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

Kihong Park
park@cs.purdue.edu

Wei Wang
wangw@cs.purdue.edu

CSD-TR 97-038
Dept. of Computer Sciences
Purdue University

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—i.e., without special network support—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.

In the first part of the paper, 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, which plays the role of optimal operating point, is directly determined by the stringency of the application's QoS requirements, making its achievement potentially perilous.

The second part of the paper presents experimentation and simulation results—the former in the context of an AFEC implementation for transporting real-time MPEG video—that confirm the qualitative dynamics predicted by the analysis. We study the efficacy of AFEC under structural perturbations and bursty traffic conditions including self-similarity, and show under what conditions AFEC is effective. We quantitatively estimate the redundancy-recovery rate function which relates redundancy to the QoS rendered at the receiver. We show that the curve's shape is, in general, unimodal and its curvature is determined by whether a connection is packet loss or delay dominated.

*Supported in part by NSF grant NCR-9714707.

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 [3, 32, 33]. 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 [4, 6, 8, 19, 34, 42, 45, 56, 58]. Much of the previous research has concentrated on the issue of implementing efficient error correcting codes at the software level from Reed-Solomon codes [40, 42] to other variants of polynomial codes over Galois Fields [2, 39, 34, 56], to ATM and TCP/IP over ATM implementation issues [4, 19]. 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—i.e., in the sense of having tractable analytical models—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, given a fixed level of redundancy or overcode, the higher the erasure probability (even if independent), the higher the probability that information recovery will fail due to too many missing packets within a code block. This corresponds to an increase in channel error rate in the bit-level FEC model. Burstiness, however, is more generally characterized by the presence of *dependencies* in packet loss and queueing delay processes which 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 [37, 23, 53, 27, 14] that show that network traffic—including individual VBR video sources—exhibit self-similarity and long-range dependence (LRD), and that this is an ubiq-

uitous 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 [1, 29, 38, 44, 49].

Describing "bursts" using the packet train approach [30], the effect of burstiness on FEC performance has been studied in a seminal paper [6] in the context of ATM networks. In [6], a *system* approach is taken 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 [22], rendering ARQ-based or hybrid schemes more effective in such situations with respect to achieving higher throughput. Only recently, however, 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 [58] simulation is used to show that burstiness can lead to recovery properties significantly different from those that stem from assuming independence. This has been confirmed and further analyzed by Cidon et al. [9, 11, 12], 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.

The idea of adaptively employing FEC is not new. In [34], an "adaptive" scheme is proposed which assuming *known*, *independent* packet loss probability computes the overcode size needed to achieve a desired recovery rate. It constitutes calculating the probability that no more than h packets out of a total of $n > h$ get lost—a simple binomial sum due to independence—and is applicable to bit-level FEC scenario under independent-error channel assumption. However, it does not address dependency issues which are at the heart of the packet-level FEC problem. In [5], a fault-tolerant information recovery scheme supporting real-time constraints is proposed using a variant of Rabin's Information Dispersal Algorithm (IDA) [55]. Assuming *known* delay distributions to a set of information dispersal sites, schedules are devised as a function of the latency distributions such that recovery of a document through the timely arrival of a sufficient number of code blocks is achieved with high probability. The impact of redundancy on network

network state and its implications to performance are not considered.

This paper's contribution is threefold. First, we formulate a model of the adaptive forward error correction problem and present a comprehensive analysis of the associated optimal control problem. We propose a specific control algorithm called *Adaptive Forward Error-Correction* (AFEC) protocol and analyze its behavior with respect to optimality and stability. Our optimality criterion is based on *individual* optimality assuming the network characteristics, albeit stochastic, are fixed. The goal of the AFEC-controlled application is to achieve a desired recovery rate—the QoS requirement specified by the user—when subject to time-varying and nonstationary network conditions and the influence of its own control actions on network state. We show that instabilities can arise from two distinct sources—the target operating point and network delay—and we provide solutions to handle them.

Second, 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 further light is shed by investigating the conditions under which AFEC is effective. We quantitatively observe the shape of the redundancy-recovery rate function which relates redundancy to the quality of service attained at the receiver. We show that the curve's shape is, in general, *unimodal*—i.e., bell-shaped—its curvature is determined by whether a system is packet loss or delay dominated (defined later). We compare the performance of AFEC against a *static* (i.e., nonadaptive) FEC protocol in which the redundancy factor is held fixed, and we show that AFEC exhibits superior performance when the network is subject to structural changes that persist for nonnegligible durations. Under short-range dependent traffic conditions, however, AFEC is able to closely match the performance of an *optimum* static FEC but not exceed it. This phenomenon is related to the fact that a feedback control is only as responsive as the time lag associated with the feedback loop allows it (i.e., round trip time): burst activities generated by short-range traffic sources, by definition, are isolated in time with respect to their correlation structure, and either anticipating or exploiting the occurrence of such bursts is an intrinsically difficult problem for any feedback control algorithm.

Third, we describe an implementation of AFEC in the context of supporting QoS-sensitive transport of real-time MPEG video called *MAFEC* end-to-end (i.e., without specialized network support), and we provide performance measurements of MAFEC in a LAN environment where traffic streams are multiplexed at a configurable router. Both the source traffic (i.e., VBR video) and background traffic are allowed to be bursty and the conclusions advanced by analysis and simulation are confirmed and further extended. MAFEC was further customized to exploit the peculiarities of its payload—MPEG video—by structuring its control in a receiver-oriented manner. In particular, in addition to the adaptive control provided via AFEC by the underlying system, the user is given the option to specify its QoS requirements both at start-up and dynamically through interaction with the receiver-side control module while the application is in progress. This allows for a more intimate accommodation of the user's QoS needs. A minor aspect of this customization is an option by the user to assign different redundancy levels to the three frame types—I, P, B—constituting an MPEG video stream thus providing differentiated protection to each frame type. This is one of the core ideas underlying Albanese et al.'s PET system [2, 35] where static FEC coding is applied to MPEG video with the highest priority (i.e., redundancy) given to I frames

followed by P frames and B frames. This is an obvious thing to do due to the uni- and bi-directional dependencies of P and B frames, respectively. Our work is completely different from Albanese et al.'s work in its two principal aspects, adaptively varying redundancy as a function of network state for facilitating QoS-sensitive transport of real-time data including MPEG video, and studying the efficacy of FEC and AFEC under bursty traffic conditions. Lastly, we remark that MAFEC was implemented entirely in software without specialized hardware or kernel support.

The remainder of the paper is organized as follows. In Section 2 we give the network model including the basic set-up of the adaptive forward error correction problem. This is followed by a description of the AFEC protocol and analysis of its optimality and stability properties (Section 3). Section 4 presents basic performance results obtained from controlled simulations that confirm the results of the analysis. Section 5 describes an implementation of AFEC for QoS-sensitive transport of real-time MPEG video called MAFEC and presents performance measurements obtained from testing the system in a local-area network environment. We conclude with a discussion of high-level issues and future work.

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. What makes the problem nontrivial is the fact that blindly increasing the level of redundancy will cause a congested system to become even worse, thus ultimately bringing forth a net decrease in the recovery rate. Furthermore, this positive feedback loop has the potential—depending on the control algorithm—to further increase the level of redundancy which only worsens the recovery rate. This is fundamentally different from bit-level FEC where the channel error rate is decoupled from redundancy although the error process itself is allowed to be correlated.

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 [40], IDA [55]). In Section 5 we will describe an implementation of packet-level FEC based Rabin's Information Dispersal Algorithm [55]. Figure 1 shows a depiction of packet-level FEC in a network environment.

2.2 Real-Time Constraints

Figure 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

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

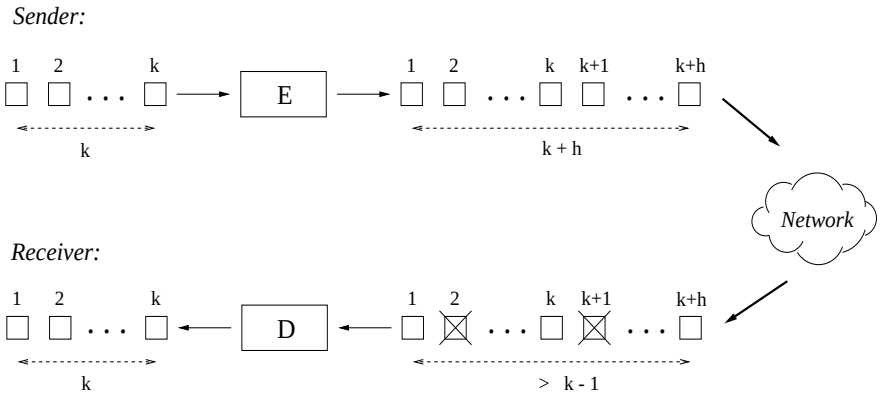

Figure 1: A block of k packets are encoded at the sender using FEC as $k+h$ packets. If the number of dropped (or significantly delayed) packets does not exceed h, the original k data packets can be recovered at the receiver.

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 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.

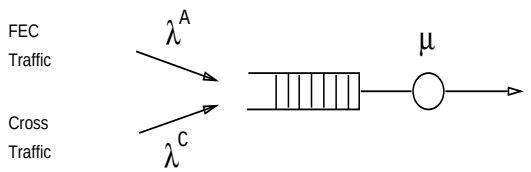

Figure 2: Shared queue at bottleneck link with FEC-controlled application traffic λ^A and cross traffic λ^C.

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 later sections by stability related mechanisms, MPEG transport related customizations, and QoS enhancing features.

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]

$$\frac{d\mathcal{G}}{dh} = 0 \quad \text{at } h = h^*, \qquad \frac{d\mathcal{G}}{dh} > 0 \quad \text{for } h < h^*, \qquad \text{and} \quad \frac{d\mathcal{G}}{dh} < 0 \quad \text{for } h > h^*.$$

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 depicts the unimodal redundancy-recovery rate relation $\gamma = \mathcal{G}(h)$. We remark that h^*, in general, can be an interval $[h_l^*, h_r^*]$, $h_l^* < h_r^*$.

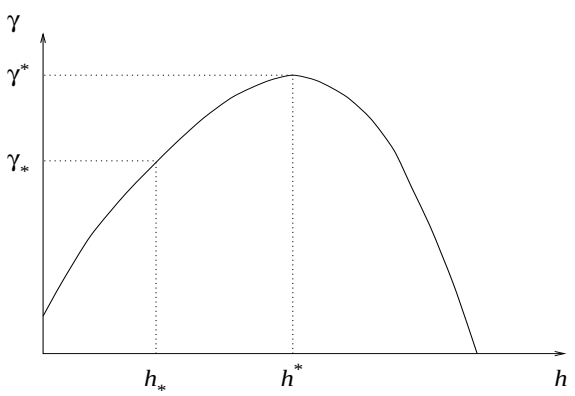

Figure 3: Unimodal redundancy-recovery rate function $\gamma = \mathcal{G}(h)$ with optimal recovery rate γ^* and target recovery rate γ_*.

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

$$\frac{dh(t)}{dt} = \epsilon(\gamma_* - \gamma(t-\tau)) \tag{1}$$

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

[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.

