# QoS-Directed Error Control of Video Multicast in Wireless Networks [1]

Dongyan Xu, Baochun Li, Klara Nahrstedt
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801

## *Abstract*

*In this paper we present a QoS-Directed Error Control scheme (QDEC) for video multicast in wireless networks. In order to provide multimedia services to mobile users, it is necessary to have not only sufficient bandwidth, but also effective control over wireless channel errors, which are characterized as bursty and location-dependent. The QDEC scheme includes (1) a video QoS differentiation specification to direct the error control; and (2) three error control algorithms with corresponding channel error conditions under which the algorithms should be chosen. Through analysis and simulation, we show that QDEC has the following key properties: (1) high error recovery rate for QoS-essential video frames, (2) excellent scalability, (3) high transmission efficiency, and (4) no QoS degradation for receivers who observe no channel errors.*

## I. INTRODUCTION

With rapid advances in wireless packet networking technology, it is becoming possible to provide mobile users not only with data, voice, but also with video communication services. Many emerging mobile applications require the playback of video on the mobile hosts (MHs), and today's wireless networks are getting ready to provide sufficient bandwidth for support of compressed video transmission. In this paper we are especially interested in the issue of wireless video multicast, in which there may be hundreds and thousands of MHs receiving the same video from one sender. Besides sufficient bandwidth, we need effective and efficient error control mechanism to deal with the following problems.

The first problem is that the wireless channel error pattern is different from that of the wired networks. The wireless channel error is characterized as bursty and location-dependent [2]. Bursty channel errors will cause MHs to suffer from back-to-back packet losses, unlike the more random packet losses on a wired link. Also, it is possible that MHs at different locations may observe different channel states at the same time. For wireless video transmission, we have one more concern: not all lost packets should be recovered with the same effort, because compressed video data are usually encoded as frames of different importance. Therefore, error control should be performed in a differentiated manner.

The second problem is that we need to achieve both good scalability (with respect to number of MHs) and transmission efficiency. Automatic Repeat reQuest (ARQ) has been proposed as effective technique for reliable multicast in wired networks, as well as for reliable unicast in wireless networks. However, for wireless video multicast with real-time constraints, ARQ may have the following disadvantages: (1) the sender may experience NACK (Negative ACK) implosion, and the sender may have to retransmit the same packet multiple times; (2) the receivers may contend with each other to send NACKs uplink, causing unnecessary latency; and (3) it is possible that the wireless channel is unidirectional, or the receivers are simple passive (receive-only) devices. No or few feedbacks will be generated in this case. Therefore, it is desirable to have error control techniques whose transmission efficiency is not affected by the increasing number of receivers.

In this paper, we present *QDEC*, a *QoS-Directed scheme of Error Control* for wireless video multicast. QDEC includes the following two aspects: (1) a simple specification of video QoS differentiation to direct the error recovery, (2) three error control algorithms (*FEC_only*, *FEC_Retrans*, and *Retrans_only*) and the channel error conditions under which the algorithms should be chosen accordingly. The key properties of the algorithms are the following: (1) high error recovery rate for QoS-essential video frames, (2) excellent scalability, (3) high transmission efficiency, and (4) no QoS degradation for receivers who observe no channel errors. The rest of the paper is organized as follows. Section II discusses related work. Section III describes the wireless channel error model. Section IV presents the QDEC scheme: Section A suggests the use of video frame differentiation to perform efficient error recovery; Section B presents the three error control algorithms of QDEC and their applicable conditions; and Section C discusses further improvements in QDEC. Section V presents simulation results. Section VI concludes this paper.

[1]This work was supported by the National Science Foundation under contract number 9870736, the Air Force Grant under contract number F30602-97-2-0121, National Science Foundation Career Grant under contract number NSF CCR 96-23867, NSF PACI grant under contract number NSF PACI 1 1 13006, NSF CISE Infrastructure grant under contract number NSF EIA 99-72884, NSF CISE Infrastructure grant under contract number NSF CDA 96-24396, and NASA grant under contract number NASA NAG 2-1250.

