AFEC: An Adaptive Forward Error Correction Protocol for End-to-End Transport of Real-Time Traffic*

Kihong Park† Wei Wang
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907
{park,wangw}@cs.purdue.edu

Abstract

This paper presents an adaptive protocol for packet-level forward error correction in dynamic networks. The objective is to facilitate end-to-end transport of real-time traffic whose timing constraints rule out the use of retransmission-based congestion control and quality of service (QoS) provision schemes. The degree of redundancy injected into the network is adjusted as a function of network state, decreasing when the network is well-behaved and increasing when it is not. The control problem is nontrivial due to the fact that increased redundancy, beyond a certain point, can backfire resulting in self-induced congestion which impedes the timely recovery of information at the receiver.

First, we give a comprehensive analysis of the control problem associated with adaptive forward error correction, concentrating on the dynamics of a particular protocol called Adaptive Forward Error Correction (AFEC). We show that instabilities can arise from two distinct sources—the desired operating point and network delay—and we give solutions to handle them. The first factor is determined by the stringency of the application's QoS requirements, making its achievement potentially perilous.

Second, we present simulation results showing the dynamics of AFEC and comparing its performance against static FEC. We study the efficacy of AFEC under structural perturbations and bursty traffic conditions, and we quantitatively estimate the redundancy-recovery rate function which relates redundancy to the QoS rendered at the receiver. We show that the curve is unimodal, its curvature being determined by whether a connection is packet loss or delay dominated.

*Supported in part by NSF grant ANI-9714707.
†Additionally supported by NSF grant ESS-9806741, and grants from PRF and Sprint.

1. Introduction

Forward error correction (FEC) is a well-studied reliable communication technique which has been successfully used, primarily at the bit-level, in a number of application domains from space communication to reliable data storage on compact disks [2, 13, 14]. The idea of FEC is simple: given a communication channel with known error probabilities, use error correcting codes to inject redundant information into the transmission such that a desired recovery rate—i.e., probability of successful decoding at the receiver of code blocks—is achieved.

In the context of supporting multimedia traffic with real-time constraints over high-speed wide-area networks, packet-level FEC has generated renewed interest due to the ARQ mechanism's—i.e., retransmission of lost messages—inherent inadequacy at handling timing constraints when subject to packet drops and large end-to-end network latencies [3, 4, 5, 9, 15, 20, 22, 28, 29]. Much of the previous research has concentrated on the issue of implementing efficient error correcting codes at the software level from Reed-Solomon codes [19, 20] to other variants of polynomial codes over Galois Fields [1, 18, 15, 28], to ATM and TCP/IP over ATM implementation issues [3, 9]. Due to the low bit error rates associated with modern communication media, the assumption is made that decoding is mainly impeded by packet loss—i.e., *erasures*—caused by network congestion and the resultant buffer overflow and queueing delay. For the real-time applications—including MPEG video and audio—that we aim to support, delayed packet arrivals that violate the receiver's timing constraints are equally as useless as if they had been dropped by the network. That is, the applications are *hard* real-time systems with respect to their quality of service (QoS) contraints either being met or not, but they are potentially *soft* in the sense that a fraction of constraint violations—i.e., *over time*—are tolerated.

The effectiveness of FEC is determined by two fundamental factors, one, in having adequate information about a communication channel such that the amount of redundancy or overcode can be appropriately set, and two, in having a system that is resilient to the effects of *traffic burstiness* or *packet loss/delay dependency*. In the former, if a communication channel or—in general—the network state is stationary, then parameters defining the network state can be estimated once and for all and subsequently used to determine the amount of redundancy to be injected into future transmissions. However, in general networks, the network state is far from being stationary with new connections arriving and departing continuously, and structural perturbations including host/link failures and congestion effects amplifying the variability of system state.

In the latter, burstiness can cause *dependencies* in packet loss and queueing delay processes which, in turn, can lead to *concentrated* periods of packet drops that can significantly impede successful decoding—i.e., error correction or *recovery*. This problem is further aggrevated by recent discoveries stemming from network traffic measurements [16, 11, 25] that show that network traffic—including individual VBR video sources—exhibit self-similarity and long-range dependence (LRD), and that this is an ubiquitous phenomenon. This is of import to FEC since it has been shown that queueing behavior, under LRD input, can significantly differ from that of Markovian input processes such as Poisson [17, 21].

In [4], a *system* approach is taken to the study of FEC in the sense that a group of FEC- and non-FEC sources and their interaction is investigated, and aggregate performance is evaluated as a function of the specific group make-up and their individual traffic characteristics. A zero-sum law is shown to be at play whereby the least effective group configuration is one where all sources use FEC resulting in a small or even negative aggregate gain.

It was a well-known fact that *bit-level* FEC is ineffective for dependent-error channels [10], rendering ARQ-based or hybrid schemes more effective in such situations with respect to achieving higher throughput. Only recently have studies been carried out for *packet-level* FEC which possess further structure due to the fact that dependencies can stem from queueing effects as well as the characteristics of the traffic source itself. In [29] simulation is used to show that burstiness can lead to recovery properties significantly different from those assuming independence. This has been confirmed and further analyzed by Cidon et al. [6, 7], albeit excluding LRD input processes.

This paper studies two central questions: (a) adaptivity to changes to network state such that appropriate redundancy can be dynamically computed and (b) the impact of burstiness—in particular, scale-invariant burstiness—on FEC performance. Note that the second question is made more difficult by the first question since dynamically varying redundancy as a function of network state—inclusive queue length—creates further dependencies due to feedback. The basic premise of (a) is that, to minimize coding overhead, it is reasonable to adjust the degree of redundancy as a function of network state such that the overcode is decreased when the network is well-behaved and it is increased when it is not. The control problem is nontrivial due to the fact that increased redundancy, beyond a certain level, can "backfire" resulting in a form of self-induced congestion which further impedes the timely recovery of information at the receiver. In other words, an *increase* in redundancy causes the receiver's recovery rate to *decrease*. This is hardly ever the case for systems investigated in the context of bit-level FEC and, hence, (a) constitutes a new problem that needs to be addressed.

