네트워크 상태와 데이타 중요도에 기반한 패킷 손실 제어 기법

(A Packet Loss Control Scheme based on Network Conditions and Data Priority)

최 태 욱 † 정 기 동 ††
(Tae-Uk Choi) (Ki-Dong Chung)

요 약 본 연구는 erasure 코드를 이용한 응용계층 FEC 기법에 대해 설명한다. Erasure 코드는 디코딩 알고리즘이 간단하여 응용수준에서 패킷 단위 에러 복구에 효율적이다. 그러나 많은 양의 패리티 패킷을 보내는 것은 에러 복구율을 높일 수 있지만 네트웍 혼잡 상황을 악화시킬 수 있다. 따라서 네트웍 상태에 적응적으로 패리티 패킷의 양을 조절할 수 있는 부가정보조절 기법이 필요하다. 또한, 비디오 데이타와 같이 우선순위가 있는 데이타의 경우 마땅히 높은 우선순위의 데이타가 더 많은 부가정보를 가져야 한다. 본 연구는 네트웍 상태(손실 정보와 혼잡 정보)와 데이타 중요도에 기반한 패킷 손실 제어 기법을 제안하고 단순링크와 혼잡링크에서 그 성능을 평가한다.

키워드 : FEC, Erasure 부호, 에러 제어, 패킷 손실, 멀티미디어 응용

Abstract This study discusses Application-layer FEC using erasure codes. Because of the simple decoding process, erasure codes are used effectively in Application-layer FEC to deal with packet-level errors. The large number of parity packets makes the loss rate to be small, but causes the network congestion to be worse. Thus, a redundancy control algorithm that can adjust the number of parity packets depending on network conditions is necessary. In addition, it is natural that high-priority frames such as I frames should produce more parity packets than low-priority frames such as P and B frames. In this paper, we propose a redundancy control algorithm that can adjust the amount of redundancy depending on the network conditions and depending on data priority, and test the performance in simple links and congestion links.

Key words : FEC, Erasure Code, Error Control, Packet Loss, Multimedia Application

1. Introduction

Real-time video transmission over the Internet suffers due to packet loss, delay and jitter. Packet loss severely degrades video quality, and excessive network delay lowers user interactivity between end-to-end users. In addition, because most video compression standards such as MPEG I, II and H.263 use motion estimation and compensation, a small loss in a frame can be propagated to subsequent frames. Many techniques to prevent error propagation have been proposed[2]-[6]. Choi et al. classified these techniques of preventing error propagation into codec-level and network-level schemes[1]. Codec-level schemes such as ET(Error Tracking) and RPS(Reference Picture Selection) prevent error propagation during encoding and decoding frames. Network-level schemes such as FEC and ARQ(Automatic Repeat Request) control the errors during sending and receiving packets. FEC sends the original data along with the redundant data and tries to recover the lost data from the redundant data when the original data is lost. Because FEC does not require additional

· 본 연구는 한국과학재단 목적기초연구(R05-2002-000-00354-0)지원으로 수행되었음
† 비 회 원 : 부산대학교 전자계산학과
tuchoi@melon.cs.pusan.ac.kr
†† 종신회원 : 부산대학교 전자계산학과 교수
kdchung@melon.cs.pusan.ac.kr
논문접수 : 2003년 2월 11일
심사완료 : 2003년 9월 15일

retransmission delay, it is preferred to ARQ in real-time multimedia applications.

FEC can be implemented in a Link-layer or Application-layer protocol. Link-layer FEC recovers bit errors, which can occur during data transmission. It uses error correcting codes[7]-[10] such as the BCH(Bose-Chaudhuri-Hocquenghem), RS (Reed-Solomon), Viterbi and Turbo codes. In fact, the Viterbi code has been used in CDMA(IS-95), and the Viterbi and Turbo codes have been used in IMT-2000. Application-layer FEC recovers erasures which means unreadable symbols. It uses erasure codes such as the RSE(Reed-Solomon Erasure) code[11]. The difference between erasure codes and error correcting codes is the awareness of the location of lost information. Because applications can locate the lost packets using packet sequences, they can utilize erasure codes to deal with packet-level errors. Erasure codes can be implemented easier than error correcting codes since they allow a simple decoding process resulting from the awareness of location.

Fig. 1 shows an example of Application-layer FEC[12]. One can feed k source data packets to an encoder, which produces n encoded packets ($n>k$) in such a way that any subset of k encoded packets allows reconstruction of the source data at the receiver. Such a code is called an (n,k) code. In addition, the k data packets are called the TG(Transmission Group), and the n packets are called the FEC block. In (n,k) erasure codes, by choosing suitable values for n and k, the residual error rate after decoding can be small.