## II. Related Work

A good survey of error control schemes for wireless networks can be found in [1]. It is suggested in [1] that any error control scheme should be based on both the application and the environment, which is well reflected in QDEC. In [3], an ARQ-based partially reliable protocol is presented for wireless real-time unicast communication. However, it is not scalable for wireless multicast. [4] presents a reliable multicast data distribution protocol based on packet level FEC. However, it only guarantees non real-time reliable delivery, and the size of the multicast group cannot grow too large. FEC-based error control has also been proposed for wireless multimedia [5], yet the FEC processing constraint on the MHs has received relatively less attention.

Error control in wired networks has also been intensively studied. Most of the approaches use ARQ, FEC, or a combination of both. The FEC codings are usually at packet level and implemented by software [6]. For bursty packet errors, [7] examines the relation between the choice of coding parameters and the performance of FEC, but does not suggest how to determine the optimal coding parameters. [9] and [8] provide adaptive FEC-based error control schemes for real-time multimedia transport, both assuming that the sender and the receiver always have sufficient FEC processing capacity.

## III. Wireless Channel Error Model

In this section we describe the wireless channel error model and its impact on multicast reliability. It is common to characterize the bit errors of wireless channels using the finite state Markovian model. We adopt the simple two-state Markov chain model: the time is divided into slots, and one bit is transmitted in each slot. For a MH, the channel is in either the *good* state or the *bad* state during each slot. $\lambda$ and $\mu$ are the transition probabilities from *good* to *bad* and from *bad* to *good*, respectively. Because the wireless channel errors are *location-dependent*, we assume that the observation of the channel state by each MH is *independent* of each other. It is easy to derive that the average error burst length is $\tau_e = \frac{1}{\mu}$, and the average length of good channel (i.e. the average interval length between two error bursts) is $\tau_g = \frac{1}{\lambda}$. Let $l$ be the length of each packet in *bits*, then $e = \frac{\tau_e}{l}$ is the average number of lost packets in an error burst, and $g = \frac{\tau_g}{l}$ is the average number of good packets received between two error bursts.

Let $N$ be the total number of MHs participating in a multicast session, we can estimate (details can be found in [10]) the average number of MHs who observe good channel at any slot as $N_g = N\frac{\mu}{\lambda+\mu}$. Therefore, the average number of MHs who *begin to* experience a burst of packet losses during the transmission of any packet can be expressed as:

$$\tilde{N} = N_g(1-(1-\lambda)^l) = N\frac{\mu}{\lambda+\mu}(1-(1-\lambda)^l) \quad (1)$$

For example, for a wireless multicast session with a data rate of 2Mbps, if the packet size is 1KB, $\lambda = 10^{-6}$, $\mu = 5 \times 10^{-6}$, then the average length of error burst is 100ms, and the average length of good state is 500ms (note that 1 slot $= 5 \times 10^{-4}$ms). Furthermore, from (1), if $N > 150$, then the average number of MHs who *begin to* experience packet losses during the transmission of *any* packet is greater than one. However, it is obviously not efficient to retransmit every packet, although each packet may well be lost by *at least* one MH.

## IV. QDEC Scheme

### A. Video QoS Differentiation Specification

To preserve good video quality and in the meantime to maintain transmission efficiency, we exploit the fact that compressed video playback can tolerate some degree of packet losses, as long as those frames of higher importance are reliably delivered. More specifically, we specify a subset of video frames as *QoS-essential frames*, and the other frames as *QoS-optional frames*. For example, the set of QoS-essential frames can be all *I* frames and some *P* frames for an MPEG video. Packets carrying data from QoS-essential/QoS-optional frames are tagged as QoS-essential packets and QoS-optional packets, respectively. We differentiate between the two types of packets by transmitting the QoS-essential packets *with* error control, and transmitting the QoS-optional packets *without* error control. By doing so, we aim to achieve a good tradeoff between transmission efficiency and video QoS. In Section C, we will suggest further differentiation among the QoS-essential frames/packets in order to handle the situation when different types of MHs observe *different* average lengths of packet losses.

### B. Error Control Algorithms

The first QDEC algorithm leverages packet-level Forward Error Correction (FEC) [6]. An $(n,k)$ FEC code involves the encoding (on the sender side) of $k$ source data packets into $n > k$ encoded packets, and the decoding (on the receiver side) of *any k* encoded packets to reconstruct the original source data. Results in [6] show that a software implementation of packet level FEC code is feasible and more flexible than bit level codes implemented by dedicated hardware. However, the coding/decoding throughput is inversely proportional to $k(n-k)$. Therefore, the values of $n$ and $k$ cannot be unboundedly large. We denote $k_{max}$ and $h_{max}$ as the maximum values of $k$ and $(n-k)$ that can be efficiently implemented by software (on each MH and on the base station).