We propose a specific control algorithm called *Adaptive Forward Error-Correction* (AFEC) protocol and analyze its behavior with respect to optimality and stability. We present a comprehensive controlled simulation study of AFEC under various network conditions including self-similarity. We show that the qualitative dynamics predicted by the analysis are confirmed by the simulation results, and we shed further light on the conditions under which AFEC is effective.

2. Network Model

2.1. Packet-Level FEC

The goal of adaptive forward error correction is to adjust the level of redundancy as a function of network state, decreasing it when the network is well-behaved and increasing it when the network is congested. Thus the overhead incurred by FEC is kept to a minimum.

For simplicity of discussion and to focus on the main ideas, we will assume an ATM framework where packets are of fixed size—i.e., *cells*—and k data packets are encoded as $n = k + h$ cells where $h \geq 0$ is the degree of redundancy. We will assume that the receipt of *any* k cells out of the n total cells suffices to recover the original k data cells. Encoding/decoding functions with this property clearly exist (e.g., Reed-Solomon [19], IDA [27]). Figure 2.1 shows a depiction of packet-level FEC in a network environment.

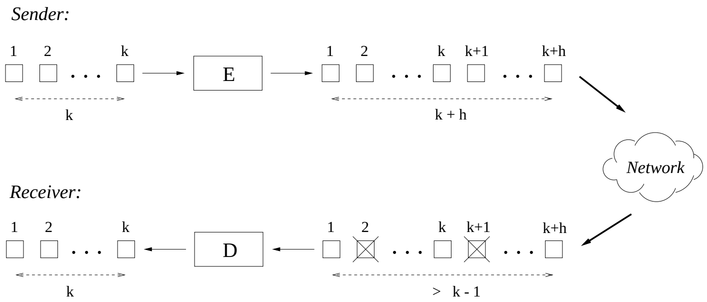

Figure 2.1. A block of k packets are encoded at the sender using FEC as $k + h$ packets. Successful recovery is achieved at the receiver if at most h packets are dropped/delayed.

2.2. Real-Time Constraints

Figure 2.2 depicts the shared output buffer at a bottleneck link where $\lambda^A(t)$ is the FEC-controlled application process, $\lambda^C(t)$ is the cross traffic, and $\mu(t)$ is the service process. The application in question is real-time constrained and the sender transmits

$$n(t_1), n(t_2), \ldots, n(t_i), \ldots \qquad (t_i < t_j \quad \text{if } i < j)$$

blocks of cells at times t_i where $n(t_i) = k(t_i) + h(t_i)$, $i = 1, 2, \ldots$. That is, $k(t_i)$ is the traffic requirement at t_i as dictated by the application and $h(t_i)$ the corresponding redundancy factor. We will model quality of

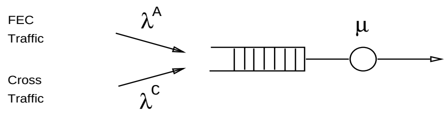

Figure 2.2. Shared queue at bottleneck link with FEC-controlled application traffic λ^A and cross traffic λ^C.

service (QoS) requirements using *hard* real-time constraints whereby we assume the existence of a monotonically increasing sequence $(t'_i)_{i \in \mathbb{Z}_+}$ of time deadlines at the receiver such that all $k(t_i)$ data cells belonging to the i'th block must be recovered before time t'_i. That is, late arrivals belonging to block i carry zero utility.

QoS is measured at the receiving end using a *recovery rate* process $\gamma(t'_i)$. It is defined to be the number of cells belonging to block i received before time t'_i. We will say there is a *hit* at time t'_i if $\gamma(t'_i) \geq k(t_i)$; i.e., decoding was timely and successful. We will extend γ's domain of definition to $\mathbb{R}_+$ by saying that for t such that $t'_i < t < t'_{i+1}$, for some $i \in \mathbb{Z}_+$, $\gamma(t)$ denotes the number of cells received belonging to block $i + 1$ before time t. Thus for all $i \in \mathbb{Z}_+$, $\gamma(t)$ is a nondecreasing function on $(t'_i, t'_{i+1}]$. The aforementioned model is general enough to represent a wide variety of real-time applications including those with periodic constraints as well as interactive applications with aperiodic constraints.

3. AFEC Protocol

3.1. Base Protocol

In this section, we describe the base protocol which is the heart of the AFEC protocol suite. The base protocol is further augmented in Section 3.3 by stability related mechanisms.

Going back to our earlier remark, increasing $h(t)$ blindly will adversely affect $\gamma(t + \tau)$, for some $\tau > 0$. That is, letting $\mathcal{G}$, $\gamma(t + \tau) = \mathcal{G}(h(t))$, denote the functional relationship between $h(t)$ and $\gamma(t+\tau)$, there is a value $h^* > 0$ such that[1] $d\mathcal{G}/dh = 0$ at $h = h^*$, $d\mathcal{G}/dh > 0$ for $h < h^*$, and $d\mathcal{G}/dh < 0$ for $h > h^*$. That is, $\mathcal{G}(h) \geq 0$ is *unimodal* with peak at $h = h^*$. Thus if the goal is to achieve maximum recovery rate, then we arrive at an optimal control problem where the optimal operating point is defined as (h^*, γ^*), $\gamma^* = \mathcal{G}(h^*)$. Figure 3.1 depicts the unimodal redundancy-recovery rate relation $\gamma = \mathcal{G}(h)$. We remark that h^*, in general, can be an interval $[h^*_l, h^*_r]$, $h^*_l < h^*_r$.

If $\gamma^* < k$, then there is a structural problem and no amount of redundancy, small or large, can yield a positive hit rate. This may be due to excessive cross traffic λ^C, long end-to-end delays, and a host of other factors.

Let us consider the case when $\gamma^* \geq k$. Assuming

[1] We use continuous notation for expositional simplicity; their discrete counterparts are straightforward. In practice, h is maintained as a continuous variable but just before passing it to a FEC encoder, it is discretized.

