이동 컴퓨팅 환경에서 실시간 데이타의 적응적 손실 복구 방법

(An Adaptive Packet Loss Recovery Scheme for Realtime Data in Mobile Computing Environment)

오 연 주 † 백 낙 훈 †† 박 광 로 ††† 정 해 원 ††† 임 경 식 ††††
(Yeunjoo Oh) (Nakhoon Baek) (Kwangroh Park) (Haewon Jung) (Kyungshik Lim)

요 약 최근 인터넷 뿐만 아니라 이동 컴퓨팅 환경에서도 멀티미디어 정보와 같은 실시간 데이타의 이용이 증가함에 따라, 실시간 데이타의 효율적이고 안정적인 전송 방법들이 요구되고 있다. 현재 실시간 데이타를 전송하기 위해서는 RTP(Realtime Transport Protocol)가 사용되는데, 이 프로토콜은 주로 UDP의 상위계층에서 동작하도록 설계되었기 때문에 안정된 전송을 보장받지 못하며, 따라서 패킷 손실을 피할 수 없다[1,2]. 특히, 제한된 무선 대역폭을 이용하는 이동 컴퓨팅 환경에서는 기존의 인터넷보다도 더 많은 패킷들이 손실될 수 있다[3]. 본 논문에서는 유선 및 무선망 환경에서의 실시간 데이타 전송시 발생되는 패킷 손실 특성을, 길버트 모델에 기초하여 확률적으로 분석하고, 이를 기반으로 새로운 복구 방법을 제안한다. 제안하는 실시간 데이타의 복구 방법에서는 수신자 측에서 분석한 패킷의 손실률에 따라, 송신자 측이 패킷에 부가하는 잉여 데이타의 양을 가변적으로 조절함으로써 사용 중인 네트워크의 패킷 손실 특성을 반영할 수 있다. 특히, 잉여 데이타들의 오프셋 값들을 비연속적으로 설정함으로써, 간헐적인 패킷 손실과 연속적인 패킷 손실 모두에 대처할 수 있는 특징이 있다. 이러한 특징은 기존의 네트워크에 비해 패킷의 손실률이 높고 급격한 트래픽 변이를 가지는 이동 컴퓨팅 환경에 잘 적용될 수 있다. 또한 본 논문에서 제안하는 방법은 확률 이론에 근거하기 때문에, 기존의 인터넷과 같은 유선망 환경에서도 비슷한 손실 특성을 가지는 경우에 적용 가능하다. 제안된 방법은 Mobile IP 및 RTP/RTCP를 이용하여 구현되었으며, 실시간 데이타의 전송에 효율적임을 실험적으로 보였다.

Abstract In these days, we have increasing demands on the real-time services, especially for the multimedia data transmission in both of wired and wireless environments and thus efficient and stable ways of transmitting realtime data are needs. Although RTP is widely used for internet-based realtime applications, it cannot avoid packet losses, due to the use of UDP stack and its underlying layers. In the case of mobile computing applications, the packet losses are more frequent and consecutive because of the limited bandwidth.

In this paper, we first statistically analyze the characteristics of packet losses in the wired and wireless communications, based on Gilbert model, and a new packet recovery scheme for realtime data transmission is presented. To reflect the transmission characteristics of the present network environment, our scheme makes the sender to dynamically adjust the amount of redundant information, using the current packet loss characteristic parameters reported by the receiver. Additionally, we use relatively large and discontinuous offset values, which enables us to recover from both of the random and consecutive packet losses. Due to these characteristics, our scheme is suitable for the mobile computing environment where packet loss rates are relatively high and varies rapidly in a wide range. Since our scheme is based on the analytic model form statistics, it can also be used for other network environments. We have implemented the scheme with Mobile IP and RTP/RTCP protocols to experimentally verify its efficiency.

† 정 회 원 : 한국전자통신연구원 네트워크연구소 연구원
oyj63246@etri.re.kr
†† 정 회 원 : 동국대학교 컴퓨터멀티미디어공학과 교수
nhbaek@dgu.ac.kr
††† 비 회 원 : 한국전자통신연구원 네트워크연구소 연구원
krpark@etri.re.kr
hw-jung@etri.re.kr
†††† 종신회원 : 경북대학교 컴퓨터과학과 교수
kslim@knu.ac.kr
논문접수 : 2000년 8월 10일
심사완료 : 2001년 7월 26일

1. 서 론

최근 인터넷 상에서는 멀티미디어 자료의 송수신이나 음성 통신과 같은, 실시간 데이타의 전송에 대한 수요가 증가하고 있다. 또한, 사용자의 이동성을 지원하는 이동 컴퓨팅 환경에서도 이러한 실시간 데이타의 송수신에 대한 요구가 증가하는 추세이다. 따라서, 유선망을 이용하는 인터넷 환경[1,2]과 무선망에 기초한 이동 컴퓨팅 환경 양쪽 모두에서 실시간 데이타 전송을 효율적으로 제공하기 위한 방법들[3,4,5]이 활발히 연구되고 있다.

현재 인터넷 상에서는 RTP(realtime transport protocol) [6]가 실시간 데이타 전송의 표준 프로토콜로 사용되고 있다. RTP는 전송되는 각 패킷에 고유 번호(sequence number)와 타임 스탬프(timestamp)를 부여함으로써, 수신 측에서 패킷 순서와 지터(jitter)를 보정하여 실시간으로 데이타를 재생할 수 있도록 해준다. 또, RTP와 함께 RTCP(realtime transport control protocol)를 이용하여 QoS(quality of service)에 대한 정보를 교환할 수 있는 기능도 제공한다. 하지만, RTP/RTCP 자체는 대부분의 경우 UDP 계층 위에서 동작하도록 설계되어 있으므로, 실시간 데이타 전송시 안정된 전송을 보장받지 못하며, 결과적으로 통신망 폭주 현상이나 네트워크 환경 특성에 의한 패킷 손실을 피할 수 없다[1]. 따라서, 응용 서비스의 종류에 따라 적합한 손실 복구 방법과 신뢰성 있는 서비스 제공을 위한 적절한 대응 방안이 필요하다[1,3,6,7].

한편, 이동 컴퓨팅 환경은 기존의 유선망에 비해 제한된 무선 대역폭을 가지며, 단말기의 이동이나 무선 링크의 특성에 의하여 트래픽 특성이 가변적으로 변화한다. 이 때문에 유선망에 비하여 전체 패킷 손실률과 패킷의 연속 손실률이 더욱 커지는 특성을 보인다.

이동 컴퓨팅 환경은 데이타 전송에 있어 특히 다음과 같은 제약점을 가진다[3,4].

- 무선 링크는 매우 빈약하고(fragile) 패킷 손실 경향이 높다. 따라서, 사용자가 만족할 만한 품질을 제공하기 위해서는 응용 계층이나 전송 계층에서 적절한 에러 복구 방법을 지원하여야 한다.
- 무선 링크는 제한된 대역폭을 가지며, 또한 이동 노드가 한 셀 내에서 사용할 수 있는 대역폭도 가변적으로 변화한다. 따라서, 이런 급격한 변화를 고려하는 손실 복구 방법이 제공되어야 한다.
- 이동 사용자는 물리적인 장애물이나 핸드오프(handoff)에 기인한 끊김 현상을 자주 느끼게 된다. 특히, 이것은 실시간 데이타 전송에 있어, 패킷 손실의 주요한 원인이 된다.

따라서, 이동 컴퓨팅 환경에서 실시간 데이타를 전송할 때에는, 급격한 트래픽 변화 및 비교적 높은 패킷 손실률에 적응성을 가지면서도 손실된 패킷들을 효과적으로 복구할 수 있는 방법이 필요하다.

본 논문에서는 이동 컴퓨팅 환경에서의 여러 특성을 고려하여, 실시간 데이타 전송 시에 발생하는 패킷 손실 특성을 확률적으로 분석하고, 이러한 확률적 접근 방법에 기초한 실시간 데이타 복구 방법을 제안하고자 한다. 제안하는 방법은 기존의 RTP/RTCP를 기반으로, 패킷 손실률이 가변적으로 변화하는 환경 및 다양한 패킷 손실의 특성에 대해서도 효율적으로 동작하도록 설계되었으며, 이에 따라 유선망 환경은 물론 이동 컴퓨팅 환경과 같은 무선망 환경에서도 적용이 가능하다.

본 논문의 2장에서는 실시간 데이타의 손실 복구 방법에 대한 기존의 연구들을 살펴본다. 3장에서는 패킷의 손실 과정 및 특성을 길버트 모델에 기초하여 확률적으로 분석하고, 분석된 결과를 이용하여 이동 컴퓨팅 환경과 같이, 다양한 패킷 손실 특성 및 전체적으로 높은 패킷 손실률을 가지는 경우에도 적용할 수 있는 손실 패킷 복구 방법을 알고리즘으로 제시한다. 4장에서는 3장에서 제시한 알고리즘을 구현하여 제안된 방법의 성능을 검증 및 분석한다. 마지막으로 5장에서 결론 및 향후 과제를 제시한다.

2. 기존 연구

본 절에서는 인터넷 상에서 실시간 데이타의 전송에 주로 사용되는 RTP/RTCP 표준에 대해 살펴보고, 전송시 발생된 패킷 손실에 대한 기존의 복구 방법들을 살펴본다.

RTP는 페이로드(payload) 타입에 따라 다양한 형태의 실시간 멀티미디어 데이타를 전송할 수 있도록 설계되었다. RTP 명세에 함께 정의된 RTCP는 RTP 데이타의 전송을 제어하거나 관련 정보를 제공할 수 있도록 설계되어 있다. RTP/RTCP에서의 실시간 데이타는 실시간으로 생성된 데이타를 다룬다는 의미이지, 전송 자체가 실시간으로 처리된다는 것은 아니다. 즉, 수신자는 전송 받은 데이타를 생성 시의 시간 간격과 동일한 간격으로 재생할 수 있음을 의미한다[6].

RTP는 단대단(end-to-end) 전달 서비스를 제공하지만, 일반적인 전송 프로토콜들이 제공하는 기능들을 모두 지원하는 것은 아니다. 예를 들어, RTP는 프로토콜의 멀티플렉싱(multiplexing)이나 체크섬(checksum) 기

능을 제공하지 않으므로, 구현 시에는 흔히 UDP를 하위 계층으로 설정하여, UDP의 해당 기능들을 이용한다. 특히, RTP 자체는 연결(connection)에 대한 개념을 사용하지 않으며, 사용되는 하위 계층에 따라 연결지향(connection-oriented) 또는 비연결(connectionless)의 어느 형태로든 사용할 수 있다. RTP의 실제 사용에 있어서는 전송의 효율을 위하여 주로 UDP를 하위 계층으로 사용하기 때문에, RTP 자체는 안정적인 전송을 보장받지 못한다. 특히 이동 컴퓨팅 환경에서는 인터넷 환경에 비하여 상대적으로 낮은 전송률과 높은 BER(bit error rate)의 링크 특성을 가지기 때문에, 실시간 데이타의 전송시 전체 패킷 손실률이 높아지는 것은 물론, 연속적인 손실의 경우도 증가할 것이다. 따라서, 이러한 패킷 손실 특성들에 따른 적절한 손실 복구 방법이 요구된다[1,2,3,4].