Using $e$ and $g$ defined in Section III, we can decide that the condition under which FEC alone is sufficient to recover from the bursty packet losses is the following:

$$e \leq h_{max} \quad (2)$$

The choice of $(n,k)$ under the condition in (2) is:

$$(n,k) = (e + min(g, k_{max}), min(g, k_{max})) \quad (3)$$

It is easy to see that by choosing $(n-k) = e$, any $e$ packets lost in an error burst can be recovered by $(n-k)$ parity packets.

To see why we choose $k = min(g, k_{max})$, notice that we try to achieve high transmission efficiency by choosing a large $k$; however, we also have to make sure that the parity packets are transmitted *before* the beginning of the next possible error burst. The algorithm *FEC_only()* is given in Figure 1.

**while** (**not** end_of_video()) {
  **do** { Packetize QoS-essential frames to collect $k$ packets
    to form a Transmission Group $TG$;
    FEC_encoding($TG$, $n$, $k$);
    Transmit $TG$;
    Transmit the $n - k$ parity packets of $TG$;
  } **until** (at least two consecutive QoS-essential frames
      have been transmitted);
  Packetize and transmit QoS-optional frames between and
  after the QoS-essential frames transmitted in this iteration;
}

Figure 1: Algorithm *FEC_only()* (used when $e \leq h_{max}$)

Our second algorithm works under the following condition:

$$h_{max} < e < k_{max} \tag{4}$$

When the average error burst length become so long that $k_{max}$ and $h_{max}$ are not large enough to completely 'cover' the error bursts, we use both FEC coding and a judicious retransmission of a subset of the data packets to recover the lost packets (note again that the error recovery is for the QoS-essential packets only). First, we decide which data packets to retransmit given a certain $(n, k)$. We will later discuss how to choose the optimal values of $n$ and $k$. For simplicity, let $h = n - k$.

To retransmit as few data packets as possible, we need to choose those that are in the *intersections* of all possible packet loss sequences that may be experienced by the MHs. We illustrate this idea by the following example. Suppose $k = 12$, $h = 3$, and $e = 5$. In Figure 2, each column represents a possible packet loss sequence. The bottom row can be viewed as the packet transmission order. For each loss sequence of length 5, since $h = 3$, we still need to retransmit $e - h = 2$ packets. It is easy to see that retransmitting the shaded packets will minimize the number of retransmitted packets. The two groups of shaded packets (packets 4, 5 and packets 9, 10) are separated by $h = 3$ packets, therefore in any loss sequence of length $e = 5$, there are exactly $e - h = 2$ shaded packets; in any loss sequence of length $e' < e$, there are exactly $max(0, e' - h)$ shaded packets.

There is still one problem: in *FEC_only()*, the parity packets are transmitted right after the TG, and it will not cause a MH to miss any parity packets it needs. However, if we do the same here, it is possible that some loss sequences can not be completely recovered. For example, in Figure 2, if parity packets are transmitted right after packet 12, then the sequences starting with 10, 11, and 12 (the last three columns)

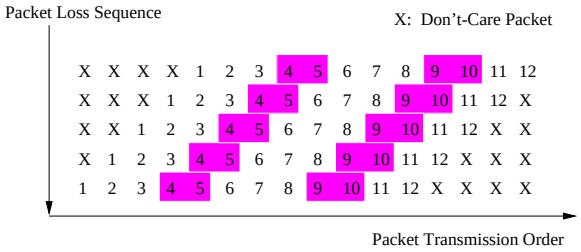


Figure 2: Possible packet loss sequences of length 5 ($k = 12$, $h = 3$, and $e = 5$)

cannot be recovered. The solution is shown in Figure 3: right after the TG, we retransmit the first group of shaded packets, then we transmit the parity packets (let them be $a$, $b$, and $c$), followed by the second group of shaded packets. In fact, this solution is correct in general: the number of the first group of retransmitted packets is $(e - h)$, and it is safe to transmit them right after the TG because each of them is separated by at least $e$ packets from its first transmission. Then the first parity packet is separated by $(e - h)$ retransmitted packets from the last packet in TG, and there are $h$ parity packets. Hence all loss sequences will capture the number of parity packets they need. Now, each of the second group of retransmitted packets is separated by at least $(e - h) + h = e$ packets from its first transmission, therefore it is safe to retransmit them right after the parity packets, and so on. The algorithm *FEC_Retrans()* is given in Figure 4.