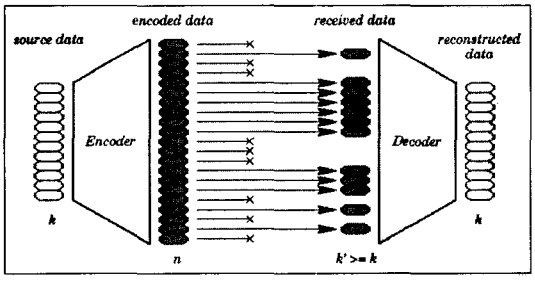


Fig. 1 The decoding and encoding process of the erasure code: the encoder takes k data packets to encode n packets. The decoder is able to recover the source data even if any $n-k$ packets are lost.

How many parity packets (redundancy; $n-k$) should be produced? It depends on network conditions and the priority of the data. When the packet loss rate is high, the amount of redundancy should increase to recover lost packets. However, the redundancy may cause the network situation to be worse when the network is congested. Thus, the congestion information as well as network loss conditions should be considered to determine the number of parity packets. In addition, the priority of the data should be considered. It is natural that high-priority frames such as I frames should produce more parity packets than low-priority frames such as P and B frames. In this paper, we consider Application-layer FEC using erasure codes, and propose a redundancy control algorithm to determine the optimal n and k values of the erasure code. The redundancy control algorithm can adjust the number of parity packets depending on the network conditions and the priority of the frames.

The paper is structured as follows: in Section 2, we analyze the performance of erasure codes using two packet loss models, which will be used for adaptability to network loss conditions. Section 3 describes two approaches for redundancy control of the erasure code. Section 4 shows experimental results of the proposed redundancy control algorithm in simple links and in congestion links. Lastly, the conclusion is drawn in Section 5.

2. Loss Models of the Erasure Code

In this section, we estimate the TG reception rate of erasure codes using two statistical network models. The one-state model assumes that packet losses are independent and network conditions are represented by the packet loss rate p. The two-state model assumes that losses are dependent or temporally correlated to and burst losses are represented by the Gilbert Model[13].

2.1 One-state Loss Model

In (n,k) erasure code, we know that as the number of parity packets $(n-k)$ increases, the probability of receiving a TG successfully becomes higher. If a receiver can estimate the loss rate from the sender to itself, it can request more parity

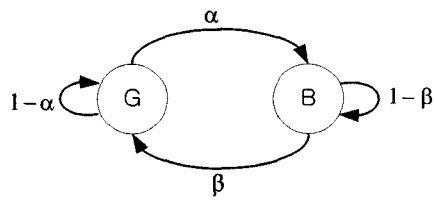


Fig. 2 Gilbert Model: In a Good state (G), errors would occur with low probability P_G while in a Bad state (B), they occur with high probability P_B. α and β represent the state transition rates. The errors occur in bursts with relatively long error- free intervals between them. The model with P_G = 0 and P_B = 1 is referred to as the simplified Gilbert Model.

packets than what it actually needs in order to protect frame loss with probabilities that are close to one. To do so, the receiver must be able to calculate the probability of receiving at least k packets out of an FEC block of n packets, or the probability that a TG is received and reconstructed successfully. We represent this value by the TG reception rate $R(n,k)$. Thus, when the loss is modeled as a temporally independent loss process with packet loss probability p, then $R(n,k)$ can be obtained using the binomial distribution as follows:

$$R(n,k)=\sum_{i=k}^{n}\binom{n}{i}(1-p)^i p^{n-i} \tag{1}$$

$R(n,k)$ is calculated quickly and recursively using equations (2) and (3) [5].

$$R(n,k)=R(n-1,k)+\binom{n-1}{k-1}(1-p)^k p^{n-k} \tag{2}$$

$$R(n,n)=(1-p)^n \tag{3}$$

2.2 Two-state Loss Model

The previous results assume that losses are not temporally correlated, and will produce inaccurate calculations for the network with burst losses. Burst losses in the Internet can be accurately modeled using the two-state loss model. Fig. 2 shows a state diagram for the two-state Markov model, which was first used by Gilbert[13]. In the model, a receiver can calculate $D(n,k)$ the probability of receiving only k packets out of n packets. Let $D(n,k|B)$ be the probability of sending n packets, receiving at least k packets and winding up in state B, and $D(n,k|G)$ be the similar probability but winds up in state G. Thus, $D(n,k)$ is calculated as follows:

$$D(n,k)=D(n,k|B)+D(n,k|G) \tag{4}$$

where

$$D(n,k|G)=D(n-1,k-1|B)\beta+D(n-1,k-1|G)(1-\alpha) \tag{5}$$

$$D(n,k|B)=D(n-1,k|B)(1-\beta)+D(n-1,k|G)\alpha \tag{6}$$

$D(n,k)$ can also be obtained recursively by using the following boundary conditions[5]:

$$D(n,0|G)=0 \tag{7}$$

$$D(n,n|G)=\pi_G(1-\alpha)^n+\pi_B\beta(1-\alpha)^{n-1} \tag{8}$$

$$D(n,n|B)=0 \tag{9}$$

$$D(n,0|B)=\pi_G\alpha(1-\beta)^{n-1}+\pi_B(1-\beta)^n \tag{10}$$

Then, in the two-state loss model, the TG reception rate $R(n,k)$ can be obtained using $D(n,k)$ as follows:

$$R(n,k)=\sum_{i=k}^{n}D(n,i) \tag{11}$$

The transition probability α and β can be estimated by the mean length of the nonloss period T_R and the mean length of the loss period T_L. Because of the state transition rate of the Markov chain, α is $(T_R)^{-1}$ and β is $(T_L)^{-1}$. π_G indicates the probability that the system initially resides in state G and is calculated by $\pi_G = \beta/(\alpha+\beta)$, and π_B indicates the probability that the system initially resides in state B and is obtained by $\pi B = \alpha/(\alpha+\beta)$.

