Redundancy-Controllable Adaptive Retransmission Timeout Estimation for Packet Video

Ali C. Begen and Yucel Altunbasak
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332 USA
{acbegen, yucel}@ece.gatech.edu

ABSTRACT

Time-constrained error recovery is an integral component of reliable low-delay video applications. Regardless of the error-control method adopted by the application, unacknowledged or missing packets must be quickly identified as lost or delayed, so that necessary timely actions can be taken by the server/client. Historically, this problem has been referred to as the retransmission timeout (RTO) estimation. Earlier studies show that existing RTO estimators suffer from either long loss detection times or a large number of pre-mature timeouts. The goal of this study is to address these problems by developing an adaptive RTO estimator for high-bitrate low-delay video applications. By exploiting the temporal dependence between consecutive delay samples, we propose an adaptive linear delay predictor. This way, our RTO estimator configures itself based on the video characteristics and varying network conditions. Our approach also features a controller that optimally manages the trade-off between the amount of overwaiting and redundant retransmission rate. The skeleton implementation shows that the proposed RTO estimator discriminates lost packets from excessively-delayed packets faster and more accurately than its rivals, which consequently enables the applications to recover more packets under stringent delay requirements.

Categories and Subject Descriptors

C.2.2 [**Computer Systems Organization**]: Computer-Communication Networks—*Network protocols*

General Terms

Algorithms, Measurement, Experimentation

1. INTRODUCTION

The Internet is a shared medium; any packet injected into the Internet has to wait for some time before it is serviced. It therefore experiences random delay. Because of the finite buffering capabilities of the intermediate routers and switching devices, it is safe to assume that a packet is lost if it has not been received or acknowledged within some time after its transmission. In TCP jargon, this duration is referred to as the retransmission timeout (RTO). It is vital that the value of the RTO is chosen large enough so that the packets experiencing long queueing delays do not trigger spurious timeouts. However, adopting an arbitrarily large RTO is impractical for delay-sensitive multimedia applications. A delayed retransmission attempt eventually recovers a missing media packet. Yet, the chances are that the retransmitted packet will be late and useless for decoding at the client side. Therefore, an RTO estimation method that quickly detects lost packets is imperative for such applications. Only then can well-timed actions be taken for error control.

Naturally, retransmission-based error-control methods are unsuitable for multimedia applications where the extra delay introduced by the retransmissions is prohibitively large. However, due to emerging broadband technologies, end-to-end delays experienced by Internet users today are comparably smaller. Consequently, retransmission-based error-control methods can be still accommodated by many of today's low-delay multimedia applications. For example, consider a video-on-demand session running over UDP between a server and a client, where the server continuously transmits video packets to the client, and the client reports missing packets to the server with negative acknowledgments (NACKs); a NACK message is generated for a packet when the client decides that the packet is lost. If the NACKs are received by the server early enough, missing packets can be retransmitted successfully before their decoding deadlines pass. As a rule of thumb, the client should not time out pre-maturely and should wait as long as it is necessary for the excessively-delayed packets, since under normal circumstances it is highly unlikely that a retransmitted packet will arrive earlier than the initially-transmitted packet.

Needless to say, the primary challenge is that the client has to decide on timeouts merely by observing the packet arrivals in the course of a streaming session. It is never a clear-cut decision whether a missing packet has been lost or delayed. Naturally, a trade-off between overwaiting and spurious timeouts is present. To address this problem, in our earlier study [1], we introduced a client-driven method that utilized packet interarrival times for RTO estimation. The proposed approach was computationally efficient, and substantially outperformed an enhanced TCP-like RTO

NOSSDAV '06 Newport, Rhode Island USA

estimator by reducing both the amount of overwaiting and the number of redundant retransmissions. In that study, however, we did not provide a formal way to compute the parameters used in the RTO estimation. Our experiments with several video streams encoded at different bitrates later showed that the best-performing set of parameters varied for each stream and there was not a global optimal solution that would work for every video traffic. This motivated us to develop an adaptive RTO estimation method that would completely configure itself based on the source video characteristics and time-varying network conditions.

