QoS-Sensitive Transport of Real-Time MPEG Video using Adaptive Forward Error Correction*

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

Abstract

This paper presents an adaptive end-to-end protocol for quality of service (QoS)-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 traffic whose timing constraints rule out the use of retransmission-based congestion control and QoS provisioning schemes.

In previous work, we proposed an adaptive packet-level FEC protocol called AFEC and analyzed 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 impedes the timely recovery of information (e.g., MPEG video frames) at the receiver.

In this paper, we extend the previous work 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 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, desired QoS.

*Supported in part by NSF grant ANI-9714707. This is an extended abstract.
†Contact author. Additionally supported by NSF grant ESS-9806741, and grants from PRF and Sprint.

1. Introduction

Motivation Forward error correction (FEC) is a reliable communication technique which has been successfully used, primarily at the *bit-level*, in a number of application domains from space communication to data storage on compact disks [2, 10, 11]. 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 ARQ mechanism's inherent inadequacy at handling timing constraints when subject to packet drops and large end-to-end network latencies [4, 5, 14, 19, 20]. 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.

Previous research has concentrated on the issue of implementing efficient error correcting codes from Reed-Solomon codes [13, 14] to other variants of polynomial codes [1, 12, 19], to ATM and TCP/IP-over-ATM implementation issues [3, 8]. 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. For the real-time applications we aim to support, delayed packet arrivals that violate the receiver's timing constraints are 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 over time can be tolerated.

The effect of burstiness on FEC performance has been studied in [4] in the context of ATM networks. It is a well-known fact that *bit-level* FEC is ineffective for dependent-error channels [9], rendering ARQ-based or hybrid schemes more effective in such situations with respect to achieving high throughput. Only recently have studies been carried out for *packet-level* FEC which possess further structure due to the fact that dependencies can stem from *queueing effects* as well as the characteristics of the traffic source itself. In [20], 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. [6, 7], albeit excluding LRD input processes.

New Contributions In previous work [16, 17], we proposed an adaptive packet-level FEC protocol called AFEC and analyzed 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 impedes the timely recovery of information at the receiver.

In this paper, we extend our previous work 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, thus exporting a constant-QoS service commensurate with user-specified, desired QoS.

A strong suit of MAFEC is its dual adaptive redundancy control which is composed of both a symmetric and asymmetric component. When a user-specified, desired video quality 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 MAFEC, 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, MAFEC never uses more network resources than is needed to achieve a user-specified target QoS, and as such, is a *cooperative* protocol in the same sense as TCP.

2. Network Model

Packet-Level FEC 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 this property include Reed-Solomon [13] and IDA [18].

Real-Time Constraints The application in question—real-time MPEG video—is *real-time constrained* in the following sense. The sender transmits $n(t_1), n(t_2), \ldots, n(t_i), \ldots$ ($t_i < t_j$ 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. $k(t_i)$ represents the number of packets needed to encode the i'th MPEG video frame—I, P, or B—which are of variable length.

We 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. For instance, given a frame rate of 30fps, successive frames must arrive within 33.3ms periodic intervals with some provision for decoding overhead at the receiving end station. QoS is measured at the receiving end 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.

3. Overview of AFEC Protocol

3.1. Symmetric Redundancy Control

Unimodal Redundancy-Recovery 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 $\gamma(t+\tau)$, $\mathcal{G}(h) \geq 0$ 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^*), $\gamma^* = \mathcal{G}(h^*)$. If $\gamma^* < k$, then there is a structural problem and no amount of redundancy, small or large, can yield a positive hit rate.

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

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

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

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 (3.1) in the form of $dh(t)/dt = \epsilon'(\nu_* - \nu(t-\tau))$ where $\nu_* \ge 1$, $\epsilon' > 0$ is an adjustment parameter, and $\nu(t-\tau) = \gamma(t-\tau)/k(t_i)$ for i such that $t'_{i-1} < t-\tau \le t'_i$. That is, i is the block number with respect to which the recovery rate $\gamma(t-\tau)$ was computed at the receiver. 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 in time to achieve recovery. The gap $\nu_* - 1$ is a measure of conservativeness.