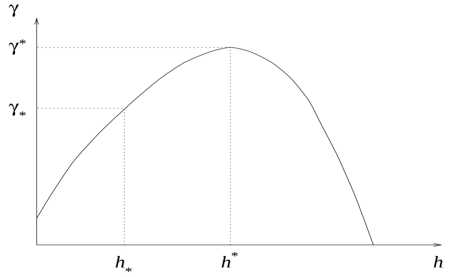

Figure 3.1. Unimodal redundancy-recovery rate function $\gamma = \mathcal{G}(h)$ with optimal recovery rate γ^* and target recovery rate γ_*.

$\gamma(t)$ is fed back to the sender from the receiver, we can formulate the following control law

$$\frac{dh(t)}{dt} = \epsilon(\gamma_* - \gamma(t-\tau)) \tag{3.1}$$

where γ_*, $k \leq \gamma_* \leq \gamma^*$, is the *target* recovery rate, $\epsilon > 0$ is an adjustment parameter, and $\tau \geq 0$ is a delay term introduced by feedback and network latency. The control algorithm as embodied by (3.1) just says that if the measured recovery rate $\gamma(t-\tau)$ is smaller than the target recovery rate γ_*, then the redundancy factor h should be increased, and vice versa.

The larger the gap $\theta = \gamma_* - k > 0$, the more resiliently—in the presence of perturbations and sudden onset of increased burstiness—adaptive redundancy control can be affected. θ can be viewed as a yardstick of *conservativeness*; at the expense of increased overhead, higher resiliency can be achieved with respect to sudden changes in packet loss and delay.

When k is variable—as determined by a stochastic application arrival process $(k(t_i))_{i \in \mathbb{Z}_+}$—we yield a companion control law for (3.1) in the form of

$$\frac{dh(t)}{dt} = \epsilon'(\nu_* - \nu(t-\tau)) \tag{3.2}$$

where $\nu_* \geq 1$, $\epsilon' > 0$ is an adjustment parameter, and $\nu(t-\tau) = \gamma(t-\tau)/k(t_i)$ for i such that $t'_{i-1} < t-\tau \leq t'_i$. That is, i is the block number with respect to which the recovery rate $\gamma(t-\tau)$ was computed at the receiver. In practice, feedback is sent only at times $t = t'_1, t'_2, \ldots$, hence we have $t - \tau = t'_i$.

Setting $\nu_* = 1$ is tantamount to trying to achieve "perfect" redundancy in the sense that $dh/dt = 0$ iff $\gamma(t-\tau) = k(t_i)$; i.e., just the right number of cells have made it through to achieve recovery. Under stochastic network conditions such as when the variance of the cross traffic process $\lambda^C(t)$ is nonnegligible, ν_* needs to be increased to achieve higher hit rates. The gap $\nu_* - 1$, analogously to $\theta = \gamma_* - k$, is a measure of conservativeness.

The implementation of AFEC in the form of (3.2) is straightforward. On the receiver side, the protocol needs to maintain a counter for $\gamma(t)$, incrementing it whenever a new cell belonging to the expected block arrives. At time t'_i, $\gamma(t'_i)$ and the block sequence identifier i are sent back to the sender and γ is reset to 0. On the sender side, given k and h at time $t = t_i$, the $k(t_i) + h(t_i)$ cells are transmitted. When a feedback control packet with $(\gamma(t'_i), i)$ is received, $h(t_{i+1})$ is updated based on $\gamma(t'_i)$ and $k(t_i)$ using the discrete counterpart of (3.2). The value $k(t_i)$ is remembered by the sender in conjunction with index i.

Since (3.1) and (3.2) are related to each other via $\epsilon = \epsilon'/k$, when analyzing the behavior of the system under average application arrival rate conditions, it suffices to study the dynamics of (3.1).

3.2. Analysis of Stability

3.2.1 Operating Point Instability

Note that the redundancy-recovery relation, $\gamma(t) = \mathcal{G}(h(t-\kappa))$, is also tied to a delay term $\kappa \geq 0$. Hence (3.1) may be rewritten as

$$\frac{dh(t)}{dt} = \epsilon(\gamma_* - \mathcal{G}(h(t-\tau'))) \tag{3.3}$$

where $\tau' = \tau + \kappa$ ignoring the processing delay at the receiver. Equation (3.3) is a nonlinear autonomous delay differential equation [8] and the study of its behavior is the subject matter at hand.

Two forms of instability may arise, one from the value of the desired operating point (h_*, γ_*), and another stemming from the delay term τ' in equation (3.3). First, assume $\tau' = 0$. Complications arising out of $\tau' > 0$—feedback always incurs a non-zero network delay—are orthogonal to the first instability consideration and will be dealt with separately.

To analyze $dh(t)/dt = \epsilon(\gamma_* - \mathcal{G}(h(t)))$, we will first linearize it and analyze the stability of the linearized system around a rest point. Assuming sufficient smoothness of $\mathcal{G}$ (mainly $\mathcal{G} \in C^1$), we can use the Stable Manifold Theorem [26] to relate the stability of the linearized system to that of the original nonlinear system in a local neighborhood of a rest point. That is, the (in)stability of the nonlinear system is implied by the (in)stability of the linearized system.

The Taylor expansion of the right-hand-side (RHS) of (3.3) around $h = h_*$ results in

$$\frac{dh}{dt} = \text{const} - \epsilon \mathcal{G}'(h_*)h + \text{higher order terms}$$

where $\mathcal{G}'(h_*) \equiv (d\mathcal{G}/dh)|_{h=h_*}$. The stability of the linearized system is dependent on the sign of $-\epsilon\mathcal{G}'(h_*)$, in particular, $-\epsilon\mathcal{G}'(h_*) < 0$. Since $\epsilon > 0$, we arrive at the stability condition of (3.3) (with $\tau' = 0$) which depends on the gradient $\mathcal{G}'(h_*)$: (3.3) is asymptotically stable at (h_*, γ_*) if $\mathcal{G}'(h_*) > 0$.

