QoS-Sensitive Transport of Real-Time MPEG Video using Adaptive Redundancy Control [1]

Kihong Park [a,2] Wei Wang [a]

[a] *Network Systems Lab, Department of Computer Sciences, Purdue University, West Lafayette, IN 47907, USA*

Abstract

This paper presents an adaptive end-to-end protocol for quality of service-sensitive transport of real-time MPEG video using packet-level forward error correction in dynamic networks. The objective is to facilitate a user-specified QoS end-to-end—i.e., without special network support—for real-time MPEG video whose timing constraints rule out the use of retransmission-based congestion control and QoS provisioning schemes. The degree of redundancy—overcode in forward error correction—injected into the network is adjusted as a function of network state, decreasing when the network is well-behaved to minimize unnecessary network resource consumption, and increasing when it is not to compensate for adverse network effects so as to maintain an invariant level of end-to-end QoS. We describe an adaptive packet-level FEC protocol called AFEC and analyze its properties with respect to optimality and stability. The optimal 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—MPEG video frames—at the receiver. We experimentally evaluate the efficacy of our end-to-end QoS control by implementing and customizing AFEC to the transport of real-time MPEG video. We realize the system as a transport/application layer protocol running on UNIX workstations, and measure its performance over controlled network environments. We show that AFEC is able to effectively hide potentially adverse network effects such as packet drops and delays stemming from traffic burstiness and nonstationary structural changes using adaptive redundancy control, exporting a constant-QoS service commensurate with user-specified QoS.

Key words: adaptive redundancy control, quality of service, forward error correction, real-time MPEG video

[1] Supported in part by NSF grant ANI-9714707.
[2] Contact author: park@cs.purdue.edu. Additionally supported by NSF grants ANI-9875789 (CAREER), ESS-9806741, EIA-9972883, and grants from PRF, Santa Fe Institute, and Sprint.

1 Introduction

1.1 Motivation

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 [3,28,29]. 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 such as real-time MPEG video 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 [4,6,9,39,51,54]. That is, by the time a lost or damaged MPEG frame is detected and retransmitted, its arrival at the receiver is too late to be useful. 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—in particular, MPEG video—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. The applications are *hard* real-time systems with respect to their quality of service (QoS) contraints either being met or not, but they are *soft* in the sense that a fraction of constraint violations over time can be tolerated.

The effectiveness of FEC is determined by two fundamental factors, one, in having adequate information about a communication channel such that the amount of redundancy or overcode can be appropriately set, and two, in having a system that is resilient to the effects of *traffic burstiness* or *packet loss/delay dependency*. In the former, if a communication channel or—in general—the network state is stationary, then parameters defining the network state can be estimated once and for all and subsequently used to determine the amount of redundancy to be injected into future transmissions. However, in general networks, the network state is far from being stationary with new connections arriving and departing continuously, and structural perturbations including host/link failures and congestion effects amplifying the variability of system state. In the latter, 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 representing an MPEG video frame. 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 [22,25,33] that show that network traffic—including *individual* compressed VBR video sources—exhibit self-similarity and long-range dependence (LRD), and that this is an ubiquitous phenomenon. This is of import to FEC since it has been shown that queueing behavior, under LRD input, can significantly differ from that of Markovian input processes such as Poisson [35,42,46].

1.2 New Contributions

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 adaptive FEC performance, in the context of transporting real-time MPEG video. The second problem is made more difficult by the first problem since dynamically varying redundancy as a function of network state—inclusive queueing at bottleneck routers—creates further dependencies due to feedback. Our contributions are as follows.

First, we construct an adaptive packet-level FEC protocol called AFEC and analyze its properties with respect to optimality and stability. The control problem is nontrivial due to the fact that increased redundancy, beyond a certain point, can "backfire" resulting in self-induced congestion which can impede the timely recovery of information—e.g., MPEG video frames—at the receiver. Instabilities can arise from two distinct sources—the desired operating point and network delay—and we give a solution in terms of the AFEC protocol. A strong suit of AFEC is its dual adaptive redundancy control which is composed of both a *symmetric* and an *asymmetric* component. When a user-specified, target QoS is readily achievable under current network conditions, only the symmetric control component comes into play thus providing smooth feedback adaptivity directed at providing a constant level of receiver perceived QoS. On the other hand, when the target QoS level is too stringent to be achievable given current network conditions, the maximal achievable QoS is rendered by AFEC, and in doing so, is forced to employ exponential backoff—hence asymmetry—in its redundancy control due to the fact that the optimal operating point is unstable. In either case, AFEC is *efficient* in that it uses just enough resources to achieve a user-specified target QoS, and as such, is a *cooperative* protocol in the same sense as TCP.

Second, we experimentally evaluate the efficacy of our end-to-end QoS control by implementing and customizing AFEC to the transport of real-time MPEG video, realizing the system as a transport and application layer protocol called MAFEC running end-to-end on UNIX workstations, and measuring its performance over controlled network environments. We show that MAFEC is able to effectively hide potentially adverse network effects such as increased packet drops and delays stemming from traffic burstiness and nonstationary structural changes using adaptive redundancy control, and thus exports a constant-QoS service commensurate with user-specified QoS. Both the source traffic—i.e., compressed MPEG VBR video—and background traffic are allowed to be bursty and the conclusions advanced by analysis are confirmed and further extended. MAFEC was customized to exploit the peculiarities of its payload—real-time 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 protocol and system, the user is given the option to specify its QoS requirements both at start-up as well as 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 secondary aspect of MAFEC's 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 PET [2,32] where static FEC coding is applied to MPEG video with the highest priority (i.e., redundancy) given to header information then I frames followed by P frames and B frames. This is a straightfoward optimization to carry out due to the uni- and bi-directional dependencies of P and B frames on neighboring I and P frames. Our work is different from PET in its two principal aspects: adaptively varying redundancy as a function of network state as as to minimize unnecessary bandwidth consumption while facilitating QoS-sensitive transport of real-time MPEG video and studying the efficacy of FEC and AFEC under bursty traffic conditions. MAFEC is implemented entirely in software at the transport and application layer without specialized hardware or kernel support.