RTCP can be used to give feedback on the QoS information. The mean length of the current nonloss periodic T_R^t and the mean length of the current loss period T_L^t are calculated and smoothed out with the smoothing factor δ by the QoS monitor every RTCP period t. This smoothing diminishes oscillation of T_R and T_L.

$$T_R=\delta T_R^t+(1-\delta)T_R \tag{12}$$

$$T_L=\delta T_L^t+(1-\delta)T_L \tag{13}$$

After obtaining T_R and T_L, the transition probability α and β, and stationary probability π_G and π_B are recalculated at regular RTCP intervals.

3. Adaptive Packet Loss Control depending on Network Conditions and Data Priority

In this section, we describe a redundancy control algorithm for Application-layer FEC using erasure codes in order to reduce bandwidth overhead of redundant data and improve the performance of FEC. The goal of redundancy control is to determine the optimal n and k values of the erasure code. Fig. 3 shows the proposed system architecture for redundancy control. The source encoder and decoder deal with the source data. The FEC encoder produces the parity packets using the erasure code, and the FEC decoder reconstructs lost data packets using parity packets. The QoS monitor collects the QoS information including loss, delay, T_R and T_L. The redundancy controller determines the values of k and n based on RTCP packets containing the QoS information from the receiver as well as the priority of frames from the source encoder.

3.1 Adaptation to Network Conditions

One approach to redundancy control is to adjust the number of parity packets depending on packet loss conditions. The number of parity packets should increase when the packet loss rate is high and decrease when the packet loss rate is low. As shown in Fig. 3, a video frame can be encoded through the source encoder and packetized through the FEC encoder. To make an encoded frame into a FEC block, the k value can be obtained by dividing the encoded frame size by the packet size.

$$k = \left\lceil \frac{the\ frame\ size}{the\ packet\ size} \right\rceil \quad (14)$$

The erasure code proposed by McAuley[2] uses a symbol size m with m=8 or m=32. When a packet size P is larger than a symbol size m, we need to choose a $P = S \times m$ where S is an integer[6]. So, we selected P to be among multiples of m=8 or m=32.

Through the analysis of the previous section, we can find out that if k is fixed and n is very high, $R(n,k)$ is extremely close to one. So, an appropriate n that satisfies $R(n,k) \geq T$ can be determined so that a TG is received with a probability larger than any $T<1$. That is, when a tolerable TG loss rate (τ) is given, n is obtained by the smallest t that satisfies $R(t,k) \geq 1-\tau$ so that a TG can be received with a probability larger than $1-\tau$.

$$n = \min_{k \leq t \leq MAX} \{t \mid R(t,k) \geq 1-\tau\}, \quad where \quad 0 < \tau < 1 \quad (15)$$

However, the redundancy can make the network situation worse when losses are due to network congestion. So, congestion information should be considered to produce the parity packets. When the network becomes congested, TCP controls the transmission rate using its AIMD(Additive Increase Multiplicative Decrease) algorithm. Because RTP has no rate control mechanism, most multimedia applications using RTP should control the transmission rate based on TCP-friendly congestion control algorithms[15][16]. Similarly, we employ a TCP-friendly rate adaptation method to prevent network congestion. This method estimates B_{TCP}, the transmission rate of TCP flows, which competes with RTP flows in the network path, then adjusts the τ value depending on the estimated TCP rate. B_{TCP} can be given by[17].

$$B_{TCP} = \frac{C \times M}{t_{RTT} \times \sqrt{loss}} \quad (16)$$

where M is the maximum packet length, t_{RTT} is the round-trip delay, and C is a constant which is

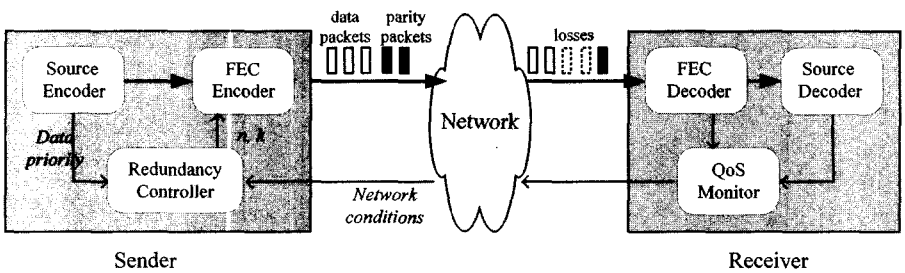


Fig. 3 Simple Video Streaming Architecture that uses Application-layer FEC

generally 1.22 or 1.36. As shown in equation (15), the redundancy decreases as τ increases. Thus, τ should increase if the current transmission rate (B_{RC}) is larger than B_{TCP}, while τ should decrease if B_{RC} is less than B_{TCP}. B_{RC} is given by

$$B_{RC} = \sum_{i=1}^{s} n_i \cdot P \tag{17}$$

where s is the frame rate (fps), n_i is the n value of the i-th frame, and P means the packet size.

