MINIMIZING DELAY IN LOSSLESS SEQUENTIAL DATA STREAMING

Sanjeev Mehrotra, Jin Li

Microsoft Research,
Redmond, WA, USA
{sanjeevm,jinl}@microsoft.com

Ying-zong Huang

Massachusetts Institute of Technology,
Cambridge, MA, USA
zong@mit.edu

ABSTRACT

There is an ongoing explosion of interactive Internet applications. By nature, these applications require responsive client-server data exchange and lossless, in-order delivery. In previous work, we have shown that a hybrid FEC-ARQ protocol which combines sending original data packets with forward error correction (FEC) packets is effective in reducing the latency caused by retransmissions of lost packets. However, the prior scheme only sends FEC packets when there are no original packets pending transmission. In this paper, we further expand the investigation of the hybrid FEC-ARQ protocol and show that sometimes, the transmission latency can be further reduced by preempting original data packets with FEC packets. We have formulated the decision of whether to send new original data packets, FEC packets, or resend original data packets as a transmission policy. An optimal transmission policy is selected to minimize the delay experienced by the application subject to a constraint on the amount of waste. By using this optimal policy, we significantly improve the delay performance over straightforward FEC schemes while controlling the amount of waste due to FEC.

Keywords— Real-time communications, lossless data streaming, forward error correction

1. INTRODUCTION

With the rapid penetration of broadband networks, online interactive applications are flourishing. Web-based applications, which use the browser as a thin client, are proliferating as the software can be installed and maintained on centralized servers as opposed to distributing the software on potentially millions of client computers. Some examples of web applications are wikis, online auctions, web-mail, and online retail sales. As another example, online games are rapidly expanding as well. Many online games have associated online communities, making online games a form of social activity beyond single player games. Also, the rising popularity of Flash and Java has led to an Internet revolution providing a unified platform to deliver streaming audio, video, and other forms of interactivity to the client.

All these interactive applications demand responsiveness. Whenever the client sends an input (e.g keyboard/mouse commands), the requests must be sent to the server in a distant data center, which processes the incoming commands, and then sends updated data, audio, or video back to the client for rendering. The responsiveness of the application is directly related to the timely interchange of the request and the response between the client and the server.

Table 1. Media streaming vs Interactive app vs File delivery

Media streaming	Interactive App	File delivery
Strict deadline	Delay sensitive	No deadline
Best effort	Reliable delivery	Reliable delivery
No ordering	In-order	In-order
Low delay	Delay-aware	Delay agnostic

Unlike interactive multimedia applications, such as VoIP and video conferencing, most interactive software applications operate as a state machine. Therefore, the data has to be delivered losslessly and in-order so that the client and server state are in sync. TCP (Transmission Control Protocol) provides reliable and ordered delivery of content over the network and thus is commonly used. However, TCP was designed from the start to handle bulk data transfer (file download / static web page download), and therefore optimizes throughput, while making no attempt minimize the delay experienced by individual packets. Its use of packet retransmission upon loss (ARQ) leads to higher delay on individual packets when loss is present. This can lead to poor performance for interactive applications. We summarize the differences of the quality of service requirements between interactive software applications vs. that of file delivery and media streaming in Table 1.

The primary cause of increased delay for interactive applications is packet loss, as an unexpected lost packet has to be retransmitted. An effective technology to reduce this delay is FEC, that is, sending additional encoded packets to protect the data packets. FEC has been promoted widely in media (audio and video) streaming applications, e.g., in [6, 3, 1]. In [4], Rizzo and Vicisano have used FEC to support reliable multicast. In [5], Sundararajan *et. al.* describe how to modify TCP by using random linear codes to protect against packet loss over the network. The idea is to retain all existing TCP mechanisms for congestion control and triggering of retransmission, but apply FEC (more specifically, random linear codes across all data in the window) at the sender and receiver, thereby masking loss at the

978-1-4244-7493-6/10/$26.00

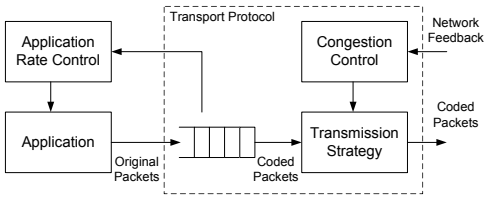

Fig. 1. Block Diagram.

network and improving responsiveness. In [2], we develop a hybrid FEC-ARQ protocol for optimized sequential (in-order) delivery. Whenever a transmission opportunity arises, the protocol will retransmit a lost packet, send a new packet (if present), or send a FEC packet (in that order of importance). Through analysis, it is shown that for low loss rate networks, the hybrid FEC-ARQ protocol is equivalent to simply opportunistically sending FEC packets which are linear combinations of all unacknowledged source packets whenever there is a "free" transmission opportunity, which is one when no new source packet or no negatively acknowledged (timed out) packet is present.