The remainder of the paper is organized as follows. The next section summarizes related work. In Section 3 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 with respect to both its symmetric control and asymmetric control components. Section 5 describes MAFEC and the details underlying the implementation of AFEC for QoS-sensitive end-to-end transport of real-time MPEG video. Section 6 describes the experimental set-up underlying our performance measurements, and shows performance results that demonstrate the behavior of MAFEC and its efficacy at facilitating end-to-end QoS control. We conclude with a discussion of some high-level issues and future work.

2 Related Work

2.1 Packet-level Forward Error Correction

Previous work in packet-level FEC has concentrated on the issue of implementing efficient error correcting codes at the software level from Reed-Solomon codes [38,39], to variants of polynomial codes over Galois Fields [2,37,51], the fast linear time—albeit probabilistic—tornado code [36], to fragmentation issues in ATM and TCP-over-ATM implementations [4,19], to applications of fast coding to reliable multicast [12].

Various forms of "adaptively" employing packet-level FEC have been explored in [5,8–10,31,44,45]. In [31], an adaptive scheme is proposed which assuming *known*, *independent* packet loss probability computes the overcode 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 dropped—a simple binomial sum due to independence—and is applicable to bit-level FEC under independent-error channel assumption. The main weaknesses are the assumptions on a priori knowledge and independence which does not address *dependency* issues—arising both from traffic correlation and queueing—which are at the heart of *packet-level* FEC. In [5], a fault-tolerant information recovery scheme supporting real-time constraints is proposed using a variant of Rabin's Information Dispersal Algorithm [49]. 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 state and its influence on performance are not considered.

2.2 Adaptive FEC and QoS

In [8–10], an adaptive FEC control is proposed for packet audio where the amount of overcode is calibrated as a function of observed loss rate in the network. The Gilbert channel is used as the loss model where interleaving with increased redundancy results in a nondecreasing recovery rate—i.e., reduced loss after recovery—and a delay constraint representing the maximal block size is incorporated to formulate and solve a constrained optimization problem. The two principal weaknesses of the control framework are, one, the Gilbert channel may serve as a first-order approximation of loss processes for audio traffic which—even in a highly compressed form such as MPEG Audio Layer 3—exhibits limited variability (smoothing over a 2–3 frame window results in near CBR traffic) which is not the case for compressed VBR video, and two, the consequent FEC control framework leads to the unrealistic property that increased redundancy monotonically improves recovery rate. As shown in [44,45], injecting "too much" redundancy where the threshold is a function of network state results in the oppositive effect of decreasing the probability of successful—i.e., timely—decoding and recovery. In [1], the conditions under which redundancy helps/hurts with respect to increased/decreased recovery rate for Markovian input is discussed. It is shown that both conditions coexist. This paper solves the more general and complex redundancy control problem with increasing and decreasing recovery rate conditions in the context of facilitating end-to-end QoS control of real-time traffic.

In [53], a method for designing an integrated control encompassing an error-resilient coder and a QoS-aware transport protocol is introduced and studied. Based on a Hidden Markov Model (HMM) assumption of the network channel, the channel state is estimated and used to dynamically control the number of packets allocated to encode a GOP. As with [10,8], the essential difference with our work, from a control-theoretic perspective, is that the unimodal property of the redundancy-recovery relation is not explicitly incorporated which imparts a different twist to the optimal control problem. The HMM approach to modeling network traffic, although powerful as a modeling construct assuming a sufficient number of states, is somewhat outdated give the empirically verified self-similar nature of Internet traffic, its well-understood physical basis, and performance implications [46]. At a more abstracted, middleware support oriented level is the work by Li and Nahrstedt [34] which advances an adaptive middleware architecture whose key features follow a control-theoretic framework to achieving QoS control. An important characteristic of the middleware control framework is its generality—i.e., application-independence assuming QoS requirements can be expressed in the language of "resource requests"—which is achieved by separating the controller from the application by a well-defined interface. In both works, local analysis is used to show conditions for asymptotic stability of the underlying control system at target operating points. The control systems, however, are different in that unimodality of redundancy-recovery injects additional nonlinear destabilizing influence whose analysis, solution, and implementation is the principal control-theoretic contribution of this paper in the context of adaptive redundancy control [3].

[3] Different control problems, even in the context of QoS, can have varying properties that prevent application of an universal control. For example, adaptive label

2.3 FEC and Queueing

The effect of burstiness on FEC performance has been studied in a seminal paper [6] in the context of ATM networks. A *system* approach is undertaken 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 is a well-known fact that *bit-level* FEC is ineffective for dependent-error channels [21], 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 additional structure due to the fact that dependencies can stem from queueing effects as well as the characteristics of the traffic source itself. In [54] 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.* [15,16], albeit excluding LRD input processes. In most recent work [57], we have extended adaptive redundancy control in the framework of multiple time scale traffic control [56] to exploit large time scale predictability structure of self-similar traffic which imparts additional proactivity. An application to TCP congestion control can be found in [43].

3 Network Model

3.1 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, so as to preserve a constant level of rendered QoS at the receiver while minimizing network resource usage. 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

control [50] which shapes end-to-end QoS over differentiated services networks is an intrinsically simpler problem than adaptive redundancy control or congestion control with unimodal performance curves.

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.

Assume k data packets representing an MPEG video frame are encoded as $n = k + h$ packets where $h \geq 0$ is the degree of redundancy. We will assume that the receipt of *any* k packets out of the n total packets suffices to recover the original k data packets and hence the corresponding video frame. Of course, if the MPEG frame in question is a P or B frame and its reference frame(s) is missing then recovery can still fail at the MPEG decoding stage. FEC encoding/decoding functions with the k-out-of-n property exist (e.g., Reed-Solomon [38], IDA [49]). In Section 5 we will describe an implementation of packet-level FEC based Rabin's Information Dispersal Algorithm [49] customized to MPEG video frames. Figure 1 shows a depiction of packet-level FEC in an end-to-end network environment.