(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 (1) in the form of

$$\frac{dh(t)}{dt} = \epsilon'(\nu_* - \nu(t-\tau)) \tag{2}$$

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 (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 (2). The value $k(t_i)$ is remembered by the sender in conjunction with index i.

Since (1) and (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 (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 (1) may be rewritten as

$$\frac{dh(t)}{dt} = \epsilon(\gamma_* - \mathcal{G}(h(t-\tau'))) \tag{3}$$

where $\tau' = \tau + \kappa$ ignoring the processing delay at the receiver. Equation (3) is a nonlinear autonomous delay differential equation[2] [18] and the study of its behavior is the subject matter at hand.

[2]Delay differential equations are also known as retarded functional differential equations.

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). 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 [54] 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) around $h = h_*$ results in

$$\frac{dh}{dt} \;=\; \text{const} - \epsilon\mathcal{G}'(h_*)h \;+\; \text{higher order terms} \tag{4}$$

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) (with $\tau' = 0$) which depends on the gradient $\mathcal{G}'(h_*)$: (3) is asymptotically stable at (h_*, γ_*) if

$$\mathcal{G}'(h_*) > 0. \tag{5}$$

By the unimodality of $\mathcal{G}$, this immediately gives us the corollary that (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 4 (left) shows the phase portrait in the local neighborhood of (h^*, γ^*); we observe that for $h > h^*$ there is divergence. Figure 4 (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 4 (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) 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 [18], 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

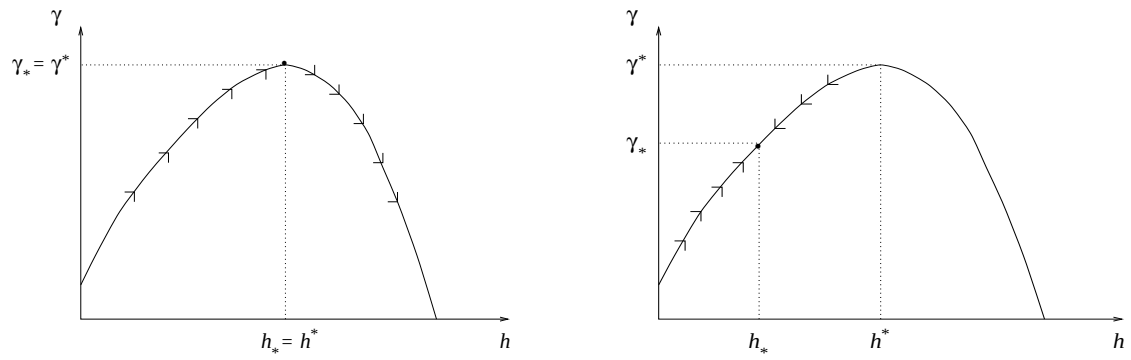

Figure 4: Left: Phase portrait around the unstable fixed point $\gamma = \gamma^*$. Right: Phase portrait around the stable equilibrium $\gamma = \gamma_* < \gamma^*$.

boundedness on $\mathcal{G}$. The stability of the linearized system corresponding to (3) in the neighborhood of $(h_*, \gamma_*) \in H_L$ is completely determined by the stability of the reduced linear system

$$\frac{dh(t)}{dt} = -\epsilon\, \mathcal{G}'(h_*) h(t - \tau'). \tag{6}$$

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 $\xi + ae^{\xi\tau} = 0$ [26]. 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 (6) 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 $0 < a < \pi/2$ [26]. Let $s = t/\tau'$. Then,

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

from which we get

$$\frac{d\tilde{h}(s)}{ds} = -\tau' \epsilon\, \mathcal{G}'(h_*) \tilde{h}(s-1) \tag{7}$$

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) via its relation to (7),

$$0 \;<\; \tau' \epsilon\, \mathcal{G}'(h_*) \;<\; \frac{\pi}{2}. \tag{8}$$

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 [52]. 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 (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 (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.

The augmented AFEC algorithm is given by

$$\frac{dh(t)}{dt} = \begin{cases} \epsilon(\gamma_* - \gamma(t-\tau)), & \text{if } d\gamma(t-\tau)/dh > 0, \\ -ah, & \text{otherwise.} \end{cases} \tag{9}$$

Here $a > 0$ is a positive constant. Hence (9) follows (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.

We remark that a control law such as

$$\frac{dh}{dt} = \epsilon(h_* - h) \tag{10}$$

would render *all* $(h, \mathcal{G}(h))$ asymptotically stable hence obviating the need for the augmented AFEC protocol. However, whereas (1), (2) are feasibly implementable control algorithms since both k and γ are known and measurable quantities, respectively, (10) depends on a more elusive quantity h_* whose estimation is contingent upon quantitatively knowing the redundancy-recovery rate function $\mathcal{G}$, a questionable assumption in the context of dynamic networks and on-line control.

4 Basic Performance Measurements

4.1 Simulation Set-Up

The following sections illustrate the basic performance characteristics of AFEC and its efficacy at facilitating end-to-end transport of real-time data confirming the analysis. To achieve a maximally controlled test environment, we use simulation to measure AFEC's performance. Section 5 shows

an implementation of AFEC for end-to-end transport of real-time MPEG data. Performance measurements based on a working system—entirely in software without special hardware support—are shown there.

The simulation set-up is based on the network configuration shown in Figure 5. 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. 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.

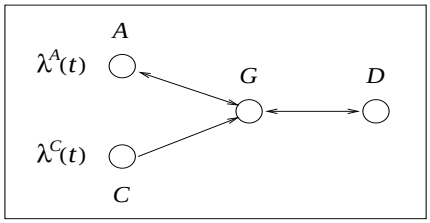

Figure 5: Bottleneck network set-up with FEC-controlled data source A, cross traffic source C, gateway G with outputbuffered switch, and destination node D; λ^A, λ^C represent the application traffic process and cross traffic process, respectively.

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-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 $(t_{i-1}, t_i]$. A function $\pi : [t, t+T] \to \{0, 1\}$ such that $\sum_{s \in [t, t+T]} \pi(s) = h$ will be called a *time-redundancy schedule* over $[t, t+T]$ with weight h. Our results are based on uniform π which, to some extent,

achieve decorrelation to counteract the tendency of packet drops to occur in bursts. A third scheduling variable is to use *dispersity routing* [41]—a form of *space-redundancy scheduling*—to achieve decorrelation over space; the latter, however, requires assistance from the internetworking layer to access routing decisions. In this paper, we concentrate on pure end-to-end mechanisms.

4.2 Redundancy-Recovery Rate Relation

4.2.1 Unimodal Redundancy-Recovery and Packet Loss

Figure 6 (left) 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 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, not AFEC, 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 6 (left) shows a unimodal redundancy-recovery rate relation where the hit rate (i.e., normalized recovery 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.

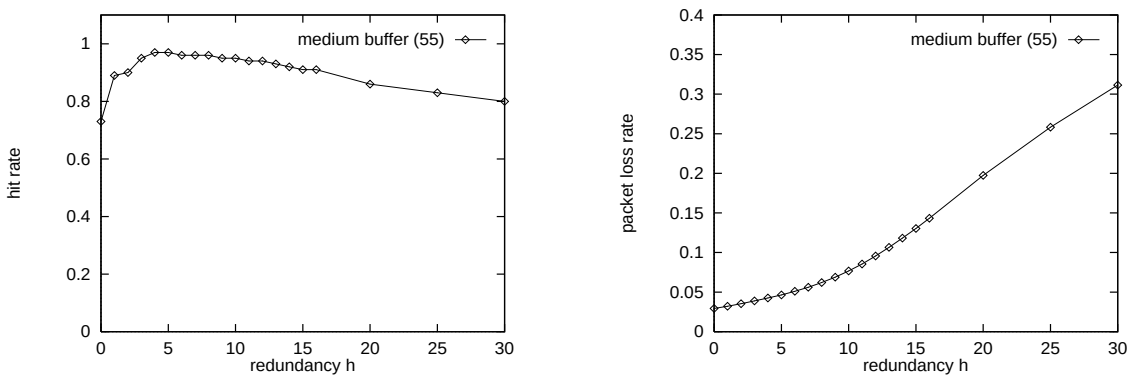

Figure 6: Left: Measured redundancy-hit rate function $\nu = \mathcal{G}(h)$ for medium bottleneck buffer ($B = 55$) condition. Right: Corresponding packet loss rates.

Figure 6 (right) shows the corresponding packet loss rate measurements as h is varied. The packet loss rate—as a function of redundancy h—is *monotonically* increasing, as expected, even over the interval $[0, 5]$ over which $\mathcal{G}(h)$ is *increasing*. Thus bandwidth is being traded against 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 7 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 $\mathcal{G}$—cf. Figure 3—or the measurements shown in Figure 6 (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 particular, 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 7. The implications of small buffer capacities and time horizons on network performance under long-range dependent traffic conditions have been investigated in [25, 57].

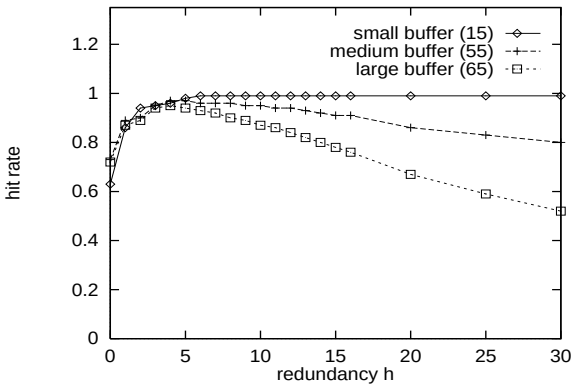

Figure 7: Measured redundancy-hit rate relation $\nu = \mathcal{G}(h)$ (note, $\nu = \gamma/k$) measured under three bottleneck buffer capacity (in cells) conditions: 15 cells (small), 55 cells (medium), and 65 cells (large).

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"[3]), 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. The curvature of $\mathcal{G}$ further increases with B thus justifying the unimodal shape assumption qualitatively shown in Figure 3.

[3]Note that there is only a size 10 gap between the "medium" and "large" buffer sizes whereas there is a size 40 difference between the "medium" and "small" sizes. The sensitivity of the curvature of $\mathcal{G}$ on buffer capacity B increases with B.

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 will call *delay dominated*. Conversely, systems where the opposite is true we will 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. That is, the unstable region $H_R = \emptyset$. Nonetheless, dynamic control is needed to achieve the desired QoS level without unnecessarily wasting resources. When $H_R \neq \emptyset$, the costs incurred can be higher if the application's desired QoS level is close to (or exceeds) the maximum recovery rate γ^*.

Finally, we remark that there is always a practical bound on h (even for packet loss dominated systems) since h must satisfy $(k+h)/L \leq T$ where L is the link speed at the sender-side.

4.3 AFEC vs. Static FEC

Figure 8 shows a typical scenario where the relative advantage of adaptive FEC 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 including empirical measurements, assume the application was privy to the information that a redundancy rate fixed at $h = 3$ would suffice to achieve the desired hit rate of 93%. For present purposes, the $h = 3$, $\nu = 93\%$ correspondence for this particular network configuration was obtained by extensive measurement, as was done in Figure 7, to estimate the shape of the redundancy-recovery rate function $\mathcal{G}$.