In this paper, we further extend the hybrid FEC-ARQ protocol. We show that only sending FEC packets when there are "free" transmission opportunities is not an optimal solution. In certain situations, e.g., in networks with high packet loss ratio, or when the traffic is bursty, or in congested cases (when the maximum application rate is higher than the network capacity), it sometimes makes sense to preempt sending a new source packet with a FEC packet of previously sent source packets. Although the source packets waiting in the sender queue get delayed, the overall delay experienced by the application can be reduced. We also show that sometimes it makes sense to only create a FEC packet of the first few packets rather than all the unacknowledged packets. We formulate the problem of figuring which packet to send as an optimized transmission policy problem, where for each transmission opportunity, we can choose to send one of three types of packets: 1) a new source packet, 2) a FEC packet, or 3) a resent packet. The optimized transmission policy explicitly minimizes average packet delay, subject to a constraint on the amount of waste, which is defined as the number of FEC packets minus the lost packets divided by the number of original packets.

The rest of the paper is organized as follows. In Sec. 2, we go over the transmission strategy in detail. The cost function is explained in Sec. 2.2 and a computation of the wasted packets is discussed in Sec 2.4. In Sec. 3, we show simulations to demonstrate the effectiveness of the proposed scheme.

2. TRANSMISSION STRATEGY

Fig. 1 shows a block diagram of a typical network setup of interactive applications. The sender application produces original source packets to send to the receiver. These packets typically come in a burst and consist of data which the receiver will process in order. The packets are sent to the transport module. The transport module typically has a buffer to temporarily hold the packets. The packets leave the buffer only when they have been acknowledged by the receiver. If the sending buffer is full, the sending application receives feedback of this event from the transport module and reduces its sending rate. For example, for an application that is sending audio/video, it can re-compress the audio/video at a lower bit rate. For game applications, it can reduce the game status update interval to reduce the sending rate. However, once the packets enter the transport module's buffer, they must be delivered losslessly to the receiver.

The transport module consists of two components. One is the congestion control module which estimates the available bandwidth in the communications channel, determines the current sending rate, and backs off (reduces sending rate) when congestion is detected. It tries to find a fair share of the bandwidth for the sending application while trying to minimize self congestion induced loss and queuing delay. The hybrid FEC-ARQ protocol developed in this paper can work with many existing congestion control modules, e.g., TFRC rate control. The second module is a transmission strategy module. It determines which type of packet to send at each transmission opportunity.

Since delay is the most important factor in determining the perceived user performance of interactive applications, the overarching goal for the transport module is to minimize the *expected* delay incurred by each packet while ensuring reliable in-order delivery. The delay incurred by the packets has several components – e.g., waiting time in the sender's queue, propagation delay, network queuing delay, retransmission delay, and decoding delay if a coding scheme is used. The requirement of in-order delivery can also cause additional delay as a packet may need to wait for prior missing packets to be delivered or decoded.

For the following discussion, we define *original packets* as the data packets which the application wishes to send from the sender to the receiver. For a stream with an in-order reliable delivery requirement, original packet i is defined to be *sequentially decodable* (i.e. usable) if and only if it and all prior packets $j \leq i$ are delivered or decoded. Let *sequential decodability delay* (SDD) refer to the time span between when a packet enters the sender queue (from the application) to the time it becomes sequentially decodable. This delay is important for interactive applications.

Let *coded packets* refer to the packets that actually enter the network. These packets can be original, FEC packets, or resent packets. Let *transmission delay* be the delay sending these coded packets from the sender to the receiver. This delay consists of the network propagation delay and queuing delay. The SDD on the original packets is a function of transmission delay incurred by the coded packets as well as loss rate suffered by the coded packets and the coding strategy being used.