Fig. 4 shows the proposed redundancy control algorithm (RC-I) that has adaptability to network conditions. It can adjust the number of parity packets considering network congestion as well as network loss conditions. Note that the n and k values are computed when a frame is encoded, and τ_i is recalculated every RTCP period since the B_{TCP} can be computed using the feedback information of RTCP packets. An increase factor (INC) is used to increase τ_i and a decrease factor (DEC) is used to decrease τ_i. Through experiments, we used INC=2 and DEC=0.2.

```
while (running) {
    /* depending on congestion state */
    if (new RTCP period) {
        if ( B_RC > B_TCP ) τ_i = τ_{i-1} × INC
        else τ_i = τ_{i-1} × DEC
    }
    /* depending on network loss conditions */
    if (new encoded frame) {
        k = ⌈the frame size / the packet size⌉
        t = k
        while ( R(t,k) < 1-τ_i ) { t = t+1 }
        n = t
    }
}
```

Fig. 4 RC-I algorithm: Redundancy control algorithm considering network conditions

3.2 Adaptation to Data Priority

Another approach to control the redundancy is to utilize data priority. For MPEG video, if I frames have more parity packets than P frames or B frames, the loss probability of I frames will decrease to prevent error propagation. In case of H.263, most frames are P frames, but key frames can be chosen to give priority. Choi[1] proposed Periodic FEC (PFEC) that divides video frames into *periodic frames* and *nonperiodic frames*, and produces parity packets for only periodic frames. This scheme can reduce the amount of redundancy much less than conventional FEC while it keeps video quality similar to conventional FEC.

As shown in Fig. 5, a periodic frame makes reference to its previous periodic frame, and a nonperiodic frame depends on its previous periodic frame. It means that the effect of the loss of a nonperiodic frame is limited to the period only if the periodic frame has no error in that period. However, if a periodic frame is lost, the error will propagate to the next period. Thus, the periodic frames should be protected with higher priority than nonperiodic frames. Fig. 6 shows PFEC using the RSE(Reed-Solomon Erasure) code. Because there are many more parity packets in periodic frames, the loss probability of periodic frames

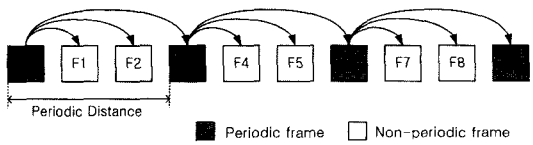


Fig. 5 Periodic referencing of PFEC: the shaded blocks denote periodic frames and the white blocks denote nonperiodic frames. Periodic Distance is the distance between two consecutive periodic frames.

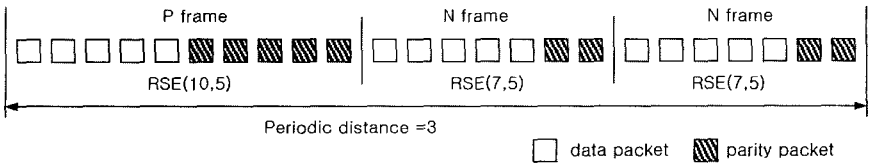


Fig. 6 PFEC using the erasure code. The periodic frame has more parity packets than the nonperiodic frame.

becomes smaller than that of nonperiodic frames.

In order to allow the differential amount of parity packets, we use two tolerable loss rates at the receiver: one is the tolerable TG loss rate for periodic frames τ_i^p and the other is the tolerable TG loss rate for nonperiodic frames τ_i^n. τ_i^n should be larger than τ_i^p. Fig. 7 shows the proposed redundancy control algorithm (RC-II) that has adaptability to data priority as well as network conditions. First, the algorithm recalculates τ_i^p and τ_i^n depending on the estimated TCP bandwidth (B_{TCP}) every RTCP period. Second, the threshold value (τ) is selected depending on frame types. Then, the k value is determined, and the n value is obtained by searching the smallest t that satisfies the condition of equation (15).

```
while (running) {
    if (new RTCP period)  {  /* depending on congestion state */
        if ( B_RC > B_TCP ) {
            τ_i^p = τ_{i-1}^p × INC  and  τ_i^n = τ_{i-1}^n × INC
        } else {
            τ_i^p = τ_{i-1}^p × DEC  and  τ_i^n = τ_{i-1}^n × DEC
        }
    }
    if (a new encoded frame) {
        if  (periodic frame)  τ = τ_i^p     /* depending on
        else  τ = τ_i^n                      the priority of the frame */
        k = ⌈the frame size / the packet size⌉
        t = k
        While ( R(t,k) < 1-τ ) { t = t+1 } /* depending on network
        n = t                                loss conditions */
    }
}
```