The top row of Figure 8 shows 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 black 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 π to spread out the redundancy in time. We observe a few concentrated periods of packet drops here and there—one especially prominent near the end of the run—which lead to corresponding recovery failures at the receiver.

The bottom row of Figure 8 shows 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 weakness of applying FEC statically.

Figure 9 (top row) shows the corresponding packet drop and hit traces for the same network

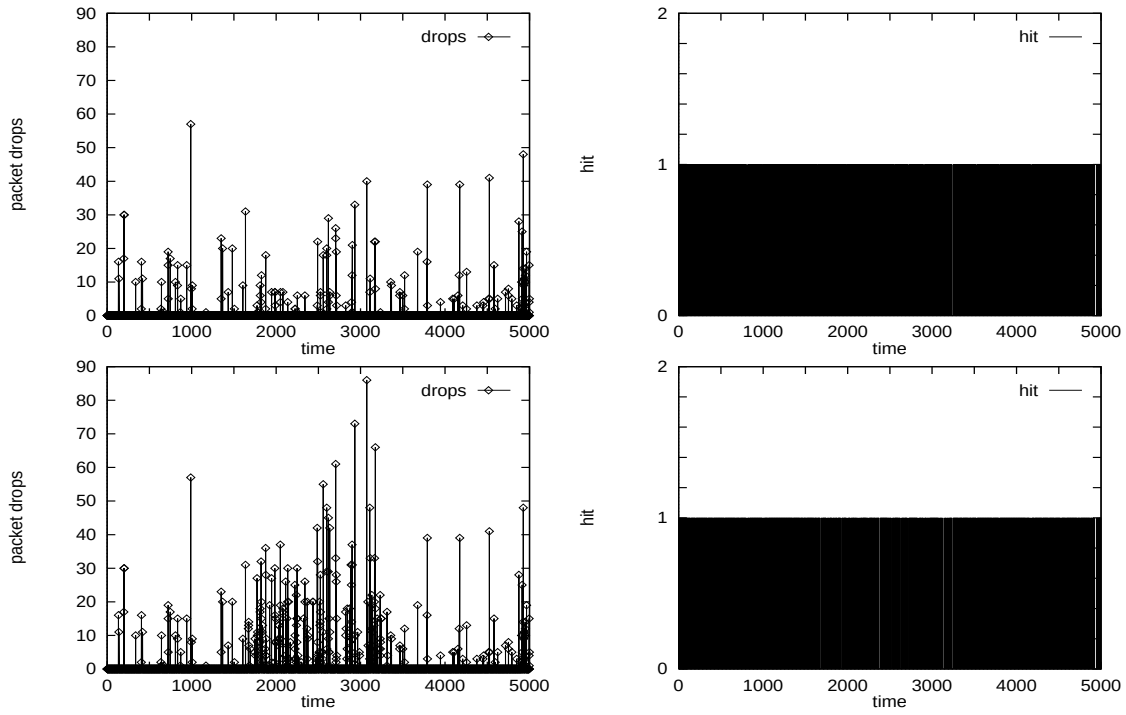

Figure 8: Static FEC run with constant redundancy $h = 3$. Top row: Packet loss trace (left) and hit occurrence trace at receiver (right) for 5000 time steps. Bottom row: Same as above but with the difference that the background traffic (λ^C) is increased at time $t = 1700$ and decreased to its original level at $t = 3400$. The *white* stripes in the right-column figures indicate instances at the receiver when its real-time constraints were violated.

configuration when the traffic level was increased at $t = 1700$ and returned to its original level at $t = 3400$. The hit trace clearly shows a marked improvement over the corresponding trace of the static FEC run in Figure 8 (bottom row, 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 9 (bottom row, right) which shows redundancy rate h as a function of time[4]. 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.

[4]Note that h is maintained as a real variable even though redundancy can only be discrete, i.e., a nonnegative integer. This is to achieve better control resolution; the h value is rounded before determining the size of the actual overcode.

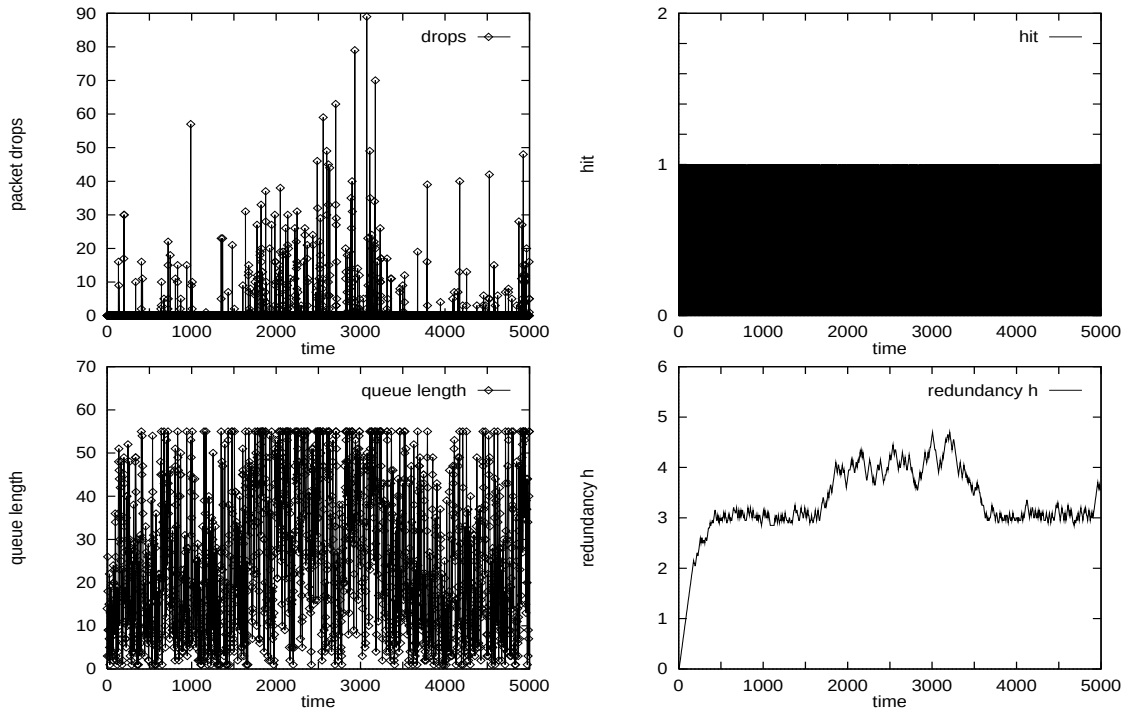

Figure 9: AFEC run. Top row: Packet loss trace (left) and hit occurrence trace at receiver (right) when the background traffic is increased at time $t = 1700$ and decreased to its original level at $t = 3400$. Bottom row: Buffer occupancy trace (left) and redundancy rate h trace (right) for the same run.

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.

4.4 Performance Envelop

Figure 10 (left) shows the hit rate ν of the AFEC protocol as a function of cross traffic intensity λ^C, $\lambda^C = 0, 1, \ldots, 10, 15, 20, 30, 40$ under three different bottleneck buffer capacity conditions, $B = 15, 55, 75$. When cross traffic intensity is low, the three performance curves coincide indicating an underutilized system where the packet loss rate is sufficiently small so that losses or delay fluctuations can be almost entirely compensated by a suitable infusion of redundancy. As the cross traffic intensity increases, however, adaptivity becomes less effective, especially for the large buffer case ($B = 75$) due to the pronounced unimodal shape of its redundancy-recovery rate function. For extremely high cross traffic intensity values, the performance graphs converge again. Even For best-effort real-time applications capable of tolerating nonnegligible miss rates, a hit rate below—say 80%—may be too small to be useful.

To discern how much of the performance loss is due to intrinsic limitations imposed by the

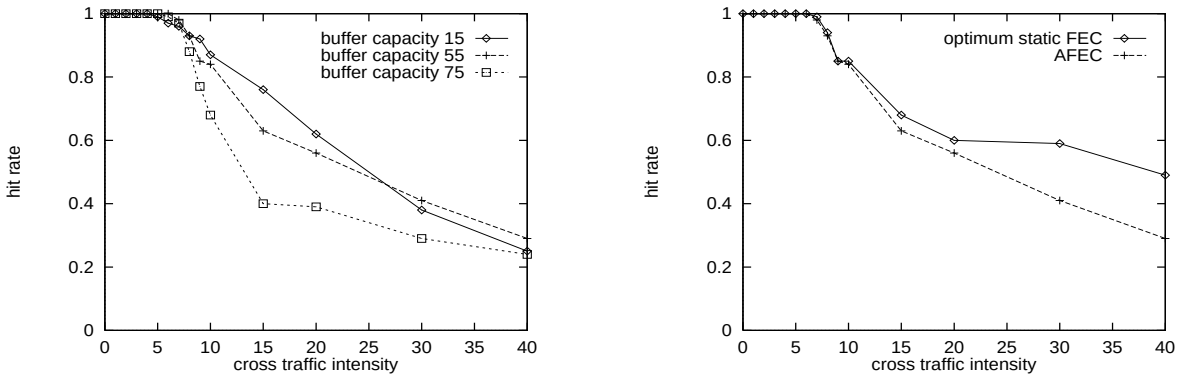

Figure 10: Left: Hit rate ν of AFEC as a function of traffic intensity $\lambda^C(t)$ under three buffer capacity conditions $B = 15, 55, 75$. Right: Same set-up where AFEC is compared against the *best* performance (over initial h) of static AFEC for $B = 55$.

increase in cross traffic intensity, we have measured the *best* performance achieved by the static FEC protocol where "best" refers to taking the maximum over the set of initial h values. Figure 10 (right) plots the optimum static FEC performance curve against AFEC's performance curve for $B = 55$. For very large cross traffic intensities we observe divergence setting in; however, for the postulated hit rates of interest (e.g., 80%), the two curves remain close together.

One may view the optimum static FEC performance graph as representing an *envelop curve* that delimits the boundaries of what is achievable. Although it is possible for AFEC to match the performance of optimum static FEC, it is difficult to exceed it under present cross traffic conditions—Poisson arrivals with rate λ^C—due to the short-range dependent nature of the cross traffic source. That is, since AFEC is a feedback control protocol, it must observe the network state before it can adjust to it. Short-range dependent sources have the property that the occurrence of intense activity at time t bears little correlation to the activity level at time $t + T$ for $T > 0$ "large." Indeed, given control computations including encoding/decoding that must be carried out at the millisecond or submillisecond level, network latencies can be hundreds of milliseconds long, and the short-range correlation structure may be too brief relative to T (e.g., round-trip time) to be highly effective for adaptation purposes[5]. AFEC is most effective vis-à-vis static FEC when network changes are persistent as shown in Section 4.3.

Returning to Figure 10 (right), based on the previous discussion, for the hit rates of interest, AFEC may be achieving performance that is close to what is, in the best of circumstances, possible.