Stability Since the above control laws are related via $\epsilon = \epsilon'/k$, when analyzing the behavior of the system under average application arrival rate conditions, it suffices to study the dynamics of (3.1). Let $H_L = \{(h,\gamma) : h = \mathcal{G}(h),\ h < h^*\}$, $H_R = \{(h,\gamma) : h = \mathcal{G}(h),\ h \ge h^*\}$. In [16, 17] we show that target operating points belonging to $(h_*, \gamma_*) \in H_L$ are asymptotically stable whereas those belonging to $(h_*, \gamma_*) \in H_R$ are unstable.

3.2. Asymmetric Redundancy Control

Instability ensues when "too much" redundancy is applied at the sender which is tantamount to shooting oneself in the foot. To achieve stability [15], we augment the control law given in (3.1) via a *directional check* given by the sign of $d\gamma/dh$. If $d\gamma/dh > 0$, then the system is in the stable region ($h < h^*$) and (3.1) is applied as usual. If, on the other hand, $d\gamma/dh \le 0$, then the system finds itself in the unstable region ($h \ge h^*$) 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

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

Here $a > 0$ is a positive constant. Hence the augmented control follows (3.1), as it should, when $h < h^*$ (i.e., $(h,\gamma) \in H_L$), and it performs drastic, asymmetric backoff only when $h \ge h^*$ ($(h,\gamma) \in H_R$). The backoff mechanism, $dh/dt = -ah$, is *exponential* leading to a decay of h of the form e^{-at}.

4. MAFEC

4.1. 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. 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.

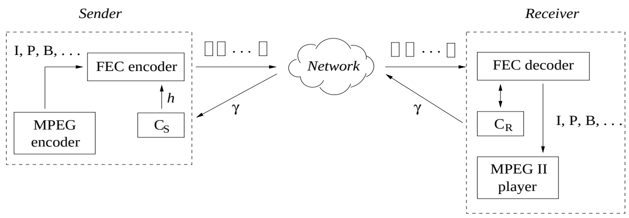

Figure 4.1. Overall structure of MAFEC.

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 uncompressed frames which are then rendered on the terminal as part of the video stream. This forward flow structure is depicted in Figure 4.1.

The receiver-side controller, when processing incoming packets and determing timeliness 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.

4.2. Forward Error Correction using IDA

Encoding We use Rabin's IDA [18] as the forward error correction code. IDA is a simple, efficient error correcting code over Galois fields where encoding and decoding involve matrix multiplication and inversion, 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 $s = \ell/k$. 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. Such matrices are easy to precompute once and for all. The code vectors are obtained by computing $\mathbf{X} = \mathbf{A}\,\mathbf{B}$ where the rows of $\mathbf{X}$—$\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$—constitute the code vectors.

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 the independence property of $\mathbf{A}$, $\mathbf{A}_k^{-1}\,\mathbf{X}_k = \mathbf{B}$, and the original message M (represented as $\mathbf{B}$) is recovered.

Processing Overhead We use Gaussian elimination over the finite field $\mathbb{Z}_p$ to compute $\mathbf{A}_k^{-1}$. 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. 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.

4.3. Receiver-Oriented QoS Control

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. As the application starts up, the user interacts with the receiver-side control module C_R 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." If AFEC determines through its instability detection mechanism (i.e., $d\gamma/dh < 0$) 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.

Stringency Control The *stringency parameter* $W \geq 0$ is a local time window over which the measured recovery rate is computed by taking the time average. 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.

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. Indeed, this is a central observation that underlies the PET system of Albanese et al. [1] 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.

5. Performance Measurements

5.1. Experimental Set-Up

Network Configuration Our experiments were carried out in the test environment shown in Figure 5.1. MAFEC, consisting of the AFEC sender and receiver, sent its traffic over a router where cross traffic stem-

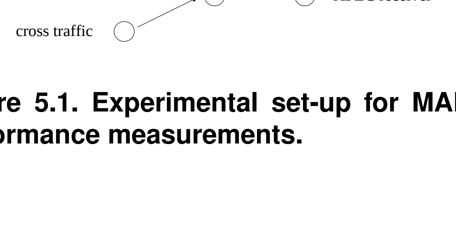

Figure 5.1. Experimental set-up for MAFEC performance measurements.