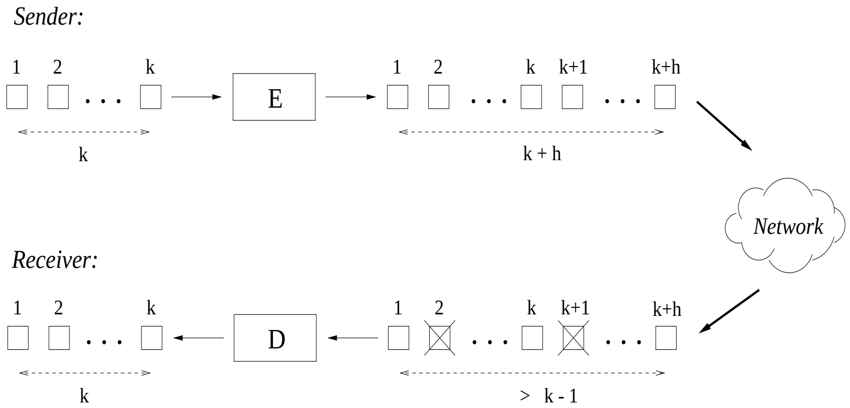


Fig. 1. A block of k packets representing an MPEG video frame are encoded at the sender using FEC as $k + h$ packets. If the number of dropped (or significantly delayed) packets does not exceed h, the original k data packets—and hence the corresponding video frame—can be recovered at the receiver.

3.2 Real-time Constraints

The application in question—real-time MPEG video—is *real-time constrained* in the following sense. The sender transmits

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

blocks of packets 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. For simplicity of exposition [4], assum-

[4] The actual system works with variable size packets but this feature is inessential to the discussion at hand.

ing fixed-size packets, $k(t_i)$ represents the number of packets needed to encode the i'th MPEG video frame—I, P, or B—which are of variable length. 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 packets belonging to the i'th block must be recovered before time t'_i. That is, late arrivals belonging to block i carry zero utility. For instance, given a frame rate of 30 fps, successive frames must arrive within 33.3 ms periodic intervals with some provision for decoding processing cost at the receiving end station.

QoS is measured at the receiver using a *recovery rate* process $\gamma(t'_i)$. It is defined to be the number of packets 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 of the i'th frame 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 packets 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—in addition to real-time MPEG video—including those with periodic constraints as well as interactive applications with aperiodic constraints.

4 Structure of AFEC Protocol

4.1 AFEC Protocol: Symmetric Redundancy Control

4.1.1 Unimodal Redundancy-recovery Relation

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 [5]

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

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

[5] We use continuous notation for expositional simplicity; their discrete counterparts are straightforward. In practice, h is maintained as a continuous variable but just

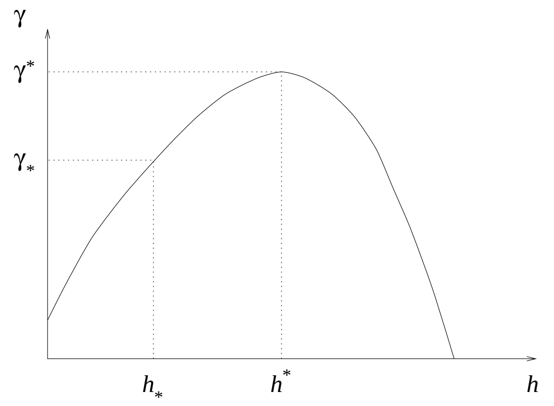


Fig. 2. Unimodal redundancy-recovery rate function $\gamma = \mathcal{G}(h)$ with optimal recovery rate γ^* and target recovery rate γ_* (here $\gamma_* < \gamma^*$).

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 and network contention, long end-to-end delays, and a host of other factors.

4.1.2 Symmetric QoS Control

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) 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 $(k(t_i))_{i \in \mathbb{Z}_+}$ corresponding to compressed VBR video—we yield a companion control law for (1) in the form of

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

---

before passing it to a FEC encoder, it is discretized.

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 packets have made it through to achieve recovery. Under stochastic network conditions such as when the variance of cross traffic 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 packet 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 $t = t_i$, $k(t_i) + h(t_i)$ packets 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.

4.1.3 Stability Analysis

Two forms of instability can arise, one from the value of the desired operating point (h_*, γ_*), and another stemming from the delay term τ. 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.

Operating Point Instability 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 [48] 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. Taylor expansion of the right-hand-side (RHS) around $h = h_*$ yields

$$\frac{dh}{dt} \;=\; c - \epsilon\, \mathcal{G}'(h_*)\, h \;+\; \text{higher order terms} \tag{3}$$

where $c \geq 0$ is a constant and $\mathcal{G}'(h_*) \equiv (d\mathcal{G}/dh)|_{h=h_*}$. 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: the symmetric control is asymptotically stable at (h_*, γ_*) if $\mathcal{G}'(h_*) > 0$. By the unimodality of $\mathcal{G}$, this immediately gives us the corollary that the control is asymptotically stable at $(h_*, \gamma_*) \in H_L$ and unstable at $(h_*, \gamma_*) \in H_R$ where

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

In particular, the "optimal" operating point (h^*, γ^*), from the perspective of maximum recovery rate, is unstable. Figure 3 (left) shows the phase portrait in the local neighborhood of (h^*, γ^*); we observe that for $h > h^*$ the flow diverges. Figure 3 (right) shows the corresponding phase portrait for $(h_*, \gamma_*) \in H_L$. We observe that the flow in the neighborhood of (h_*, γ_*) is convergent. In the case of the rest of the unstable set, $(h_*, \gamma_*) \in H_R \setminus \{(h^*, \gamma^*)\}$, the flow is again divergent for $h > h_*$, analogous to Figure 3 (left) which is omitted.