손실된 패킷을 복구하기 위해 적용할 수 있는 대표적인 방법들에는 ARQ(Automatic Repeat Request)와 FEC(Forward Error Correction) 방식, 잉여 정보를 이용한 부가전송(redundant data) 방식, 그리고 인터리빙(interleaving) 방식이 있다[1,2,8,11]. ARQ 방식은 손실된 패킷을 송신 측에서 재전송하는 것을 기반으로 하는 폐루프 메카니즘(closed-loop)이기 때문에 완전한 데이타의 복구를 지원한다는 장점이 있지만, 재전송으로 인한 지연이 증가하게 되므로 실시간 데이타의 복구 방법으로는 부적합하다[1,2,9,10,11,12]. FEC 방식은 손실된 패킷을 복구하기 위하여, 송신 측에서 원래의 정보를 가지고 XOR(eXclusive ORing) 연산으로 생성한 잉여 패킷을 추가로 전송하고, 수신 측에서는 이 잉여 패킷을 이용하여 손실된 원시 데이타 패킷을 복구하는 형태의 개루프(open-loop) 메카니즘이다. FEC는 최소한의 지연만으로 에러를 복구할 수 있지만, 패킷 손실이 연속적으로 발생하는 경우에는 패킷 복구율이 그리 높지 않다는 단점이 있다[1,9,10,11,12].

잉여 데이타 정보를 이용하는 부가 전송 방식에서는, 어떤 특정 데이타를 전송하는 패킷이 전송 도중에 손실될 경우를 고려하여, 다른 패킷에 해당 데이타를 원시 데이타보다는 좀 더 간략화된 잉여 데이타의 형태로 부가하여 전송한다. 그림 1에서 보는 바와 같이, n번째 패킷을 전송할 때, 앞서 전송된 $(n-1)$, $(n-2)$번째 패킷의 원시 데이타를 좀더 간략한 형태로 생성한, 잉여 데이타들을 함께 전송한다. 수신자 측에서는, n번째 패킷이 정상적으로 도착한 경우에는 해당 원시 데이타를 그대로 사용한다. 반면에 n번째 패킷이 손실된 경우에는 $(n+1)$, $(n+2)$번째 등의 패킷에 들어있는, n번째 패킷에 해당하는 잉여 데이타를 사용한다. 따라서, n번째 패킷이 손실되더라도 해당 잉여 데이타를 가진 다른 패킷들 중의 하나만 제대로 도착하면 n번째 패킷에 대응되는 부분은 복구가 가능하다. RTP에 잉여 데이타 방식을 적용시키는 경우에는 RTP 데이타 패킷의 헤더 확장(header extension) 부분에 잉여 데이타를 실어서 전송하는 것이 일반적이다[1,6,8]. 이 방식은 송신 측이 별도의 시간 지연 없이 패킷을 전송할 수 있고, 수신 측에서는 최악의 경우에도 다음에 도착할 몇 개의 패킷만을 기다리면 복구가 가능하다는 장점이 있다. 반면에 같은 데이타의 중복 전송에 따른 시스템의 추가 처리 시간과 전송량이 증가하고, 연속적인 패킷 손실에 대해서는 복구율이 그리 높지 않다는 단점이 있다.

인터리빙 방식은 전송할 패킷을 $m \times n$ 행렬 형태의 블럭으로 구성한 다음, 각 블럭 내의 패킷을 열 우선(column-major) 순서로 전송한다. 예를 들어, 4×4 블록은 1, 5, 9, 13, 2, 6, 10, 14, …의 순서로 패킷을 전송한다. 인터리빙 방식은 연속적인 패킷 손실을 분산시키는 효과는 가지지만, 송신 측에서는 패킷을 $m \times n$ 블럭으로 구성할 경우에 최대 $m \times n-(m+n)+1$ 개의 패킷이 전송 버퍼에서 대기 상태에 놓여 있어야 하는 단점을 가진다. 즉, 4×4 블럭을 쓰는 경우에는 최대

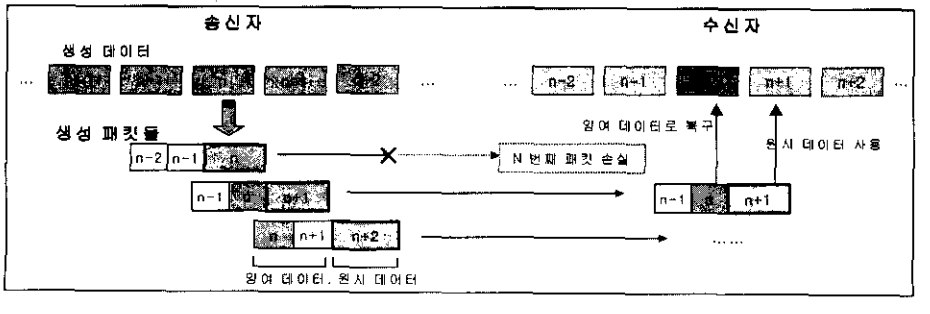


그림 1 잉여 데이타를 이용한 손실 패킷 복구 방법

$4\times4-(4+4)+1=9$ 개의 패킷들이 전송 버퍼에 대기 상태로 놓여 있게 된다. 또한 수신 측에서도, 수신 후 $\{m\times n-(m+n)+1\}+m\times n$ 만큼의 재생 지연 시간이 필요하다[1,11]. 이 사실은 인터리빙 방식이 특히 실시간 대화형 응용들에서 사용되기에는 부적합한 원인으로 지적되어 왔다[6,11,12].

이들 두 가지 기법을 함께 사용하는 방법들도 가능하다. 특히, 순수한(pure) 인터리빙 기법은 연속적인 패킷 손실률을 감소시킬 수는 있지만, 손실이 일어난 패킷 자체를 복구할 수는 없다. 따라서, 인터리빙 방식으로 전송되는 패킷들에 잉여 데이타를 부가함으로써, 패킷 자체의 복구도 가능하게 하는 시도가 있었다[11]. 이 경우에는 손실된 패킷의 복구율을 높일 수는 있지만, 인터리빙 방식의 한계점인 재생 지연 문제를 피할 수는 없다[10,11,12].

3. 이동 컴퓨팅 환경을 고려한 실시간 데이타의 손실 복구

3.1 패킷 손실 특성 분석을 위한 확률 모델의 도입

최근 들어, 네트워크 상에서 발생하는 패킷 손실을 확률적으로 모델링하고자 하는 시도들이 있어 왔다. 그 중에서 비교적 정확한 확률 모델로 평가받는 것이 길버트 모델(Gilbert model)이다. 길버트 모델을 사용하면 간헐적 패킷 손실과 연속적인 패킷 손실 모두를 잘 근사시킬 수 있다는 것이 실험적으로 검증되었으며[7,9,10], 최근에는 이를 이용한 연구 결과들이 등장하고 있다[2].

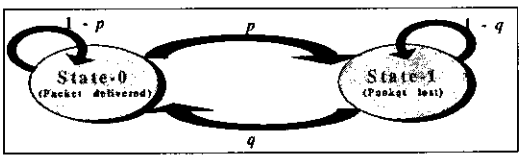


그림 2 길버트 모델

그림 2에서와 같이, 길버트 모델에서는 패킷의 손실 여부를 2-state 마코프 체인(Markov chain)으로 설명한다. 임의의 패킷이 무사히 수신자 측에 도달하는 경우를 하나의 스테이트(state-0)로 보고, 반대로 패킷이 손실되는 경우는 다른 스테이트(state-1)로 본다. 이 때, 패킷 손실 특성은 각 스테이트 간의 전이 확률 p와 q로 표현할 수 있다. 즉, n번째 패킷이 도착한 경우, $(n+1)$번째 패킷이 손실되는 것은 state-0에서 state-1으로의 전이를 의미하고, 그 확률은 p로 표현한다. 반대로 $(n+1)$번째 패킷이 무사히 도착하는 확률은 $1-p$가 된다. 같은 방식으로, n번째 패킷이 손실된 경우에는 $(n+1)$번째 패킷이 무사히 도착할 확률과 손실될 확률이 각각 q, $1-q$가 된다.

n번째 패킷이 어느 스테이트에 있는지를 표현하는 확률 변수를 X_n이라 하면, $X_n=0$은 n번째 패킷이 수신자 측에 도착한 경우이고, $X_n=1$은 n번째 패킷이 전송 혹은 수신 과정에서 손실된 경우를 의미한다. 전이 확률 p, q는 확률 변수 X_n을 이용하여 다음과 같이 표현할 수 있다.

$$p=\Pr[X_{n+1}=1|X_n=0]$$
$$q=\Pr[X_{n+1}=0|X_n=1]$$

연속된 패킷들이 같은 스테이트를 지속적으로 유지할 확률은, 전이 확률 p, q에 대한 기하 분포(geometric distribution)를 보인다[8]. 따라서, 임의의 패킷이 state-0 또는 state-1에 있을 확률은 기하 분포의 특성에 따라, 다음과 같이 계산된다.

$$\Pr[X_n=0]=\frac{q}{p+q} \quad (1)$$

$$\Pr[X_n=1]=\frac{p}{p+q} \quad (2)$$

길버트 모델은 기존의 패킷 손실률 계산 과정을 포함하고 있다. 예를 들어, RTP/RTCP 표준에서는 일정 시간 간격동안에 전송된 패킷들 중에서 손실된 패킷의 비율을 손실비(fraction lost) 값으로 표현한다. 이 값은 길버트 모델의 관점에서는 임의의 패킷이 state-1에 있을 확률을 의미하고, $\Pr[X_n=1]$의 계산 결과와 일치한다.

또한, 길버트 모델에서는 전이 확률을 이용하여, 간헐적인 패킷 손실은 물론, 연속적인 손실이 일어날 확률도 구할 수 있다는 장점이 있다. 즉, n, $n+1$, $n+2$의 3개 패킷이 연속적으로 손실될 확률은 다음과 같이 계산할 수 있다.

$$\begin{aligned}\Pr[X_n=1,X_{n+1}=1,X_{n+2}=1] &= \Pr[X_{n+2}=1|X_{n+1}=1]\cdot\Pr[X_{n+1}=1|X_n=1]\cdot\Pr[X_n=1] \\ &= (1-q)\cdot(1-q)\cdot\frac{p}{p+q} \\ &= (1-q)^2\frac{p}{p+q}\end{aligned} \quad (3)$$

본 논문에서는 간헐적인 패킷 손실과 연속적인 패킷 손실 모두를 감안하기 위해, 길버트 모델에 기초한 패킷 손실률을 계산하고자 한다. 다음 절에서는 이러한 패킷 손실률 계산을 바탕으로 본 논문이 제안하는 손실 패킷 복구 방법을 설명한다.

3.2 복구 방법의 설계

본 절에서 설계하는 실시간 데이타의 복구 방법은 다음과 같은 특징을 가진다.

- 본 절에서 설계하는 복구 방법은 이동 컴퓨팅 환경을 대상으로 한다. 따라서, 유선망에 비해 전체 패킷 손실률과 연속적인 패킷 손실률이 높은 경우를 고려한다.
- 길버트 모델의 전이 확률 p, q를 이용하여 패킷 손실 특성을 확률적으로 분석한다.
- 복구 방법은 잉여 데이타 부가 전송 방식을 기반으로 한다. 이 때, 기존의 정적인 부가 방식들[10]과는 달리 동적인 부가 방식을 채택함으로써, 네트워크 상태 변화를 반영한다.
- 부가 전송하는 잉여 데이타들의 개수는 완전 손실률을 고려하여 결정한다. 또한, 연속적인 패킷 손실률 감소를 위해, 비연속적이고 큰 오프셋 값을 사용한다.
- 패킷 손실 특성에 대한 정보는 RTCP APP 패킷을 이용하여 전송한다.

2절에서 언급한 바와 같이, 손실된 패킷을 복구하는 방법으로는 인터리빙 방식과 잉여 데이타를 이용하는 부가 전송 방식이 대표적이다. 그러나, 인터리빙 방식은 재생 지연이라는 제약때문에 실시간 데이타를 다루는 경우에는 잘 쓰이지 않는다. 같은 이유로, 본 논문이 제안하는 패킷 복구 방법도 잉여 데이타를 이용하는 부가 전송 방식에 근거한다. 반면에, 본 논문이 제안하는 방법에서는 패킷 손실 특성을 확률적으로 분석한 후, 그 특성에 따라 부가하는 잉여 데이타의 양을 적절하게 조절하여 전송하는 점이 특징이다. 이렇게 함으로써, 가변적으로 변화하는 네트워크 특성을 반영하여 대역폭 증가를 최소화할 수 있다.