In this study, we devise a novel RTO estimation method that involves two main steps. In the first step, an adaptive linear delay predictor produces the best estimate in terms of the mean-squared error criterion by exploiting the temporal dependence among the packet delay samples. In the second step, on the other hand, a controller optimally manages the trade-off between the amount of overwaiting and redundant retransmission rate by regulating the bias to be added to the estimate produced in the previous step. We refer to this two-step method as the *redundancy-controllable adaptive RTO estimation*. The contribution of this method is two-fold: First, an adaptive delay predictor is proposed. A large number of multimedia protocols such as packet scheduling algorithms and adaptive buffer management techniques can potentially benefit from this predictor [2, 3]. Second, an optimal bias controller is derived. This controller allows the applications to maximize their error-recovery capability under any given redundant rate budget. To the best of our knowledge, this feature has not been previously offered by existing RTO estimation methods.

One of the earliest RTO estimation methods is the Jacobson's algorithm [4], which basically uses an exponentially-weighted moving average (EWMA) approach. Currently, TCP employs this algorithm with some modifications [5]. This class of RTO estimators have been thoroughly examined by Loguinov and Radha in the context of a video streaming application [6]. Their empirical study concluded that EWMA-based RTO estimation was not quick enough to detect lost packets. The authors also suggested using jitter samples in fine-tuning the estimations. Although [6] presents important findings, its scope is rather limited, since the study primarily focuses on a low-bitrate video streaming application with a large playout buffer. On the TCP end, other proposals for replacing [4] are [7–10]. However, these approaches are not suitable for low-delay applications either, due to their conservative estimates and slow adaptation to time-varying network conditions.

In a more recent study [11], Sinha and Papadopoulos proposed a timerless retransmission protocol that eliminated the pitfalls of round-trip time (RTT) estimation and timer-triggered timeouts. In this protocol, a lost packet can only be identified upon detection of a gap in the received packets. Hence, when a batch of packets are lost or excessively delayed, this protocol has to wait indefinitely until a new packet is received, which potentially impedes the timely recovery of bursty losses. Previously, Papadopoulos and Parulkar used an algorithm similar to [4] for real-time streaming [12]. A different approach was later proposed by Rhee [13], where retransmission decisions were based on multiples of frame durations. However, these approaches are not adaptive and cannot perform well when streaming high-bitrate video under low-delay requirements.

In the sequel, Section 2 provides an overview of different RTO estimation methods. In Section 3, we discuss the details of the adaptive linear delay prediction and timeout estimation. Results from Internet experiments are presented in Section 4. We conclude the paper with directions for future work in Section 5.

2. OVERVIEW OF RTO ESTIMATORS

In this section, we briefly summarize three different classes of previously proposed RTO estimators. Later, in Section 4, we compare our approach with these estimators in terms of their performances.

2.1 TCP-Like RTO Estimators

The RTO estimation algorithm used in current TCP implementations is based on Jacobson's algorithm [4], which was later modified in [5]. In TCP, the TCP sender records a new RTT measurement when it receives an unambiguous acknowledgment packet. Let $\mathbf{r}[n]$ denote the RTT observation corresponding to packet n. Jacobson's algorithm predicts the RTT of the subsequent packet (denoted by $\tilde{\mathbf{r}}[n+1]$) by computing the moving average:

$$\tilde{\mathbf{r}}[n+1] = \frac{7}{8}\tilde{\mathbf{r}}[n] + \frac{1}{8}\mathbf{r}[n]. \quad (1)$$

The TCP sender also keeps track of the variation in the observed RTT values, which is computed by

$$\sigma_{RTT} = \frac{3}{4}\sigma_{RTT} + \frac{1}{4} \times |\mathbf{r}[n] - \tilde{\mathbf{r}}[n+1]|. \quad (2)$$

Subsequently, the value of RTO is set by using

$$\text{RTO} = \max\left(RTO_{min}, \tilde{\mathbf{r}}[n+1] + \max\left(G, k \times \sigma_{RTT}\right)\right), \quad (3)$$

where $k = 4$ and G is the clock granularity. In practice, RTO_{min} is set to one second [5] to reduce spurious timeouts. In addition, current TCP variants implement Karn's algorithm [14], which suggests doubling the RTO value when a timeout occurs. Employing exponential timer backoff as well as adopting a large RTO_{min} are essential for TCP's congestion control algorithm and network-friendliness. However, such measures are naturally too costly for delay-sensitive applications. Therefore, in our comparisons, we will use an *enhanced TCP-like RTO estimator*, where RTO_{min} is set to zero and the exponential timer backoff is disabled.