2.1. Choice of Packets and Policies

The transmission strategy can send one of three types of packets: original packet, FEC packet, or resent packet. The FEC packets consist of linear combinations (over a Galois field) of existing unacknowledged (undecodable) packets in the sender queue. Let $\mathbf{x}[l]$ be the lth original source packet which is rep-

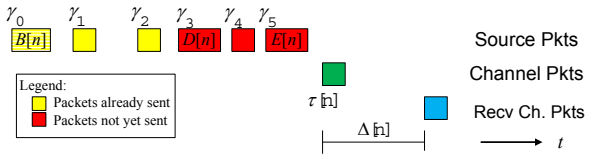

Fig. 2. Timeline - First three packets have been sent, but not necessarily decodable. Last three have not yet been sent. At time $\tau[n]$, the nth coded packet is generated which reaches the receiver after $\Delta[n]$ time.

resented as a vector of bytes, each of which is an element in $GF(2^8)$. Then, if $\mathbf{y}[k]$ is the kth packet sent from the sender to the receiver, it can be written as $\mathbf{y}[k] = \sum_{l=b[k]}^{e[k]} f_{k,l}\mathbf{x}[l] = \mathbf{f}_k^*\mathbf{x}$, where $f_{k,l}$ are coefficients from $GF(2^8)$. If an original packet is sent, then $\mathbf{y}[k] = \mathbf{x}[b[k]]$, for some $b[k]$ and $e[k] = b[k]$. Because of the in-order requirement, it can be shown that for FEC packets, without loss of optimality, $b[k]$ can be assumed to be the index of the first undecoded original packet in the sender queue. The transmission strategy chooses from amongst the following three transmission policies.

- Sending a new source packet without coding.
- Sending a FEC packet of only the first certain number of undecoded packets.
- Resending an already sent packet which has timed out or been negatively acknowledged.

2.2. Cost Function Used to Decide Amongst Policies

At any given transmission opportunity, the cost that we use to decide amongst the various policies is to minimize the expected SDD. For our discussion, we define the following terms, which are shown in the timeline in Fig. 2.

- n is the current transmission opportunity.
- $B[n]$ is the index of the first unacknowledged packet in the sender queue prior to transmission n.
- $E[n]$ is the index of the last packet in the sender's queue.
- $D[n] \leq E[n]$ is the index of the first packet which has not yet been sent.
- $\tau[k]$ is the time when coded packet k leaves the sender.
- $\Delta[k]$ is the transmission delay experienced by coded packet k (propagation delay plus queuing delay).
- γ_l is the time original packet l enters the sender queue.

The expected SDD for original packet l can be written as $D_l = \sum_{\delta\in\mathcal{D}} \delta \text{ Prob}(\text{SDD} = \delta)$, where $\mathcal{D}$ is the set of possible values for SDD given by $\tau[k] + \Delta[k] - \gamma_l$ over k. The probability of achieving this SDD is $p_l[k] - p_l[k-1]$, where $p_l[k]$ is the probability that all original packets up to and including l are decodable (i.e. packet l is sequentially decodable) using transmissions up to and including transmission k. This probability can be computed exactly with reasonable complexity as shown in [2]. This gives

$$D_l = \sum_{k=0}^{\infty}(p_l[k] - p_l[k-1])(\tau[k] + \Delta[k] - \gamma_l). \tag{1}$$

The SDD is affected by the transmission delay through the term $\Delta[k]$, the time spent in the sender queue by $\tau[k] - \gamma_l$, and the network loss and coding strategy by $p_l[k]$.

We assume that the congestion control module is able to achieve a smooth transmission rate and queuing delay, Thus $\tau[k+1] - \tau[k] = T$ (the time between successive transmission opportunities is relatively constant) and $\Delta[k] = \Delta$ (transmission delay is stable and approaches the network propagation plus queuing delay). Then, rearranging terms in (1), we get

$$D_l = (\tau[s_l] + \Delta - \gamma_l) + \sum_{k=s_l}^{\infty}(1 - p_l[k])T, \tag{2}$$

where s_l is the first packet transmission opportunity that comes after packet l enters the queue, that is $s_l = \min\{j : \tau[j] \geq \gamma_l\}$. We can view this expected delay in terms of waiting times. With probability 1, packet l waits until the first transmission opportunity that comes after it enters the queue plus the network transmission delay. With probability $1 - p_l[k]$ it waits an additional time of T for the next transmission opportunity. At a given transmission opportunity n for M original packets, γ_l and $\tau[s_l]$ are the same for all transmission policies. We can remove these common terms to obtain the cost function to be optimized as