By the unimodality of $\mathcal{G}$, this immediately gives us the corollary that (3.3) is asymptotically stable at $(h_*, \gamma_*) \in H_L$ and unstable at $(h_*, \gamma_*) \in H_R$ where

$$\begin{aligned} H_L &= \{(h, \gamma) : h = \mathcal{G}(h),\ h < h^*\}, \\ H_R &= \{(h, \gamma) : h = \mathcal{G}(h),\ h \geq h^*\}. \end{aligned}$$

In particular, the "optimal" operating point (h^*, γ^*), from the perspective of maximum recovery rate, is unstable.

Figure 3.2 (left) shows the phase portrait in the local neighborhood of (h^*, γ^*); we observe that for $h > h^*$ there is divergence. Figure 3.2 (right) shows the corresponding phase portrait for $(h_*, \gamma_*) \in H_L$. Clearly, the flow to (h_*, γ_*) is convergent. In the case of the rest of the unstable set, $(h_*, \gamma_*) \in H_R \setminus \{(h^*, \gamma^*)\}$, the flow is again divergent for $h > h_*$, analogous to Figure 3.2 (left); it is omitted here.

3.2.2 Delay Instability

We now return to the issue of incorporating the effect of network delay as captured by $\tau' > 0$ in equation (3.3) with respect to stability. We will show that the presence of network delay, in general, can lead to oscillatory behavior even for $(h_*, \gamma_*) \in H_L$ for which we have shown asymptotic stability when $\tau' = 0$ (Section 3.2.1). If τ' is sufficiently large, then the oscillation can become unbounded leading to a different causal aspect of instability.

We use a modified form of the Stable Manifold Theorem, adopted to the case of nonlinear autonomous delay differential equations [8], to relate the stability of the nonlinear system to the stability property of the linearized system around a rest point. We assume sufficient smoothness and boundedness on $\mathcal{G}$. The stability of the linearized system corresponding to (3.3) in the neighborhood of $(h_*, \gamma_*) \in H_L$ is completely determined by the stability of the reduced linear system

$$\frac{dh(t)}{dt} = -\epsilon\,\mathcal{G}'(h_*)h(t - \tau'). \tag{3.4}$$

A linear delay differential equation of the form $dx(t)/dt = -ax(t - \tau)$, $a > 0$, has a sinusoidal solution which is determined by the complex roots of the characteristic equation $\xi + ae^{\xi\tau} = 0$. The system is asymptotically stable if for all roots of the characteristic equation, the real parts lie in the negative half plane. The latter, in turn, is related to the relative magnitude of a and τ which is shown next.

By a change of variables, we further reduce (3.4) to a normal form where $\tau' = 1$. This is useful because the stability condition of $dx(t)/dt = -ax(t-1)$ can be shown to be $0 < a < \pi/2$. Let $s = t/\tau'$. Then,

$$\frac{dh(\tau' s)}{ds} = -\tau'\epsilon\,\mathcal{G}'(h_*)h(\tau'(s-1)),$$

from which we get

$$\frac{d\tilde{h}(s)}{ds} = -\tau'\epsilon\,\mathcal{G}'(h_*)\tilde{h}(s-1) \tag{3.5}$$

where we have redefined $\tilde{h}(s) \equiv h(\tau' s)$. Thus we arrive at the asymptotic stability condition for the original nonlinear system (3.3) via its relation to (3.5),

$$0 \,<\, \tau'\epsilon\,\mathcal{G}'(h_*) \,<\, \frac{\pi}{2}. \tag{3.6}$$

Hence, the larger the network delay τ', the smaller the adjustment factor ϵ needs to be to preserve stability. The gradient $\mathcal{G}'(h_*)$ stands at a similar trade-off relationship with ϵ with respect to stability.

3.3. Augmented AFEC Protocol: Exponential Backoff if $(h, \gamma) \in H_R$

Section 3.2.1 has shown that instability ensues when "too much" redundancy is applied at the sender which is tantamount to shooting oneself in the foot. In fact, the reasons underlying the instability of (h^*, γ^*) are conceptually similar to the ones underlying a *congestion susceptible* system—a two-dimensional system—which has been studied in the context of congestion control in [23]. Whereas in the latter the natural goal is to achieve maximum utilization, in the FEC context, (h^*, γ^*) carries with it a redundancy overhead and the true optimal operating point may be at some value $h_* < h^*$ which meets the desired hit rate requirement at minimum redundancy overhead.

To achieve a form of stability at (h^*, γ^*) which may be needed when the difference $\gamma^* - k$ is not large or the variance of the system high, one can augment the control law given in (3.1) via a *directional check* given by the sign of $d\gamma/dh$. If $d\gamma/dh > 0$, then the system is in the stable region $(h < h^*)$ and (3.1) is applied as usual. If, on the other hand, $d\gamma/dh \leq 0$, then the system finds itself in the unstable region $(h \geq h^*)$ and a *backoff mechanism*—$dh/dt < 0$—is instituted until $d\gamma/dh > 0$ at which time we find ourselves again in the stable regime.

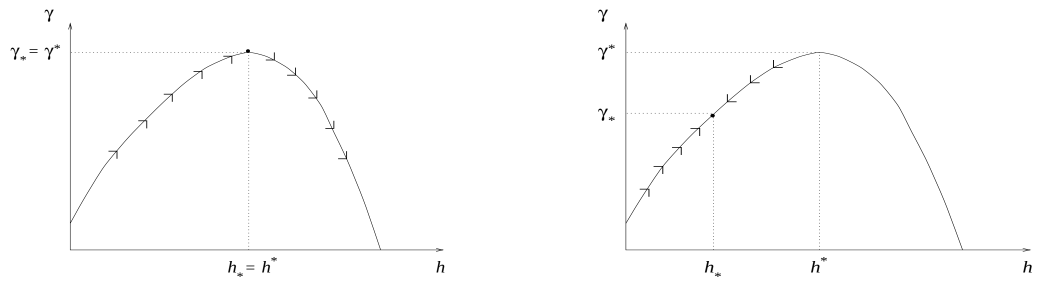

Figure 3.2. Left: Phase portrait around the unstable fixed point $\gamma = \gamma^*$. Right: Phase portrait around the stable equilibrium $\gamma = \gamma_* < \gamma^*$.