2.2 Recursive Weighted Median Filtering

Recursive weighted median (RWM) filtering was recently proposed by Ma *et al.* to improve Jacobson's algorithm [10]. The idea is to compute the RTT estimate by taking the weighted median of the last K estimates and last J observations. That is,

$$\tilde{\mathbf{r}}[n] = \text{WM}\left(\left[\tilde{\mathbf{r}}[n-k]|_{k=1}^{K}, \mathbf{r}[n-j]|_{j=1}^{J}\right], \mathbf{W}\right), \quad (4)$$

where $\mathbf{W}$ is the weight vector. The study suggests the following values: $K = 1$, $J = 5$ and $\mathbf{W} = [\frac{1}{2}, (\frac{7}{8})^0, (\frac{7}{8})^1, (\frac{7}{8})^2, (\frac{7}{8})^3, (\frac{7}{8})^4]$. Once an RTT estimate is computed, the value of the RTO is determined by scaling the RTT estimate, where the scale factor depends on the mean absolute deviation among the RTT samples. Improvements over Jacobson's algorithm are reported through Internet experiments [10].

2.3 Percentile-Based RTO Estimators

Empirical studies usually try to fit a well-known distribution to packet delays experienced in IP networks [15–17]. Such statistical models are particularly useful in laying out a mathematical framework for building application-layer protocols [2]. For example, [15] suggests that RTT values follow a shifted Gamma distribution, whereas [16, 17] show that packet delay distributions are heavy-tailed and can be characterized by a Pareto distribution. The pertaining distribution model parameters can be estimated from the collected samples by using Maximum Likelihood Estimation. Alternatively, one can collect the delay samples and generate a distribution on the fly. Having a history of the delay samples, the delay of the next packet can be predicted by computing the p^{th}-percentile of the delay histogram, where p is selected depending on the maximum redundant retransmission rate tolerable by the application.

3. PROPOSED APPROACH

Let us start our discussion with the problem scenario. Consider a low-delay video application where the client runs an RTO estimation method to determine the *best* time to request a retransmission for a missing packet. Upon receipt of a packet, the client measures its delay and predicts the delay for the subsequent packet. The client then computes the amount of additional waiting to take into account the delay variability. In the implementation, we identify each packet with a unique number, which can be associated with the *Sequence Number* field in the RTP header [18]. Since a retransmission cannot be distinguished from the initial transmission, the delay measurements for the retransmitted packets are ignored to avoid any ambiguity.

Our Internet experiments suggest that the majority of the lost packets can be recovered with a single retransmission and doing two or more retransmissions is rarely necessary. Furthermore, the low-delay requirement of our target applications severely limits the possibility of multiple retransmissions. Thus, our discussion will focus only on the single-retransmission case[1].

3.1 Adaptive Linear Delay Prediction

Statistical models, briefly discussed in Section 2.3, are only useful for characterizing the general properties of packet delays, and fall short in describing the temporal dependence among packet delays. Previously, Jiang and Schulzrinne investigated the conditional delay distributions [16], and found significant correlation between the adjacent delay samples. Here, we exploit this correlation through autoregressive models for delay prediction.

Consider a stochastic process $\mathbf{s}$ and let $\mathbf{s}[n-k]$, $k \geq 1$ denote the past samples of this process. The operation of linear prediction expresses the value of $\mathbf{s}[n]$ as the linear combination of the samples $\mathbf{s}[n-k]$. The estimate based on the N most recent values is given by

$$\tilde{\mathbf{s}}_N[n] = \tilde{E}\left\{\mathbf{s}[n] | \mathbf{s}[n-k], 1 \leq k \leq N\right\} = \sum_{k=1}^{N} \alpha_{k,N}\mathbf{s}[n-k]. \tag{5}$$

[1]Note that the methods summarized in Section 2 were originally designed for server-side RTO estimation. In our comparisons (Section 4), we adopt them for use at the client side.