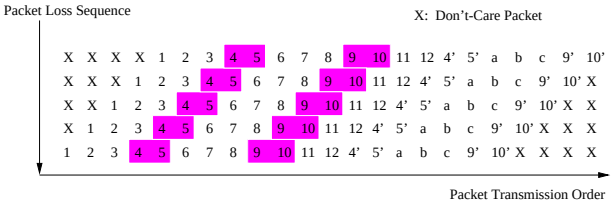


Figure 3: Packet transmission order in *FEC_Retrans()* ($k = 12$, $h = 3$, and $e = 5$)

We now move on to the issue of choosing the proper $h$ and $k$ values, so that the ratio of the number of redundant packets per TG to $k$, the size of TG, is minimum (the redundant packets include both parity packets and retransmitted packets). This ratio can be expressed as the following function:

$$F(k, h) = \frac{h + h_{Retrans}}{k} \tag{5}$$

$h_{Retrans}$ is the number of retransmitted packets. Generalized from the example above, we can derive that

$$h_{Retrans} = (e - h)\lfloor \frac{k}{e} \rfloor + max(0, k - e\lfloor \frac{k}{e} \rfloor - h) \tag{6}$$

Hence the problem is reduced to finding $h$ and $k$ such that

$$F(k, h) = \frac{h + (e - h)\lfloor \frac{k}{e} \rfloor + max(0, k - e\lfloor \frac{k}{e} \rfloor - h)}{k} \tag{7}$$

will achieve the minimum under the following constraints:

$$k \leq k_{max} \quad and \quad h \leq h_{max} \quad and \quad g \geq k + e - h \tag{8}$$

**while** (**not** end_of_video()) {
  **do** { Packetize QoS-essential frames to collect $k$ packets to form a Transmission Group $TG$;
    FEC_encoding($TG$, $n$, $k$);
    Transmit $TG$;
    Retransmit $(e-h)$ packets starting from the $(h+1)$th packet in $TG$;
    Transmit the $n-k$ parity packets of $TG$;
    $i=1$;
    **while** $((j=i*e+h+1)\leq k)$ {
      Retransmit $min(e-h, k-j+1)$ packets starting from the $j$th packet in $TG$;
      $i=i+1$;
    }
  } **until** (at least two consecutive QoS-essential frames have been transmitted);
  Packetize and transmit QoS-optional frames between and after the QoS-essential frames transmitted in this iteration;
}

Figure 4: Algorithm *FEC_Retrans()* (used when $h_{max} < e < k_{max}$)

The last condition in (8) is required, because by using our retransmission scheme, the longest packet interval from the end of a loss sequence to the end of its complete recovery is $k+e-h$ (for example, in Figure 3 the first loss sequence is recovered after packet $a$ is sent $k+e-h=14$ packets later). We have to make sure that there will not be another error burst before the recovery. The following theorem decides the right values of $h$ and $k$. Its proof can be found in [10].

**Theorem 1** *If $h_{max} < e < k_{max}$, then for algorithm FEC_Retrans(), function $F(k,h)$ will reach the minimum under the constraints in (8) when $h = h_{max}$ and*

$$k = \begin{cases} k_0, & \text{if } F(k_0, h_{max}) < F(k_1, h_{max}); \\ k_1, & \text{if } F(k_1, h_{max}) \leq F(k_0, h_{max}); \end{cases} \quad (9)$$

*where $k_0 = min(k_{max}, g-e+h)$, and $k_1$ is the largest integer such that*

$$k_1 < k_0 \ \ and \ \ k_1 = h + ie, \ \ i \ is \ an \ integer \quad (10)$$

For example, suppose $e=9$, $g=32$, $h_{max}=6$, and $k_{max}=30$. Then $h=h_{max}=6$. Figure 5 shows the values of $F(k,6)$. Also $k_0=29$ and $k_1=6+2\times 9=24$. From Theorem 1 we choose $k=24$, which is confirmed by Figure 5.