The augmented AFEC algorithm is given by

$$\frac{dh(t)}{dt} = \begin{cases} \epsilon(\gamma_* - \gamma(t-\tau)), & \text{if } d\gamma(t-\tau)/dh > 0, \\ -ah, & \text{otherwise.} \end{cases} \tag{3.7}$$

Here $a > 0$ is a positive constant. Hence (3.7) follows (3.1), as it should, when $h < h^*$ (i.e., $(h, \gamma) \in H_L$), and it performs a backoff when $h \geq h^*$ ($(h, \gamma) \in H_R$). The backoff mechanism, $dh/dt = -ah$, is *exponential* leading to a decay of h of the form $O(e^{-at})$. Implementing directional detection—i.e., estimating derivatives, in general—can be tricky; we use information maintained at the sender in the form of a local history $(i, h(t_i))$ in conjunction with information contained in the feedback $(i, \gamma(t_i'))$ for the estimation of $sgn\,(d\gamma(t-\tau)/dh)$. There is again an observation delay incurred whose general effect is studied in Section 3.2.2.

4. Performance Measurements

4.1. Simulation Set-Up

The simulation set-up is based on the network configuration shown in Figure 4.1. A is the application node which generates the FEC-controlled source traffic $\lambda^A(t)$, C is the cross or background traffic node with arrival process $\lambda^C(t)$, and G is a gateway implementing an output-buffered switch with FIFO service, as shown in the logical diagram, Figure 2.2. D is the destination node at which the quality of service with respect to the hit rate ν is measured. The links are unidirectional (digraph) with each edge $X \leftrightarrow Y$ representing two separate links (X, Y) and (Y, X). Quantities of interest include the traffic flow on the bottleneck link (G, D), the queueing behavior at its output buffer, and the arrival times of the $\lambda^A(t)$ process packets at D which determines the QoS received by the application.

The FEC-controlled application—a pair of processes, i.e., the sender at A and the receiver at D—constitutes a full-duplex end-to-end connection from A to D and vice versa. The application is modeled as a periodic hard real-time system whereby $k(t_1), k(t_2), \ldots, k(t_i), \ldots$ data packets ($t_{i+1} = t_i + T$ for some $T > 0$) are generated at A, and the $k(t_i)$ data packets must be recovered at D before time t_i' where $t_{i+1}' = t_i' + T$. An example would be a video-

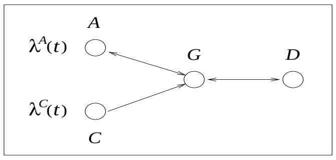

Figure 4.1. Bottleneck network set-up with FEC-controlled data source A, cross traffic source C, gateway G, and destination D.

conferencing application—compressed (i.e., VBR) or uncompressed (CBR)—whereby to meet the f frames per second requirement, $k(t_i)$ data packets must be recovered successfully during the interval $(t_{i-1}', t_i']$. Moreover, $t_i' \geq t_i + \ell$ where $\ell > 0$ is the minimum network latency from A to D.

With respect to scheduling, sender A has two degrees of freedom. One, when communicating the information contained in the i'th block of $k(t_i)$ packets, A may choose to use an error correcting code yielding an amplified block consisting of $n(t_i) = k(t_i) + h(t_i)$ packets where $h(t_i) \geq 0$ represents the redundancy level or overcode. The only requirement is that the decoding function must be able to reconstruct the original $k(t_i)$ data packets from *any* $k(t_i)$ of the $n(t_i)$ code packets transmitted (cf. Section 2). If $X_i = (x_{i1}, x_{i2}, \ldots, x_{in_i})$, $n_i = n(t_i)$, denotes the sequence of packets generated at A belonging to block i, and $Y_i = (y_{i1}, y_{i2}, \ldots, y_{im_i})$,

$y_{ij} \in X_i$, denotes the packets received at D during $(t'_{i-1}, t'_i]$, we will say a *hit* has occured at time t'_i if $m_i \geq k(t_i)$.

Two, A may choose to schedule the $n(t_i)$ packets *in time*, i.e., over the interval $(t_{i-1}, t_i]$. Our results are based on the uniform time schedule which, to some extent, achieves decorrelation to counteract the tendency of packet drops to occur in bursts.

4.2. Redundancy-Recovery Rate Relation

4.2.1 Unimodal Redundancy-Recovery

Figure 4.2 (top) shows the measured redundancy-recovery rate relation $\gamma = \mathcal{G}(h)$ for the previous network configuration, expressed as the normalized quantity $\nu = \mathcal{G}(h)$ where $\nu = \gamma/k$ is the *hit rate*. The cross

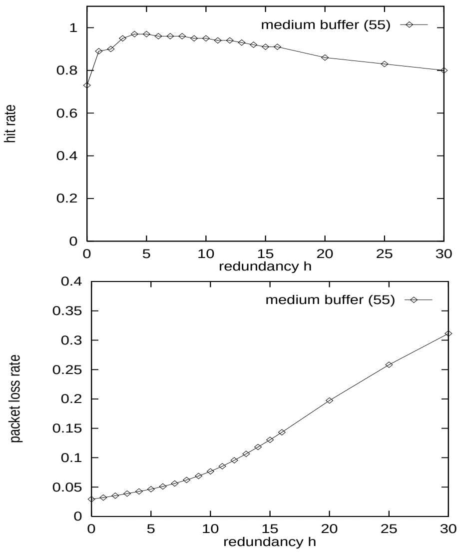

Figure 4.2. Top: Measured redundancy-hit rate function $\nu = \mathcal{G}(h)$ for medium bottleneck buffer ($B = 55$) condition. Bottom: Corresponding packet loss rates.

traffic $\lambda^C(t)$ was a Poisson process with rate $\lambda^C = 10$ cells per unit time, service rate $\mu = 12$ cells/unit time, code block size $k = 2$, application receive period $T = 4$, and link latency 1. The processing time at the nodes was assumed to be fixed (0 in this case). Each measurement was obtained from a simulation run of 3000 time units. Keeping the output buffer B at gateway G fixed, traces were collected when the redundancy factor h was held *constant* at the start of the run. That is, a *static* (or *nonadapative*) FEC protocol was used for the measurement process. h was varied in the range $h = 0, 1, 2, \ldots, 15, 20, 25, 30$, and the collection of runs corresponding to a bottleneck capacity of $B = 55$ packets are shown.