4.5 Self-Similar Traffic and Burstiness

4.5.1 Impact of Self-Similarity

One of the most important considerations when evaluating the effectiveness of FEC-based QoS provision schemes including AFEC is their performance under "bursty" traffic conditions. FEC is known to perform best if a connection can be modeled as an independent error channel where packet

[5]This is also related to the delay-bandwidth product which increases proportionately in broadband networks.

drops (i.e., erasures) are uncorrelated. Recent simulation [58, 7] and analytical studies [9, 10, 12] have shown that FEC performance can significantly deteriorate under bursty traffic conditions where packet drops occur in a concentrated manner.

An extreme form of burstiness is *scale-invariant burstiness* in which a traffic process' variability does not dampen out as the traffic process is aggregated in time. Recent measurements of network traffic [37, 23, 53, 27, 14] have shown that network traffic—including individual VBR video sources—exhibit this precise phenomenon as captured by self-similarity and long-range dependence. There is some debate as to how significantly long-range dependence (LRD) impacts performance. LRD, in self-similar time series, can be measured by the *Hurst parameter* H with range $0.5 \leq H \leq 1$ where $H \approx 1$ means that the time series is more correlated and bursty. There exist standard techniques for estimating the Hurst parameter of a time series and a discussion of the various techniques and relative merits can be found in [59].

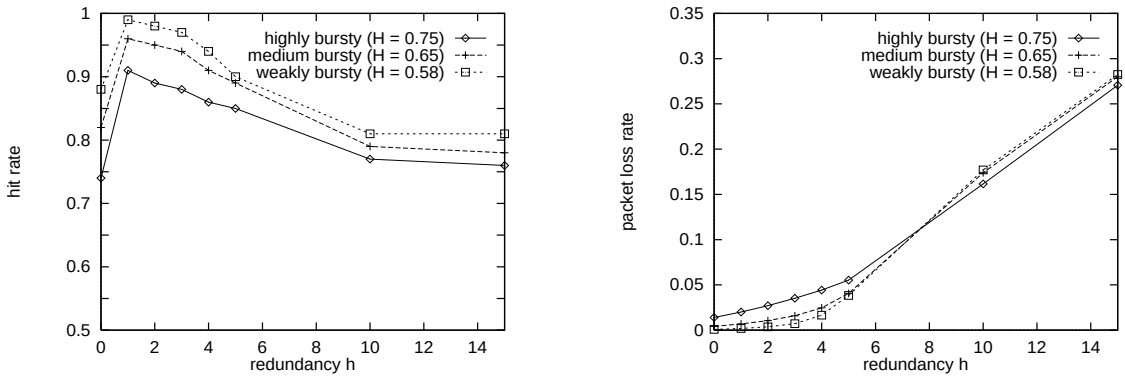

Figure 11: Left: Measured redundancy-hit rate function $\nu = \mathcal{G}(h)$ for three three levels of cross traffic burstiness conditions $H = 75, 65, 58$. Right: Corresponding packet loss rates.

Figure 11 (left) shows the redundancy-recovery rate relation under three different burstiness conditions of the cross (or background) traffic. The self-similar background traffic was obtained by multiplexing several traffic streams with heavy-tailed ON/OFF times which is known to yield self-similar traffic series with varying LRD depending on the degree of heavy-tailedness [60]. The mean traffic intensity was held constant to achieve comparability such that performance difference is solely attributable to traffic variability[6]. In fact, the mean traffic intensity of the most bursty traffic ($H = 0.75$) was *smaller* than that of the intermediate case ($H = 0.65$) which, in turn, had a smaller mean traffic intensity than the least bursty input ($H = 0.58$): 89.5 packets/s vs. 91.6 packets/s vs. 92.1 packets/s. In spite of this bias, Figure 11 (left) shows that the hit rate of $H = 0.75$ cross traffic is smaller than that of $H = 0.65$ and $H = 0.58$. This ordering relation holds over the entire redundancy spectrum $0 \leq h \leq 15$ tested. Thus, FEC performance under bursty traffic is indeed adversely affected manifesting itself as a downward shift in the redundancy-recovery performance curve.

Figure 11 (right) shows the corresponding packet loss rates. We observe that up until $h \approx 7$, the

[6]It should be noted that the more heavy-tailed a distribution, the more its sample mean (even with a large number of samples) will *underestimate* the true population mean. Thus care has to be given in the sampling process.

packet loss rate associated with the most bursty traffic ($H = 0.75$) is higher than that of $H = 0.65$ and $H = 0.58$. Again, this is in spite of the mean traffic intensity of $H = 0.75$ being *lower* than that of the other two. As H increases further, however, the packet loss of $H = 0.75$ traffic becomes smaller than that of $H = 0.65$, $H = 0.58$. This is due to the fact that as redundancy increases the buffer capacity (fixed at 20 packets) gets saturated by h which acts as a "DC component," and the variability associated with the least bursty traffic is sufficient to cause repeated buffer overflow. The concentratedness of packet drops for $H = 0.75$ traffic remains, and that is why the ordering relation in Figure 11 (left) stays invariant.

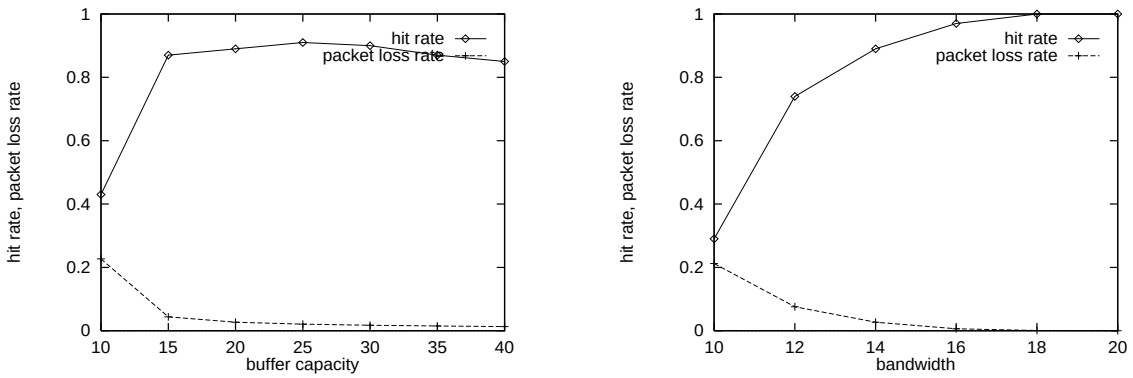

Figure 12: Left: Measured hit rate and packet loss rate as a function of bottleneck buffer capacity. Right: Measured hit rate and packet loss rate as a function of bottleneck bandwidth.

4.5.2 Impact of Network Resources

Figure 12 (left) shows the performance characteristics of the same configuration as in Figure 11 except that now the buffer capacity is varied in the range 10–40. We observe that up until $B \leq 25$ the hit rate continues to improve. However, after $B = 25$, the hit rate begins to decline—albeit gradually—as increased buffer capacity turns the system more and more into a delay dominated state. The packet loss rate decrease with increased buffer capacity as expected. One of the implications of LRD in queueing systems is that the queue length distribution decays hyperbolically and not exponentially as in Markovian traffic sources such as Poisson [44, 38]. In the case of FEC, this makes the system more prone to be delay dominated thus adversely affecting performance. Thus buffer capacity is not an effective resource for facilitating QoS provision under FEC.

Figure 12 (right) shows the performance characteristics when varying bottleneck bandwidth while keeping buffer capacity fixed at 20. We observe that as bandwidth is increased the hit rate converges to 1. That is, a two-fold increase in link speed is more effective at facilitating FEC performance than a two-fold or n-fold increase in buffer capacity. The relative merits of buffer capacity vs. bandwidth for FEC under self-similar traffic conditions are consistent with those observed for non-real-time performance conditions [48, 49].

4.6 AFEC Dynamics

4.6.1 Operating Point Instability

In this section, we show the instability ensueing from applying control law (1) without the backoff mechanism incorporated in the augmented AFEC control equation (9). Figure 13 (left) shows the observed redundancy-hit rate function with network configuration settings where the bottleneck buffer capacity was set relatively high ($B = 85$). The maximum hit rate achievable by the static FEC protocol is 0.85. The augmented AFEC protocol was able to achieve a hit rate of 0.84. The sensitivity of the curve near its peak and its early rise leave little room for adjusting h; i.e., $\gamma^* - k$ is very narrow.

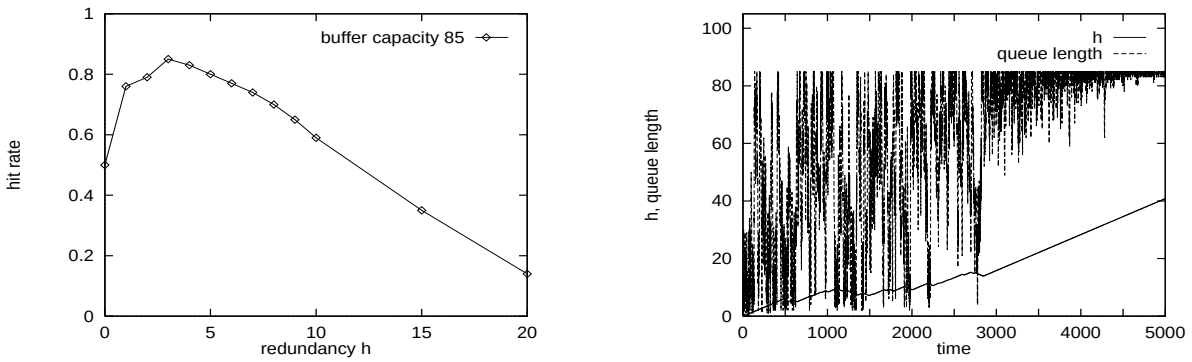

Figure 13: Left: Observed redundancy-hit rate function $\nu = \mathcal{G}(h)$ for large bottleneck buffer ($B = 85$) condition. Right: Redundancy h and buffer occupancy trace for the same system with a too large target recovery rate γ_*, $\gamma_* \approx \gamma^*$.

Figure 13 (right) shows the h and buffer occupancy trace for a AFEC run without the backoff mechanism of (9). With the backoff mechanism turned off, we see a positive drift in h—still fluctuation up and down—until around time $t = 2750$ at which point the positive feedback loop kicks in with force, prompted by the magnitude of h becoming the dominating term with respect to total traffic influx. The queue quickly reaches saturation resulting in further decreases in the hit rate ν as predicted by the sensitive unimodal shape of the network's redundancy-hit rate function (Figure 13 (left)). This, in turn, provokes further increases in h by control law (1) sustaining the positive feedback loop since (1) is oblivious to the fact that it finds itself in the unstable region.

4.6.2 Delay Instability

Figure 14 shows the adverse effect of increased network delay in stabilizing system behavior as predicted by the analysis of Section 3.2.2. Figure 14 (left) shows the redundancy rate h trace for a typical network configuration where the feedback link (D, G) from the destination D to the router G was set at 10. The adjustment factor ϵ in control law (1) was fixed at $\epsilon = 0.15$.