This estimate is called the one-step forward predictor of order N. The process $\tilde{\mathbf{s}}_N[n]$ is the response of the forward predictor filter

$$\mathbf{H}_N(z) = \sum_{k=1}^{N} \alpha_{k,N} z^{-k} \tag{6}$$

to the input $\mathbf{s}[n]$. Our objective in prediction is to determine the constants $\alpha_{k,N}$ so as to minimize the mean square value

$$P_N = E\{\boldsymbol{\epsilon}_N^2[n]\} \tag{7}$$

of the forward prediction error $\boldsymbol{\epsilon}_N[n] = \mathbf{s}[n] - \tilde{\mathbf{s}}_N[n]$. From the orthogonality principle, we know that the prediction error, *i.e.*, $\boldsymbol{\epsilon}_N[n]$, is orthogonal to all the data used to generate the prediction, *i.e.*, $\mathbf{s}[n-m]$, where $1 \leq m \leq N$. Mathematically, we have

$$E\left\{\left(\mathbf{s}[n] - \sum_{k=1}^{N} \alpha_{k,N}\mathbf{s}[n-k]\right)\mathbf{s}[n-m]\right\} = 0 \quad 1 \leq m \leq N, \tag{8}$$

which yields a set of linear equations known as the Yule-Walker equations. The coefficients $\alpha_{k,N}$ of the predictor filter $\mathbf{H}_N(z)$ can be computed from

$$R[m] - \sum_{k=1}^{N} \alpha_{k,N} R[m-k] = 0 \quad 1 \leq m \leq N, \tag{9}$$

where $R[q]$ represents the lag-q autocorrelation of $\mathbf{s}$. Alternatively, one can use Durbin-Levinson recursion [19] by virtue of the toeplitz nature of the linear system given in (9). The resulting mean-squared prediction error equals to

$$P_N = R[0] - \sum_{k=1}^{N} \alpha_{k,N} R[k]. \tag{10}$$

As the order of prediction, N, increases, the value of the mean prediction-error power, P_N, decreases or else remains the same. Since prediction-error power is always positive, we have

$$P_1 \geq P_2 \geq \ldots \geq P_N \xrightarrow[N\to\infty]{} P \geq 0. \tag{11}$$

The implication of (11) is that as we increase the order of the predictor filter $\mathbf{H}_N(z)$, we successively reduce the correlation between the adjacent samples of the input process until we ultimately reach a point at which increasing the order of prediction any further does not reduce the prediction-error power. At this point, the error is a white noise process and consists of purely uncorrelated samples.

Suppose that $P_{M-1} > P_M$ and $P_M = P_{M+1} = \ldots = P$. By definition, the process $\mathbf{s}$ is called an M^{th} order autoregressive, denoted by AR(M), process or a wide-sense Markoff process of order M. For this process, the M^{th} order predictor, $\tilde{\mathbf{s}}_M[n]$, equals to its Wiener predictor:

$$\tilde{E}\left\{\mathbf{s}[n] | \mathbf{s}[n-k], 1 \leq k \leq M\right\} = \tilde{E}\left\{\mathbf{s}[n] | \mathbf{s}[n-k], k \geq 1\right\}. \tag{12}$$

Wiener predictors produce the best fit to the observed data by exploiting the existing correlation completely. Generally, it is desirable to have the values predicted by a model to be close to the actual data values. However, an overfitted model cannot distinguish the systematic effects of the process

from its random effects, thus, suffers from a low predictive accuracy. Therefore, Wiener predictors are usually not used in practice. Next, we study the issue of model selection by examining two delay traces.

In the following numerical results, we benefit from the one-way delay traces collected by simulating a moderate-sized Internet topology [20] in *ns*-2 network simulator [21][2]. We generated two delay traces from video streams encoded at 300 and 600 Kbps. These traces will be referred with the notation of $\Delta T = 40$ ms and $\Delta T = 20$ ms, respectively, where ΔT denotes the average transmission interval at the server.