Figure 4.2 (top) shows a unimodal redundancy-recovery rate relation where the hit rate increases as the redundancy level h increases up to a certain point after which further increases in h produce a *decrease* in the hit rate. The curvature of the unimodal redundancy-recovery rate function depends on further properties of the system configuration; it is discussed in the next section. Figure 4.2 (bottom) shows the corresponding packet loss rate measurements as h is varied. The packet loss rate—as a function of redundancy h—is *monotonically* increasing even over the interval $[0, 5]$ over which $\mathcal{G}(h)$ is *increasing*. Thus bandwidth is being traded for improved performance as measured by the hit rate even though performance, as measured by packet loss rate, is monotonically decreasing.

4.2.2 Packet Loss vs. Delay Domination

Figure 4.3 shows the measured redundancy-recovery rate relation $\gamma = \mathcal{G}(h)$ for the previous network configuration for three buffer capacities $B = 15, 55, 65$. For $B = 15$ ("small buffer size"), we observe the hit rate ν increasing as h is increased, saturating thereafter. That is, contrary to the unimodal shape assumed of

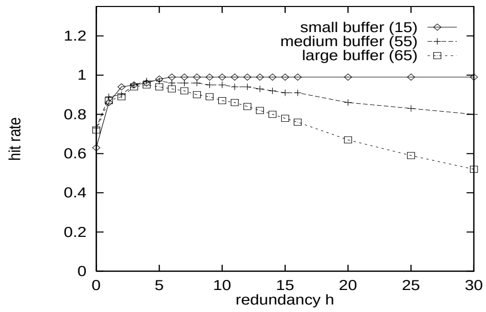

Figure 4.3. Curvature of measured redundancy-hit rate relation $\nu = \mathcal{G}(h)$ under three bottleneck buffer capacity (in cells) conditions.

$\mathcal{G}$ (cf. Figure 3.1) or the measurements shown in Figure 4.2 (left), neither the hit rate, nor the recovery rate γ, decreases for large h.

This phenomenon stems from the fact that small buffer capacities imply a form of "forgetfulness"—i.e., the system has limited memory—which induces an extreme case of many-to-one mapping from the space of traffic characteristics to the output buffer state. In par-

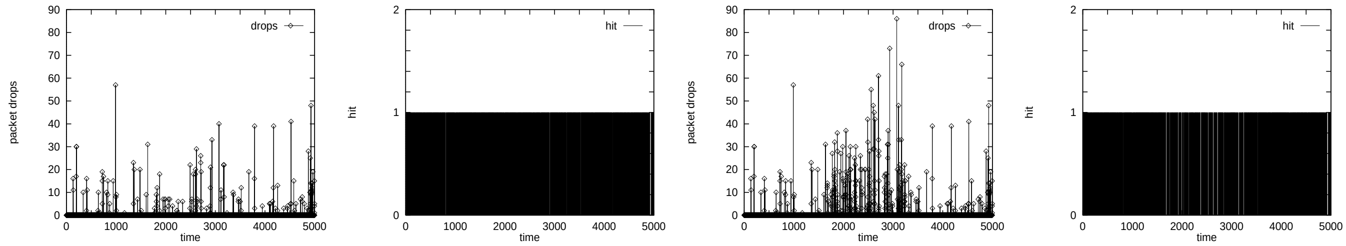

Figure 4.4. Static FEC run with constant redundancy $h = 3$. Left two columns: Packet loss trace (left) and hit occurrence trace at receiver (right) for 5000 time steps. Right two columns: Same as before but with the background traffic (λ^C) increased at time $t = 1700$ and decreased to its original level at $t = 3400$. The *white* stripes in the hit traces indicate instances at the receiver when real-time constraints were violated.

ticular, increasing the traffic rate via h beyond a certain point saturates the queue, and since the sender-side of the application transmits new blocks every T time units, the net recovery rate at the receiver-side saturates as well, yielding the flat hit rate curve observed in Figure 4.3. For $B = 55$ ("medium buffer size"), we see a decline in the hit rate as h is increased beyond a certain point, $h = 5$, continuing to decrease thereafter. This is even more pronounced for $B = 65$ ("large buffer size") where for $h = 30$ a hit rate is achieved that is even smaller than the hit rate achieved with no redundancy. For $B = 55$, 65, the mean sojourn time in the queue has become significant vis-à-vis the application's receive period T so that further increases in h boost the probability that a packet belonging to block i is overly delayed in the queue by packets belonging to block $i-1$. This, in turn, adversely affects the average timeliness of packets—even if the packets in question have not been dropped—causing a decline in the recovery rate.

System configurations where the sum of link latencies and maximum sojourn times at routers along a path is "large" vis-à-vis a real-time application's QoS requirement as represented by its receive time T, we call *delay dominated*. Conversely, systems where the opposite is true we call *packet loss dominated*. From a control perspective, the optimal control problem associated with packet loss dominated systems is significantly easier since feedback control does not suffer under instability problems as shown in Section 3.2. Nonetheless, dynamic control is needed to achieve the desired QoS level without unnecessarily wasting resources.

4.3. AFEC vs. Static FEC

Figure 4.4 shows a typical scenario where the relative advantage of AFEC vs. static or nonadaptive FEC can be discerned. Assume we are given an application that can tolerate a hit rate of $\nu = 93\%$. The network was configured with bottleneck buffer capacity $B = 55$ cells, Poisson cross traffic with rate 12, service rate 13, receive period $T = 4$, and application block size $k = 2$. Using a priori information about the network state, assume the application was privy to the information that a redundancy fixed at $h = 3$ would suffice to achieve the desired hit rate of 93%.