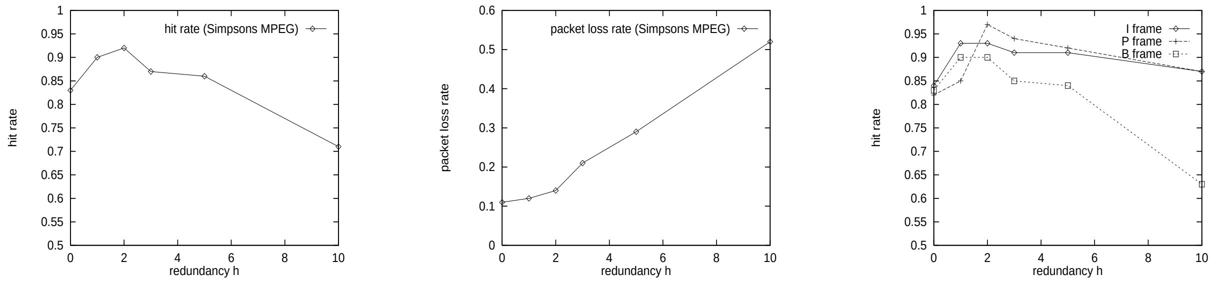

Figure 5.2. Left: Measured unimodal redundancy-recovery function for Simpsons MPEG I clip. Middle: Corresponding packet loss rate function. Right: Measured redundancy-recovery function for Simpsons MPEG clip partitioned for I, P, B frames.

ming 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 I videos at real-time frame rates 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 and FEC decoding processing overhead.

The topology depicted in Figure 5.1 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. 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. MAFEC was also tested over an ATM LAN network as well as over IP internetwork environments with favourable results.

Benchmark Data The performance results are obtained using *actual* MPEG I video data rather than frame size traces. Thus, the cost of FEC encoding and decoding, control actions, and MPEG II player's processing overhead are all reflected in the performance results. Benchmark instances tested include Simpsons cartoon, Red's Nightmare, and Terminator MPEG I video clips. For example, the Simpsons cartoon video 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 $IBBPBB$. 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.

5.2. Performance Results

5.2.1 Unimodal Redundancy-Recovery

Figure 5.2 (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). The observed function is *unimodal* with peak at $h = 2$.

Figure 5.2 (middle) shows the corresponding packet loss rates which, as expected, are monotonically increasing. Figure 5.2 (right) shows the redundancy-recovery relation for I, P, and B frames individually. As with the aggregate redundancy-recovery function, the shapes are unimodal with the function for B frames exhibiting the most curvature.

5.2.2 AFEC vs. Static FEC

Figure 5.3 (left) 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 5.3 (left, first) 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 (left, second) figure shows the corresponding hit trace which shows the detrimental effect of packet drops on the recovery rate.

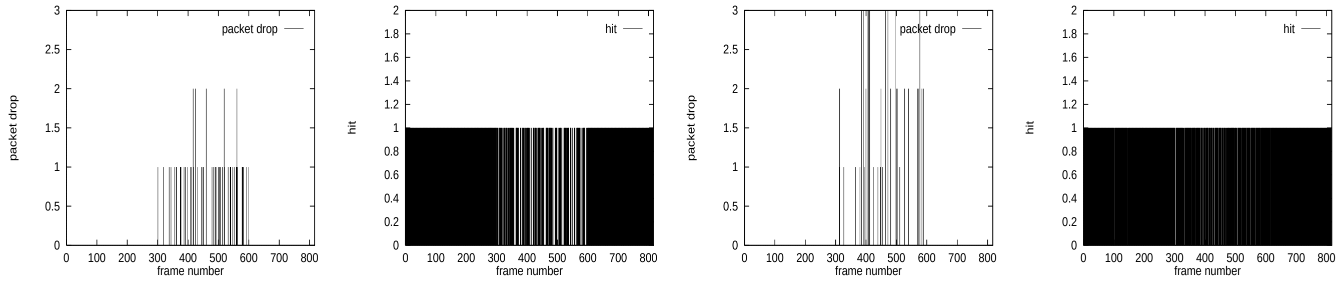

Figure 5.3. Static FEC vs. AFEC run when cross traffic is active during $[300, 600]$ for Simpsons MPEG clip. Packet drop trace and corresponding hit trace for static FEC with $h = 0$ (left) and AFEC (right).