RTP/RTCP와 같은, 기존의 실시간 데이타 전송 방법에서는 송신자 측과 수신자 측이 일정 시간 간격 동안 데이타 패킷을 교환한 후, 수신자 측이 이 기간 동안 전송된 패킷들의 손실률에 대한 정보를 수신자보고(receiver report)의 손실비(fraction lost) 값으로 송신자 측에게 피드백한다. 이 손실비 값으로써, 전송된 패킷들이 손실되는 정도는 알 수 있지만, 간헐적으로 또는 연속적으로 발생하는 패킷 손실의 특성을 구분할 수는 없다. 본 논문이 제안하는 복구 방법에서는 좀 더 정확한 패킷 손실 특성을 파악하기 위해, 3.1절에서 설명한 길버트 모델을 이용한다. 수신자 측은 손실비와는 별도로 전이 확률 p, q의 값을 전송하고, 송신자 측은 이 전이 확률 값들을 이용하여 간헐적이거나 연속적인 패킷 손실에 대한 특성을 파악할 수 있다. 송신자 측이 좀더 정확한 패킷 손실 특성을 파악할 수 있다는 점은, 송신자 측에서 적절한 손실 복구 방법을 제공하기 위한 중요한 요소가 된다.

본 논문이 제안하는 복구 방법은 잉여 데이타를 이용한 부가 전송 방식에 근거한다. 따라서, 송신자 측에서는 부가 전송하는 잉여 데이타의 개수와, 각 잉여 데이타가 원시 데이타를 기준으로 어느 정도의 간격을 가지는 지를 의미하는 오프셋(offset) 값들에 따라 다양한 형태로 잉여 데이타를 부가시킬 수 있다. 전이 확률 p, q 값을 통하여 패킷 손실 특성을 파악할 수 있는 경우에는 이들 중에서 적절한 것을 선택할 수 있다.

우선, 패킷 손실이 적어 수신자 측에서 충분한 품질을 얻을 수 있다면, 잉여 데이타를 부가 전송하지 않음으로써 불필요한 대역폭 낭비를 피할 수 있다. 패킷 손실이 많은 경우에는 패킷 손실의 연속성 여부를 고려하여 잉여 데이타의 개수와 오프셋 값들을 결정할 수 있다. 기존의 RTP/RTCP에서 사용하는 손실비 값이나 누적 패킷 손실 개수(number of cumulative packet lost)가 증가하더라도 패킷의 손실 특성이 대부분 간헐적인 경우라면, 비교적 적은 오프셋 값들을 가지는 잉여 데이타만을 사용하더라도 해당 데이타들을 복구하는 것이 가능하다. 그러나, 전체적으로는 패킷 손실률이 낮더라도, 손실된 패킷들이 연속적인 부분에 집중되는, 패킷 손실의 연속성이 높은 경우에는 잉여 데이타들의 오프셋 값을 증가시킴으로써 연속적인 손실에 대한 복구율을 높일 수 있다.

잉여 데이타를 이용한 패킷 복구 방법들에서는 잉여 데이타들의 오프셋을 어떻게 설정하느냐도 복구율을 결정하는데 있어 중요한 요소가 된다[2,11]. 최근까지의 잉여 데이타를 이용한 패킷 복구 방법들은 잉여 데이타가 원시 데이타로부터 -1, -2, -3과 같이, 비교적 작고 서로 연속된 오프셋 값들을 가지도록 설정하였다[2,8]. 이러한 연속된 오프셋 값들은 한 두개의 패킷이 간헐적으로 손실되는 경우에 대해서는 복구율이 높지만, 여러 개의 패킷이 연속적으로 손실되는 경우에 대해서는 복구 가능성이 급격히 줄어든다. 이에 대한 보완책으로 잉여 데이타를 포함하는 패킷들을 인터리빙 방식으로 전송하여 연속적인 손실에 대비할 수 있다는 연구 결과도 나와 있다[11]. 반면에, 이 방법은 인터리빙 자체의 한계인 시간 지연 문제를 극복하기 어려운 단점이 있다.

$m \times n$ 블록을 사용하는 인터리빙 방식은, 원래 연속적이던 데이타들을 m 개 패킷 만큼의 간격을 가지도록 해 줌으로써, m 개까지의 연속적인 패킷 손실을 분산시켜 간헐적인 패킷 손실로 바꾸어 주는 역할을 한다. 이 때, -1의 오프셋을 쓰는 잉여 데이타를 사용하는 방법도 함께 사용하면, $m \times n$ 인터리빙 방식은 패킷의 연속

적인 손실을 m 개까지 복구할 수도 있다. 즉, k번째 패킷에 들어간 원시 데이타는 $(k+m)$번째 패킷에 잉여 데이타 형태로 들어가므로, k번째 패킷에 들어간 데이타가 완전 손실될 확률은 다음과 같이 계산된다.

$$\Pr[X_k = 1, X_{k+m} = 1] = \Pr[X_{k+m} = 1 | X_k = 1] \cdot \Pr[X_k = 1] \quad (4)$$

그림 3에서는 -1 오프셋의 잉여 데이타 부가 방식을 인터리빙 방식에 접목시킨 경우를 보여준다. 3×5 배열의 인터리빙 방식에서, 데이타 12번은 전송되는 6번째 패킷에 원시 데이타로 포함되어 전송되고, (6 + 3) = 9 번째 패킷에 잉여 데이타로 부가되어 전송된다.

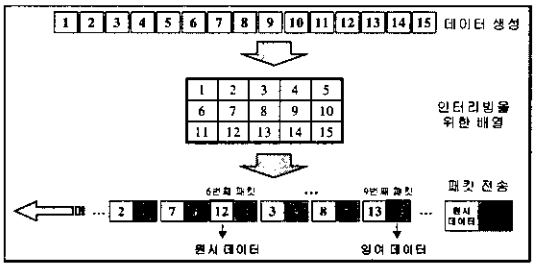


그림 3 -1 오프셋 값의 잉여 데이타 부가 방식을 결합한 인터리빙 방식

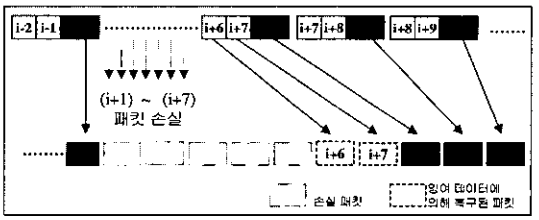
잉여 데이타만을 사용하는 부가 전송 방식에서는 이와 동등한 효과를 얻기 위해, $-m$의 오프셋을 가지는 잉여 데이타를 사용할 수 있다. 즉, k번째 패킷에 들어간 원시 데이타는 $(k+m)$번째 패킷에서 $-m$ 오프셋을 가지는 잉여 데이타 형태로 들어간다. 따라서, k번째 패킷에 들어간 데이타가 완전 손실될 확률은 다음과 같이 계산된다.

$$\Pr[X_k = 1, X_{k+m} = 1] = \Pr[X_{k+m} = 1 | X_k = 1] \cdot \Pr[X_k = 1] \quad (5)$$

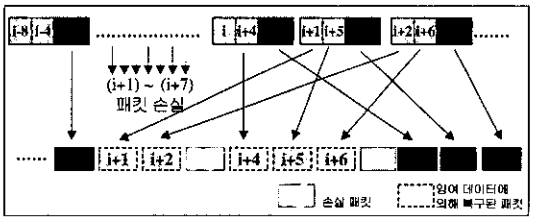
따라서, 비교적 큰 m 값에 대하여, $-m$ 오프셋을 사용하도록 하는 잉여 데이타 방식은 -1 오프셋을 사용하는 $m \times n$ 블록을 사용하는 인터리빙 방식과 동등하다고 볼 수 있다. 본 논문에서는 이 점에 착안하여, 이제까지의 잉여 데이타 방식들과는 달리, 비교적 큰 오프셋 값들을 사용한다.

이동 컴퓨팅 환경에서는 전체 패킷 손실률과 연속 패킷 손실률이 모두 큰 특성이 있다. 손실률이 클수록 더 많은 잉여 데이타를 사용하는 것이 효율적이지만, 네트워크의 대역폭을 감안하면 어느 한계 이상으로 잉여 데이타의 개수를 늘릴 수는 없다. 이러한 이유로, 본 논문에서는 잉여 데이타의 최대 개수를 4개로 설정한다.

그리고, 패킷의 연속적인 손실을 줄이기 위해, 오프셋 값들을 -1, -2, -4, -8과 같이, 비연속적으로 설정한다. -1, -2의 오프셋 값들은, 기존의 잉여 데이타를 이용한 패킷 복구 방법들에서와 같이, 간헐적인 패킷 손실을 복구할 수 있는 기능을 수행한다. 반면에, -4, -8의 비교적 큰 오프셋 값들은 연속적인 패킷 손실의 복구를 가능하게 한다. 예를 들어, 7개의 패킷이 연속적으로 손실되는 경우, 그림 4의 (a)에서 보는 바와 같이 (-1, -2)의 오프셋으로는 복구에 한계가 있지만, 그림 4의 (b)에서처럼 (-4, -8)의 오프셋을 사용하는 경우에는 상당한 양의 데이타를 복구할 수 있다. 패킷의 연속적인 손실에 대비한다는 측면에서는 더 큰 오프셋 값이 유리하지만, 데이타의 실시간 처리를 보장하기 위해서는 어느 한계치 이상의 오프셋 값들은 사용이 불가능하다. 예를 들어, 8000Hz로 샘플링된 음성 데이타는 20 msec 단위로 패킷이 전송되는 것이 일반적인데, -8 오프셋의 경우에는 최대 160 msec의 시간 지연을 가져올 수 있다. 실시간성의 특성을 지닌 음성 데이타의 경우, 150 ~ 200 msec 이상의 시간 지연은 사용자가 감지할 수 있는 정도로 품질이 떨어지므로[8], 이 이상의 오프셋 값을 쓰는 것은 적합하지 않다.



(a) (-1, -2) 오프셋 잉여 데이타를 가지는 3개의 패킷으로 복구한 경우



(b) (-4, -8) 오프셋 잉여 데이타를 가지는 3개의 패킷으로 복구한 경우

그림 4 잉여 데이타를 이용한 손실 복구 방법

이상에서 보는 바와 같이, 잉여 데이타를 이용하여 데이타를 복구하는 경우에는 패킷 자체의 손실보다는

잉여 데이타를 이용하더라도 복구되지 못하는 데이타의 비율이 더 중요하게 된다. 특히, 간헐적인 패킷 손실의 경우에는 비록 원시 패킷이 손실되더라도 잉여 데이타를 이용하여 원시 데이타 복구가 가능한 경우가 많으므로, 단순히 패킷의 손실률만을 따지는 것은 한계를 가지게 된다. 따라서, 본 논문에서는 전체 패킷들의 손실률 외에, 특정 데이타가 잉여 데이타를 이용하더라도 복구되지 못하는, 완전 손실률을 고려하여 어떤 형태의 잉여 데이타를 사용할 것인지를 결정한다.