Figure 14 (middle) shows the same system with the only difference that the link latency on the feedback link (D, G) was increased from 10 to 100. The marginal effect of increasing network delay is clearly discernible where the amplitude of the oscillation has increased several-fold, leading to a

wildly fluctuating, ill-behaved system.

To further illustrate the stability condition captured in relation (8), i.e., $0 < \tau\epsilon < \text{const}$, we have kept everything the same as in the network configuration of the middle figure, except that the adjustment factor ϵ was decreased from 0.15 to 0.03. Its effect is shown in Figure 14 (right). As predicted by the stability condition (8), clearly the amplitude of the oscillation has damped down to a similar level as that of Figure 14 (left). One difference is the diminished responsiveness of the resulting system to changes in network state which is a price that must be paid to achieve stability.

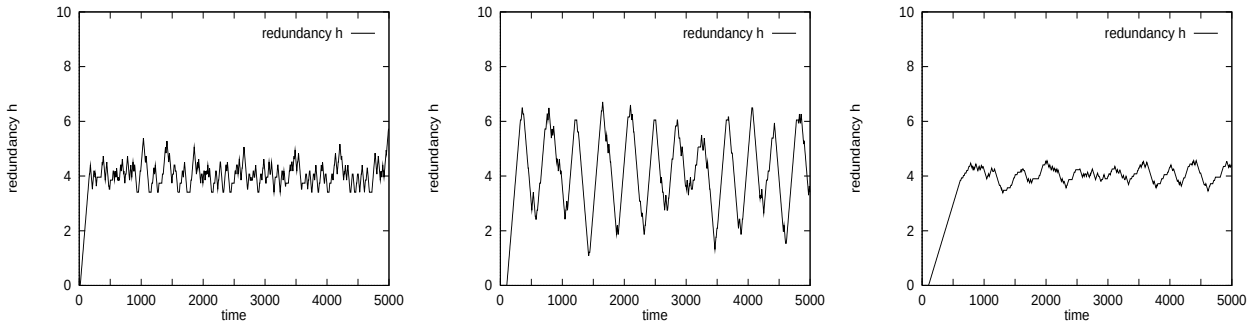

Figure 14: Left: Redundancy h trace for a network configuration with a small end-to-end network latency. Middle: Redundancy trace for the same network configuration except that the end-to-end link latency has been increased from 10 to 100. Right: Same network configuration as in the middle figure except that the adjustment factor ϵ was decreased.

5 Real-Time MPEG Video Transport

5.1 System Description

We have built an implementation of AFEC customized for the transport of real-time MPEG video and tested it in an end-to-end network environment. This section describes this system which is called MAFEC.

5.1.1 Overall Structure

MAFEC consists of a number of modules including the FEC encoder, FEC decoder, sender-side controller C_S, and receiver-side controller C_R. On the sender side, a stream of I, P, B frames are generated by an MPEG encoder at some fixed frame rate f (usually 25 or 30 frames per second). The FEC encoder applies forward error correction on each frame, first, by segmenting it into k number of packets of fixed MTU size s, then, using the redundancy level h passed on from the sender-sider controller C_S, producing a sequence of $n = k + h$ packets which are submitted to the network.

The receiver, upon receiving a sequence of packets belonging to the same frame (or block) checks if a sufficient number (i.e., at least k) of packets have arrived and consults with the receiver-side controller C_R to determine timeliness. If the conditions are successfully met, k of the packets are

forwarded to the FEC decoder proper—the checking or preprocessing module is considered to be part of the FEC decoder—which decodes the packet stream to recover the original k data packets constituting the sender's frame. The I, P, or B frame is then forwarded to the MPEG II player which applies its own decoding to produce the uncompressed frames which are then rendered on the terminal as part of the video stream. This forward flow structure is depicted in Figure 15.

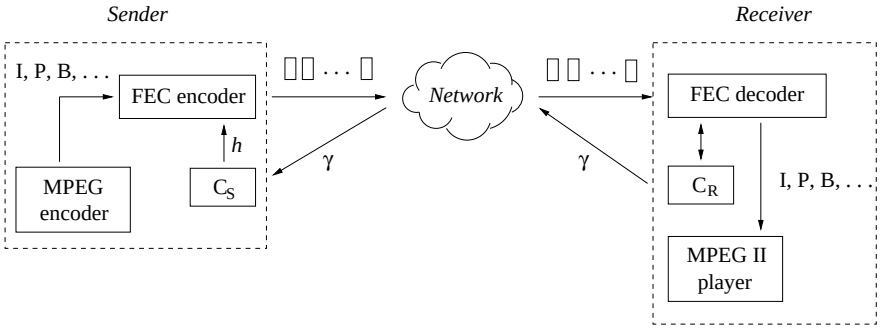

Figure 15: Overall structure of MAFEC. At the sender, I, P, B frames forwarded from an MPEG encoder are fed into the FEC encoder. The latter is controlled by a sender-side controller C_S. At the receiver, the coded packets are fed into the FEC decoder which is controlled by a receiver-side controller C_R. The decoded I, P, B frames are forwarded to a MPEG-II player.

The receiver-side controller, when processing incoming packets and determing timeliness—be they successful or not—computes certain control information including the number of timely arrived packets γ which is then fed back via a control packet over the network to the sender. The details of the AFEC control algorithm as captured by C_S, C_R which are customized toward handling MPEG data are described in Section 5.1.3. All of the modules are implemented in software, and it is completely end-to-end such that the system can be deployed over any network.

5.1.2 Forward Error Correction using IDA

Encoding We use Rabin's IDA [55] as the forward error correction code. IDA is a simple, efficient error correcting code over Galois fields where encoding and decoding involve straightforward matrix multiplication and inversion, respectively, albeit over the finite field $\mathbb{Z}_p$ ($p = 257$). Given a message M (i.e., I, P, or B frame) of ℓ bytes, it is segmented into a sequence of k vectors each of size $s = \ell/k$ (the last vector is padded if necessary). This yields a $(k \times s)$ matrix $\mathbf{B}$ where the k vectors correspond to the rows of the matrix.

Let $\mathbf{A}$ be any $(N \times k)$ matrix where $N \geq k$ such that any subset of k vectors obtained from its rows are linearly independent. From a practical perspective, N needs to satisfy $N \geq k + h$ where h is the variable redundancy factor. In fact, for VBR traffic such as MPEG video in our case, k can be variable itself since ℓ is variable (more on this later). Hence N is chosen a priori to be sufficiently large. Such matrices are easy to precompute once and for all. The code vectors are obtained by computing

$$\mathbf{X} = \mathbf{A}\,\mathbf{B} \tag{11}$$

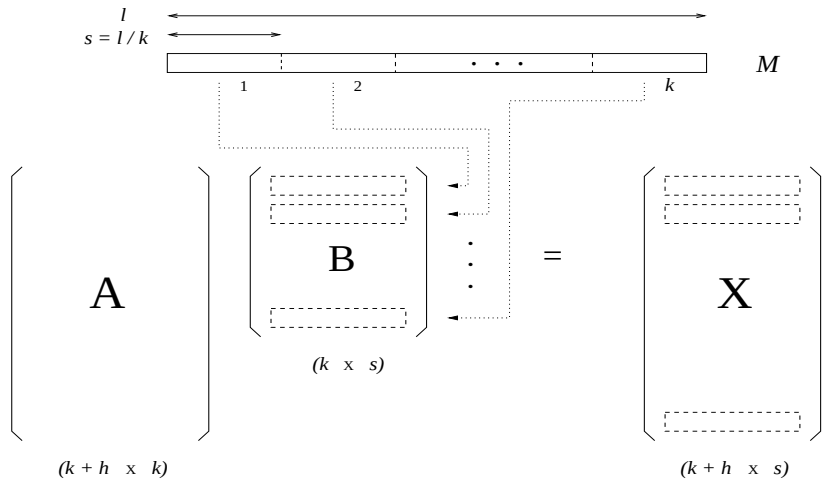

Figure 16: IDA encoding process: Message M (of length ℓ) is segmented into k pieces giving $\mathbf{B}$. Matrix $\mathbf{B}$ is then multiplied with $\mathbf{A}$ to give the code matrix $\mathbf{X}$.

where the rows of $\mathbf{X}$—$\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$—constitute the code vectors. Note that if $k + h < N$, then only the truncated matrix of $\mathbf{A}$ consisting of the first $n = k+h$ rows, $\mathbf{A}_n$, need be used in the matrix multiplication (11) yielding $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$. That is, $\mathbf{X} = \mathbf{A}_n \mathbf{B}$. The code vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ are then submitted to the network. The encoding process is depicted in Figure 16.

Decoding Assume at least k of the code vectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ are (timely) received at the receiver. Denote them by $\mathbf{x}_{i_1}, \mathbf{x}_{i_2}, \ldots, \mathbf{x}_{i_k}$ and the resulting matrix $\mathbf{X}_k$. Let $\mathbf{A}_k$ be the $(k \times k)$ square matrix consisting of the rows of $\mathbf{A}$ indexed by $i_1, i_2, \ldots, i_k$. Then by (11) and the independence property of $\mathbf{A}$,

$$\mathbf{A}_k^{-1} \mathbf{X}_k = \mathbf{B} \tag{12}$$

and the original message M (represented as $\mathbf{B}$) is recovered. Thus decoding reduces to inverting the matrix $\mathbf{A}$.

Processing Overhead We use Gaussian elimination over the finite field $\mathbb{Z}_p$ to invert $\mathbf{A}_k^{-1}$. The time complexity of encoding and decoding is $O(sk^2)$ for matrix multiplication and $O(k^3)$ for matrix inversion. Since k is the dominating variable and $s = \ell/k$, it is advantageous to keep k small at the cost of making s large and thus incurring a space or coding inefficiency cost. Indeed, for k large, we find that the FEC decoder can become the bottleneck causing buffer overflows and timeliness violations at the receiver even when the network and router are well-behaved. Thus when implementing MAFEC in software, in real-time, on general-purpose workstations, processing cost becomes a nonnegligible factor which needs to be carefully controlled.

To this end, we implemented a dynamic update mechanism which, in the presence of variable ℓ (i.e., variable I, P, B frame sizes), tries to maintain a small, constant k (e.g., 2) by suitably adjusting the packet size s. A typical configuration we have used is $k = 2$ and $s \leq 4200$ bytes (upper bound packet size). Thus, at the sender, when performing encoding, if the frame size

$\ell \leq ks$, s was set to $\ell/k = \ell/2$; otherwise, k was set to $\ell/s = \ell/4200$. For the MPEG I video traces we tested, the above parameters were found to be adequate.

We note that in internetworks, in particular, ATM networks where packets are of small fixed size (53 B), increasing s to achieve a small k is accompanied by a fragmentation cost. Thus, in general, if the processor speed on the end stations is sufficient, choosing k such that coding efficiency is increased and s matches the network's MTU is an appropriate policy.

5.1.3 Receiver-Oriented Control

The adaptive redundancy control algorithm employed for MPEG video transport is fundamentally the same as the one described in Section 3. However, there exist subtle differences including customizations specific to MPEG payloads which are explicated next.