Fig. 7 RC-II algorithm: Redundancy control algorithm considering data priority and network conditions

4. Experiment

We implemented a prototype system of Fig. 3 to evaluate the proposed redundancy control algorithms. The video encoder and decoder were implemented by modifying the H.263 standard source. Erasure codes are implemented using Vandermonde matrices described in [12]. The other modules such as Redundancy Controller and QoS Monitor were programmed using Visual C++ 6.0. For the experiment, we distinguish the packet loss in *simple link* from the loss in *congestion link* because the proposed RC-I and RC-II schemes depend on the characteristic of the network link. Packet loss in simple link is due to bit errors, but the loss in congestion link mainly originates from the congestion node. The video sequence "akiyo" was employed which has 176×144 QCIF and 10 fps. The average encoded frame size was between 100 and 400 bytes, and a packet size of 80 bytes was used.

Table 1 Characteristics of simple links based on the Gilbert model

Channel No.	α	β	Avg. loss rate (ρ)
1	0.095	0.769	0.109
2	0.118	0.599	0.164
3	0.160	0.569	0.219
4	0.193	0.469	0.291
5	0.224	0.450	0.332

4.1 Performance in Simple Links

We used the simplified Gilbert model at the packet level to demonstrate the simple links. As shown in Fig. 2, the Gilbert channel has two states: the good state (G) and the bad state (B). A packet is transmitted correctly when the channel is in the good state and errors occur when the channel is in the bad state. The average error rate (ρ) produced by the Gilbert channel is computed as [18]

$$\rho = P_G\pi_G + P_B\pi_B = \frac{P_G\alpha + P_B\beta}{\alpha+\beta},$$

where P_G is a loss probability in the good state, and P_B is a loss probability in the bad state. Because of P_G=0 and P_B=1 in the simplified Gilbert model, the average packet loss rate (ρ) of the simple links is $\rho = \alpha/(\alpha+\beta)$. We tested the proposed schemes varing the α and β values of the Gilbert channel. Table 1 shows state transition probabilities (α and β) and average loss packet loss rate(ρ) in five selected simple links. We compared RC-I and RC-II with Yuk's scheme (Yuk). Yuk is a media-independent scheme that estimates P_{DL}, the mean data packet loss rate after

recovery and controls the number of parity packets so that the condition (P_{DL} < target loss rate) is satisfied [2].

Fig. 8 shows average transmission rates of each scheme in simple links. The transmission rates of RC-I, RC-II and Yuk increase as the average loss rates increase. This is because the redundancy of each scheme becomes larger to protect frame losses depending on the network loss conditions. We see that the amount of transmitted data in RC-I is larger than in Yuk. Since a FEC block in Yuk may not consist of a frame but rather several frames, the total number of FEC blocks required to produce the redundancy in Yuk can be smaller than in RC-I. So, the transmission rate in Yuk is lower than in RC-I. It also shows that the transmission rate of RC-II is less than that of Yuk. This is because RC-II considers data priority while Yuk does not. That is, RC-II reduces the amount of redundancy not adding parity to nonperiodic frames.

Fig. 9 shows average frame loss rates in simple links. We can see that RC-I, RC-II and Yuk are able to control the frame loss rates. In addition, the loss rate of RC-I is lower than that of Yuk, while the loss rate of RC-II is higher than that that of Yuk. This is due to the fact that the frame loss rate is dependant on the amount of redundancy. Fig. 10 shows average PSNR in simple links. We also observe that the PSNR of RC-I is higher than that of Yuk, while the PSNR of RC-II is lower than that of Yuk. This is because the PSRN is dependent on the frame loss rate.