The relation between the mean prediction-error power and the order of prediction is given in Fig. 1 for both delay traces. Based on the definition given in (12), the order of Wiener prediction for the $\Delta T = 20$ ms and $\Delta T = 40$ ms traces is found to be 32 and 60, respectively. Clearly, we require a higher order of prediction for larger ΔT. This is not surprising since the correlation between the adjacent delay samples reduces with ΔT. Fig. 1 shows that the mean prediction-error power gradually decreases with the order of prediction. However, based on the AICC (Akaike's Information Corrected Criterion) scores [19] the predictive accuracy first shows an increasing and then a decreasing trend. Specifically, the predictive accuracy for the $\Delta T = 20$ ms and $\Delta T = 40$ ms traces reaches its global maxima at $N = 9$ and $N = 12$, respectively. These values are comparably smaller than the ones corresponding to the Wiener prediction, signifying that Wiener predictors are indeed overfitted and have sub-optimal predictive accuracy.

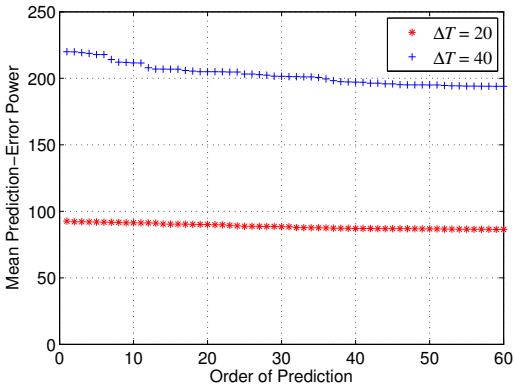

Figure 1: Variation of the mean prediction-error power with the order of prediction.

Generally speaking, the AICC scoring system suggests good models that provide sufficient insight into the process being analyzed, while leaving out the random effects. In non-time-critical tasks, the computational complexity is of a less important issue. Thus, the models suggested by the AICC approach can be facilely employed without hampering the performance of the system. However, if the prediction is carried out in real time, low-complexity models have to be used to sustain the system feasibility. The main objective is, thus, to select a computationally-efficient yet intuitively plausible prediction model that adequately captures the dynamics in the packet delay process.

A *naive* approach is the AR(1) model, where the next delay estimate is solely determined by the last observation, *i.e.*, $\tilde{\mathbf{s}}_1[n] = \mathbf{s}[n-1]$. The phase diagrams plotted in Fig. 2 clearly indicate the existence of a significant lag-1 correlation among the delay samples and support the AR(1) prediction model. However, this predictor is not capable of distinguishing whether packet delays are increasing, decreasing, or remaining the same, and therefore, does not serve our goal.

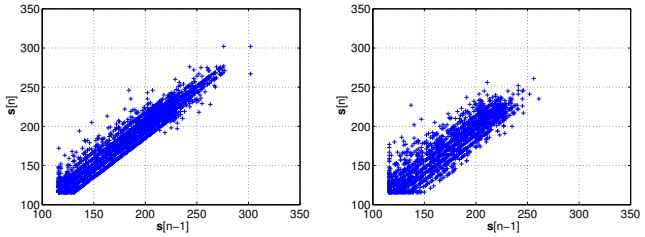

Figure 2: Phase diagrams for $\Delta T = 20$ ms (on the left) and $\Delta T = 40$ ms (on the right).

A more elaborate model is the AR(2) model. AR(2) model bases its estimation on the last two observations. By definition, we have

$$\tilde{\mathbf{s}}_2[n] = \alpha_{1,2}\mathbf{s}[n-1] + \alpha_{2,2}\mathbf{s}[n-2], \tag{13}$$

which can be rewritten as

$$\tilde{\mathbf{s}}_2[n] = (\alpha_{1,2} + \alpha_{2,2})\mathbf{s}[n-1] + \alpha_{2,2}\left(\Delta T - \Delta t[n-1]\right), \tag{14}$$

where $\Delta t[n]$ denotes the interarrival time for packet n, *i.e.*, the time difference between the arrivals of packets n and $n-1$. The interpretation of (14) is that the AR(2) model takes into consideration not only the last delay sample but also its deviation from the previous sample.

To understand how well an AR(2) predictor compares to its Wiener counterpart, we plot the prediction-error autocorrelation functions (ACF) for both predictors. Since Wiener predictors completely model the data, the resulting error samples are guaranteed to be uncorrelated, which is, however, not necessarily true for AR predictors of lower orders. Nevertheless, Fig. 3 shows that the correlation left out by the AR(2) predictors is rather insignificant, implying that AR(2) predictors have sufficient prediction accuracy for practical purposes.