앞에서 언급한 바와 같이, 본 논문에서는 -1, -2, -4, -8의 오프셋을 가지는 잉여 데이타를 사용한다. 이들 외에 잉여 데이타를 사용하지 않는 경우를 포함하여, 총 5가지 경우의 잉여 데이타 부가 방식을 설정하고, 이들 각각을 R0-Method, R1-Method, R2-Method, R3-Method, R4-Method라고 표현한다. R0-Method는 잉여 데이타를 사용하지 않음을 의미하고, R1-Method, R2-Method, R3-Method, R4-Method는 각각 (-1), (-1, -2), (-1, -2, -4), (-1, -2, -4, -8)의 오프셋을 가지는 잉여 데이타들을 부가하여 전송한다. 즉, R3-Method에서는 n번째 데이타를 전송하는 패킷에 (n-1), (n-2), (n-4)번째 데이타에 대한, 총 3개의 잉여 데이타를 실어서 전송한다. 특히, R3-Method와 R4- Method에서는 비교적 큰 오프셋을 가지는 잉여 데이타를 전송함으로써, 인터리빙에 의한 전송 효과 즉, 연속적인 패킷 손실률의 감소 효과를 가져올 수 있다. 연속적인 패킷 손실의 특성은 전체 패킷 손실률과 밀접한 관계를 가지므로, 오프셋의 조합을 선택할 때, 간헐적으로 발생하는 패킷 손실을 복구하기 위한 -1, -2의 오프셋들이 항상 포함되어야 한다. 이 때문에, 본 논문에서는 (-1, -2, -4, -8)의 오프셋을 위와 같은 조합으로 선택하였다.

이제 송신자 측의 입장에서는 R0-Method, R1-Method, R2-Method, R3-Method, R4-Method 등의 총 5가지 잉여 데이타 부가 방식을 사용할 수 있고, 수신자 측의 입장에서는 각각의 잉여 데이타 부가 방식에 따라 손실된 데이타의 복구 정도가 달라진다. 각 잉여 데이타 부가 방식이 어떤 상황에서 사용될 수 있는지를 분석하기 위해서는, 전이 확률 p, q 값을 이용하여 길버트 모델로써 완전 손실률을 다음과 같이 계산할 수 있다. 이 완전 손실률은 전이 확률 p, q 값만으로 계산이 가능하기 때문에, p, q 값만을 전달받은 송신자 측에서도 계산할 수 있다.

가) R0-Method

R0-Method에서는 잉여 데이타를 실어 보내지 않으므로, 임의의 패킷이 손실되는 확률 자체가 해당 데이타의 완전 손실 확률과 같다.

$$LossR0 = \Pr[X_n=1] = \frac{p}{(p+q)} \quad (6)$$

나) R1-Method

R1-Method에서는 n번째 패킷에 n번째 생성된 데이타를 원시 데이타 형식으로, (n-1) 번째 생성된 데이타를 잉여 데이타 형식으로 함께 실어보낸다. 전송된 패킷들을 표현하면 그림 5와 같다. 그림 5에서 2번째 패킷은 2번째 생성된 원시 데이타와 1번째 생성된 데이타의 잉여 데이타를 가진다.

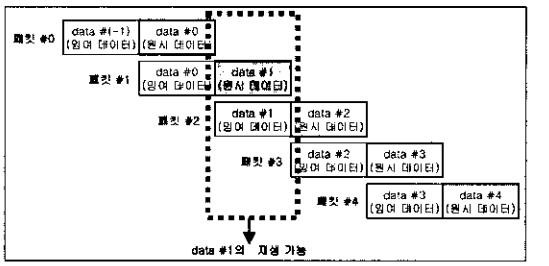


그림 5 R1-Method에서의 전송 패킷과 생성 데이타

n번째 생성된 데이타의 입장에서는, n번째 패킷이 손실되면 자신의 원시 데이타를 잃어버리게 되고, (n+1)번째 패킷이 손실되면 자신의 잉여 데이타를 잃어버린다. 따라서, 데이타의 완전한 손실은 n번째 패킷과 (n+1)번째 패킷이 모두 손실된 경우에 일어나고, n번째 생성된 데이타가 완전히 손실될 확률을 수식으로 표현하면 다음과 같다.

$$\begin{aligned} LossR1 &= \Pr[X_n=1, X_{n+1}=1] \\ &= \Pr[X_{n+1}=1 \mid X_n=1] \cdot \Pr[X_n=1] \\ &= (1-q) \cdot \frac{p}{p+q} \end{aligned} \quad (7)$$

다) R2-Method

R2-Method에서는 n번째 패킷에 n번째 생성된 원시 데이타에 (n-1), (n-2)번째 생성된 데이타의 잉여 데이타 형식을 부가해서 전송한다. 전송된 패킷들을 표현하면 그림 6과 같다. 그림 6에서 3번째 패킷은 3번째 생성된 원시 데이타와 1, 2번째 생성된 데이타의 잉여 데

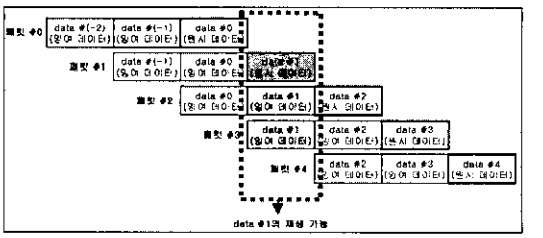


그림 6 R2-Method에서의 전송 패킷과 생성 데이타

이타로 구성된다.

R1-Method에서와 같은 과정을 통해, n번째 생성된 데이타가 완전 손실될 확률을 계산하면 다음과 같다.

$$\begin{aligned} LossR2 &= \Pr[X_n=1, X_{n+1}=1, X_{n+2}=1] \\ &= \Pr[X_{n+2}=1 \mid X_{n+1}=1] \\ &\quad \cdot \Pr[X_{n+1}=1 \mid X_n=1] \cdot \Pr[X_n=1] \\ &= (1-q)\cdot(1-q)\cdot\frac{p}{p+q} \\ &= (1-q)^2\cdot\frac{p}{p+q} \end{aligned} \quad (8)$$

라) R3-Method

R3-Method에서는 n번째 패킷에 n번째 생성된 원시 데이타에 $(n-1)$, $(n-2)$, $(n-4)$번째 생성된 데이타의 잉여 데이타 형식을 부가해서 보낸다. 전송된 패킷들을 표현하면 그림 7과 같다. 그림 7에서 4번째 패킷은 4번째 생성된 원시 데이타와 0, 2, 3번째 생성된 데이타의 잉여 데이타로 구성된다.

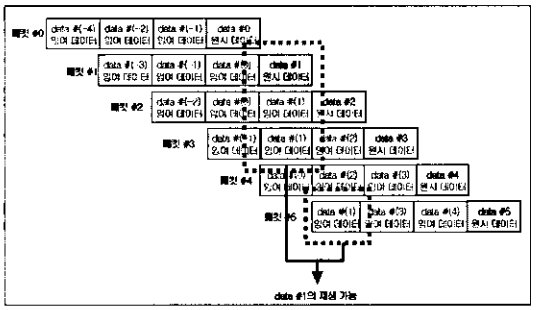


그림 7 R3-Method에서의 전송 패킷과 생성 데이타

따라서, n번째 생성된 데이타가 완전히 손실될 확률을 수식으로 표현하면 다음과 같다.

$$\begin{aligned} LossR3 &= \Pr[X_n=1, X_{n+1}=1, X_{n+2}=1, X_{n+4}=1] \\ &= \Pr[X_{n+4}=1 \mid X_{n+2}=1] \\ &\quad \cdot \Pr[X_{n+2}=1 \mid X_{n+1}=1] \\ &\quad \cdot \Pr[X_{n+1}=1 \mid X_n=1] \cdot \Pr[X_n=1] \\ &= \{(1-q)^2+pq\}\cdot(1-q)\cdot(1-q)\cdot\frac{p}{p+q} \\ &= \{(1-q)^2+pq\}\cdot(1-q)^2\cdot\frac{p}{p+q} \end{aligned} \quad (9)$$

마) R4-Method

R4-Method에서는 n번째 패킷에 n번째 생성된 원시 데이타와 $(n-1)$, $(n-2)$, $(n-4)$, $(n-8)$번째 생성된 데이타를 잉여 데이타 형식으로 부가해서 보낸다. 전송된 패킷들을 표현하면 그림 8과 같다. 그림 8에서 8번째 패킷은 8번째 생성된 원시 데이타와 0, 4, 6, 7번째 생성된 데이타의 잉여 데이타로 구성된다.

따라서, n번째 생성된 데이타가 완전히 손실될 확률을 수식으로 표현하면 다음과 같다.

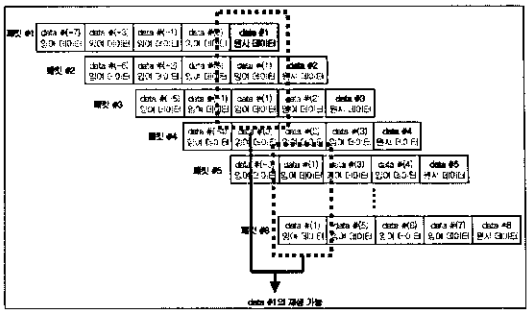


그림 8 R4-Method에서의 전송 패킷과 생성 데이타

$$\begin{aligned} LossR4 &= \Pr[X_n=1, X_{n+1}=1, X_{n+2}=1, \\ &\quad X_{n+4}=1, X_{n+8}=1] \\ &= \Pr[X_n=1 \mid X_{n-4}=1] \\ &\quad \cdot \Pr[X_{n-4}=1 \mid X_{n-6}=1] \\ &\quad \cdot \Pr[X_{n-6}=1 \mid X_{n-7}=1] \\ &\quad \cdot \Pr[X_{n-7}=1 \mid X_{n-8}=1] \\ &\quad \cdot \Pr[X_{n-8}=1] \\ &= \{(1-q)^4+pq\cdot(6-4p+p^2-8q+3pq+3q^2)\} \\ &\quad \cdot \{(1-q)^2+pq\}\cdot(1-q)^2\cdot\frac{p}{p+q} \end{aligned} \quad (10)$$

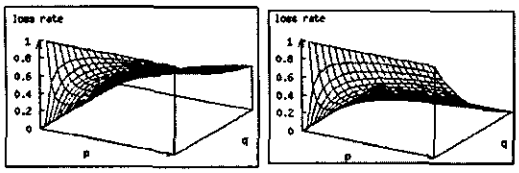
위에서 계산한 각 잉여 데이타 부가 방식들의 완전 손실률들은 잉여 데이타의 갯수가 높아짐에 따라, 그 손실률이 단조 감소(monotone decreasing)하는 특성을 가진다, 즉, 전이 확률 p, q 값들이 0에서 1 사이의 값을 가지면, 항상 다음의 관계식을 만족시킨다.

$$LossR0 \geq LossR1 \geq LossR2 \geq LossR3 \geq LossR4 \quad (11)$$

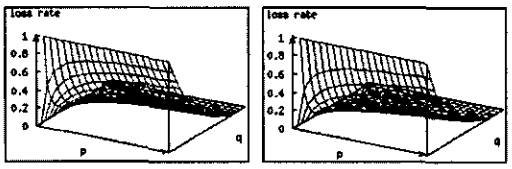
이렇게 단조 감소하는 특성은 손실률을 계산하거나, 이 손실률을 바탕으로 특정 부가 방식을 선택하는 과정에서 좀더 효과적인 처리가 가능하도록 한다.

다음 그림 9는 위에서 계산한 각 잉여 데이타 부가 방식의 완전손실률을 p, q 값의 변화에 따라 3차원 그래프로 그린 것이다. 그래프에서 x축은 p, y축은 q, 그리고 z축은 각 부가 방식의 완전 손실률 값을 나타낸다. 그림 9에서 보는 바와 같이, 완전 손실률은 p의 증가에 따라 커지고, q가 증가할수록 급격히 감소한다. 또한, 각 부가 방식들은 단조 감소의 특성에 따라, 더 많은 잉여 데이타를 사용하는 방식일수록 완전 손실률은 낮아진다.