Our last algorithm works under the the worst (and less likely) condition:

$$e \geq k_{max} \quad (11)$$

In this case, the average length of wireless channel error burst will be too long to be 'covered' by any FEC coding that the sender and receivers can afford. Even if we choose the $(h_{max}+k_{max}, k_{max})$ FEC code, since $e \geq k_{max}$, the Transmission Group (TG) of $k_{max}$ packets may well be all lost. In order to recover from the error burst, we need to retransmit $(k_{max}-h_{max})$ data packets in addition to the $h_{max}$ parity packets. This is equal to retransmit all $k_{max}$ data packets and even worse, there is the additional cost for FEC coding. Therefore, in this situation, the only solution may be just to retransmit the QoS-essential packets. By doing so we can at least eliminate the FEC coding cost. Algorithm *Retrans_only()* is given in Figure 6.

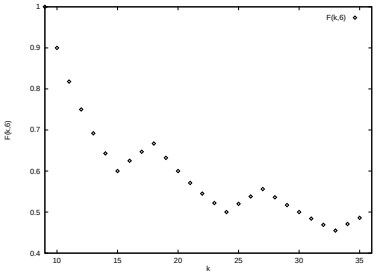


Figure 5: Values of $F(k,6)$, ($k \geq 9$ and $k_{max}=30$)

**while** (**not** end_of_video()) {
  Packetize sufficient frames to collect $e$ packets;
  Transmit the QoS-essential packets in it;
  Transmit the QoS-optional packets in it;
  Retransmit the QoS-essential packets in it;
}

Figure 6: Algorithm *Retrans_only()* (used when $e \geq k_{max}$)

## C. Discussion

We first evaluate the overall transmission efficiency achieved by QDEC. Let $V_{Q-ess}$ and $V_{Q-opt}$ be the volumes (in number of packets) of QoS-essential frames and QoS-optional frames in a video, respectively. We will derive $\eta$, the ratio of $V = V_{Q-ess} + V_{Q-opt}$ to the total number of packets transmitted during the video multicasting.

If $e \leq h_{max}$, algorithm *FEC_only()* can achieve

$$\eta = \frac{V}{V + e\lceil \frac{V_{Q-ess}}{k} \rceil} \quad (12)$$

If $h_{max} < e < k_{max}$, algorithm *FEC_Retrans()* can achieve

$$\eta = \frac{V}{V + kF(k,h)\lceil \frac{V_{Q-ess}}{k} \rceil} \quad (13)$$

If $e \geq k_{max}$, algorithm *Retrans_only()* can achieve

$$\eta = \frac{V}{V + V_{Q-ess}} \quad (14)$$

The QDEC scheme can be further improved in the following aspects: (1) So far we have assumed that the value of $e$ is the same for every MH. In real life, some MHs may observe a smaller $e$ ($e_s$), while others may observe a larger $e$ ($e_l$), depending on their locations, moving speeds, and other factors. It is possible to further differentiate among the QoS-essential frames, such that some QoS-essential frames should be reliably delivered to every MH, while the other QoS-essential frames be reliably delivered only to the MHs

observing $e_s$. Correspondingly, in the error control algorithms, the two subsets of QoS-essential packets are transmitted using *separate* FEC codes - their parameters are decided by $e_l$ and $e_s$, respectively. (2) To further improve coding and transmission efficiency, we can apply the technique of *interleaving*: within each iteration of the algorithms, we first interleave the transmissions of the two subsets of QoS-essential packets; then we interleave the transmission of the QoS-essential packets with that of the QoS-optional packets in between. This will reduce the number of parity and/or retransmitted packets needed by each Transmission Group. The optimal selection of FEC coding parameters after the revisions of (1) and (2) is still an open problem for future work. (3) For a large multicast group, it is possible that at least one MH will begin to experience packet losses during the transmission of *any* packet (such as the example in Section III, in which case ARQ does not save). However, when the number of MHs becomes smaller, it may be more efficient to fall back to ARQ or hybrid ARQ. This leads to a revised scheme in which the sender periodically estimates the number of receivers ([11] provides a scalable approach for such estimation), and it works in QDEC mode when the number of receivers is beyond a threshold, and in ARQ mode when the number is below the threshold.

We did not present the receiver-side operations under the QDEC scheme, due to lack of space. The receivers should be able to (1) discard redundant packets not needed, (2) re-construct and re-order video frames based on packet sequence numbers and frame numbers, and (3) configure their FEC decoders by setting proper parameters according to the FEC encoder on the sender side.