3.2 Timeout Estimation

From the point of view of (7), an underestimate that is marginally smaller than the actual value is as good as an overestimate that is marginally larger than the actual value. However, in the context of RTO estimation, underestimations trigger pre-mature timeouts whereas overestimations eliminate them. In this section, we formulate an elegant way to compute the minimum amount of additional waiting that is required to keep the probability of a pre-mature timeout below a desired value.

In Fig. 4, we plot the prediction-error distributions for both delay traces. We notice that each of these distributions (particularly, the tail parts) can be approximated by a Gaussian distribution whose mean and standard deviation

[2]In a real environment, one-way delay measurement requires a clock synchronization between the end-points. See [22] for details.

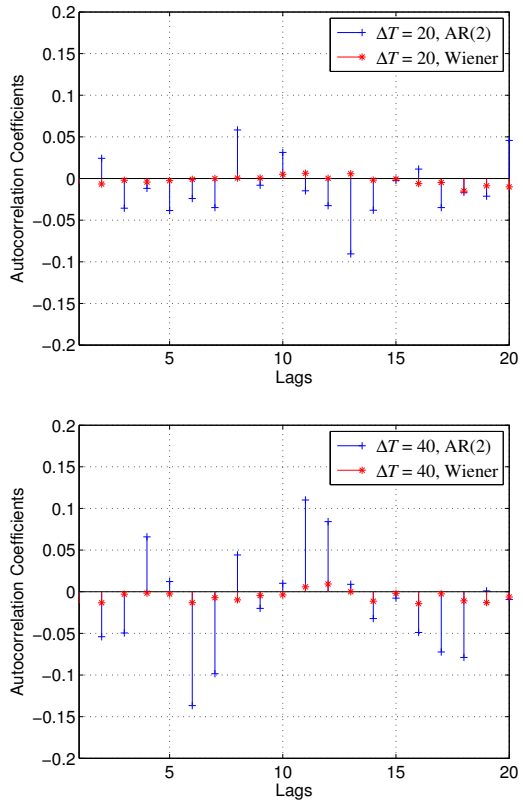

Figure 3: ACFs of the prediction errors produced by Wiener and AR(2) predictors for $\Delta T = 20$ ms (above) and $\Delta T = 40$ ms (below).

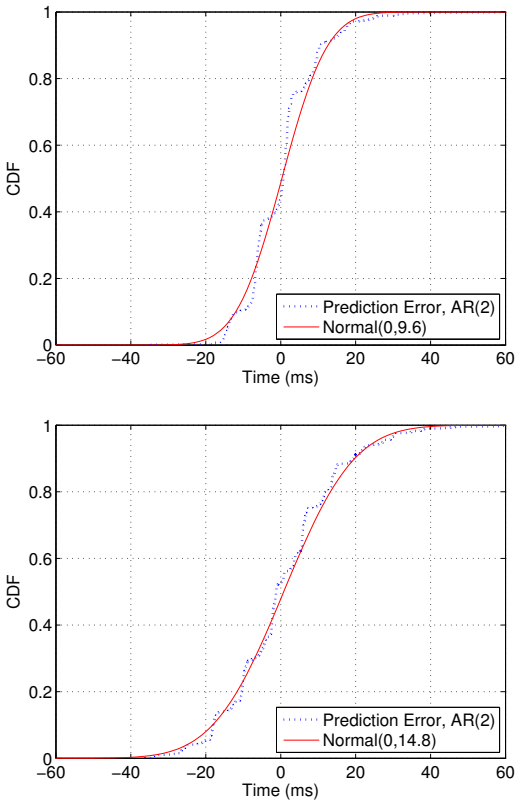

Figure 4: Prediction-error distributions for $\Delta T = 20$ ms (above) and $\Delta T = 40$ ms (below). Plots do not include the lost packets.

(σ) are equal to those of the corresponding prediction-error distribution. Statistically, Gaussian-distributed samples of a white noise process are independent of each other. In the light of Fig. 3, we infer that AR(2) predictors produce error samples that are independent. This result has two important implications: First, a sequence of independent random variables is not predictable by linear or non-linear models. Thus, if packet delay sampling is sufficiently dense, the delay process can be almost completely characterized by an AR(2) model. Second, Gaussian-distributed processes are easy to work with and a rich set of mathematical tools is available.