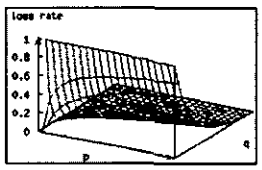
이동 컴퓨팅 환경과 같이, 비교적 손실률이 높으면서도 대역폭이 제한되는 상황에서는 전송된 데이타를 모두 수신하려는 것보다는 어느 정도의 손실을 허용하는 것이 전체적으로 더 효과적일 수 있다. 특히, 멀티미디어 데이타의 실시간 처리에서는 일정 비율까지의 데이타 손실은 정보 전달의 입장에서는 그리 큰 문제가 아닌 경우가 많다[11]. 손실된 데이타들을 복구하기 위해 송신자가 재전송을 시도하는 경우, 재전송으로 인한 시



(a) R0-Method (b) R1-Method



(c) R2-Method (d) R3-Method

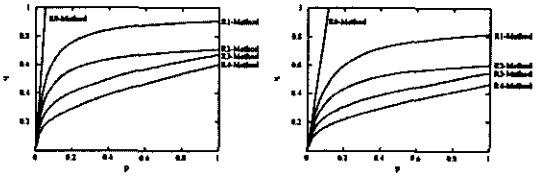


(e) R4-Method

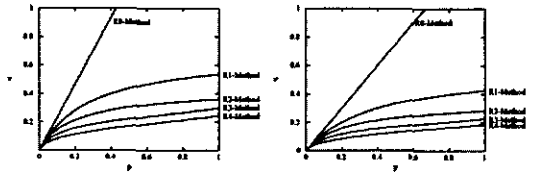
그림 9 잉여 데이타 부가 방식들의 완전 손실률 그래프

간 지연이 오히려 멀티미디어 데이타의 품질을 급격히 떨어뜨릴 수 있다. 이러한 상황에서는 최소한의 품질을 유지하기 위하여 허용할 수 있는 손실률에 대한 임계값을 도입하여, 이 임계값보다 작거나 같은 손실률은 허용하도록 함으로써, 전송에 사용되는 대역폭을 최소화하면서도 어느 정도의 품질을 보장할 수 있게 된다. 그림 10은 임의의 손실률 허용 임계값(α)에 따른 부가 방식들의 경계들을 보여준다. 그림 9의 3차원 그래프를 이용하면, 완전 손실률이 손실률 허용 임계값보다 낮은 p, q 값들의 영역을 추출할 수 있다. 이 영역은 2차원 그래프 상에서 표현될 수 있고, 각 영역의 경계는 그림 10에서와 같이 나타난다. 각 부가 방식들은 해당 경계보다 왼쪽 위 방향에 속하는 모든 p, q 값들에 대해 손실률 허용 임계값보다 낮은 완전 손실률을 가진다. 따라서, p, q 값이 정해지면, p, q 값의 위치보다 오른쪽 아래 방향에 그 경계가 위치하는 부가 방식들을 사용하여 손실률 허용 임계값을 만족시킬 수 있다.

그림 10에서 각 그래프를 비교해 보면, 손실률 허용 임계값(α)이 커질수록, 각 부가 방식의 경계도 오른쪽 아래 방향으로 옮겨감을 알 수 있다. 이것은 특정 부가 방식을 기준으로 보면, α가 커질수록 허용할 수 있는 p의 최대값은 더 커지고 q의 최소값은 작아진다는 것을



(a) α = 5%인 경우 (b) α = 10%인 경우



(c) α = 30%인 경우 (d) α = 40%인 경우

그림 10 손실률 허용 임계값에 따른 각 메소드 경계 그래프

의미한다. 1-q는 패킷의 연속적인 손실 특성을 반영하므로, q의 최소값이 작아진다는 것은 연속적인 패킷 손실 확률이 커진다는 것을 의미한다. 한편, 특정 p와 q를 기준으로 본다면, α가 커질수록 적용될 수 있는 부가 방식들이 많아진다고 할 수 있다. 예를 들어, p = 0.2, q = 0.6일 때, α가 5%인 경우에는 R2-Method이상의 부가 방식들만이 손실률 허용 임계값보다 낮은 완전 손실률을 가지게 되는데 비해, α가 30%인 경우에는 R0-Method, R1-Method도 α 값보다 낮은 완전 손실률을 가진다.

실시간 데이타의 전송 시, 최대로 허용할 수 있는 손실률에 대한 임계값이 설정되면, 송신자 측에서는 위에서 계산한 각 잉여 데이타 부가 방식의 손실률을 이용함으로써, 어느 방식을 적용하여 전송할 지를 선택할 수 있다. 다음 3.3절에서는 동적으로, 현재 손실 특성에 따라 가장 적절한 잉여 데이타 부가 방식을 선택하는 방법에 대해 설명한다.

3.3 이동 컴퓨팅 환경에서의 복구 알고리즘 구현

본 절에서는 이동 컴퓨팅 환경에서 실시간 데이타의 전송 도중에 손실된 패킷들을 복구하기 위한 동적 잉여 데이타 부가 전송 알고리즘을 보인다. 본 논문이 제안하는 복구 알고리즘은 송신자와 수신자 측에서의 동작으로 구성된다. 우선, 수신자 측은 수신된 패킷들을 처리하는 과정에서 잉여 데이타를 이용하여 손실된 데이타를 복구한 후 재생하는 작업을 수행한다. 또한, 일정한 시간 간격이 지나면 그 동안 수신된 패킷들의 손실 특성을 검사하여, 송신자에게 전이 확률 p, q의 형태로 전송한다. 송신자는 이 전이 확률 값들을 이용하여, 3.2절에서 제시한 각 잉여 데이타 부가 방식의 손실률을 계

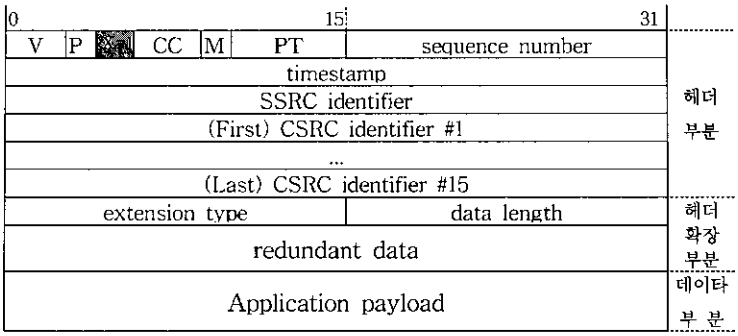


그림 11 RTP 확장 헤더 부분에 잉여 데이타를 넣은 RTP 패킷 형식

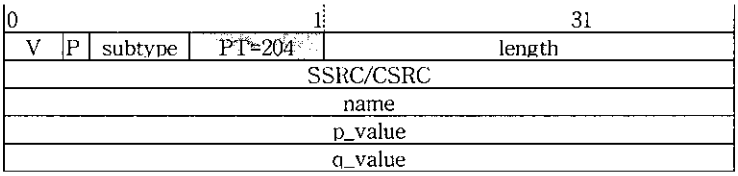


그림 12 확장된 RTCP APP 패킷 형식

산하여 적합한 부가 방식을 선택한 후, 선택된 부가 방식에 따라 필요한 잉여 데이타들을 부가하여 패킷을 전송한다. 송신자 측과 수신자 측은 일정한 시간 간격마다 잉여 데이타 부가 방식의 선택을 위한 정보 교환을 반복하기 때문에, 네트워크 상에서의 패킷 손실 특성이 가변적으로 변화하더라도 동적인 대처가 가능하다.

본 논문이 제시하는 복구 알고리즘은 실시간 데이타의 처리에 촛점을 맞추었으며, 4절에서 제시하는 바와 같이, 실제로 구현하여 검증하였다. 이제부터는, 기존의 실시간 데이타 전송 프로토콜인 RTP/RTCP를 기반으로 구체적인 알고리즘과 구현 방법을 제시한다.

가) RTP/RTCP 패킷의 확장

본 논문이 제안하는 복구 방법에서는 실시간 데이타를 전송할 때, 잉여 데이타를 부가 전송하기 위한 방법과 수신자 측에서 전이 확률 p, q 값을 송신자에게 보낼 수 있는 방법이 필요하다. 따라서, 기존의 RTP/RTCP 명세에 근거하여, RTP 데이타 패킷과 RTCP APP 패킷을 이용하고자 한다.

RTP/RTCP에서 실시간 데이타는 RTP 데이타 패킷을 이용하여 전송된다. RTP 데이타 패킷은 전송되는 실시간 데이타 이외에도 응용 프로그램에서 부가적으로 필요한 정보들을 헤더 확장(header extension) 부분에 넣을 수 있도록 설계되어 있다. 즉, RTP 헤더의 X 필드가 "1"의 값을 가지면, 헤더와 전송되는 데이타 블록 사이에 헤더 확장 부분이 있음을 의미하고, 이 헤더 확장 부분에 들어있는 정보는 해당 응용 프로그램에서 사용할 수 있다[6]. 그림 11은 RTP 헤더 확장 부분에 잉여 데이타를 넣은 형식을 보여준다. 잉여 데이타의 크기는 헤더 확장 부분의 length 필드에서 설정할 수 있고, 4절의 구현 예에서는 잉여 데이타 하나의 크기를 80 바이트로 설정하여, 하나의 RTP 데이타 패킷은 선택된 부가 방식에 따라, 헤더 확장 부분이 없거나 80~320 바이트의 잉여 데이타를 가지도록 하였다. 송신자 측과 수신자 측은 RTP 헤더 확장 부분을 통하여 잉여 데이타를 전송하거나 받는다.

RTP 헤더 확장 부분에서 사용된 필드들은 다음과 같다.

-extension type : 부가 방식의 종류 (1 ~ 4).
-data length : 잉여 데이타 블록의 크기.
-redundant data : 잉여 데이타가 저장되는 블록.

RTP/RTCP에서는 수신자 측이 송신자 측에게 보내는 정보는 기본적으로 RTCP의 RR(receiver report) 패킷을 사용하지만, 응용 프로그램에서 추가로 필요한 정보가 있을 때는 APP(application) 패킷을 사용할 수 있다. 본 논문에서 제안하는 복구 방법에서는 수신자 측이 전이 확률 p, q의 값을 송신자 측에게 보낼 필요가 있는데, RR 패킷 내에는 이 값들을 넣을 부분이 없으므로, APP 패킷을 사용한다. APP 패킷 내부의 포맷은 응

용 프로그램에서 설정할 수 있으므로, 본 논문에서는 수신자 측이 그림 12와 같은 형식으로 확장한 APP 패킷을 송신자 측에게 보내도록 한다. APP 패킷 내부에서 사용하는 필드들은 다음과 같다.

- name : 수신 상태에 대한 확률 정보를 담고 있음을 표시함.(= "PVAL")
- p_value : 전이 확률 p의 값.
- q_value : 전이 확률 q의 값.

나) 수신자 동작 알고리즘

수신자 측에서는 실시간 데이타를 재생하기 위해 재생 버퍼 블록을 두어야 한다. 이 버퍼 블록은 비순차 패킷의 재배열과 지터 보상의 기능을 담당하기 위한 것이다. 수신자 측은 송신자 측으로부터 RTP 패킷을 받을 때마다 크게 두 가지 동작을 한다. 먼저 원시 데이타는 재생 버퍼 블록에 저장하고, RTP 패킷 헤더에 들어 있는 확장 필드(extension field)가 1인지를 본다. RTP 패킷 헤더의 확장 필드가 1이면, 잉여 데이타가 RTP 헤더 확장 부분에 들어 있다는 뜻이므로, RTP 헤더 확장 부분에 들어 있는 잉여 데이타를 가져온다. 그리고, 이 잉여 데이타에 해당하는 재생 버퍼 블록을 살펴보고 이미 그 블록에 데이타가 들어있다면, 이 잉여 데이타는 불필요하므로 무시한다. 해당 재생 버퍼 블록이 아직 비어있으면 이 잉여 데이타를 저장한다. 또, 재생 버퍼에 들어있는 패킷들의 배열 상태를 탐색하여, 전이 확률 p, q를 계산한다.