$$C = \sum_{l=0}^{M-1}\sum_{k=\max(s_l,n)}^{\infty}(1 - p_l[k]). \tag{3}$$

To simplify further, we only consider source packets starting from $l = B[n]$ (all other packets have already been decoded) and ending at $E[n]$ which is the last packet entering the sender queue. We could also consider other packets past $E[n]$ that will enter the sender's queue, but this will be application-specific. For each packet n, we only consider certain terms in the summation over k. For packets which currently have non-zero probability of decodability ($p_l[n-1] \neq 0$), we only consider the first term in the summation, and for original packets which have $p_l[n-1] = 0$, we look at the first L_l terms which is defined to be the expected time till p_l becomes non-zero. This gives,

$$C \approx \sum_{l=B[n]}^{D[n]-1}(1 - p_l[n]) + \sum_{l=D[n]}^{E[n]}\sum_{k=n}^{n+L_l-1} 1. \tag{4}$$

L_l can also be estimated as the expected number of transmission opportunities needed to successfully deliver all packets prior to original packet l. L_l can be computed using the current expected number of missing packets Q_n and the current loss estimate ϵ as $L_l = (Q_n + l - D[n])/(1-\epsilon)$. The expected number of missing packets can easily be computed from the probabilities p_l. If we remove common terms and simplify, we get the following cost functions for sending a FEC and an original packet respectively

$$C_{FEC} = \sum_{l=B[n]}^{D[n]-1}(1 - p_l[n]) + \frac{(E[n] - D[n] + 1)(Q_n + 1)}{1-\epsilon},$$

$$C_{ORIG} = \sum_{l=B[n]}^{D[n]}(1 - p_l[n]) + \frac{(E[n] - D[n])Q_n}{1-\epsilon}. \tag{5}$$

$p_l[n]$ is the new probability of sequential decodability if that packet is sent and Q_n is the new value for the expected number of missing packets up to the last packet sent – if an FEC packet is sent, the last packet sent stays at $D[n]$ and if an original packet is sent, it increases to $D[n]+1$. Using (5), we compute the cost for each possible FEC packet (each value of $e[k] = B[n], B[n]+1, \ldots, D[n]-1$, with $b[k] = B[n]$) and for an original packet ($b[k] = e[k] = D[n]$) and send the one with minimum cost. The case when $b[k] = e[k] = B[n]$ is evaluating the benefit of retransmitting the first packet in the sent queue, and in cases when packets in the sent queue have timed out, the algorithm will choose such a strategy.

The value for ϵ used by (5) is estimated using a sliding window of certain number of packets into the past. The loss for this window is computed (ϵ_W) and the overall loss rate is updated using $\epsilon \leftarrow \eta\epsilon + (1-\eta)\epsilon_W$ using some weight η.

2.3. Example of Cost Computation

As an example, consider $\epsilon = 0.1$, $B[n] = 1$, $D[n] = 4$, $E[n] = 6$. That is, there are six packets in the burst, out of which four have been sent but not yet acknowledged as being decodable. The values for $p_l[n-1]$ would be the following.

l	1	2	3	4	5	6
p_l	0.9000	0.8100	0.7290	0.6561	0	0

Then, depending on whether we send a FEC packet which encompasses all the original source packets or whether we send a new original packet (packet 5), p_l would become

FEC	0.9656	0.9412	0.9258	0.9185	0	0
ORIG	0.9000	0.8100	0.7290	0.6561	0.5905	0.

For FEC, the expected number of missing packets with index ≤ 4 would become $Q_n = 0.0905$, and for an original packet the expected number of missing packets with index ≤ 5 would be $Q_n = 0.5000$. Using (5), we would get $C_{FEC} = 3.8838$, and $C_{ORIG} = 2.4255$. Thus, in this case between the two, we should send the original packet. As another example, suppose the loss rate is still 0.1, but now $B[n] = 1$, $D[n] = 10$, and $E[n] = 11$. That is, almost the entire burst has been sent, but no packets have yet been acknowledged. Using the same computations as above, the costs are $C_{FEC} = 5.3622$ and $C_{ORIG} = 6.0465$, and thus here we preempt a source packet to send a FEC packet. In general, the advantage of sending a FEC packet increases as loss rate increases and as the ratio of unacknowledged packets (in-flight) to waiting packets (in sender queue) increases.