Let τ denote the additional amount of waiting to be added to the initial delay predicted by (14), and let $\Phi(\tau)$ denote the underestimation probability. By definition,

$$\Phi(\tau) = P\left\{\tilde{\mathbf{s}}_2[n] + \tau < \mathbf{s}[n]\right\}, \tag{15}$$

which is a non-increasing function of τ. We seek the minimum value for τ that satisfies

$$\Phi(\tau) \leq p_f, \tag{16}$$

where p_f is the desired probability of timing out pre-maturely. By rewriting $\Phi(\tau)$ as $P\{\tau < \boldsymbol{\epsilon}_2\}$, we compute τ from

$$\tau = F^{-1}_{\boldsymbol{\epsilon}_2}(1 - p_f), \tag{17}$$

where $F_{\boldsymbol{\epsilon}_2}$ is the cumulative density function of $\boldsymbol{\epsilon}_2$. A nice feature of the Gaussian distribution is that its inverse cumulative function can be directly calculated from the first and second-order moments. For example, to limit the rate of pre-mature timeouts to 5%, τ should be set to $1.65 \times \sigma$, which is 25 ms for the $\Delta T = 40$ ms trace. While 25 ms can be mistakenly seen insignificant, τ quickly increases for lower p_f values, *e.g.*, for $p_f = 0.1\%$, the required amount increases to 46 ms.

The adverse impact of large τ values is the increase in the time required to detect lost packets. To quantify the detection time of a lost packet, we use the delay of the last successfully-received packet as the hypothetical delay for the lost packet. The loss detection time is then given by the difference between the predicted and the hypothetical delays. That is,

$$w[n] = \tilde{\mathbf{s}}_2[n] + \tau - \mathbf{s}[n^*], \quad \forall n : \mathbf{s}[n] = \infty, \tag{18}$$

where n^* is the last successfully-received packet. The average loss detection time and the pre-mature timeout probability are the benchmarks that characterize the performance of an RTO estimator. In Fig. 5, we plot several $(\overline{w}, p_f)$ points for both delay traces. We notice that the client detects lost packets faster when streaming at higher bitrates. This result is consistent with Fig. 1 and 3 where we showed that the AR(2) prediction is more accurate for smaller ΔT values.

4. INTERNET EXPERIMENTS

4.1 Setup

In order to assess the performance of the proposed RTO estimation method, we developed a skeleton implementation

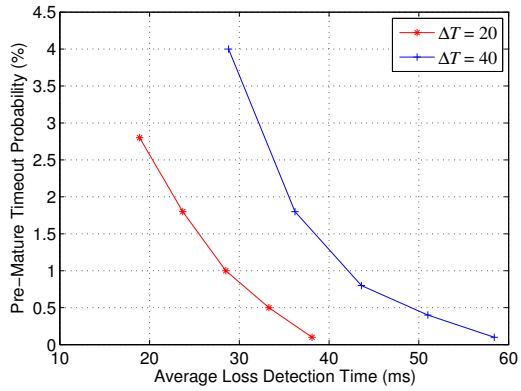

Figure 5: Performance analysis of the proposed RTO estimator.

and established an experimental platform on the Internet. On this platform, we emulated a real-time video streaming application over UDP between a broadband client in Ankara, Turkey and a broadband server in Atlanta, GA USA[3]. The client simultaneously streamed video packets from the server and carried out the RTO computation. When a packet was identified as lost, a retransmission request was sent to the server. If this request was successfully received, the server immediately retransmitted the requested packet. The delay measurements for the retransmitted packets were ignored by the client in order to avoid any potential ambiguity.