전이 확률 p, q는 3.1절에서 설명한 바와 같이, 마코프 체인에서의 이론적인 확률 값이지만, 패킷들이 수신된 상태를 살펴보고, 길버트 모델에서의 상태 전이에 대응되는 각각의 경우의 수를 구한 후, 이를 이용하여 p, q의 값을 계산할 수 있다.

그림 13은 수신된 패킷들을 시퀀스 번호에 따라 배열한 후, 상태 전이에 따른 경우의 수를 분석하는 예이다. 길버트 모델에서의 전이확률 p, q, $1-p$, $1-q$에 대응되는, 상태 전이에 대한 경우의 수를 P, Q, P', Q'이라 하면, 그림 13에 대한 각 경우의 수는 표 1과 같다.

표 1 상태 전이에 대한 경우의 수

상태 전이	P : 0->1	P' : 0->0	Q : 1->0	Q' : 1->1
경우의 수	4	9	4	2

이들 경우의 수를 이용하여 전이 확률 p와 q를 구하면,

$$p = \frac{P}{(P+P')} = \frac{4}{13} \simeq 0.307 \quad (12)$$

$$q = \frac{Q}{(Q+Q')} = \frac{4}{6} \simeq 0.667 \quad (13)$$

이 된다. 이 때, RTCP에서의 fraction lost 값은 다음과 같이 계산된다.

$$\Pr[X_n = 1] = \frac{p}{p+q} = \frac{12}{38} \simeq 0.316 \quad (14)$$

수신자 측에서는 RTP 데이타 패킷을 받을 때마다, p, q에 대응되는 경우의 수들을 갱신하고, RTCP APP 패킷을 보내는 시점에 p, q의 확률 값들을 계산하도록 한다. 이 계산 과정의 시간 간격은 RTP/RTCP에서 RR 패킷을 전송하는 시간 간격으로 제안되어 있는 평균 5초, 즉 8000Hz의 음성 데이타인 경우에 약 500 여개의 패킷을 받을 수 있는 정도의 시간으로 설정한다. 그림 14는 수신자 측이, 전달받은 RTP 패킷에 대한 처리 및 RTCP APP 패킷을 이용하여 송신자에게 p, q 값을 전송하는 과정을 알고리즘으로 표현한 것이다.

수신자 측에서는 p, q 확률 값들을 계산하기 위하여, 먼저 일정한 시간 간격을 설정해야 한다. RTP/RTCP의 fraction lost 값을 계산하는 경우에는 RTCP RR 패킷의 전송 간격을 기준으로 한다. 즉, 지난번 RR 패킷을 보낸 이후 현재 RR 패킷을 보내는 시간 간격 동안에 전달받은 패킷들에 대해 fraction lost 값을 계산한다 [1]. 본 논문에서는 p, q 값 계산을 위해 이 기준을 적용하였다. 다음 4절의 실험에서도, 지난번 RTCP RR/APP 패킷을 보낸 이후 현재 RR/APP 패킷을 전송할 때까지의 시간 간격을, p, q 값 계산을 위한 하나의 주기로 사용한다.

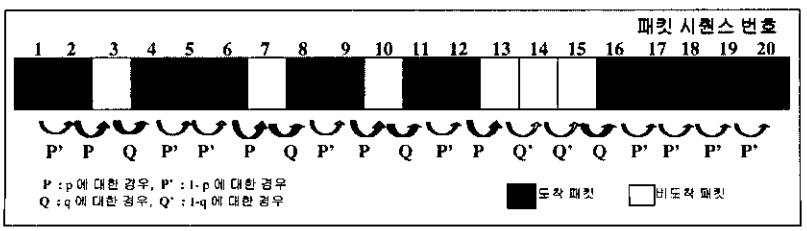


그림 13 전송된 패킷들의 배열

```
Algorithm ReceiverProcessing
  while the RTP session is active do
                :
    /* 1. receive data packets */
     원시 데이타 부분은 재생 버퍼로 이동
     if RTP header extension field == 1 then
        for each redundant data in the RTP header extension do
              if corresponding play buffer is empty then
                 잉여 데이타를 원시 데이타 형태로 변환
                  변환된 원시 데이타 형태를 재생 버퍼로 이동.
              endif
        endfor
     endif
                :
    /* 2. reply RTCP APP packets */
     if time to send RTCP APP packet then
        수신된 데이타들의 손실 특성으로부터 p, q 값 계산
          RTCP APP 패킷에 p, q 값을 실어서 전송
     endif
                :
  endwhile
```

그림 14 수신자 동작 알고리즘

다) 송신자 동작 알고리즘

송신자는 수신자로부터 RTCP APP 패킷을 받게 되면, 패킷에 들어있는 p, q 값들을 가지고서 각 잉여 데이타 부가 방식의 완전 손실 확률을 계산한다. 완전 손실 확률이란 길버트 모델의 패킷 손실 특성인 p, q 값으로부터 앞으로의 손실률을 예측한 기대값이다. 즉, 해당 부가 전송 방식을 적용하였을 때 이 기대값만큼의 손실은 결코 복구하지 못한다는 의미이다. 예를 들어, 손실률의 기대값이 0.06이 나올 경우, 어떤 임의의 패킷 하나에 대해 다른 패킷에 들어 있는 잉여 데이타를 이용하더라도 결코 복구하지 못할 확률이 6%임을 말한다. 각 잉여 데이타 부가 방식마다 잉여 데이타의 개수 및 원시 데이타와의 오프셋 값들이 틀리므로, 이 손실률들은 다르게 계산된다. 이 값들 중에서 손실률 허용 임계값 α보다 작은 값을 갖는 잉여 데이타 부가 방식을 선택한다. 임계값 α는 네트워크 환경 및 트래픽 그리고, 실시간 데이타의 특성을 고려하여 송·수신자 사이에 결정된, 손실률에 대한 최대 허용치이다. 만일 두 개 이상의 잉여 데이타 부가 방식이 α보다 작은 값을 가지는 경우에는, 부가하는 잉여 데이타의 개수가 적은 쪽을 선택한다. 즉, R2-Method와 R3-Method의 손실 확률이 모두 α보다 작은 경우, R2-Method를 선택한다. 이렇게 함으로써, 완전 손실률은 α보다 작도록 유지하면서도, R3-Method를 선택하는 경우에 비해 전체 패킷의 크기를 감소시킴으로써 네트워크 대역폭 증가를 최소화하게 된다. 최종 선택된 부가 방식은 수신자 측으로부터 다음 번 RTCP APP 패킷을 전달받을 때까지 수행된다. 그림 15는 송신자 측이, 패킷 손실 특성 p, q 값을 가지고서, 수신자의 패킷 손실 상태에 따른 최적의 잉여 데이타 부가 방식을 선택하는 알고리즘이다. 잉여 데이타 부가 방식의 선택에 있어서는 각각의 손실률이 단조 감소한다는 성질을 이용하여 불필요한 손실률 계산을 줄였다. 즉, *LossR1*이 α보다 적다면, *LossR2*, *LossR3*, *LossR4*는 *LossR1*보다 더 낮은 확률 값을 가지므로, 더 이상의 계산이 불필요하다.

```
Algorithm SelectMethod

    input α = (사용자가 정의한 손실률 허용 임계값)
    /* 1. activate R2-method as the default */
    activate R2-method

    /* 2. select the optimal method with respect to p and q
       values */
      while the RTP session is active do
                wait for the RTCP APP packet
                get the p, q values from the RTCP APP
                packet
                if ( LossR0 ≤ α )
                        then activate R0-Method.
                   else if ( LossR1 ≤ α )
                        then activate R1-Method.
                   else if ( LossR2 ≤ α )
                        then activate R2-Method.
                   else if ( LossR3 ≤ α )
                        then activate R3-Method.
                   else if ( LossR4 ≤ α )
                        then activate R4-Method.
                   else {
                        activate R4-Method.
                        printf ("All Methods do not satisfy
                        the α value").
                   }
                endif
        endwhile
```

그림 15 패킷 손실 특성에 따른 잉여 데이타 부가 방식 결정 알고리즘

4. 실험 및 분석

4.1 실 험

본 절에서는 3절에서 제안한 동적인 부가전송 방법의 성능을 평가하기 위해, 실시간 전송 프로토콜인 RTP를 이용하여 ARAT(Adaptive Realtime-Audio Transmission) 시스템을 구현하였다.

ARAT 시스템은 송신자 프로그램인 "SndSender"와 수신자 프로그램인 "SndReceiver"로 구성된다. "SndSender"는 실시간 데이타를 전송할 때, 수신자 측으로부터 피드백 정보를 받을 때마다, 이 정보를 이용하여

적절한 잉여 데이타 부가 방식을 선택하여 수행하고, "SndReceiver"는 송신자 측으로부터 전달받은 오디오 데이타를 재생하면서, 주기적으로 수신상태를 송신자에게 피드백하게 된다. 두 프로그램은 GCC 컴파일러를 사용하여 리눅스 운영체제에서 구현되었다.

실험 환경은 리눅스 운영체제가 동작하는 두 대의 시스템이, 서로 다른 서브넷을 가지는 환경에서 전송할 수 있도록 구성되었다. 송신자 측은 펜티엄 PC 기반의 리눅스 환경에 RTP 계층의 모듈들을 포함한 송신자 응용 프로그램인 SndSender 프로그램이 UDP/IP 계층 위에서 동작한다. 실험은 간결성을 위해 송신자와 수신자간의 유니캐스트 방식으로 행해졌으며, 수신자 측에서는 지터보상 및 패킷 순서 재배열을 위하여 재생버퍼를 두었다. 수신자 측은 128M 바이트 메모리를 가지는 펜티엄 III 450MHz 급의 리눅스 플랫폼 노트북에 스피커 및 사운드 카드를 설치하였다. 여기에는 RTP 프로토콜 모듈과 Mobile IP를 포함한 수신자 응용 프로그램인 SndReceiver 프로그램이 동작한다. RTP 프로토콜 모듈은 RFC1889 문서에 기반하여 현재 공개되어 있는 소스인 rtptools[13], NeVoT[14]을 참조하여 구현되었고, Mobile IP는 HP사의 리눅스용 소스 코드[15]를 수정하여 사용하였는데, 이것은 사용자의 이동에 대한 투명성을 제공[16]하기 위한 것이다.

실험에서 사용되는 기본 코덱으로는 64Kbps의 전송률을 갖는 PCM 방식을 이용하였다. 원시데이타의 형식은 8000Hz의 샘플링 비율의 PCM 형식이다. 그리고, 잉여 데이타의 부가 전송으로 인한 대역폭 증가를 최소화하기 위해, 잉여 데이타의 형식은 4000Hz의 샘플링 비율을 갖도록 하였다. 잉여 데이타는 RTP 패킷의 헤더 확장부분에 삽입되어 전송되도록 하였고, 수신자 측에서는 패킷 손실에 대한 정보를 RTCP APP 패킷에 저장한 후, RTCP RR 패킷과 결합하여(compound RTCP RR packet) 전송하도록 하였다.

우선, 이동 컴퓨팅 환경을 고려한 패킷 전송 지연 및 손실 특성을 반영하기 위하여, 이동 컴퓨팅 환경에서의 평균적인 패킷 손실률을 설정하고, 그에 따른 손실 특성 p, q의 영역을 결정하였다. 이 영역에 속하는 (p, q) 값들을 30여 개 샘플링한 후, 부가 전송을 적용하지 않은 경우(R0-Method), R1-Method, R2-Method, R3-Method, R4-Method 만을 정적으로 사용한 경우, 그리고 3.3절에서 제안한 동적 부가 전송 방법을 사용한 경우들에 대해 각각 동일한 조건 하에서 실험하였다. 이동 컴퓨팅 환경에서의 패킷 손실 특성을 시뮬레이트하기 위하여, 패킷 전송 시에 송신자 측에서 길버트 모델을 적용한 랜덤 패킷 손실을 발생시켰다. 그리고, 수신자 측에서는 각 경우에 대한 완전 손실률을 비교 분석하였다.