Receiver-Initiated QoS Control Noting that the consumer of the QoS-sensitive real-time video is the receiver, control actions—although ultimately affected at the sender—are initiated at the receiver. First, as the application starts up, the user interacts with the receiver-side control module C_R (cf. Figure 15) to indicate its *desired* QoS level, in particular, the target hit rate ν_* (here $0 \leq \nu_* \leq 1$). This is then used as a *parameter* by the AFEC control algorithm when making its adjustments. Although ν_* is, by default, held fixed after start-up, the user is given the option of changing ν_* while the application is in progress via a "tuning knob" that resets the value of ν_*. This allows the user to interactively adjust video quality.

Second, if AFEC determines through its instability detection mechanism (i.e., $d\gamma/dh < 0$) which is used in the triggering of exponential backoff that the target hit rate is unattainable—i.e., $\nu_* > \nu^*$—the user is notified of this fact by the application such that corrective actions can be taken (decrease ν_*) should the user choose to do so. Instability detection is carried out at the sender side where the correspondence between changes in h and the resultant changes in ν is kept track of. There is a further parameter that the user can provide to the application through which more intimate QoS control is facilitated. This is discussed next.

Stringency Control The *stringency parameter* $W \geq 0$ is a local time window over which the measured or observed recovery rate is computed by taking the time average. That is, the recovery rate $\gamma(t)$ fed back to the sender at time t is given by $\mathbf{E}_{\tau \in [t-W,t]}\{\gamma(\tau)\}$. The larger the stringency parameter W, the more variation is allowed in a sample path of the system which nevertheless meets the desired recovery rate in the mean. Making W small ensures that QoS violations are more tightly controlled on a smaller time scale. Setting W too small, however, can lead to instabilities arising from overly responsiveness to noise-like perturbations. Currently, W is interpreted as the W most recent observations in the past (i.e., W is not an absolute time measure) and it is set to a default value of 10 unless otherwise specified.

Differentiated Weighting It is a well-known fact that, other things being equal, losing an I frame is more detrimental to perceived visual quality than losing P or B frames due to the latters uni- and bi-directional dependencies. Indeed, this is the main observation that underlies the PET system by Albanese et al. [2, 35] where a user-specified weighting is used to apply higher coding overhead or redundancy to I frames followed by P frames and B frames.

MAFEC supports a similar mechanism whereby the user specifies three weighting parameters $\omega_I, \omega_P, \omega_B \geq 1$ that are then multiplied with h to yield the weighted redundancy. Thus at the sender side, C_S passes on $\omega_I, \omega_P, \omega_B$ (by default set to 1) to the FEC encoder at start time, and these values are subsequently used to amplify the redundancy variable h (output from C_S) pending on the type of the frame.

5.2 Experimental Set-Up

5.2.1 Network Configuration

Our experiments were carried out in the test environment shown in Figure 17. MAFEC, consisting of the AFEC sender and receiver, sent its traffic over a router where cross traffic stemming from a separate cross traffic source was multiplexed with the application traffic. By varying the cross traffic characteristics as well as the resources (e.g., buffer capacity) at the router a wide spectrum of contention levels could be produced. To facilitate a controlled environment for measurement purposes, the camera and MPEG I encoder at the sender were replaced by an emulator that fed the frames of *stored* MPEG videos at real-time frame rates (25 or 30 frames per second) to the AFEC sender. Thus, as far as MAFEC was concerned, its functioning was not affected since in either case I, P, B frames were input to the FEC encoder at real-time speeds. The MPEG II player at the receiver was always run, and the performance measurements reflect the overhead incurred by this additional processing overhead. Thus modulo *where* the I, P, B frames came from at the sender-side, the system depicted in Figure 15 was faithfully implemented with all the components implemented in software without specialized hardware support.

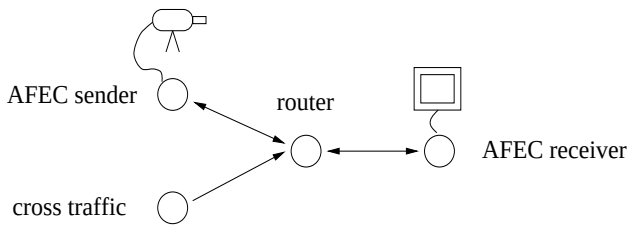

Figure 17: Experimental set-up for MAFEC performance measurements.

The logical topology depicted in Figure 17 was realized over a private FastEthernet LAN environment with the AFEC sender/receiver, cross traffic source, and router running on four separate Sun UltraSparc 1 workstations. Thus all traffic first transited through a FastEthernet hub before reaching the next hop destination. Without our configurable router, we found that the 100 Mbps bandwidth of FastEthernet was too large relative to the data rate of MPEG I video (1.5+ Mbps), even with our cross traffic source active, to cause significant contention. With an UltraSparc 1 node acting as a configurable router implementing FIFO scheduling, a wide range of contention levels could be created under controlled conditions. The AFEC protocol as well as the cross traffic source ran on top of UDP, and MAFEC was realized as an application layer process portable to other UNIX environments with standard multimedia kit support for video capture and MPEG encoding. MAFEC was also tested over non-LAN internetwork environments with favourable results, but due

to the non-controlled environment, the results are not shown here.

5.2.2 Benchmark Data

We stress that the present performance results are obtained using actual MPEG I video data rather than frame size traces used in other simulation and experimentation set-ups. Thus, the cost of FEC encoding and decoding, control actions, and MPEG II player's processing overhead are all reflected in the performance results. The cost of FEC decoding which is dominant when the coding efficiency is high (i.e., high k) has been handled by the dynamic control mechanism described in Section 5.1.2.

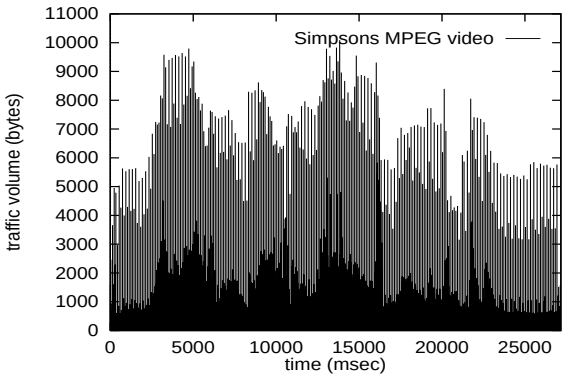

Figure 18: Traffic series of a Simpsons MPEG video clip.

Figure 18 shows the frame size trace of a Simpsons cartoon MPEG I video clip which consists of 816 frames which at 30 f/s frame rate lasts for about 27 seconds. There are 137 I frames, 136 P frames, and 543 B frames, and the GOP sequence is given by $IBBPBB$. Other benchmark instances tested include Red's Nightmare and Terminator MPEG I video clips. They yield similar performance results as Simpsons clip and are not shown here. The MPEG video clips used have a Hurst parameter estimate in the 0.8+ range and thus possess varying degrees of long-range dependence and burstiness. There is an on-going debate as to how relevant LRD is to network performance when resources are limited [23, 25, 27, 57, 49]. VBR video can be modeled using a variety of models (short-range and long-range) including such exotic schemes as random maps on a circle [31]. What is commonly accepted in the midst of this diversity and controversy is that compressed video is "bursty," certainly much more so that Poisson traffic, and it is only this aspect that will be reflected here.

5.3 Performance Measurements

5.3.1 Unimodal Redundancy-Recovery Relation

Figure 19 (left) shows the measured redundancy-recovery function for the Simpsons MPEG clip when using static FEC with h fixed in the range 0–10. The frame rate was set at 25 fps and the cross traffic source was generating Poisson background traffic with $\lambda = 5$ packets per second (MTU 1 kB). Clearly, the observed function is *unimodal* consistent with the simulation results.

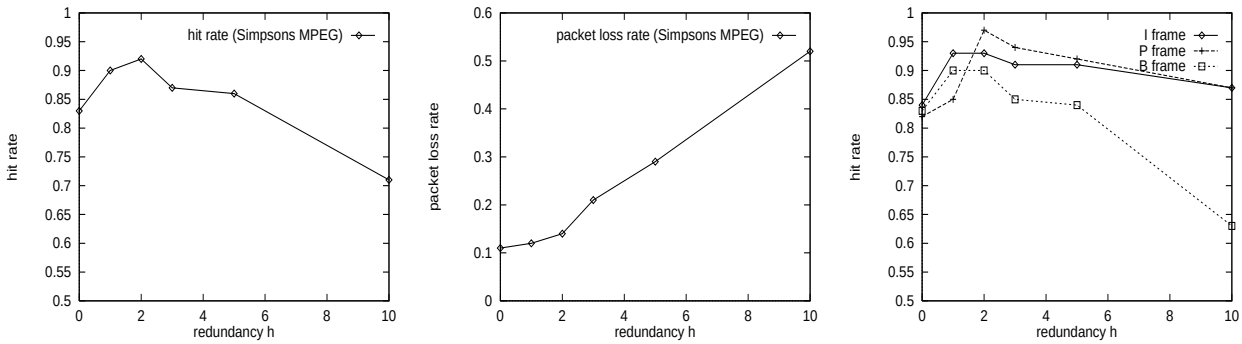

Figure 19: Left: Measured unimodal redundancy-recovery function for Simpsons MPEG clip. Middle: Corresponding packet loss rate function. Right: Measured redundancy-recovery function for Simpsons MPEG clip partitioned for I, P, B frames.

Figure 19 (middle) shows the corresponding packet loss rates which, as expected, are monotonically increasing. Figure 19 (right) shows the redundancy-recovery relation for I, P, and B frames individually. As with the aggregate redundancy-recovery function, the shapes are again unimodal with the function for B frames exhibiting the most curvature. The latter stems from the fact that the frequency of B frames is dominant, and when h is high, commensurately suffers to a greater extent.

5.3.2 AFEC vs. Static FEC

Structural Adaptivity Figure 20 (top) shows the trace of a static FEC run with h set at 0 when a Poisson background traffic is turned on during the interval $[300, 600]$ while it remains off during the other times. Here, time is represented using frame numbers. Figure 20 (top, left) shows the packet drop trace with no drops occurring when the background traffic is idle—in spite of the application traffic itself being bursty—and packet drops occurring during the interval when the cross traffic source is active. The (top, right) figure shows the corresponding hit trace which shows the detrimental effect of packet drops on the recovery rate.

Figure 20 (bottom) shows the trace of an AFEC run for the same configuration as above. The left figure shows that due to adaptive changes in h, the packet drops in the center interval have increased over their static FEC counterpart. The hit trace, however, shows a reverse effect, i.e., an improvement in the hit rate due to redundancy control even though a higher packet loss is being incurred. The white miss line in Figure 20 (bottom, right) is due to system buffer overflow at the receiver host which occasionally occurs due to the FEC decoding overhead discussed in Section 5.1.2.

Figure 20, in conjunction with the corresponding figures for the simulation-based runs in Section 4.3, show that static FEC is inferior to AFEC in the sense that without prior knowledge of the network state and its stationarity, it is unlikely that a preset redundancy level will remain effective during the lifetime of a connection. Thus either a sacrifice in QoS or a wastage of resources will be incurred. In the context of Figure 20 (top), if h is set to 0 then QoS suffers during the center third interval when the network undergoes structural perturbations such as arrival of new connections.