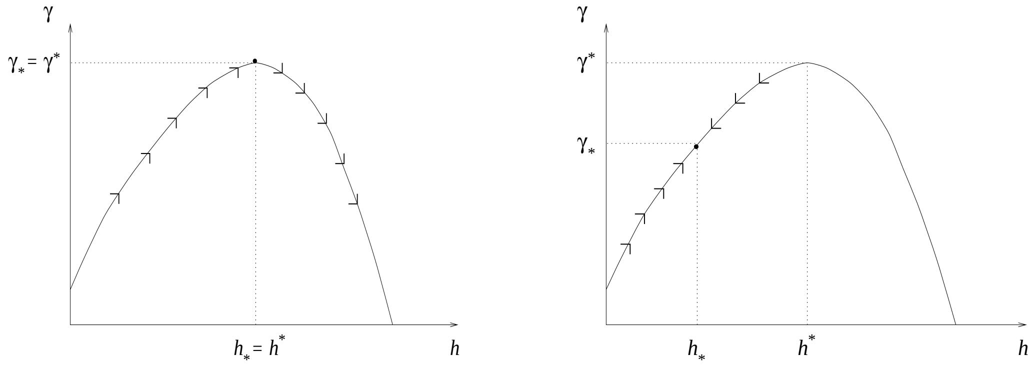


Fig. 3. Left: Phase portrait around the unstable fixed point $\gamma = \gamma^*$. Right: Phase portrait around the stable equilibrium $\gamma = \gamma_* < \gamma^*$.

Delay Instability We return to the problem of incorporating the effect of network delay as captured by $\tau > 0$. The presence of network delay, in general, can lead to oscillatory behavior even for $(h_*, \gamma_*) \in H_L$. 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 boundedness on $\mathcal{G}$. The stability of the linearized system in the neighborhood of $(h_*, \gamma_*) \in H_L$ is determined by the stability of the reduced linear system

$$\frac{dh(t)}{dt} = -\epsilon\, \mathcal{G}'(h_*) h(t - \tau). \tag{4}$$

A linear delay differential equation of the form $dx(t)/dt = -ax(t - \tau)$, $a > 0$, has a sinusoidal solution [24] which is determined by the complex roots of the characteristic equation $\xi + ae^{\xi\tau} = 0$. The system is asymptotically stable if for all roots of the characteristic equation, the real parts lie in the negative half plane. The latter, in turn, is related to the relative magnitude of a and τ. By a change of variables, we further reduce (4) to a normal form where $\tau = 1$.

This is useful because the stability condition of $dx(t)/dt = -ax(t-1)$ is given by $0 < a < \pi/2$. Let $s = t/\tau$. Then,

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

from which we get

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

where we have redefined $\tilde{h}(s) \equiv h(\tau s)$. Thus we arrive at the asymptotic stability condition of the original nonlinear system

$$0 \;<\; \tau\epsilon\,\mathcal{G}'(h_*) \;<\; \frac{\pi}{2}. \tag{5}$$

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.

4.2 AFEC Protocol: Asymmetric Redundancy Control

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 [47]. Related results for stability analysis of congestion control include [11,20,30,40,55]. We refer the reader to [58] for a classification and taxonomy of congestion controls. Whereas in congestion control the natural goal is to achieve maximum throughput or 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. Note that $d\gamma/dh > 0$ iff $(h, \gamma) \in H_L$. The augmented AFEC algorithm containing *both* the symmetric and asymmetric control components is given by

$$\frac{dh(t)}{dt} = \begin{cases} \epsilon(\gamma_* - \gamma(t-\tau)), & \text{if } d\gamma(t-\tau)/dh > 0, \\ -ah, & \text{otherwise.} \end{cases} \tag{6}$$

Here $a > 0$ is a positive constant. Hence (6) follows (1), as it should, when $h < h^*$—i.e., $(h, \gamma) \in H_L$—and it performs drastic, asymmetric backoff when $h \geq h^*$ (i.e., $(h, \gamma) \in H_R$). The backoff mechanism, $dh/dt = -ah$, is *exponential* leading to a decay of h of the form $O(e^{-at})$. Implementing directional detection is straightforward; 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 $sgn\,(d\gamma(t-\tau)/dh)$.

5 Implementation of AFEC for Real-time MPEG Video

5.1 System Description of MAFEC

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. The next sections describe the architecture of the system and its control components.

5.1.1 Overall Structure

MAFEC—**AFEC** for **M**PEG video—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 (by default 25 frames per second). The FEC encoder applies forward error correction on each frame, first, by segmenting it into k 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 an MPEG II player which applies its own decoding to produce the

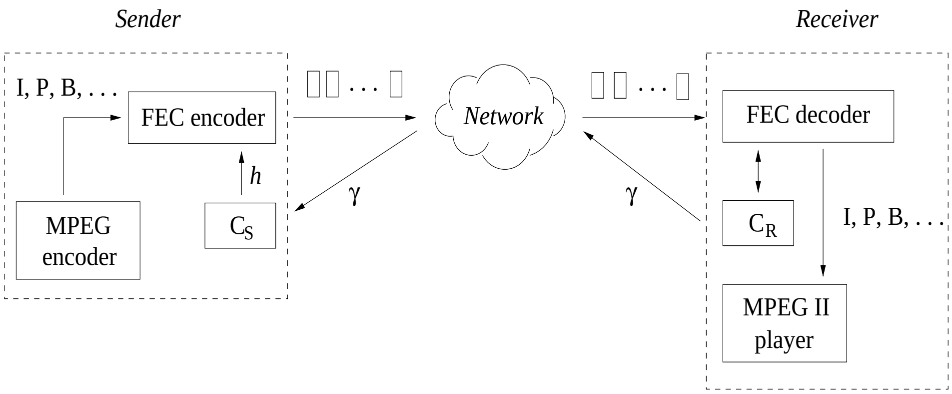


Fig. 4. Overall structure of MAFEC. At the sender, I, P, B frames forwarded from an MPEG encoder are fed into the FEC encoder. The latter is controlled by a sender-side controller C_S. At the receiver, the coded packets are fed into the FEC decoder which is controlled by a receiver-side controller C_R. The decoded I, P, B frames are forwarded to an MPEG-II player.

uncompressed frames which are then rendered on the terminal as part of the video stream. This forward flow structure is depicted in Figure 4. 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 with packets transported using UDP, and it is completely end-to-end such that the system can be deployed over any IP network.