Figure 5.3 (right) shows the trace of an AFEC run for the same configuration. The (right, first) 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 is due to system buffer overflow at the receiver host which occasionally occurs due to the FEC decoding overhead discussed in Section 4.2.

Figure 5.3 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.

5.2.3 AFEC Dynamics

Behavior in the Stable Regime Figure 5.4 (left) shows the average hit rate 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. Figure 5.4 (left, second) shows the continued changes in h during the entire run caused by the contention level being applied via the cross traffic process throughout.

Behavior in the Unstable Regime Figure 5.4 (right) show the corresponding performance measurements when the target hit rate was raised to 0.96 while keeping everything else the same. Figure 5.4 (right, first) 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 case, 0.935.

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 5.4 (right, second) shows a much wider range in h than its corresponding trace for target hit rate 0.92. 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 exponential backoff which causes an even steeper decline in h. This is also evidenced by h adjustments in the 0.92 target hit rate case being carried out predominantly by the symmetric component of MAFEC's control whereas in the 0.96 case, the asymmetric component is activated frequently.

6. 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. 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. 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 is not directly tied to the presence or absence of packet drops. In fact, if the desired QoS level falls into the stable regime, then exponential backoff is rarely invoked with symmetric control affecting both upward

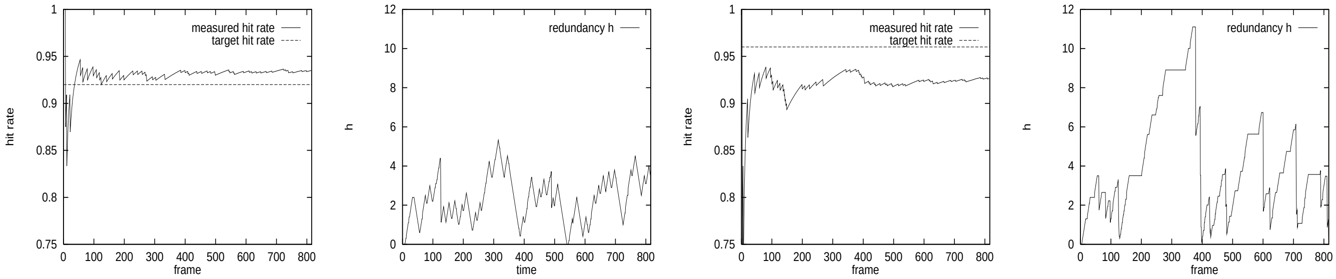

Figure 5.4. AFEC dynamics. Measured hit rate and h trace as a function of time (expressed as frame number) for target hit rate 0.92 (left) and target hit rate 0.96 (right).

and downward adjustments in h.

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. First, AFEC never consumes more resources than it has to to achieve a user-specified target hit rate which may involve not having to incur any packet drops at all. Second, if AFEC detects that increasing redundancy is counter-productive, it engages in exponential backoff just like TCP to prevent instability. A noncooperative protocol would exploit the fact that other protocols would be backing off, and by not doing so, soak up resources abandoned by others. The "gentlemanly" or cooperative behavior mode intrinsic to AFEC is also the modus operandi of TCP, and in this sense, their resource sharing behavior is fundamentally the same.

References

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

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

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

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

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

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

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

[8] 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.

[9] 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.

[10] 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.

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

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

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

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

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

[16] K. Park. AFEC: an adaptive forward error-correction protocol and its analysis. Technical Report CSD-TR-97-038, Department of Computer Sciences, Purdue University, 1997.

[17] 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.

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

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

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