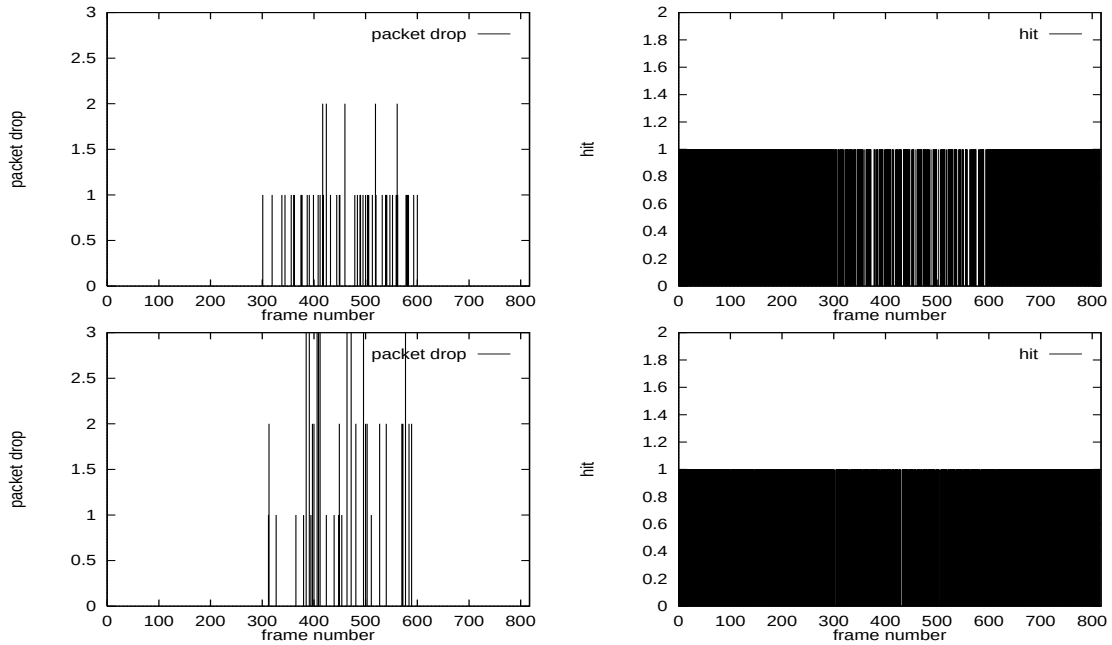

Figure 20: Static FEC vs. AFEC run when cross traffic is active in over interval $[300, 600]$ for Simpsons MPEG clip. To row: Packet drop trace (left) and corresponding hit trace (right) for static FEC with $h = 0$. Bottom row: Same as above but with AFEC active.

If h were set to 2 or 3, then QoS would remain minimally affected during the active cross traffic period, however, while the cross traffic is inactive, bandwidth would be unnecessarily wasted.

Structural Instability Figure 21 (left) shows the unimodal redundancy-recovery relation obtained using static FEC for a fixed configuration and the performance achieved by AFEC when the target hit rate was set at 0.96. The rendered hit rate 0.963 is higher than the desired QoS and this is possible due to the target hit rate being below the peak hit rate 0.972 of the redundancy-recovery function, thus putting the desired operating point into the stable region (cf. discussion in Section 3.2).

Figure 21 (right), on the other hand, shows the target hit rate 0.98 falling above the peak hit rate, and the desired QoS is impossible to achieve under the given system configuration. The structural instability associated with the desired operating point will cause AFEC to overshoot—recall that AFEC will continuously attempt to attain the desired hit rate—which then triggers the exponential backoff mechanism to preserve stability. This repeated overshoot and backoff causes performance to suffer thus achieving a hit rate of 0.964 that is smaller than the static FEC's peak hit rate.

5.3.3 AFEC Dynamics

Behavior in the Stable Regime Figure 22 (top) shows the average hit rate, hit, and redundancy trace for an AFEC run for the Simpsons MPEG clip when the target hit rate is set to 0.92 and Poisson cross traffic is applied throughout the application's lifetime. The average measured hit rate

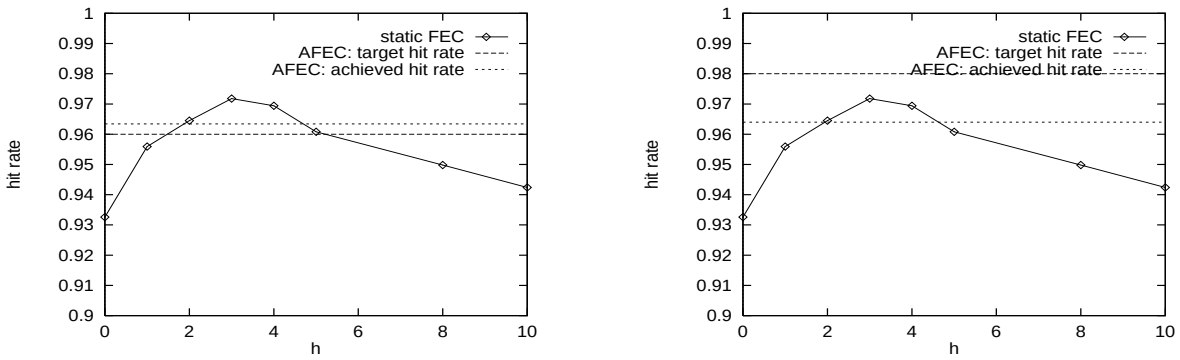

Figure 21: Left: AFEC hit performance vs. static FEC performance when target hit rate is 0.96. Right: AFEC hit performance vs. static FEC performance when target hit rate is 0.98.

shows that the desired QoS level is being satisfied (modulo short-term variation effects) and this is reflected in the well-behaved hit trace plot shown in Figure 22 (top, middle). Figure 22 (top, right) shows the continued changes in h during the entire run caused by the contention level being maintained via the cross traffic process throughout the run.

The bottom row figures show the corresponding performance measurements when the target hit rate was raised to 0.96 while keeping everything else the same as above. Figure 22 (bottom, left) shows that the desired hit rate 0.96 is not attainable for the present configuration (i.e., 0.96 falls into the unstable region) and a final hit rate of 0.926 is attained that is even lower than the hit rate achieved for the 0.92 target, 0.935 (although by not much). The hit trace—Figure 22 (bottom, middle)—shows no marked differences with its counterpart in the top row except for a concentrated bad period near frame 400.

Behavior in the Unstable Regime The latter, however, is a sign of an important difference in the two dynamics which is more clearly visible in the h trace plot. Figure 22 (bottom, right) shows a much wider range in h than its corresponding trace Figure 22 (top, right), but more importantly, it shows the steep climb and even steeper descent of h that is characteristic of feedback control behavior near unstable points. That is, the steep climb is due to the onset of instability—albeit temporary—that is then tamed by the exponential backoff which causes an even steeper decline in h. We can observe this "stairway-to-abyss" shape being approximately repeated at different scales throughout the run.

The presence of frequent backoffs—i.e., exponential backoffs due to instability and *not* the regular backoffs associated with control law (2) (or the first part of (9))—can be more clearly seen in Figure 23 which shows the time instances when an exponential backoff was triggered at the sender. We observe that in the target hit rate 0.92 run (Figure 23 (left)) only two exponential backoff instances occur, one just after frame 100, and another just before frame 500. Looking carefully at Figure 22 (top, right), one can observe a miniature version of the stairway-to-abyss shape at the corresponding points.

Figure 23 (right), on the other hand, shows significantly more exponential backoff instances for the 0.96 target hit rate run which is consistent with the redundancy pattern observed in Fig-

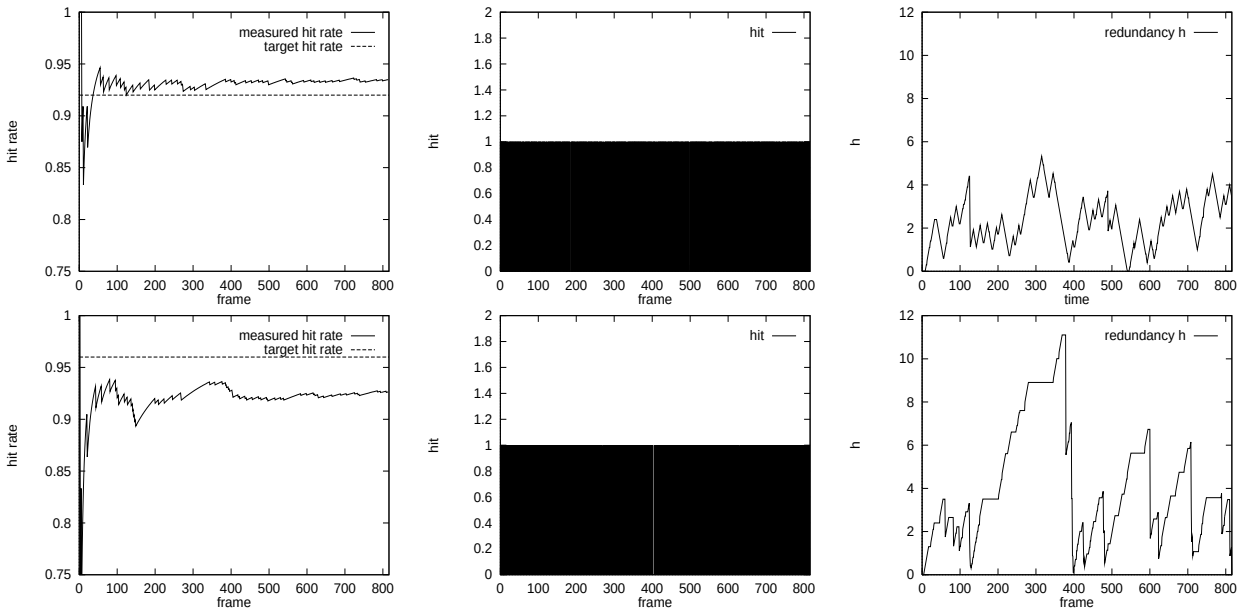

Figure 22: AFEC dynamics. Top row: Target hit rate 0.92. Measured hit rate (left), hit trace (middle), h trace as a function of time (expressed as frame number). Bottom row: Target hit rate 0.96. Measured hit rate (left), hit trace (middle), h trace as a function of frame number.

ure 22 (bottom, right). We emphasize that in AFEC, there are two types of "backoff" mechanism that cause a decrease in the h values. One is the *exponential backoff* mechanism that kicks in (second part of (9)) only when $d\gamma/dh < 0$ is detected—i.e., *increasing* h causes the recovery rate to *decrease* which corresponds to the unstable region. The other backoff mechanism is the *natural* backoff due to control law (2) which takes charge whenever $d\gamma/dh \geq 0$, i.e., h and γ move in the *same* direction. If the target hit rate lies in the stable region, exponential backoff rarely kicks in and the system undergoes "smooth" changes in h. This facilitates a more stable, quiet network state. On the other hand, if the desired QoS level falls outside the stable region, then AFEC is forced to engage in control actions that lead to a more variable network state. As mentioned in Section 5.1.3, whenever exponential backoffs are triggered—i.e., instability detected—this information can be forwarded to the user such that corrective actions in the form of lowering of the target hit rate can be induced if the user chooses to do so.