5.1.2 Forward Error Correction using IDA

Encoding We use Rabin's IDA [49] as the forward error correction code. IDA is a simple, space-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

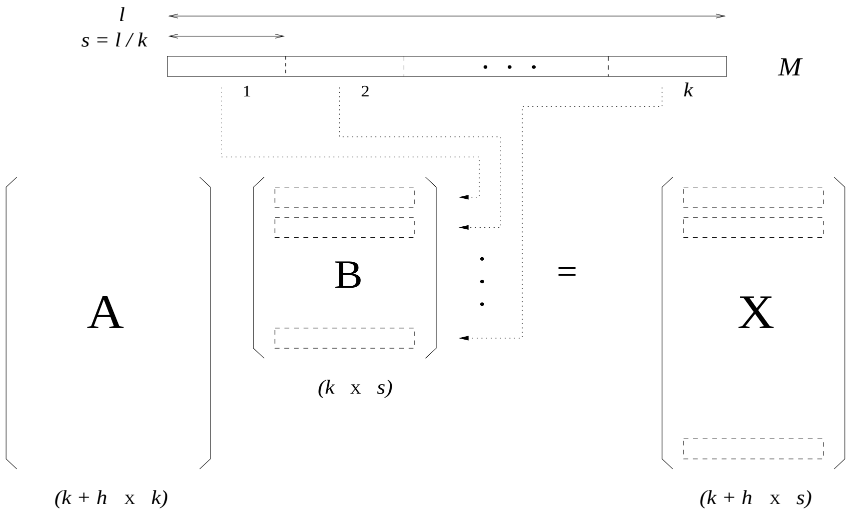


Fig. 5. IDA encoding process: Frame or message M (of length ℓ) is segmented into k pieces giving $\mathbf{B}$. Matrix $\mathbf{B}$ is then multiplied with $\mathbf{A}$ to give the code matrix $\mathbf{X}$.

code vectors are obtained by computing

$$\mathbf{X} = \mathbf{A}\,\mathbf{B} \tag{7}$$

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 (7) 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 5.

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 (7) and the independence property of $\mathbf{A}$,

$$\mathbf{A}_k^{-1}\,\mathbf{X}_k = \mathbf{B} \tag{8}$$

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 is well-behaved and packets arrive at the end station in a timely fashion. Thus when implementing MAFEC in software, in real-time, on generalpurpose 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 in reducing the decoding processing cost. 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 effective policy.

5.1.3 Receiver-Oriented QoS Control

The adaptive redundancy control algorithm employed for MPEG video transport is fundamentally the same as the one described in Section 4. 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 4) to indicate its *desired* QoS level, in particular, the target hit rate ν_* ($0 \leq \nu_* \leq 1$). This is then used as a *parameter* by the AFEC control algorithm when making its adjustments. Although ν_* is, by default, held constant after start-up, the user is given the option of changing ν_* while the application is in progress via a "tuning knob" that changes 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 structurally 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.

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 target 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 by the user.

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 uni- and bi-directional dependencies of the latter. Indeed, this is the main observation that underlies the PET system by Albanese *et al.* [2,32] 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 by 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.

6 Performance Measurements

The next sections describe the physical performance evaluation environment and performance measurements from benchmark experiments.

6.1 Experimental Set-up

6.1.1 Network Configuration

Our experiments were carried out in the test environment shown in Figure 6. 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 substituted by an emulator that fed the frames of *stored* MPEG I videos at real-time frame rates (by default 25 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. Similar results are obtained when running the sender on a `x86` PC equipped with an Optibase real-time MPEG I video/audio encoder under Windows NT, connected to a Sony DCR-VX 1000 digital video camera. The MPEG II player at the receiver was always run, and the performance measurements reflect the overhead incurred by this and FEC decoding processing overhead. Thus modulo *where* the I, P, B frames came from at the sender-side—video compression board or stored video file paced at real-time speed—the system depicted in Figure 4 was faithfully implemented with all the components implemented in software without specialized hardware support.

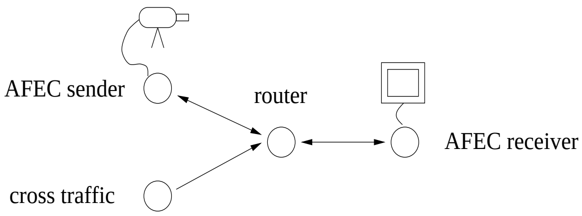


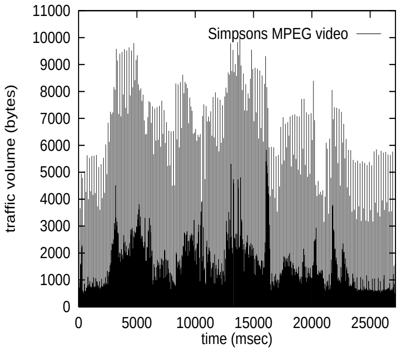
Fig. 6. Experimental set-up for MAFEC performance measurements.

The FEC-controlled application—a pair of processes, 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 $T > 0$ (by default 40 ms), are generated at A, and the $k(t_i)$ data packets must be recovered at D before time t'_i where $t'_{i+1} = t'_i + T$. In the MAFEC context, we have a video-conferencing application sending compressed (i.e., VBR) video real-time 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.

The topology depicted in Figure 6 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. Without the configurable router, we found that the 100 Mbps bandwidth of FastEthernet was too large relative to the data rate of MPEG I video ($\sim$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/drop tail packet 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 an ATM LAN as well as over non-LAN internetwork environments with favourable results. In the latter, due to the non-controlled nature of the test environment, the results are context dependent, not systematic, and omitted here. In the following results, the adjustment constant ϵ of the symmetric component of AFEC is set to 0.1, and the asymmetric backoff constant of AFEC is set to $a = 2$ (binary exponential backoff).