The left two columns of Figure 4.4 show the packet drop trace at the bottleneck buffer (left) and hit trace at the receiver (right) for a 5000 time unit trace. Note that the presence of a unit impulse at $t = t'_i$ in the hit trace implies that at least $k(t_i)$ cells were received at the destination D before time t'_i. That is, decoding was timely and successful. Although there is a strong correlation between packet drops and occurrence of *misses*—i.e., recovery failure—at the receiver, an isolated burst of packet drops need not necessarily affect the recovery of the corresponding data block at the receiver since we employ a uniform time-redundancy schedule to spread out the redundancy in time.

The right two columns of Figure 4.4 show the same set-up except that the cross traffic $\lambda^C(t)$ was perturbed during the middle third of the run. In particular, at time $t = 1700$, λ^C was increased from 12—its previous level—to 15 thus increasing both its mean and variance. $\lambda^C(t)$ was returned to its previous characteristics at time $t = 3400$. The packet drop profile shows a marked increase in the loss rate during the $[1700, 3400]$ period, and a commensurate negative impact on the recovery rate during the same period. The static FEC continues to maintain its redundancy rate at $h = 3$, and the resulting hit rate—88%—significantly violates the application's QoS requirement. Perturbations such as these are common and to be expected in dynamic, shared network environments, and it points toward a

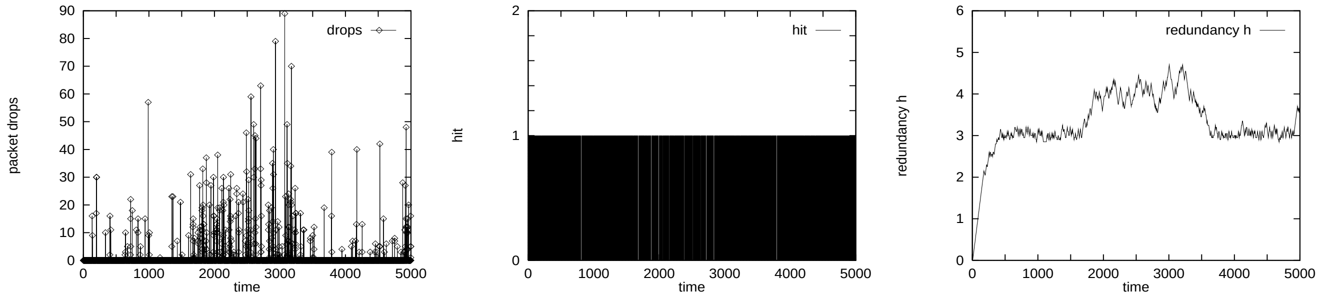

Figure 4.5. AFEC run. Packet loss trace (left), hit occurrence trace (middle), and redundancy rate h trace (right) at receiver when the background traffic is increased at time $t = 1700$ and decreased to its original level at $t = 3400$.

weakness of applying FEC statically.

Figure 4.5 shows the corresponding packet drop and hit traces for the same network configuration for AFEC when the traffic level was increased at $t = 1700$ and returned to its original level at $t = 3400$. The hit trace (Figure 4.5 (middle)) clearly shows a marked improvement over the corresponding trace of the static FEC run in Figure 4.4 (right). The packet drop traces look similar due to the dominance of the cross traffic process $\lambda^C(t)$ in determining variability; the application's variability, in the current set-up, is restricted to changes in h. The same cross traffic process—i.e., sample path—was used in both runs for comparability. Upon closer examination, we detect an elevation in the packet drop sequence during the interval $[1700, 3400]$ for the AFEC run vis-à-vis the static FEC run due to a higher value of h.

The reason for the improved performance can be gleamed from Figure 4.5 (right) which shows redundancy rate h as a function of time. During the "busy" interval $[1700, 3400]$, h is increased accordingly to compensate for the increased packet loss rate. Note that the increase in h, as noted above, further *increases* the overall packet loss rate during $[1700, 3400]$. However, the recovery rate at the receiver is in the range where it can still benefit from increased redundancy, thus resulting in an improved performance with respect to recovery rate vis-à-vis static FEC. This is also reflected in the hit rate which was 94% in the case when there was no sudden increase in the cross traffic, and in the case when increased cross traffic perturbation is applied, AFEC is still able to achieve a hit rate of 93%. This is a marked improvement compared to static FEC where the hit rate had dropped from 93% to 88%. Thus, AFEC is able to achieve a measure of constancy in the presence of structural perturbations to the network state.

5. Conclusion

We have presented a study of the packet-level adaptive forward error correction problem for the QoS-sensitive transport of real-time data end-to-end. Since AFEC is intended to be used in the current Internet environment, it is worthwhile explicating how the protocol—as a member of a family of operational protocols including TCP—fits into the overall picture from a system perspective.

On the one hand, since AFEC trades bandwidth for QoS, it is certainly a more aggressive consumer of resources than, say, TCP[2]. Moreoever, while TCP engages in exponential backoff at the onset of suspected packet drops or significant increase in queueing delay detected via timeouts, AFEC's backoff point—both "natural" and exponential—is not directly tied to the presence of absence of packet drops. In fact, if the desired QoS level (i.e., hit rate) falls into the stable regime, then exponential backoff is rarely invoked with the symmetric control law (3.3) affecting both upward and downward adjustments in h. Only when the target operating point falls into the unstable regime does exponential backoff kick in since the "optimal" operating point (with respect to recovery rate)—i.e., the peak of the unimodal redundancy-recovery relation—is asymptotically unstable.