## V. Simulation Results

In this section we present the simulation results to demonstrate the performance of QDEC. We simulate a wireless LAN with a total bandwidth of 1.6Mbps. All data packets have a fixed size of 1KB. The sender transmits two videos simultaneously. The video trace in our simulation is the *Star War* MPEG-1 trace (available via anonymous FTP from Bellcore [12]), and its frame pattern is IBBPBBPBBPBB. For simplicity, we use the first half of the movie trace as Video 1, and its second half as Video 2. The number of receivers is 2000 - half are receiving Video 1 and the other half are receiving Video 2. We set $k_{max} = 32$ and $h_{max} = 6$ for the sender and receivers. The following three experiments are performed to simulate the video multicast under different channel error conditions.

**Experiment 1** We set $e = 4$ and $g = 25$ - the average error burst length is relatively short. The occurrences of error bursts experienced by each MH are *independent of* those experienced by others. The QoS requirement specifies all the I frames and P frames as QoS-essential. Since $e \leq h_{max}$, algorithm *FEC_only()* is used. By (3), we choose a $(4 + 25, 25)$ FEC code. Figure 7 shows the percentages of I, P, and B frames correctly reconstructed by the receivers (Video 1 for receivers 0 - 999 and Video 2 for receivers 1000 - 1999). The high percentage of I and P frames reconstructed indicates the algorithm's effectiveness in recovering the QoS-essential frames. We also simulate the brute force video multicast without QDEC, under the same error condition. The results are shown in Figure 8.

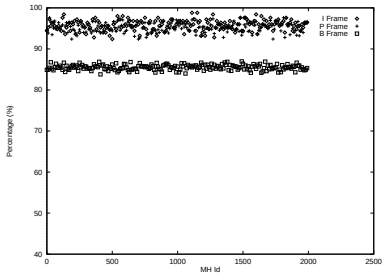


Figure 7: Percentage of Frames Correctly Reconstructed in Experiment 1 - using QDEC

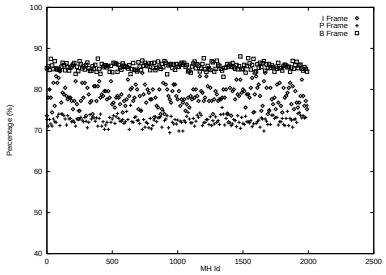


Figure 8: Percentage of Frames Correctly Reconstructed in Experiment 1 - without QDEC

**Experiment 2** We set $e = 12$ and $g = 60$ - the average error burst length is longer than Experiment 1. Since $h_{max} < e < k_{max}$, algorithm *FEC_Retrans()* is used. By Theorem 1, a (6+30, 30) FEC code is chosen. We also adjust the QoS requirement: all the I frames, and the first and second P frames in each GOP are QoS-essential. The results are given in Figure 9. We notice an (expected) decrease in the percentage of P frames reconstructed. The results of multicast without QDEC under the same channel error condition are shown in Figure 10.

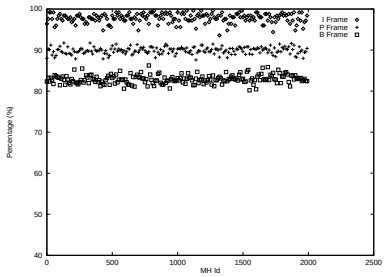


Figure 9: Percentage of Frames Correctly Reconstructed in Experiment 2 - using QDEC

**Experiment 3** We set $e = 40$ and $g = 300$ - the average error burst is very long. Since $e > k_{max}$, algorithm

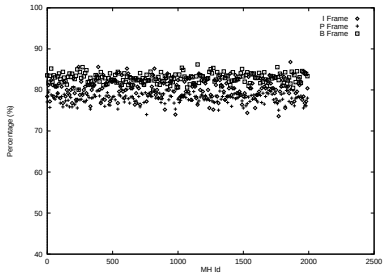


Figure 10: Percentage of Frames Correctly Reconstructed in Experiment 2 - without QDEC

*Retrans_only()* is chosen. Under such an unfavorable channel condition, we have to further lower the QoS requirement: all the I frames, and only the first P frame in each GOP are QoS-essential. Figure 11 shows the results. Again as a comparison, the results under the same error condition but without QDEC are shown in Figure 12.