- 이동 컴퓨팅 환경을 고려한 p, q 의 영역 결정

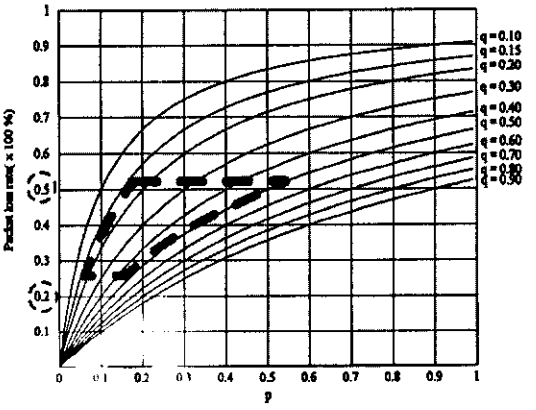


그림 16은 특정 q 값에 대해, p 값의 변화에 따른 패킷 손실률을 보인 그래프이다. 이 그래프는 식(4)의 패킷 손실률 계산식으로부터 구할 수 있고, 패킷 손실률과 패킷 손실 특성과의 관계를 보여준다. 이동 컴퓨팅 환경에서는 대체적으로 패킷 손실률과 패킷의 연속적인 손실이 유선망에 비해 높게 발생하므로, 패킷 손실률이 20~50%, 연속적으로 손실되는 패킷의 평균 개수가 2~7개라고 가정한다면, 그림 16의 그래프를 참조하여 패킷 손실률과 q 값으로부터 p와 q 각각의 최대·최소값을 결정할 수 있다.

q 값은 길버트 모델에서 state 1의 평균 거주 시간(Mean Residence Time)을 이용하여 구할 수 있다. 평균 거주 시간(MRT)이란 일단 손실상태(state-1)에 들어갔을 때, 평균적으로 머무는 시간을 의미하고, 이 값을 확률 계산식으로 유도하면, $\frac{1}{q}$이 된다[9,10]. 연속적으로 손실되는 패킷의 평균 개수가 2~7개이므로, $2 \le \frac{1}{q} \le 7$이고, 따라서 q 값은 $0.143 \le q \le 0.50$의 영역을 가지게 된다. 그림 16에서 패킷 손실률이 20~50%인 부분과의 교차 영역을 p 축으로 내리면, p의 영역은 q가 0.15일 때 $0.02 \le p \le 0.14$, q가 0.50일 때 $0.06 \le p \le 0.50$의 영역을 가지게 된다. 이렇게 계산된 p, q의 영역을 그림 10(a)의 경계값 그래프에 옮겨서 적용하면, 그림 19에서의 점선으로 그려진 사각 부분이 된다. 이때, 사각 부분의 외부에 있는 부가 방식은 손실 허용 임계값(α)을 만족시키지 못하므로 사용할 필요가 없다고 볼 수 있다.

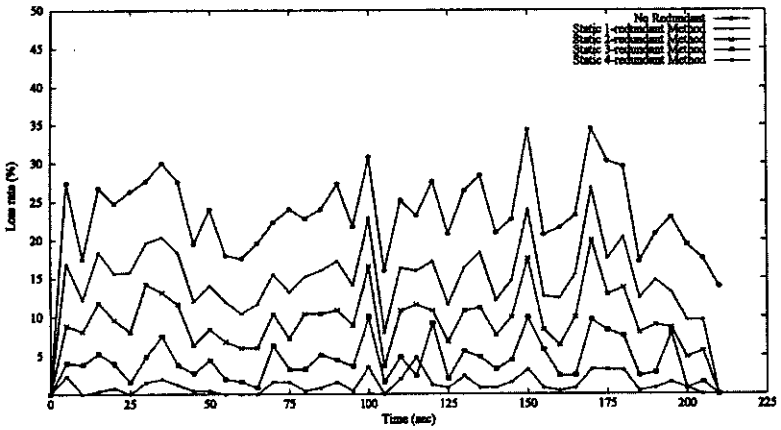


그림 17 정적 부가 방식들을 이용한 손실률 측정 (p=0.12, q=0.35, α=5%)

본 실험에서는 손실 허용 임계값을 5%로 두었는데, 그 이유는 일반적으로 20ms 크기의 오디오 패킷 10%를 손실할 경우, 음성 인식의 불편함을 느낀다고 알려져 있기 때문이다[2,8].

4.2 패킷 손실 측정 및 실험 결과 분석

그림 17과 그림 18은, 샘플링된 여러 (p, q)들에 대한 실험 측정 결과들 중에서, 패킷 손실률이 25.5%이고 (p, q) = (0.12, 0.35)인 경우에 대해 측정된 완전 손실률들을 비교하기 위하여 나타낸 것이다. 측정 시간 간격은 RTP/RTCP에서의 RR 패킷 전송 시간 간격과 유사하게 평균 5초로 두었다.

그림 17은 R0-Method ~ R4-Method의 부가 방식들을 정적으로 적용시킨 경우에 대한 각 방식들의 완전 손실률을 시간의 흐름에 따라 측정한 것이다. 이들 측정 결과를, 3.2절에서 계산한 이론 결과 값과 비교하면 표 2와 같다.

표 2 (p, q)=(0.12, 0.35)에서의 완전손실률

전송 방식	실험 결과(%)	이론적 계산 결과(%)
R0-Method(비부가 전송)	24.0	25.5
R1-Method(정적 전송)	15.4	16.6
R2-Method(정적 전송)	9.8	10.8
R3-Method(정적 전송)	4.4	5.0
R4-Method(정적 전송)	1.2	1.6

측정된 실험 결과들은, 표 2에서 알 수 있듯이, 이론적 계산 결과들에 거의 유사한 완전 손실률을 보인다. 다만, 실험할 당시의 전체 패킷 손실률은 이론적으로 계산된 전체 패킷 손실률보다는 1.5%정도 낮았다. 따라서, 측정된 각 부가 방식들의 완전 손실률도 이론 계산 값보다는 낮은 결과를 나타내었다.

실험 결과들을 비교해 보면, 부가 전송을 사용하지 않는 R0-Method의 경우에는 평균 24.0%의 손실률을 보인 반면에 R1-Method와 R2-Method는 각각 15.4%, 9.8%, 그리고 R3-Method와 R4-Method는 각각 4.4%, 1.2%의 완전 손실률을 보였다. 따라서, R3-Method와 R4-Method는 손실률 허용 임계값보다 낮은 손실률을 가진다는 것을 알 수 있다. 그러므로, 3.3절에서 제안한 동적 부가 전송 방법은, 주로 R3-Method과 R4-Method를 사용하여 손실률 허용 임계값에 근접한 완전 손실률을 유지하면서도, 일시적으로 패킷 손실이 적게 발생하는 경우에는 R2-Method를 사용함으로써 부가 전송으로 인한 추가 대역폭 증가를 줄이고자 하였다.

그림 18은 동일한 실험 환경에서 비부가 방식의 경우와 동적 부가 전송 방법을 사용한 경우에 대한 완전 손실률을 측정한 것이다. 그래프에서 보는 바와 같이, 동적 부가 전송 방법을 사용한 경우의 손실률은 대략 5%를 중심으로 변화하였다. 이것은 동적 부가 전송 방법을 사용한 경우, 시간이 경과하더라고 임계값과 거의 유사하게 손실률을 유지한다는 것을 보여 준다.

동적 부가 전송 방법에서 실제로 적용된 부가 방식들을 측정해 보면, R2-Method, R3-Method, R4-Method가 각각 9.5%, 57.2%, 33.3% 적용되었음을 알 수 있었다. 이 결과는 3.2절에서 구한 그림 10의 경계 그래프를 이용하여 분석 가능하다. 손실률 허용 임계값(α)이 5%인 경우는 그림 10 (a)에 해당되고, 편의상 동일한 경계 그래프를 그림 19에 나타내었다. (p, q) = (0.12, 0.35)는 그림 19에서의 ①점이 된다. ①점은

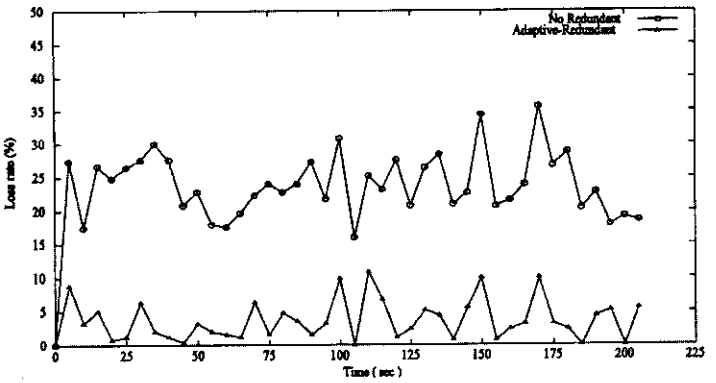


그림 18 동적 부가 전송 방법의 적용 여부에 따른 손실률 비교
(p =0.12, q =0.35, α= 5%)

R3-Method의 경계 근처에 위치하므로, R3-Method나 이 점보다 아래에 위치한 R4-Method를 사용하면 손실률 허용 임계값(α)보다 같거나 낮은 완전 손실률을 가지지만, 이보다 위쪽에 위치한 R2-Method 혹은 R1-Method로는 해당 임계값 α를 만족시킬 수 없다는 것을 의미한다. 그러나, 만일 R4-Method만을 사용하면 α보다는 훨씬 낮은 손실률을 가질 수 있지만, 네트워크 상황이 동적으로 변화하여 패킷 손실이 적은 경우에는 불필요하게 대역폭을 낭비하게 된다. 따라서, 네트워크 대역폭의 증가를 최소화시키면서도 해당 손실률 허용 임계값보다 작은 손실률을 가지기 위해서는 실제 측정 결과와 같이, R2-Method, R3-Method, R4-Method가 복합적으로 사용되어져야 함을 알 수 있다.

본 논문에서 제안하는 복구 방법은 손실률 허용 임계값에 가장 근접한 손실률을 유지하면서도, 잉여 데이타 부가 전송으로 인한 대역폭 요구를 최소화하고자 하는 의도에 부합하도록 수행되었다. 즉, R2-Method와 R3-Mehod 그리고 R4-Method를 복합하여 적용함으로써, 손실률 허용 임계값에 근접한 손실률을 유지하게 하였다. 특히 R3-Method를 중점적으로 적용하면서도, 송·수신자 사이의 네트워크 상황이 동적으로 변화하는 경우, 수신자로부터 피드백된 패킷 손실 특성에 따라 다른 부가 방식들도 간헐적으로 적용함으로써, 부가 전송으로 인한 오버헤드와 해당 손실률 허용 임계값의 보장이라는 두 가지 측면을 적응성있게 조절함을 알 수 있었다. 실험을 위해 샘플링한 (p, q)들에 대한 결과들을 분석해 보면, 대체적으로 위와 유사한 결과를 나타내었다. 다만, 그림 19의 ②와 같이, (p, q)의 위치가 R4-Method의 경계보다 아래에 있는 경우에는 모든 부가 방식들이 해당 손실률 허용 임계값(α)보다는 큰 결과를 보였는데, 이 결과는 어떠한 부가 방식들을 적용하더라도 해당 α 값을 만족시키지 못함을 나타낸다. ②의 위치는 (p, q) = (0.2, 0.2)이므로, 이 때의 패킷 손실률은 50.0%이다. 즉, 전송된 패킷의 $\frac{1}{2}$이 손실되므로, 손실률 허용 임계값 5%를 만족시키지 못하는 상황이다. 그러나, 동적 부가 전송 방법의 경우에는 R4-Method를 적용하여, 완전 손실률을 R4-Method만큼 낮추도록 수행시킴으로써, 비부가 전송의 경우에 비해서는 훨씬 낮은 손실률을 유지하도록 하였다.