2.4. Computing FEC waste

When we are allocating the transmission rate amongst source and FEC packets, one method is to allocate some percentage of it to the needed source rate (accounting for loss) and make sure that the the percentage of non-innovative packets (waste) does not take more than the remainder. The waste can be computed using a deterministic term (based upon feedback) and a probabilistic term for the in-flight coded packets (those which have not been acknowledged or timed out). At a given transmission opportunity n, let w be the number of packets that are known to the sender to have been useless by the receiver (from feedback), and let t be the total number of packets received (from feedback). We can compute the expected fraction of packets which are waste (more than the needed amount defined as FEC packets minus lost packets) as

$$u = \frac{w + \sum_{k\in\mathcal{F}} p_{e[k]}[k-1]}{(t+|\mathcal{F}|) - (w + \sum_{k\in\mathcal{F}} p_{e[k]}[k-1])}, \quad (6)$$

where $\mathcal{F}$ is the set of in-flight coded packets, and $|\mathcal{F}|$ is the number of such packets. The probability $p_{e[k]}[k-1]$ is the probability that we were already able to decode up to $e[k]$ given transmissions up to $k-1$, and thus is the probability that the kth coded packet with ending position $e[k]$ was useless.

For any given packet that we are considering on transmitting at n, we can update the set $\mathcal{F}$, and can calculate an updated value of u. We control the amount of waste u to below a certain threshold U_{MAX}. If sending a FEC packet causes u to be above this threshold, we do not send it.

We believe that this definition of waste – as a fraction of wasted packets to needed packets – is more useful than the typical definition of redundancy which is the fraction of FEC packets to source packets. For example, if the loss rate is 5% and if 5% of the total packets are FEC, then most of the FEC packets are actually used to recover lost packets, and thus there is no waste.

3. EXPERIMENTAL RESULTS

In this section, we show the performance of the proposed transmission policy optimization. We simulate the setup as shown in Fig. 1. The notation used in the experiments is as follows. The application produces P packets of size B bits with an inter-burst gap of G seconds. This gives a maximum application source rate of $S = PB/G$ bits/sec. We assume that if the source rate exceeds network bandwidth and the sending buffer is full, the application rate control module will kick in, and excess packets will be dropped. The sending buffer size is Q bits. The congestion control module provides a transmission opportunity to send a single B-bit packet every T seconds giving a network transmission rate of $R = B/T$ bits/sec. We assume that the channel has a delay of D seconds (round trip time) and a loss rate of L. U_{MAX} is the maximum amount of waste that is allowed.

For all the experiments, we show two figures, one is the cumulative density function (CDF) of the sequential decodability delay (SDD), and the other is the CDF of the instantaneous application bit rate defined as the number of packets from the burst that are sent divided by the spacing between the bursts. We compare the following four transmission strategies.

- The best achievable bound. This is the performance if the sender has immediate knowledge of which packets will be

Table 2. SDD Percentiles (90%, 95%, 99%) for $Q = 16$ packets, $U_{MAX} = 10$, $S = 500$Kbps, $R = 400$Kbps. L is the loss rate, D is the round trip delay in sec. Type shows the strategy being used, "No cost" refers to the FEC strategy in [2], "Cost" refers to FEC with use of cost function, and "ARQ" refers to retransmission only. U is the actual fraction of wasted packets.

L	D	Type	90%	95%	99%	U
0.05	0.15	ARQ	0.47	0.51	0.63	0.00
0.05	0.15	No Cost	0.43	0.47	0.51	0.11
0.05	0.15	Cost	0.37	0.41	0.51	0.11
0.15	0.15	ARQ	0.63	0.71	1.01	0.00
0.15	0.15	No Cost	0.53	0.59	0.67	0.11
0.15	0.15	Cost	0.47	0.53	0.63	0.22
0.15	0.40	ARQ	1.64	1.79	2.55	0.00
0.15	0.40	No Cost	0.76	0.80	1.46	0.29
0.15	0.40	Cost	0.56	0.58	0.68	0.39

Table 3. SDD Percentiles and fraction of wasted packets for $Q = 16$ packets, $U_{MAX} = 10$, $S = 300$Kbps, $R = 400$Kbps.