6.1.2 Benchmark Data

We emphasize that the performance results are obtained using actual MPEG I video data as opposed to frame size traces commonly used in 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 dealt with by the dynamic control mechanism described in Section 5.1.2.



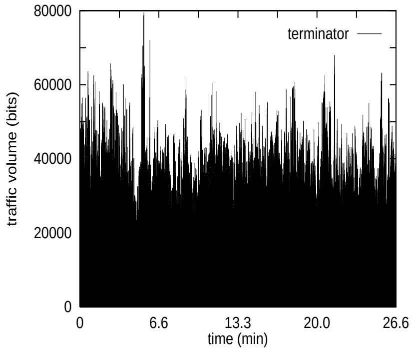


Fig. 7. Traffic series of a Simpsons MPEG I video clip (left) and Terminator clip (right).

Figure 7 (left) 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. Figure 7 (right) shows the trace of the Terminator clip which has a much longer playing time ($\sim$30 mins). They yield similar performance results as the Simpsons clip and are not shown separately. The MPEG video clips used have a Hurst parameter estimate in the $\sim$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 [22,23,25,52]. 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 [27]. What is commonly accepted in the midst of this diversity is that compressed video is "bursty," much more so than Poisson traffic.

6.2 *Performance Results*

6.2.1 *Unimodal Redundancy-recovery Relation*

Figure 8 (left) shows the measured redundancy-recovery function for the Simpsons MPEG I 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). We observe that the empirical—i.e., measured—redundancy recovery function is *unimodal* consistent with the expected shape of $\mathcal{G}$.

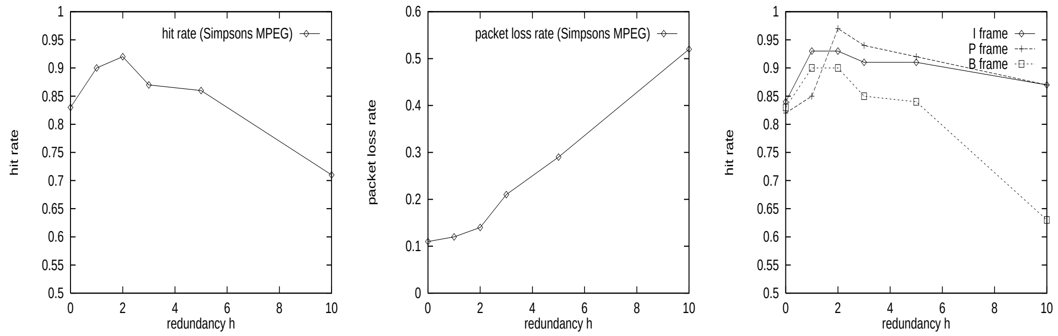


Fig. 8. Left: Measured unimodal redundancy-recovery function for Simpsons MPEG I clip. Middle: Corresponding packet loss rate function. Right: Measured redundancy-recovery function for Simpsons MPEG clip partitioned for I, P, B frames.

Figure 8 (middle) shows the corresponding packet loss rates which, as expected, are monotonically increasing. Figure 8 (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 in this sequence, and when h is high, commensurately suffers to a greater extent.

6.2.2 *AFEC vs. Static FEC*

Structural Adaptivity Figure 9 (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 9 (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 9 (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 $[300, 600]$ 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—h increases during the $[300, 600]$ interval—even though a *higher* packet loss is being incurred during the same period. The white miss line in Figure 9 (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.

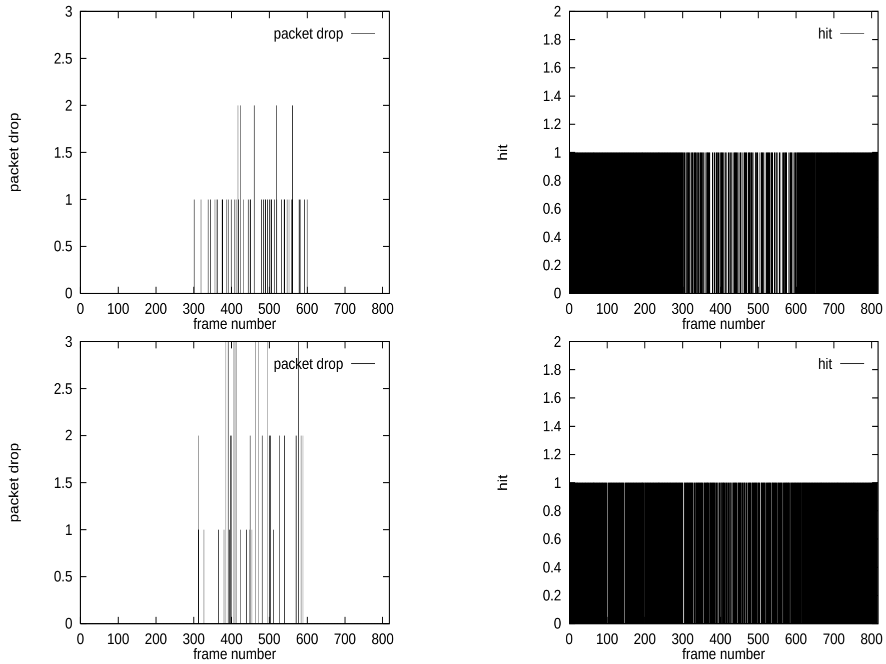


Fig. 9. Static FEC vs. AFEC run when cross traffic is active in over interval $[300, 600]$ for Simpsons MPEG clip. To row: Packet drop trace (left) and corresponding hit trace (right) for static FEC with $h = 0$. Bottom row: Same as above but with AFEC active.

Figure 9 shows 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 9 (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. If h were set to 2 or 3, then QoS would remain minimally affected during the active cross traffic period, however, during the periods when cross traffic is inactive, bandwidth would be unnecessarily wasted.