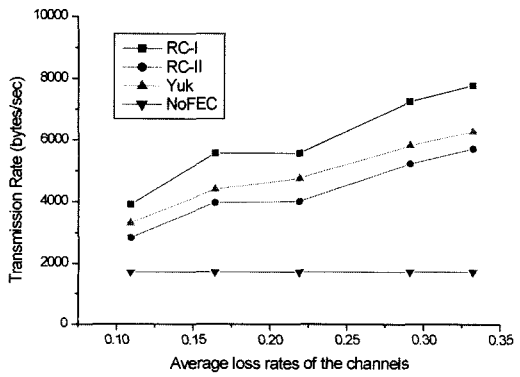


Fig. 8 Average transmission rate in simple links

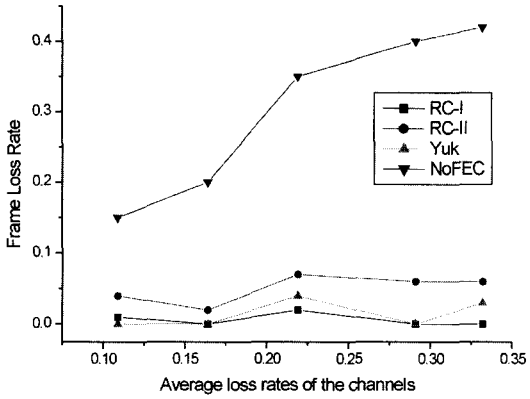


Fig. 9 Average frame loss rates in simple links

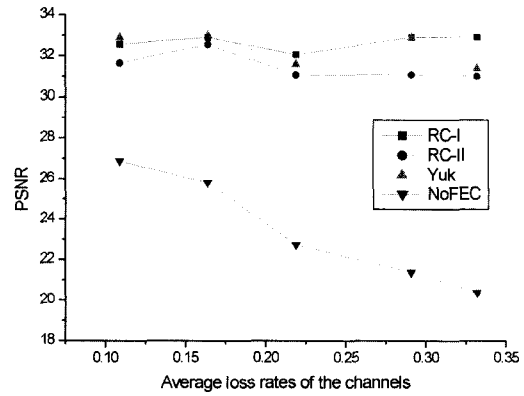


Fig. 10 Average PSNR in simple links

4.2 Performance in Congestion Links

In the Internet, packet losses mainly originate from congested nodes. We used the NS-2 network simulator to generate the network traffic in congestion links. Fig. 11 shows the topology of NS-2. All packets pass through the bottleneck link whose bandwidth is 500Kbps and propagation delay is 50ms. The number of end nodes is controlled to produce the various traffics. The traffics are monitored at one of the end nodes. Table 2 shows the characteristics of the five traces.

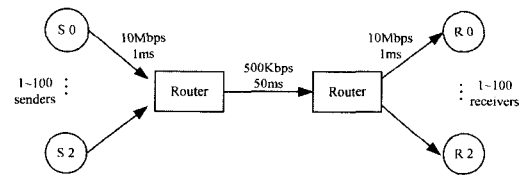


Fig. 11 Topology of NS-2 network simulator

Table 2 Characteristics of network traces generated by the NS-2 simulator

Trace No.	Number of Nodes	Avg. Loss Rate	Avg. Delay(ms)
1	5x2	0.055	90.2
2	10x2	0.090	102.4
3	20x2	0.170	112.9
4	40x2	0.225	115.3
5	80x2	0.299	123.4

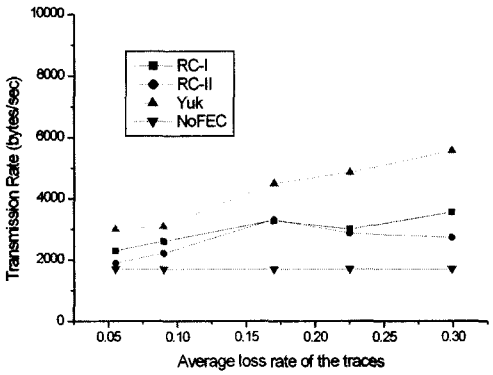


Fig. 12 Average transmission rate in congestion links

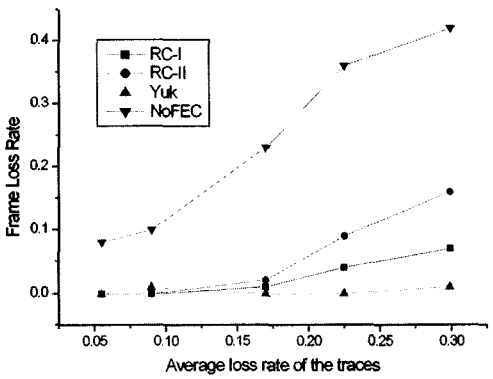


Fig. 13 Average frame loss rate in congestion links

Fig. 12 shows the average transmission rate in congestion links. The transmission rate of Yuk increases according to the average loss rates of the traces because Yuk produces parity packets without regards to other TCP traffic. However, RC-I and RC-II can control the transmission rate as high as the TCP bandwidth. So, the transmission rate of RC-I and RC-II do not increase depending on network loss conditions. We also see that RC-II produces a little less redundancy than RC-I.

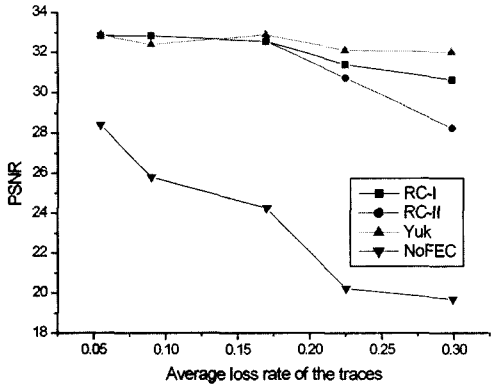


Fig. 14 Average PSNR in congestion links

Fig. 13 shows the frame loss rate in congestion links. We can see, compared to Fig. 9, that the frame loss rates of all schemes increase. Yuk achieves the lowest frame loss rate. The loss rate of RC-I and RC-II is higher than that of Yuk. It also shows that the loss rate of RC-I is rising faster than that of RC-II. These mean that the frame loss rate is dependent on the amount of redundancy.