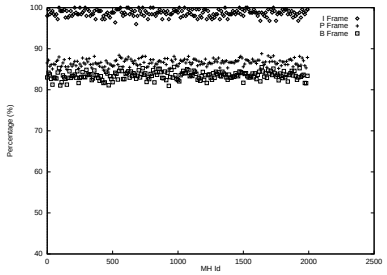


Figure 11: Percentage of Frames Correctly Reconstructed in Experiment 3 - using QDEC

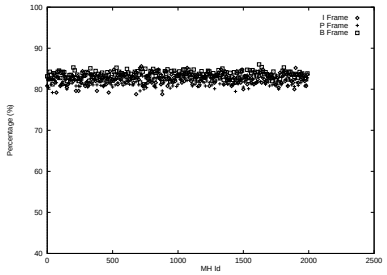


Figure 12: Percentage of Frames Correctly Reconstructed in Experiment 3 - without QDEC

Finally, we calculate the transmission efficiency $\eta$ for Video 1 and Video 2 in the three experiments, by applying (12), (13), and (14), respectively. The $\eta$ values are given in Table 1, which indicate good tradeoff between video QoS and transmission efficiency under different channel error conditions.

Table 1
Values of $\eta$ in the three experiments

| Video | Experiment 1 | Experiment 2 | Experiment 3 |
|---|---|---|---|
| Video 1 | 90.6% | 82.7% | 80.9% |
| Video 2 | 90.8% | 85.7% | 78.3% |

## VI. Conclusion

In this paper, we addressed the bursty and location-dependent characteristics of wireless channel errors, and proposed QDEC, a QoS-Directed Error Control scheme for wireless video multicast. It includes (1) a differentiation specification of video QoS to direct the error recovery, and (2) three error control algorithms (*FEC_only*, *FEC_Retrans*, and *Retrans_only*) and their applicable channel error conditions. Through analysis and simulation, we showed that QDEC has the following key properties: (1) high error recovery rate for QoS-essential video frames, (2) excellent scalability, (3) high transmission efficiency, and (4) no QoS degradation for receivers who observe no channel errors.

## VII. References

[1] H. Liu, H. Ma, M. El Zarki, and S. Gupta "Error Control Scheme for Networks: An Overview," *ACM MONET, vol. 2, October 1997*

[2] S. Lu, V. Bharghavan, and R. Srikant "Fair Scheduling in Wireless Packet Networks," *Proc. ACM Sigcomm '97, September 1997*

[3] C. Wu and G. Ma "Data Transfer Scheme for Wireless Real-Time Communications," *Proc. IEEE Infocom '98, March 1998*

[4] L. Rizzo and L. Vicisano "RMDP: an FEC-based Reliable Multicast Protocol for Wireless Environments," *Mobile Computing and Communication Review, vol. 2, April 1998*

[5] E. Ayanoglu, P. Pancha, A. Reibman, and S. Talwar "Forward Error Control for MPEG-2 Video Transport in a Wireless ATM LAN," *ACM MONET, vol. 1, January 1996*

[6] L. Rizzo "Effective Erasure Codes for Reliable Computer Communication Protocols," *Computer Communication Review, vol. 27, April 1997*

[7] J. Nonnenmacher, E. Biersack, and D. Towsley "Parity-Based Loss Recovery for Reliable Multicast Transmission," *Proc. ACM Sigcomm '97, September 1997*

[8] J. Bolot, S. Fosse-Parisis, and D. Towsley "Adaptive FEC-Based Error Control for Interactive Audio in the Internet," *Proc. IEEE Infocom '99, March 1999*

[9] K. Park and W. Wang "QoS-Sensitive Transport of Real-Time MPEG Video Using Adaptive Forward Error Correction," *Proc. IEEE ICMCS '99, June 1999*

[10] D. Xu, B. Li, and K. Nahrstedt "QoS-Directed Error Control of Video Multicast in Wireless Networks," *Tech. Report, Computer Science Dept., UIUC, August 1999*

[11] J. Nonnenmacher and E. Biersack "Optimal Multicast Feedback," *Proc. IEEE Infocom '98, March 1998*

[12] M. W. Garrett and W. Willinger "Analysis, Modeling and Generation of Self-Similar VBR Video Traffic," *Proc. ACM Sigcomm '94, September 1994*