Structural Instability Figure 10 (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 4).

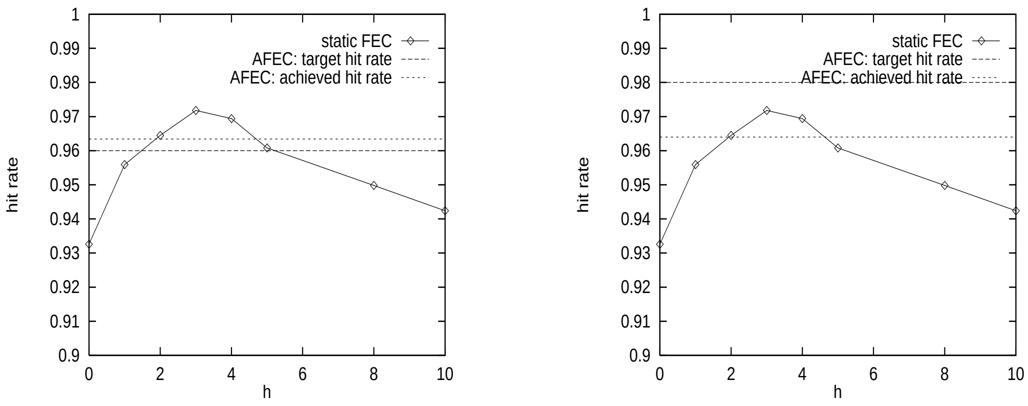


Fig. 10. Left: AFEC hit performance vs. static FEC performance when target hit rate is 0.96. Right: AFEC hit performance vs. static FEC performance when target hit rate is 0.98.

Figure 10 (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 target operating point will cause AFEC to overshoot—recall that AFEC will continuously attempt to attain the desired hit rate—which then triggers an exponential backoff of the asymmetric control component 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.

6.2.3 AFEC Dynamics

Behavior in the Stable Regime Figure 11 (top) shows the average hit rate, hit, and redundancy trace for an AFEC run for the Simpsons MPEG I 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 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 11 (top, middle). Figure 11 (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 11 (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 11 (bottom, middle)—shows no marked differences with its counterpart in the top row except for a concentrated bad period near frame 400.

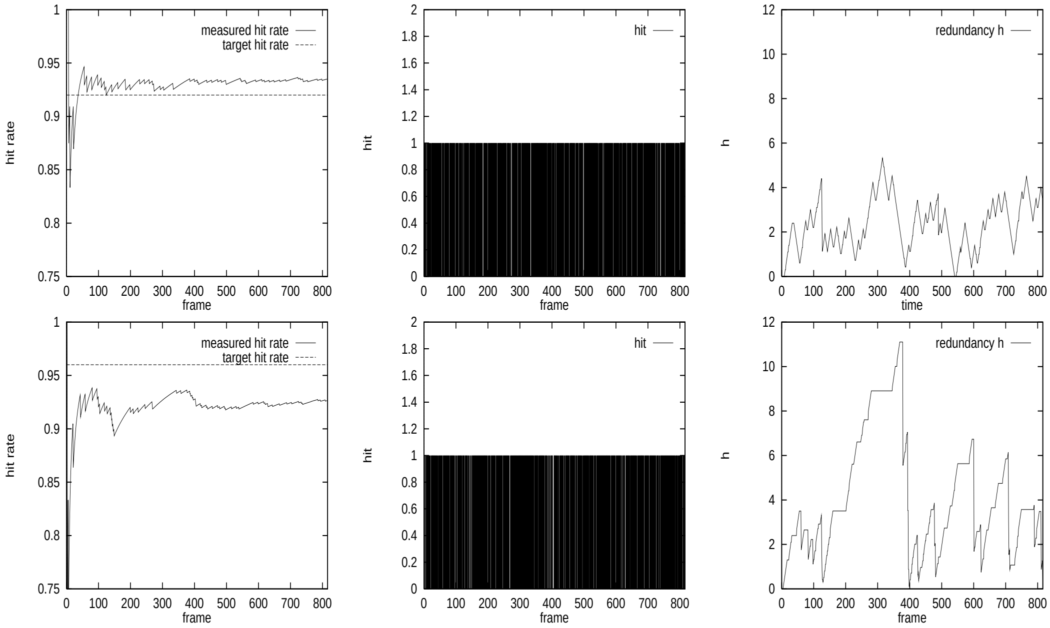
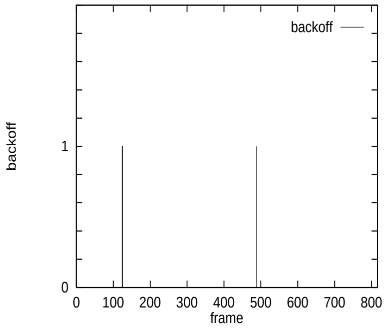


Fig. 11. AFEC dynamics. Top row: Target hit rate 0.92. Measured hit rate (left), hit trace (middle), h trace as a function of time (expressed as frame number). Bottom row: Target hit rate 0.96. Measured hit rate (left), hit trace (middle), h trace as a function of frame number.

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 11 (bottom, right) shows a much wider range in h than its corresponding trace Figure 11 (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 (6))—can be more clearly seen in Figure 12 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 12 (left)) only two exponential backoff instances occur, one just after frame 100, and another just before frame 500. Looking carefully at Figure 11 (top, right), one can observe a miniature version of the stairway-to-abyss shape at the corresponding points.

Figure 12 (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 Figure 11 (bottom, right). We emphasize that in MAFEC, 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 (6)) 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.



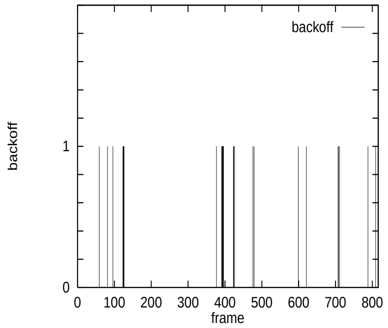


Fig. 12. Left: Backoff instances for AFEC run with target hit rate 0.92. Right: Backoff instances for AFEC run with target hit rate 0.96.