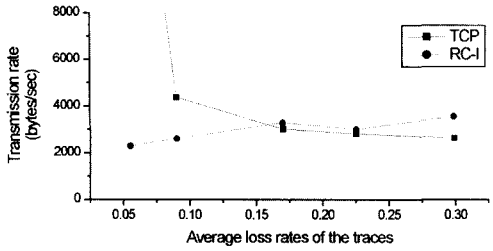
Fig. 14 shows the average PSNR of each scheme in congestion links. Yuk achieves the highest PSNR. We see that the PSNRs of RC-I and RC-II decrease faster than Yuk, and the line of RC-I goes down slower than that of RC-II. These mean that the PSNR is dependent on frame loss rates.

4.3 TCP friendliness

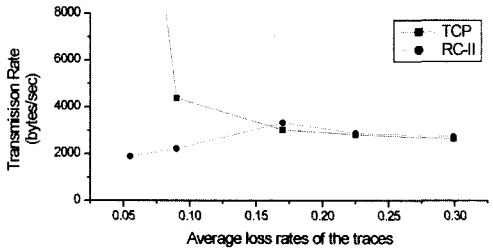
We compared the transmission rate of RC-I and RC-II with the estimated bandwidth of TCP to show TCP friendliness. Fig. 15 shows the comparison depending on the average loss rates. Fig. 16 shows the transmission rates during the initial 20 seconds when trace number 5 of Table 2 was used.

Regarding RC-I, we see in Fig. 15(a) that the transmission rate of RC-I becomes higher than that of TCP as the loss rates increase. This results from the heavy fluctuation of redundancy in RC-I (see Fig. 16(a)).

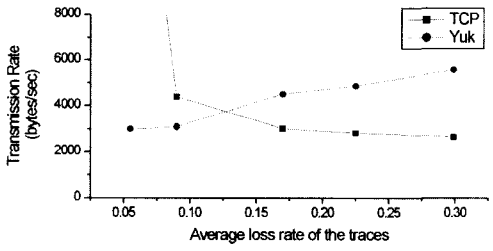
Regarding RC-II, we observe from Fig. 15(b) that RC-II keeps the transmission rate similar to TCP bandwidth as the loss rates increase. The reason is that RC-II can control the amount of redundancy more closely to TCP than RC-I, and as a result it can reduce excessive redundancy (see



(a)

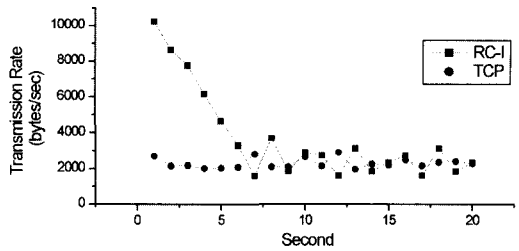


(b)

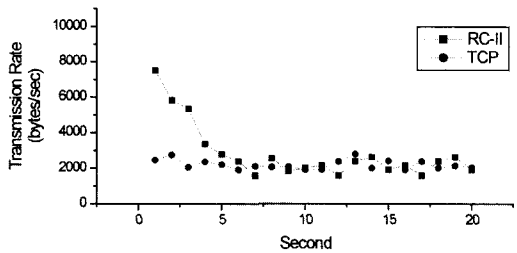


(c)

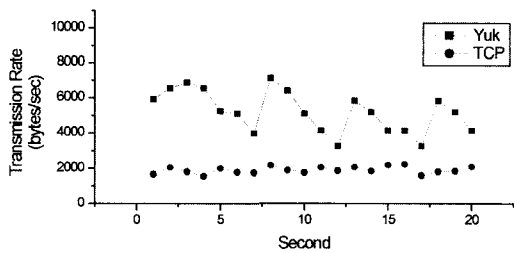
Fig. 15 Transmission rates of each scheme depending on the average loss rates.



(a)



(b)



(c)

Fig. 16 Transmission rates of each scheme depending on time (second).

Fig. 16(b)).

Regarding Yuk, we observe from Fig. 15(c) and Fig. 16(c) that, regardless of TCP bandwidth, the transmission rates of Yuk increase as the loss rate increase because Yuk does not consider the other competing TCP flows.

5. Related Work

Many error control schemes have been proposed to recover the lost packets in multimedia communication[1]-[6]. Yuk[2] controls the number of parity packets in RS-code-based real-time data transmission. In Yuk, after the k value is computed based on network delay, the n value is determined using *the packet loss rate after recovery* estimated by *two-state continuous-time Markov chain*. Its design focuses on real-time audio transmission rather than video transmission. So, Yuk is unable to control the redundancy for video frame. Moreover, it does not consider network congestion and data priority.