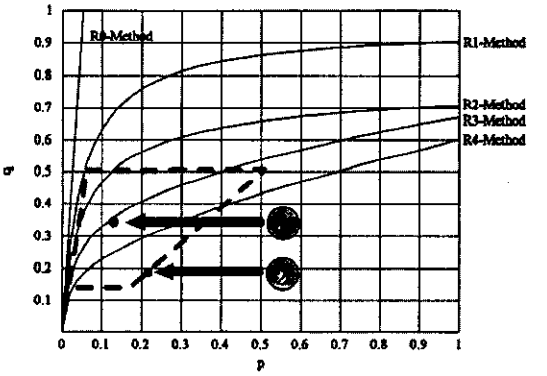


그림 19 α=5%일 때, 부가 방식들의 경계 그래프

표 3은 동일한 패킷 손실률 33%를 가지는 (p, q)들에 대해 실험하여 측정한 완전 손실률을 비교한 것이다. 표 3에서는 각각의 (p, q)마다 측정된 완전 손실률이 다름을 보여준다. 이것은 패킷 손실률 뿐만 아니라 패킷의 연속적인 손실 특성도 완전 손실률에 커다란 매개 변수

표 3 패킷 손실률 33%을 가지는 (p, q)에 따른 완전 손실률 비교(단위: %)

부가 방식 \ (p, q)	(0.1, 0.2)	(0.15, 0.3)	(0.2, 0.4)	(0.3, 0.6)
No (비부가)	31.45	32.03	32.31	33.29
R1-Method	25.23	22.51	19.27	13.28
R2-Method	20.16	15.60	11.29	5.17
R3-Method	13.60	8.41	4.75	1.85
R4-Method	6.78	3.21	1.68	0.62
Adaptive (동적 부가)	7.33	4.16	3.42	3.49

로 작용함을 의미한다. 즉, $1-q$가 패킷의 연속적인 손실 특성을 반영하기 때문에, q의 값이 낮으면 연속적인 손실의 개수가 높다는 의미이다. 따라서, 연속적인 손실의 개수가 증가할수록 q 값은 낮아지게 되는데, 만약 이 값이 각 부가 방식이 허용할 수 있는 연속 손실의 복구 한계를 초과한 경우에는, 손실된 패킷을 제대로 복구할 수 없으므로 완전 손실률은 높아지게 된다.

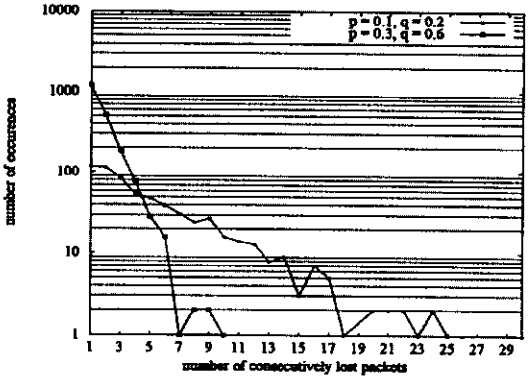


그림 20은 패킷 손실률 33%를 가지는 (p, q)들에 대한 연속적인 패킷 손실의 빈도수 분포를 그린 것이다. 두 곡선 모두 기하 분포를 따른다는 점에서는 유사하지만, (p, q)에 따라 기울기가 달라짐을 알 수 있다. (0.3, 0.6)의 경우에는 기울기가 급격하게 변하고 연속적으로 손실된 패킷이 최대 10개까지 나타난다. 그러나 (0.1, 0.2)의 경우에 기울기는 완만하지만, 연속적으로 손실되는 패킷의 개수가 최대 30개까지 나타난다. 따라서, 같은 패킷 손실률을 가지는 경우에도, 패킷 손실 특성은 다르게 발생할 수 있음을 보여준다. 이 사실은 본 논문에서 분석한 패킷 손실 특성들이, 손실 복구를 위한 잉여 데이타 부가 방식의 선택에 있어서 중요한 매개 변수임을 보여주는 것이다.

5. 결론 및 향후 과제

본 논문에서는 이동 컴퓨팅 환경에서의 실시간 데이타 전송시 발생하는 간헐적인 패킷 손실 및 연속적인 패킷 손실을 복구하기 위하여 잉여 데이타의 동적 부가 전송 방법을 제안하였다. 제안된 방법은 낮은 전송률과 링크 특성에 의한 높은 BER(Bit Error Rate) 및 사용자의 이동성에 의한 패킷 전송 지연 및 연속적인 손실 특성을 가지는 이동 컴퓨팅 환경 상에서, 전송시에 발생되는 패킷 손실 특성을 길버트 모델을 이용하여 분석하고, 패킷 손실 특성에 따라 송신자 측에서 적절하게 잉여 데이타를 부가하여 전송함으로써, 수신자 측에서는 이를 이용하여 손실된 패킷을 복구할 수 있도록 하였다. 송신자 측에서는 현재의 패킷 손실 특성에 따라 잉여 데이타의 개수를 동적으로 부가하고, 특히 연속적인 패킷 손실이 많은 경우에는 원시 데이타와 비교적 큰 오프셋 값 즉, -4, -8의 오프셋 값을 가지는 잉여 데이타들을 선택하여 인터리빙 방식의 효과를 가지도록 함으로써, 전체 패킷 손실률 뿐만 아니라 패킷 손실의 연속성도 감소시키고자 하였다. 또한, 이러한 동적인 적용은 패킷 손실이 적은 경우에는 잉여 데이타 부가 전송으로 인한 추가 대역폭 및 처리시간을 감소시킬 수 있다. 기존의 ARQ 방법 등에서는 손실률이 높은 경우에는 원시 패킷의 반복적인 재전송을 수행함으로써 대역폭이 증가할 뿐만 아니라, 재전송에 기인한 복구 지연 시간 즉, 재생 지연이 커진다는 단점을 가진다. 이에 비해 본 논문이 제안하는 방법에서는 대역폭과 복구 지연 시간의 증가를 상대적으로 줄일 수 있다.

제안된 방법은, 실시간 데이타 및 잉여 데이타의 전송을 위해 기존의 RTP 프로토콜을 이용하였으며, 수신자 측의 패킷 손실 정보를 전송하기 위하여 RTCP APP 패킷을 사용하였다. 그리고 Mobile IP를 사용함으로써 사용자의 이동성에 대한 투명성을 보장하도록 하였다. 또한 제안한 알고리즘을 구현하여, 비부가 및 정적 부가 방식들과 비교 실험 및 분석해 봄으로써, 패킷 복구율이 향상되었음을 확인하였다.

본 논문에서는 실시간성에 민감한 인터넷 전화나 음성회의와 같은 응용을 고려하여, 오프셋의 최대값을 -8로 제한하였지만, 실시간성의 제약이 적은 인터넷 방송과 같은 경우를 고려한 오프셋 값 결정에 대한 연구도 필요하다. 그리고 이동 컴퓨팅 환경뿐만 아니라 유무선 복합망 환경에서 전송시에 발생되는 트래픽 변화 및 패

킷 손실 특성에 대한 지속적인 분석 및 실험이 필요하다.

참 고 문 헌

[1] 박준서, 고대식, "저비트율 잉여오디오 정보를 이용한 손실 패킷 복구 방법의 구현 및 성능평가", 대한전자공학회 논문지, 제35권, 제7호, 1998.

[2] 강민규, 궁상환, 김동규, "RTP/RTCP를 이용한 영상회의 시스템에서 오디오 패킷 손실 보상을 위한 동적 부가 전송 메카니즘 개발 및 성능 분석", 한국정보처리학회 논문지, 제5권, 제10호, Oct. 1998.

[3] Kevin Brown and Suresh Singh, "Quality of Service Guarantees in Mobile Computing," J. Computer Communications, Vol. 19, 1996.

[4] Kevin Brown and Suresh Singh, "M-UDP : UDP for Mobile Computing," ACM Computer Communication Review, Oct. 1996.

[5] Kevin Brown and Suresh Singh, "Extensions to RTP to support Mobile Networking," 3rd Intl. Workshop on Mobile Multimedia Communication, Sept. 25-27, 1996.

[6] Internet Engineering Task Force, "RTP : A Transport Protocol for Real-Time Applications," RFC 1889, January 1996.

[7] 배정숙, 이은봉, 이재용, 신명기, 함진호, "실시간 멀티미디어 데이타 서비스를 위한 WWW 시스템 구현", 한국정보과학회 논문지(C), 제5권, 제2호, 1999.

[8] V. Hardman, A. Sasse, M. Handley, A. Handley, A. Watson, "Reliable audio for use over the Internet," *Proc. INET'95*, Honolulu, HI, pp. 171-178, June 1995.

[9] Jean-Chrysostome Bolot, Hugues Crepin, Andres Vega Garcia, "Analysis of Audio Packet loss in the Internet," Proceedings NOSSADV 95 (Network and Operating System Support for Digital Audio and Video), pp. 163-174, Durham, NH, April 1995.

[10] J-C. Bolot, S. Fosse-Parisis, D. Towsley, "Adaptive FEC-Based error control for Internet Telephony," Proc. Infocom'99, New York, March 1999.

[11] 박준서, 고대식, "인터넷상의 실시간 오디오 방송 서비스 구현", 한국통신학회 논문지, Vol. 23, No. 6, 1998.

[12] C. Perkins, O. Hodson, "A survey of packet loss recovery techniques for streaming media," *IEEE Network*, Sept. Oct. 1998.

[13] rtptools, http://www.cs.columbia.edu/~hgs/rtptools/

[14] NeVot, http://www.cs.columbia.edu/~hgs/nevot/

[15] HP Mobile IP, http://www.hpl.hp.com/personal/Jean_Tourrilhes/Mobile IP/

[16] C. Perkins, "IP Mobility Support," IETF RFC2002, Oct. 1996.

오 연 주

1998년 국립경상대학교 컴퓨터과학과(학사). 1999년 ~ 2001년 경북대학교 대학원 컴퓨터과학과(석사). 2000년 ~ 현재 ETRI 네트워크기술연구소 홈네트워킹팀(위촉연구원). 관심분야는 홈네트워킹기술, VoIP, RTP/RTCP, Mobile IP

백 낙 훈

1990년 한국과학기술원 과학기술대학 전산학과 졸업. 1992년 한국과학기술원 전산학과 공학석사. 1997년 한국과학기술원 전산학과 공학박사. 1997년 ~ 1998년 미국 죠오지워싱턴 대학 초청 연구원. 1998년 ~ 2000년 경북대학교 전자전기공학부 초빙교수. 2000년 ~ 2001년 2월 경북대학교 전자전기공학부 BK21 계약교수. 2001년 3월 ~ 현재 동국대학교 컴퓨터멀티미디어공학과 전임강사. 관심분야는 컴퓨터 그래픽스, 계산기하학, 계산 이론, 멀티미디어 시스템

박 광 로

1982년 경북대학교 전자공학과. 1985년 경북대학교 대학원. 1984년 ~ 현재 ETRI 네트워크기술연구소 홈네트워킹팀장, 책임연구원. 관심분야는 홈네트워킹기술, 무선LAN기술, Mobile IP

정 해 원

1980년 2월 한국항공대학교 항공통신정보공학과(학사). 1982년 2월 한국항공대학원 항공전자공학과(석사). 1999년 2월 한국항공대학원 항공통신정보공학과(박사). 1982년 3월 ~ 현재 ETRI 네트워크기술연구소 기가접속팀장, 책임연구원. 관심분야는 무선 LAN, 홈네트워킹, 기가비트이더넷

임 경 식

1982년 경북대학교 전자공학과(공학사). 1985년 한국과학기술원 전산학과(공학석사). University of Florida 전산학과(공학박사). 1985 ~ 1998년 한국전자통신연구원 책임연구원, 실장. 1998년 ~ 현재 경북대학교 컴퓨터과학과 조교수. 관심분야는 이동 컴퓨팅, 무선 인터넷, 홈 네트워킹, 컴퓨터통신