7 Conclusion and Discussion

We have presented a study of the packet-level adaptive forward error correction problem for the QoS-sensitive transport of real-time MPEG video end-to-end. We have studied the performance characteristics of MAFEC under bursty traffic conditions and structural perturbations, and we have shown the efficacy of AFEC vis-à-vis static FEC as well as its dynamical properties. 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. We expect our end-to-end QoS amplification control to be useful in DiffServ networks [7,17,41] where the services exported by IP—given its “soft” nature due to weak inter-flow protection—can be further fortified with respect to their QoS properties using end-to-end control.

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 [6]. 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 6.2.3)—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 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 [13,14]. In most recent work [57], we have extended AFEC to the multiple time scale traffic control framework [56]—a novel workload-sensitive traffic control paradigm—where, in addition to AFEC's adaptive feedback control, long-range correlation structure in self-similar traffic is explicitly exploited to affect further performance improvement on top of AFEC.

References

[1] O. Ait-Hellal, E. Altman, A. Jean-Marie, and I. Kurkova. On loss probabilities in presence of redundant packets and several traffic sources. *Performance*

[6] Our discussion will assume TCP Tahoe as the reference point as defined by its principal features given in [26].

Evaluation, 36:485–518, 1999.

[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.

[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] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated service. RFC 2475, 1998.

[8] J. Bolot, S. Fosse-Parisis, and D. Towsley. Adaptive FEC-based error control for Internet telephony. In *Proc. IEEE INFOCOM '99*, pages 1453–1460, 1999.

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

[10] J. Bolot and A. Garcia. Control mechanisms for packet audio in the Internet. In *Proc. IEEE INFOCOM '96*, pages 232–239, 1992.

[11] J. Bolot and A. Shankar. Analysis of a fluid approximation to flow control dynamics. In *Proc. IEEE INFOCOM '92*, pages 2398–2407, 1992.

[12] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A digital fountain approach to reliable distribution of bulk data. In *Proc. ACM SIGCOMM '98*, 1998.

[13] S. Chen and K. Park. A distributed protocol for multi-class QoS provision in noncooperative many-switch systems. In *Proc. IEEE International Conference on Network Protocols*, pages 98–107, 1998.

[14] S. Chen and K. Park. An architecture for noncooperative QoS provision in many-switch systems. In *Proc. IEEE INFOCOM '99*, pages 864–872, 1999.

[15] 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.

[16] 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.

[17] D. Clark and W. Fang. Explicit allocation of best-effort packet delivery service. *IEEE/ACM Trans. Networking*, 6(4):362–373, 1998.

[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] K. Fendick, M. Rodrigues, and A. Weiss. Analysis of a rate-based control strategy with delayed feedback. In *Proc. ACM SIGCOMM '92*, pages 136–148, 1992.

[21] 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.

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

[23] 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.

[24] J. Hale and S. Verduyn Lunel. *Functional Differential Equations.* Springer-Verlag, 1993.

[25] 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.

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

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

[28] 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.

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

[30] S. Keshav. A control-theoretic approach to flow control. In *Proc. ACM SIGCOMM '91*, pages 3–15, 1991.

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

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

[33] 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.

[34] B. Li and K. Nahrstedt. A control-based middleware framework for quality of service adaptation. *IEEE J. Select. Areas Commun.*, 17(9):1632–1650, 1999.

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

[36] M. Luby, M. Mitzenmacher, A. Shokrollahi, D. Spielman, and V. Stemann. Practical loss-resilient codes. In *Proc. ACM STOC '97*, 1997.

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

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

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

[40] A. Mukherjee and J. Strikwerda. Analysis of dynamic congestion control protocols - a Fokker-Planck approximation. In *Proc. ACM SIGCOMM '91*, pages 159–169, 1991.

[41] K. Nichols, V. Jacobson, and L. Zhang. A two-bit differentiated services architecture for the Internet. Internet Draft, 1997.

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

[43] K. Park and T. Tuan. Performance evaluation of multiple time scale TCP under self-similar traffic conditions. To appear in *ACM Transactions on Modeling and Computer Simulation*, 2000.

[44] K. Park and W. Wang. AFEC: an adaptive forward error correction protocol for end-to-end transport of real-time traffic. In *Proc. IEEE IC3N*, pages 196–205, 1998.

[45] K. Park and W. Wang. QoS-sensitive transport of real-time MPEG video using adaptive forward error correction. In *Proc. IEEE Multimedia Systems '99*, pages 426–432, 1999.

[46] K. Park and W. Willinger, editors. *Self-Similar Network Traffic and Performance Evaluation*. Wiley-Interscience, 2000.

[47] Kihong Park. Warp control: a dynamically stable congestion protocol and its analysis. In *Proc. ACM SIGCOMM '93*, pages 137–147, 1993.

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

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

[50] H. Ren and K. Park. Toward a theory of differentiated services. In *Proc. IEEE/IFIP International Workshop on Quality of Service*, pages 211–220, 2000.

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

[52] 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.

[53] Sergio Servetto. *Compression and Reliable Transmission of Digital Image and Video Signals*. PhD thesis, University of Illinois at Urbana-Champaign, 1999.

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

[55] Scott Shenker. A theoretical analysis of feedback flow control. In *Proc. ACM SIGCOMM '90*, pages 156–165, 1990.

[56] T. Tuan and K. Park. Multiple time scale congestion control for self-similar network traffic. *Performance Evaluation*, 36:359–386, 1999.

[57] T. Tuan and K. Park. Multiple time scale redundancy control for QoS-sensitive transport of real-time traffic. In *Proc. IEEE INFOCOM '00*, 2000.

[58] C. Yang and A. Reddy. A taxonomy for congestion control algorithms in packet switching networks. *IEEE Network*, 9:34–45, July/August 1995.