In [3] and [5], loss recovery techniques for real-time reliable multicast were proposed. These focus on the combination effects of forward error correction (FEC) with automatic repeat request (ARQ). They consider only the network loss conditions to control the redundancy of FEC.

Bolot[4] proposed control mechanisms for packet video in the internet. These mechanisms focus on adapting the bandwidth requirements and the resilience of packet losses of the video streams. However, they use the media-dependent FEC such as secondary encodings rather than the media-independent FEC such as the RS code.

RESCU[6] is an error recovery scheme for

real-time interactive video transmission. It focuses on eliminating error propagation rather than error recovery. In RESCU, packets arriving after their display times are not discarded but instead used to reduce error propagation. However, they do not explain on redundancy control in RESCU.

6. Conclusion

We have discussed a redundancy control problem in Application-layer FEC using erasure codes. Then, we proposed a redundancy control algorithm that can adjust the number of parity packets considering network loss and congestion as well as source data priority. According to the experiments, we found out that RC-I achieves the highest performance but produces much redundancy in simple links, while RC-II shows the best adaptability to network congestion in congestion links. We believe that this study of Application- layer FEC can be used effectively for real-time multimedia transmission over wired/wireless networks.

References

[1] T.U. Choi, M.K. Ji, S.H. Park and K.D. Chung, "An adaptive periodic FEC Scheme for Internet video applications," in Proc. IWDC, Springer LNCS 2170, pp.691~702, 2001.

[2] S.W. Yuk, M.G. Kang, B.C. Shin and D.H. Cho, "An adaptive redundancy control method for erasure-code-based real-time data transmission over the Internet," IEEE Trans. Multimedia, Vol. 3, No. 3, Sept. 2001.

[3] J. Nonnenmacher, E.W. Biersack and D. Towsley, "Parity-based loss recovery for reliable multicast transmission," IEEE/ACM Trans. Networking, Vol. 6, No. 4, Aug. 1998.

[4] J-C. Bolot and T. Turletti, "Experience with control mechanisms for packet video in the Internet," Computer Communication Review, Jan. 1998.

[5] D. Rubenstein, J. Kurose and D. Towsley, "Real-time reliable multicast using proactive forward error correction," UMASS Tech. Rep. 98~19, 1998.

[6] Injong Rhee and Srinath R. Joshi, "Error recovery for interactive video transmission over the Internet," IEEE J. Select. Areas Commun., Vol 18, No. 6, June 2000.

[7] R. E. Blahut, "Theory and Practice of Error Control Codes," Addison Wesley, MA, 1984.

[8] S. Lin, D. J. Costello, "Error Control Coding: Fundamentals and Applications," Prentice Hall, 1983.

[9] V. Press, "Introduction to Error-Correcting Codes," 2nd ed., Wiley, 1989.

[10] J. H. van Lint, "Introduction to Coding Theory," 2nd ed., Springer-Verlag, 1992.

[11] A.J. McAuley, "Reliable broadband communications using a burst erasure correcting code," in Proc. ACM SIGCOMM, Philadelphia, PA, Sept. 1990.

[12] L. Rizzo, "Effective erasure codes for reliable computer communication protocols," Computer Communication Review, Vol. 27, No. 2, pp.24~36, Apr. 1997.

[13] E.N. Gilbert, "Capacity of a burst-noise channel," Bell System Technology Journal, Sept. 1960.

[14] See http://www-mash.cs.berkeley.edu/ns/ns.html.

[15] D. Sisalem and H. Schulzrinne, "The loss-delay based adjustment algorithm: A TCP-friendly adaptation scheme," in Proc. of NOSSDAV, Cambridge England, July 1998.

[16] R. Rejaie, M. Handley and D. Estrin, "RAP: An end-to-end rate-based congestion control mechanism for realtime streams in the Internet," in Proc. of IEEE infocom, Mar. 1999.

[17] M. Mathis, J. Semake, J. Mahdavi and T. Ott, "The macroscopic behavior of TCP congestion avoidance algorithm," IEEE Network, 11(6), November 1997.

[18] James R. Yee and Edward J. Weldon, "Evaluation of the performance of Error-correcting codes on a Gilbert Channel," IEEE Trans. Communications, Vol. 43, No. 8, August 1995.

Tae-Uk Choi received the B.S degree in computer science & statistics from Dong-eui University, Pusan, Korea, in 1997, and M.S degree in computer science from Pusan National University, in 1999. He is currently pursuing the Ph.D degree in computer science at the same university. His research interests include Reliable Video Communications, Multimedia QoS, and Mobile & Ubiquitous Multimedia

Ki-Dong Chung received the B.S degree from Seoul National University, Seoul, Koera, in 1973, and M.S and Ph.D degree in computer science from the same university, in 1975 and 1986. In 1990, he was a visiting scholar at Massachusetts Institute of Technology, Cambridge, MA. Since 1987, he has been a professor of computer science at Pusan National University, Pusan, Korea. His research interests include Operating System, Parallel Processing, VOD(Video On Demand), Multimedia Systems, and Wireless & Mobile Multimedia