6 Discussion

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. We have given a comprehensive analysis of the adaptive forward error correction problem with respect to stability and optimality, and our conclusions are supported by both controlled simulations and experimentation of a prototype implementation in the context of transporting real-time MPEG video.

We have studied the performance characteristics of the AFEC protocol under bursty (i.e., self-

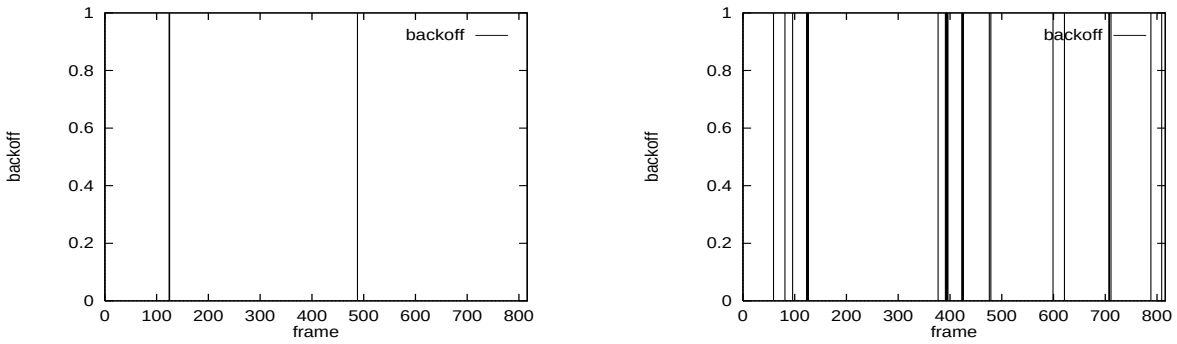

Figure 23: Left: Backoff instances for AFEC run with target hit rate 0.92. Right: Backoff instances for AFEC run with target hit rate 0.96.

similar) traffic conditions and structural perturbations, and we have shown the efficacy of AFEC vis-à-vis static FEC as well as its dynamical properties. The real-time MPEG transport system—MAFEC—was implemented entirely in software without specialized hardware or kernel support. We have shown performance measurements of MAFEC in a LAN environment where traffic streams are multiplexed at a configurable router and shown its effectiveness at facilitating QoS-sensitive transport of MPEG video.

MAFEC was further customized to exploit the peculiarities of its payload—MPEG video—by structuring its control in a receiver-oriented manner. In particular, in addition to the adaptive control provided via AFEC by the underlying system, the user is given the option to specify its QoS requirements both at start-up and dynamically through interaction with the receiver-side control module while the application is in progress. This allows for a more intimate accommodation of the user's QoS needs.

AFEC's effectiveness comes at the cost of trading bandwidth for QoS, and there are intrinsic limitations to its performance vis-à-vis reservation-based methods supported by some form of access control. This is so since sudden changes in network state—i.e., nonstationary effects—will temporarily deteriorate QoS while adaptation to the new network state is in progress. The advantage of AFEC vis-à-vis reservation-based schemes lies in the fact that a nonnegligible fraction of present and future soft real-time applications including video conferencing and Internet telephony may be tolerant to this level of QoS violations and thus render services that are satisfactory for everyday use. The state-of-the-art in guaranteed QoS provision schemes relies on the paradigm of overprovisioning [15, 16, 20, 21, 24, 36, 43, 46, 47, 17] which uses traffic shaping and GPS-based schedulers to assure stringent QoS quarantees. However, the scale-invariant burstiness associated with self-similar network traffic [37, 53, 60, 48] limit the shapability of real-time and aggregated traffic multiplexed at routers while at the same time reserving bandwidth that is significantly smaller than the peak transmission rate, thus making this approach extremely costly. Lastly, AFEC is a purely end-to-end QoS provision scheme and, as such, is well-suited to the current Internet environment—at least in the interim during which new architectures are being discussed—which has been dominated by an end-to-end paradigm [13].

Since AFEC—and systems built on top of AFEC such as MAFEC—are 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[7]. 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 (cf. Section 5.3.3)—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) 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., $d\gamma/dh < 0$), then it engages in exponential backoff just like TCP to prevent instability. This is fundamentally different from a greedy or noncooperative protocol that would exploit the fact that other protocols will be backing off when network contention arises, and by not backing off, is 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. A game-theoretic discussion of QoS provision issues can be found in [50].

Much of present work is directed in two venues, one, in the extension and application of AFEC to other real-time transport domains, in particular, MPEG audio which by virtue of its reduced data rate (at least an order of magnitude less than MPEG I video even for CD quality audio and several factors less for telephony, i.e., voice) is an obvious candidate for heterogenous wide-area network environments [8]; and two, in the extension of AFEC to incorporate predictive mechanisms to exploit the correlation structure present in self-similar traffic. The latter is part of a more general problem which is investigated in [51].

References

[1] R. Addie, M. Zukerman, and T. Neame. Fractal traffic: measurements, modelling and performance evaluation. In *Proc. IEEE INFOCOM '95*, pages 977–984, 1995.

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

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

[7] Our discussion will assume TCP Tahoe as the reference point as defined by its principal features given in [28].

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

[5] Azer Bestavros. AIDA: A bandwidth allocation strategy for distributed time-critical real-time systems. In *Proc. IEEE IPPS Workshop on Parallel and Distributed Real-Time Systems*, 1993.

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

[7] Ernst Biersack. Performance evaluation of forward error correction in an ATM environment. *IEEE J. Select. Areas Commun.*, 11(4):631–640, 1993.

[8] 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.

[9] 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.

[10] I. Cidon, A. Khamisy, and M. Sidi. Dispersed messages in discrete-time queues: delay, jitter, and threshold crossing. In *Proc. IEEE INFOCOM '94*, pages 218–223, 1994.

[11] I. Cidon, A. Khamisy, and M. Sidi. On queueing delays of dispersed messages. *Queueing Systems, Theory and Applications*, 15:325–346, 1994.

[12] 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.

[13] David Clark. The design philosophy of the DARPA internet protocols. In *Proc. SIGCOMM '88*, 1988.

[14] M. Crovella and A. Bestavros. Self-similarity in world wide web traffic: Evidence and possible causes. In *Proceedings of the 1996 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, May 1996.

[15] R. L. Cruz. A calculus for network delay, part I: network elements in isolation. *IEEE Trans. Inform. Theory*, 37(1):114–131, 1991.

[16] R. L. Cruz. Quality of service guarantees in virtual circuit switched networks. *IEEE J. Select. Areas Commun.*, 13(6):1048–1056, 1995.

[17] G. de Veciana, G. Kesidis, and J. Walrand. Resource management in wide-area ATM networks using effective bandwidths. *IEEE J. Select. Areas Commun.*, 13(6):1081–1090, 1995.

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

[19] 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.

[20] A. Elwalid and D. Mitra. Effective bandwidth of general Markovian traffic sources and admission control in high speed networks. *IEEE/ACM Trans. Networking*, 1(3), 1993.

[21] A. Elwalid and D. Mitra. Analysis, approximations and admission control of a multi-service multiplexing system with priorities. In *Proc. IEEE INFOCOM '95*, pages 463–472, 1995.

[22] 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.

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

[24] L. Georgiadis, R. Guérin, V. Peris, and K. Sivarajan. Efficient network QoS provisioning based on per node traffic shaping. *IEEE/ACM Transactions on Networking*, 4(4):482–501, 1996.

[25] M. Grossglauser and J-C. Bolot. On the relevance of long-range dependence in network traffic. In *Proc. ACM SIGCOMM '96*, pages 15–24, 1996.

[26] Jack K. Hale and Sjoerd M. Verduyn Lunel. *Functional Differential Equations*. Springer-Verlag, 1993.

[27] D. Heyman and T. Lakshman. What are the implications of long-range dependence for VBR-video traffic engineering? *IEEE/ACM Transactions on Networking*, 4(3):301–317, June 1996.

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

[29] Philippe Jacquet. Queueing with aggregated up/down arrivals, 1996. Preprint.

[30] R. Jain and S. Routhier. Packet trains—measurements and a new model for computer network traffic. *IEEE J. Select. Areas Commun.*, 4(6):986–995, 1986.

[31] P. Jelenkovic and B. Melamed. Automated TES modeling of compressed video. In *Proc. IEEE INFOCOM '95*, pages 746–752, 1995.

[32] 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.

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

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

[35] B. Lamparter and M. Kalfane. The implementation of PET. Technical Report TR-95-047, International Computer Science Institute, 1995.

[36] A. Lazar and G. Pacifici. Control of resources in broadband networks with quality of service guarantees. *IEEE Network Magazine*, pages 66–73, October 1991.

[37] 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.

[38] 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.

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

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

[41] N. Maxemchuk. Dispersity routing. In *Proc. ICC '75*, 1975.

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

[43] R. Nagarajan and J. Kurose. On defining, computing and guaranteeing quality-of-service in high-speed networks. In *Proc. IEEE INFOCOM '90*, pages 2016–2025, 1992.

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

[45] 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.

[46] A. Parekh and R. Gallager. A generalized processor sharing approach to flow control in integrated services networks: the single-node case. *IEEE/ACM Trans. Networking*, 1(3):344–357, 1993.

[47] A. Parekh and R. Gallager. A generalized processor sharing approach to flow control in integrated services networks: the multiple node case. *IEEE/ACM Trans. Networking*, 2(2):137–150, 1994.

[48] K. Park, G. Kim, and M. Crovella. On the relationship between file sizes, transport protocols, and self-similar network traffic. In *Proc. IEEE International Conference on Network Protocols*, pages 171–180, October 1996.

[49] K. Park, G. Kim, and M. Crovella. On the effect of traffic self-similarity on network performance. In *Proc. SPIE International Conference on Performance and Control of Network Systems*, pages 296–310, 1997.

[50] K. Park, M. Sitharam, and S. Chen. Quality of service provision in noncooperative networks: heterogeneous preferences, multi-dimensional QoS vectors, and burstiness. Technical Report CSD-TR-97-039, Department of Computer Sciences, Purdue University, 1998.

[51] K. Park and T. Tuan. Congestion control for self-similar network traffic. In preparation, 1998.

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

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

[54] Lawrence Perko. *Differential Equations and Dynamical Systems*. Springer-Verlag, 1991.

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

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

[57] B. Ryu and A. Elwalid. The importance of long-range dependence of VBR video traffic in ATM traffic engineering: myths and realities. In *Proc. ACM SIGCOMM '96*, pages 3–14, 1996.

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

[59] M. S. Taqqu, V. Teverovsky, and W. Willinger. Estimators for long-range dependence: an empirical study, 1995. Preprint.

[60] W. Willinger, M. Taqqu, R. Sherman, and D. Wilson. Self-similarity through high-variability: statistical analysis of Ethernet LAN traffic at the source level. In *Proc. ACM SIGCOMM '95*, pages 100–113, 1995.