L	D	Type	90%	95%	99%	U
0.05	0.15	ARQ	0.46	0.58	0.68	0.00
0.05	0.15	No Cost	0.36	0.38	0.40	0.28
0.05	0.15	Cost	0.36	0.38	0.41	0.30
0.15	0.15	ARQ	0.70	0.86	1.15	0.00
0.15	0.15	No Cost	0.41	0.54	0.66	0.18
0.15	0.15	Cost	0.46	0.50	0.62	0.24
0.15	0.40	ARQ	1.67	1.77	3.10	0.00
0.15	0.40	No Cost	0.65	0.66	0.83	0.33
0.15	0.40	Cost	0.62	0.65	0.75	0.54

lost and retransmits them immediately at the next transmission opportunity.

- The strategy using only retransmission (ARQ).
- The strategy adopted in [2]. This is referred to as the "no cost" FEC or opportunistic FEC. In this strategy, an FEC packet of all unacknowledged source packets is sent whenever the sender queue is empty. Otherwise an original packet is sent.
- The cost based transmission strategy developed in this paper.

For all simulations, we set $B = 8000$ bits (1KB) and $P = 10$ packets (the burst length). We first show the achievable performance if we are allowed to send as much FEC as the network allows (set $U_{MAX} = 10$) in a congested network, when $S = 500$Kbps, $R = 400$Kbps, $D = 0.15$sec, $L = 0.05$, and $Q = 16$packets. Note that the total sending rate is constrained by the network bandwidth, and the source application will reduce its sending rate through notification of the sending buffer being full. The results are shown in Fig. 3 and summarized in Table 3. The 90th percentile of SDD reduces by over 14% when using the cost function vs. not using the cost function, the 95th percentile by over 12%, and the 99th percentile is about the same. The percentage of wasted packets is 11.2%. We also see that the application bit rate is smoother, with the application capable of delivering a bit rate of at least 300Kbps 90% of the time, rather that 70% of the time if the cost function is not used. This is due to the fact that lowering SDD results in the sender queue being emptier.

Keeping other parameters the same, if we increase Q to 32 packets, we observe that few FEC packets are sent (regardless of whether we use the cost function or not). The reason is that since $S > T$, the buffer is almost always full, and since Q is relatively large, the penalty for sending FEC packets is high since $E[n] - D[n]$ in Eqn. (5) is large.

In the second experiment, we increase the loss rate to $L = 0.15$ (see Fig. 4 and Table 3). From Table 3, we observe that cost based transmission policy reduces 90th percentile SDD by 11%, 95th percentile SDD by 10%, and 99th percentile SDD by 6%. The application sending bit rate is also much smoother with the cost function than without. However, both of the schemes reach a median (50th percentile) bit rate of about 250 kbps.

In the third experiment, we further increase delay to $D = 0.4$seconds. From Fig. 5, we see that by using the cost function, we are able to achieve a result very close to the lower bound (in terms of SDD and application bit rate). From Table 3, we can see that this comes at the expense of increasing the percentage of wasted packets to close to 40%.

In the fourth experiment, we consider the case when the network is not congested ($S < R$). We reduce the the maximum source rate S to below capacity (300 kbps) and keep other parameters the same. We observe that the cost function based transmission policy gains no advantage over the opportunistic FEC (see Fig. 6 and Table 3). Opportunistic FEC is able to achieve a result close to the lower bound especially in the high-loss, high-delay case where we have spare capacity. This confirms the results presented in [2].

Finally, in Fig. 7, we show the effect of modifying the fraction of wasted packets allowed, U_{MAX}, in the $L = 0.15$, $D = 0.4$sec case. We see that the SDD performance keeps getting better as U_{MAX} is increased (at the expense of reducing application rate). We note that although the $U_{MAX} = 10$ case is intended to show the best case, the actual waste for this is only $U = 0.39$.

We have also observed that as P (the burst length) gets larger, the cost function based transmission strategy will have a bigger performance advantage over the opportunistic FEC one. Due to space constraints, the result is not presented here.

4. CONCLUSION

We have presented a cost based transmission strategy to optimally choose amongst transmission policies. Through extensive experimental results, we have shown that the proposed scheme achieves better delay performance than the opportunistic FEC scheme especially for cases when the application traffic is bursty, the maximum source rate exceeds that of the network capacity, the network packet loss rate is high, and/or the network delay is high.