에 대한 FEC와 제안된 중복패킷 전송 방식의 오류 복원 성능을 비교한 실험 결과이고, 그림 26은 동일한 실험을 QCIF Coast 영상에 대해서 수행한 결과이다. 그림에서 볼 수 있듯이 그림 25의 일부 패킷 손실률 구간을 제외하고는 제안된 이중전송 방식이 FEC방식보다 PSNR이 높음을 알 수 있다. Foreman 시험 영상에 대해서는 제안된 중복 전송방식이 FEC방식보다 최대 2.62dB 까지 PSNR이 높음을 알 수 있고Coast 시험 영상에 대해서는 1.35dB 정도 PSNR이 높음을 확인할 수 있다.

보다 실제적인 환경에서 성능을 평가하기 위하여 IP네트웍 시뮬레이터인 NIST Net 을 활용하여 실험

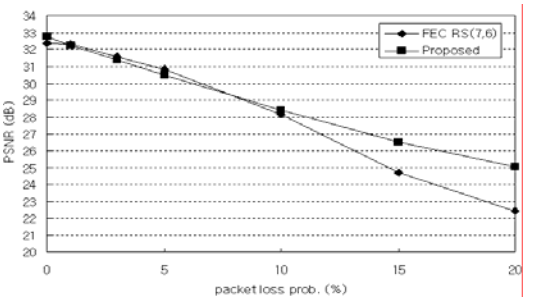



그림 25. Foreman 영상에 대한 FEC와 제안방식의 복원 성능 비교.

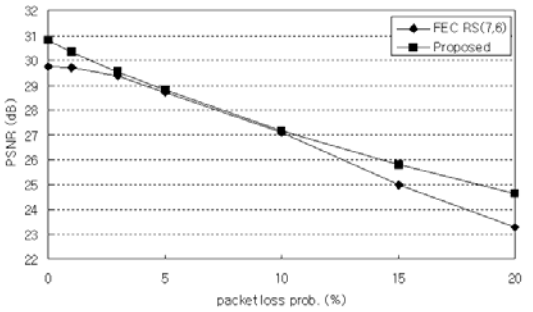



그림 26. Coast 영상에 대한 FEC와 제안방식의 복원 성능 비교.

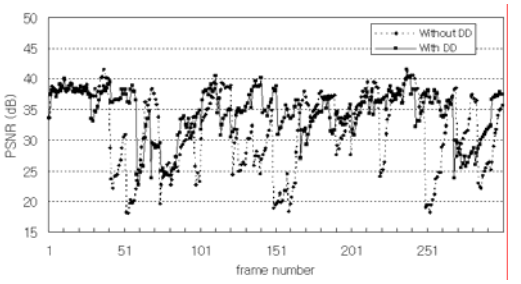



그림 27. NIST Net 상에서의 제안된 방법의 성능 비교.

을 하였다[14]. NIST Net을 이용하면 일정한 대역폭을 가지는 가상 네트웍을 생성할 수 있다. 본 실험에서는 이렇게 생성된 가상 네트웍을 통하여 영상을 송수신하고 수신된 각 프레임의 화질을 측정하였다.

NIST Net에서 네트웍의 가용 대역폭을 256kbps로 설정하였다. 또한 가상 네트웍에서 사용되는 라우터는 RED(random early detection)방식을 이용하도록 설정이 되어있다. 전송률을 가상 네트웍의 가용대역폭보다 크게 하여 전송하게 되면 패킷의 손실이 발생하게 된다. 실제 IP 네트웍에서 패킷 손실이 발생하는 많은 이유가 있지만, 본 실험에서는 이러한 이유가 패킷 손실에 가장 큰 영향을 미치는 요인으로 가정하였다. 실험에서는 실제 전송률을 263kbps(네트웍 가용대역폭의 103%)로 일정하게 하여 전송을 하였다. 여기서 전송률은 패킷 헤더(IP/UDP/RTP) 및 DD, 압축 영상 데이터를 모두 포함한 비트율이다. 전송률을 일정하게 유지하기 위해서 H.263+TMN8의 비트율 제어 방식(rate control)을 사용하였다. 테스트 영상으로는 QCIF Foreman을 이용하였다. 그림 27은 이 실험의 결과로 얻어진 수신측의 각 화면에 대한 재생 화질을 비교한 결과이다. DD를 사용하지 않은 경우 평균 PSNR은 31.8dB, 제안된 기법을 사용했을 경우 평균 PSNR은 34.5dB이다. 제안된 방법을 사용했을 경우 2.7dB의 PSNR 향상이 있음을 관찰할 수 있다. 또한 제안된 기법의 경우 화면간 PSNR 변동폭이 작아서 더 일정한 화질을 유지할 수 있기 때문에 주관적 화질면에서도 향상된 성능을 보여준다.

## Ⅵ. 결 론

실시간 영상 전송을 위한 여러가지 오류 제어 기법들이 제안되어 왔다. 비디오 계층에서는 다양한 오류내성 부호화 기법들이 있고, 네트웍 계층에서는 지연제한 재전송 기법, FEC기법, 중복 전송 방식들이 있다. 동일한 패킷을 여러 번 전송하는 중복패킷 전송 방식은 실시간성을 거의 완벽하게 보장해주는 오류 제어 방식이기는 하나 수신단의 전송률을 크게 높이게 되며, 이는 네트웍의 체증을 가중시키므로 보완이 필요하다. 영상의 복원에 큰 영향을 주면서도 데이터량이 전체 영상의 데이량에 비해 매우 작은 high priority data만을 중복 전송하게 되면 전송 데이터량을 크게 줄일 수있다. 또한 중복되는 데이터들의 전송을 위해 추가해 주어야 하는 헤더들을 효과적으로 삭제함으로써 중복 전송 방식의 성능을 크게 향상시킬 수 있다. 한 slice 정보의 중복 데이터 정보를 다음 slice와 묶어서 동일 패킷에 실어서 전송하는 방법을 사용했으며, 이를 위해 RTP 유료부하형식을 변형하였다.