Modulo these differences, however, AFEC is not fundamentally different from TCP in its behavioral characteristics with respect to how it goes about sharing network resources with other protocols. First, AFEC never consumes more resources than it has to to achieve a user-specified target hit rate which may—in some instances—involve not having to incur any packet drops at all. Second, if AFEC detects that increasing redundancy is counter-productive (i.e., $d\gamma/dh < 0$),

[2] Our discussion will assume TCP Tahoe as the reference point as defined by its principal features given in [12].

then it engages in exponential backoff just like TCP to prevent instability. This is different from a greedy or noncooperative protocol that would exploit the fact that other protocols would be backing off when network contention arises, and by not backing off, be able to soak up the resources abandoned by the others. The "gentlemanly" or cooperative behavior mode intrinsic to AFEC is also the modus operandi of TCP, and from this perspective, there are no instrinsic reasons to suspect that AFEC would not become a "well-behaved" member of the existing protocol family.

Acknowledgments We would like to thank the referees for helpful comments. A number of performance results including those pertaining to self-similar burstiness and implementation of a real-time MPEG transport system have been omitted due to space constraints. We refer the reader to [24, 30] for related works.

References

[1] A. Albanese, J. Blomer, J. Edmonds, M. Luby, and M. Sudan. Priority encoding transmission. *IEEE Trans. Information Theory*, 42(6):1737–1744, 1996.

[2] E. Berlekamp, R. Peile, and S. Pope. The application of error control in communications. *IEEE Communications Magazine*, 25(4):44–57, 1987.

[3] A. Bestavros and G. Kim. TCP Boston: a fragmentation-tolerant TCP protocol for ATM networks. In *Proc. IEEE INFOCOM '97*, 1997.

[4] E. Biersack. Performance evaluation of forward error correction in ATM networks. In *Proc. ACM SIGCOMM '92*, pages 248–257, 1992.

[5] J. C. Bolot and A. Vega-Garcia. The case for FEC-based error control for packet audio in the Internet. To appear in *ACM Multimedia Systems*, 1996.

[6] I. Cidon, A. Khamisy, and M. Sidi. Analysis of packet loss processes in high-speed networks. *IEEE Trans. Information Theory*, 39(1):98–108, 1993.

[7] I. Cidon, A. Khamisy, and M. Sidi. Delay, jitter and threshold crossing in ATM systems with dispersed messages. *Performance Evaluation*, 29(2):85–104, 1997.

[8] O. Diekmann, S. van Gils, S. V. Lunel, and H. Walther. *Delay Equations: Functional-, Complex-, and Nonlinear Analysis*. Springer Verlag, 1995.

[9] S. Dravida and R. Damodaram. Error detection and correction options for data services in B-ISDN. *IEEE J. Select. Areas Commun.*, 9(9):1484–1495, 1991.

[10] F. C. Fujiwara, M. Kasahara, K. Yamashita, and T. Namekawa. Evaluations of error-control techniques in both independent and dependent-error channels. *IEEE trans. on commun.*, COM-26(6):785–793, 1978.

[11] M. Garret and W. Willinger. Analysis, modeling and generation of self-similar VBR video traffic. In *Proc. ACM SIGCOMM '94*, pages 269–280, 1994.

[12] V. Jacobson. Congestion avoidance and control. In *Proc. ACM SIGCOMM '88*, pages 314–329, 1988.

[13] S. Kaneda and E. Fujiwara. Single bit error correcting double bit error detecting code for memory systems. *IEEE Trans. on Computers*, C-31(6):596–602, 1982.

[14] K. Kawashima and H. Saito. Teletraffic issues in ATM networks. *Computer Networks and ISDN Systems*, 20:369–375, 1990.

[15] B. Lamparter, O. Böhrer, W. Effelsberg, and V. Turau. Adaptable forward error correction for multimedia data streams, 1993. Preprint.

[16] W. Leland, M. Taqqu, W. Willinger, and D. Wilson. On the self-similar nature of ethernet traffic. In *Proc. ACM SIGCOMM '93*, pages 183–193, 1993.

[17] N. Likhanov and B. Tsybakov. Analysis of an ATM buffer with self-similar ("fractal") input traffic. In *Proc. IEEE INFOCOM '95*, pages 985–992, 1995.

[18] M. Luby, M. Mitzenmacher, M. Shokrollahi, D. Spielman, and V. Stemann. Practical loss-resilient codes. To appear in *ACM STOC '97*, 1997.

[19] F. MacWilliams and N. Sloane. *The Theory of Error Correcting Codes*. North-Holland, New York, 1977.

[20] A. McAuley. Reliable broadband communication using a burst erasure correcting code. In *Proc. ACM SIGCOMM '90*, pages 197–306, 1990.

[21] I. Norros. A storage model with self-similar input. *Queueing Systems*, 16:387–396, 1994.

[22] H. Ohta and T. Kitami. A cell loss recovery method using FEC in ATM networks. *IEEE J. Select. Areas Commun.*, 9(9):1471–1483, 1991.

[23] K. Park. Warp control: a dynamically stable congestion protocol and its analysis. *Journal of High Speed Networks*, 2(4):373–404, 1993.

[24] K. Park and W. Wang. AFEC: an adaptive forward error correction protocol for end-to-end transport of real-time traffic. Technical Report CSD-TR-98-025, Department of Computer Sciences, Purdue University, 1998.

[25] V. Paxson and S. Floyd. Wide-area traffic: the failure of Poisson modeling. In *Proc. ACM SIGCOMM '94*, pages 257–268, 1994.

[26] L. Perko. *Differential Equations and Dynamical Systems*. Springer-Verlag, 1991.

[27] M. Rabin. Efficient dispersal of information for security, load balancing, and fault tolerance. *Journal of the ACM*, 36(2):335–348, 1989.

[28] L. Rizzo. Effective erasure codes for reliable computer communication protocols. *Computer Communication Review*, 27(2):24–36, 1997.

[29] N. Shacham and P. McKenny. Packet recovery in high-speed networks using coding. In *Proc. IEEE INFOCOM '90*, pages 124–131, 1990.

[30] T. Tuan and K. Park. Congestion control for self-similar network traffic. Technical Report CSD-TR-98-014, Department of Computer Sciences, Purdue University, 1998.