또한 중복 데이터 전송 시 움직임이 적은 영상에 대한 효과를 증대시키기 위하여 움직임 크기에 적응적인 중복 데이터 생성 방식도 제안하였다. 패킷 방식의 통신에서는 FEC방식을 사용하면 오류에 대한 강인성이 증가한다. 오류 정정을 위한 지연시간도 비교적 작은 편이지만, 지연시간을 줄일수록 오버헤더가 늘어나서 실제 영상 데이터에 할당할 수 있는 비트량이 줄어든다. 그러나, 제안된 방법은 지연시간이 한 패킷 지연시간으로 고정되어 있어서 저지연 통신에 유용하며, 지연 시간을 줄이기 위해 추가적으로 더해주어야 하는 오버헤더도 적은 점에서 저전송률 통신에도 적합하다. 또한 제안 방식은 구현이 간단하고, 오류 정정을 위해 많은 연산을 수행하지 않아도 된다는 장점도 있다. 제안된 방식의 성능은 단일패킷 전송 방식, FEC 방식과의 비교를 통해 그 우수성이 확인되었으며, IP 네트웍 시뮬레이터인 NIST Net을 활용한 실험에서도 오류에 강인함이 입증되었다. 이러한 장점들을 고려했을 때 제안된 기법에 의한 중복패킷 전송 방식은 패킷 손실이 큰 네트웍 환경에서 실제적으로 적용이 가능한 방법이다.

## 참 고 문 헌


[1] J. Arnold, M. Frater, J. Zhang "Error resilience in the MPEG-2 video coding standard for cell based networks A review," *Signal Processing: Image Communication*, vol. 14, no. 6, May 1999, pp.607-633.

[2] A. Li, S. Kittitornkun, Y. Hu, D. Park, J. Villasenor, "Data partitioning and reversible variable length codes for robust video communications," *Proc. of Data Compression Conference(DCC)*, March 2000.

[3] S. Wenger, "H.264/AVC over IP," *IEEE Trans. Circuits Syst. Video technol.*, vol. 13, No. 7, July 2003.

[4] D. Wu, Y. Hou, Y. Zhang, "Transporting Real-time Video over the Internet: Challenges and approaches," *Proc. IEEE*, vol. 88, no. 12, Dec. 2000, pp. 1855-1877.

[5] K. Roh, K. Seo, J. Kim, "Data partitioning and coding of DCT coefficient based on requantization for error-resilient transmission of video", *Signal Processing: Image Communication*, vol. 17, no. 8, Sept. 2002, pp. 573- 585.

[6] J. Boyce, R. Gaglianello, "Packet loss effects on MPEG video sent over the public Internet", *ACM multimedia*, 1998, pp.181-190.

[7] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, "RTP: A transport protocol for real-time applications," RFC 1889, Jan. 1996.

[8] C. Bormann, L. Cline, G. Deisher, T. Gardos, C. Maciocco, D. Newell, J. Ott, G. Sullivan, S. Wenger, C. Zhu, "RTP payload format for the 1998 version of ITU-T Rec. H.263 video (H.263+)," RFC 2429, Oct. 1998.

[9] ITU-T, Video Coded Test Model Near-term 8 (TMN 8), 1997.

[10] ITU-T Recommendation H.263: Video coding for low bitrate communication, ITU, Geneva, March 1996.

[11] ITU-T Recommendation H.263+: Video coding for low bitrate communication, ITU, Geneva, January 1998.

[12] J. Kurose, K. Ross, "Computer Networking: A top-down approach featuring the Internet", Addison Wesley Longman Inc.

[13] I. Moccagatta, S. Soudagar, J. Liang, H. Chen, "Error-resilient coding in JPEG-2000 and MPEG-4", *IEEE J. Selected Areas Commun.*, vol. 18, no. 6, June 2000, pp. 899-914.

[14] http://www.itl.nist.gov/div892/itg/carson/nistnet/.

[15] ISO/IEC 14496-2, Coding of audio visual objects: Visual, ISO/IEC JTC1/SC29/WG11, Tokyo, March 1998.



서 만 근 (Man-keon Seo) 정회원

2002년 2월 한양대학교 전자공학과(공학사)
2004년 2월 한국과학기술원 전자전산학과(공학석사)
2004년 3월~현재 한국과학기술원 전자전산학과 박사과정
<관심분야> Scalable Video Coding, Video Communication

정 요 원 (Yo-won Jeong) 정회원

2000년 2월 한국과학기술원 전자전산학과(공학사)
2002년 2월 한국과학기술원 전자전산학과(공학석사)
2002년 3월~현재 한국과학기술원 전자전산학과 박사과정
<관심분야> Joint-Source Channel Coding, OS Optimization, Scalable Video Coding

서 광 덕 (Kwang-deok Seo) 정회원

1996년 2월 한국과학기술원 전기 및 전자공학과(공학사)
1998년 2월 한국과학기술원 전기 및 전자공학과(공학석사)
2002년 8월 한국과학기술원 전자전산학과(공학박사)
2002년 8월~2005년 2월 LG전자 선임연구원
2005년 3월~현재 연세대학교 컴퓨터정보통신공학부 조교수
<관심분야> 영상통신, 영상/음성 신호처리 및 압축, 멀티미디어 단말기 개발

김 재 균 (Jae-kyoon Kim) 정회원
한국통신학회 논문지 제 25권 제 7A호 참조
현재 한국과학기술원 전자전산학과 교수