We conducted our experiments in five sessions of 60 minutes, where we tested four different approaches: (*i*) the enhanced TCP-like RTO estimator, denoted by $\text{RTO}_{\text{E-TCP}}$, (*ii*) recursive weighted median filtering, denoted by $\text{RTO}_{\text{RWM(1,5)}}$, (*iii*) a percentile-based RTO estimator that predicts the forward-trip time (FTT) of the next expected packet by computing the p^{th}-percentile of the FTT histogram (excluding the lost packets), denoted by RTO_{PRC}, and (*iv*) the proposed RTO estimator, denoted by $\text{RTO}_{\text{AR(2)}}$. After each session, the mean delay and packet loss rate were measured to ensure that similar network characteristics were observed in all sessions. (The mean round-trip delay and mean forward-path packet loss rate were measured as approximately 250 ms and 6.0%, respectively.) In order to compensate for one-way delay jitter and allow some time for retransmissions, we employed a playout delay of 500 ms. The packets that missed their decoding deadlines, *i.e.*, late packets, were still used in the decoding process to improve the decodability of the predictively-encoded frames. We concealed missing macroblocks with the ones in the last successfully-decoded frame. This basic error-concealment technique reduced the amount of severe transitions from the frames in error.

For video quality comparison, we encoded the test sequence FOREMAN (352 × 288) with a standard H.264 codec [23] at 600 Kbps and 20 frames per second. We present our results in terms of both the percentage of on-time packets and average video quality. For the latter, we use the peak signal-to-noise ratio (PSNR) measure on the luminance (Y) channel. PSNR is computed from $\text{PSNR} = 10 \times \log_{10}(\frac{255^2}{\text{MSE}})$, where MSE stands for the mean-squared error between the original and decoded luminance frames.

[3]The reason for experimenting over an intercontinental network was to observe a wide range of packet loss and delay characteristics.

4.2 Results

An immediate result of our experiments is that without any error control, the video quality severely suffers from the lost packets; in our case, the average streaming quality barely reached 34.3 dB (5.6 dB lower than the lossless case). When $\text{RTO}_{\text{E-TCP}}$ was employed, the video quality improved by 0.7 dB at the expense of an average rate increase of 39 Kbps. However, 7% of this rate increase was redundant due to pre-mature timeouts, and 86% of it was useless since those packets missed their deadlines. The second method, $\text{RTO}_{\text{RWM(1,5)}}$, improved the video quality by only 0.2 dB while increasing the average transmission rate by 38 Kbps. We observed that 5% of the retransmissions requested by $\text{RTO}_{\text{RWM(1,5)}}$ were redundant and 93% of them were late. These results clearly indicate that the RTO estimators that are primarily designed for TCP are not suitable for low-delay applications. The overinflated estimates inevitably disrupt the timely detection of lost packets, which adversely affects the on-time retransmission performance.

In the third session, we tested $\text{RTO}_{\text{AR(2)}}$ and obtained an average video quality of 38.3 dB (4.0 dB improvement over the no-retransmission case) at an average streaming rate of 638 Kbps. The quality gain stemmed from the fact that only 5% and 30% of the total retransmissions were redundant and late, respectively. In the last two sessions, we experimented with RTO_{PRC} with two different p values. These p values were chosen by trial and error such that (*i*) RTO_{PRC} recovered as many packets on time as $\text{RTO}_{\text{AR(2)}}$, and (*ii*) RTO_{PRC} consumed an average total streaming rate equal to the one consumed by $\text{RTO}_{\text{AR(2)}}$. Table 1 summarizes our results, and shows that $\text{RTO}_{\text{AR(2)}}$ clearly outperforms all other three RTO estimators by achieving a higher video quality while streaming at an equal or smaller bitrate.

	Rate (Kbps)	Redundant Ret.	Late Ret.	PSNR (dB)
$\text{RTO}_{\text{E-TCP}}$	639	7%	86%	35.0
$\text{RTO}_{\text{RWM(1,5)}}$	638	5%	93%	34.5
RTO_{PRC}	644	18%	29%	38.3
RTO_{PRC}	638	5%	55%	36.5
$\text{RTO}_{\text{AR(2)}}$	638	5%	30%	38.3

Table 1: Experimental results for different RTO estimators.

4.3 Implementation Issues

In our implementation, the AR(2) predictor filter coefficients, $\alpha_{1,2}$ and $\alpha_{2,2}$, as well as the prediction-error standard deviation were computed from the accumulated statistics. Particularly, we employed a window-based approach and used the last 20 samples to solve (9). While $\alpha_{1,2}$ and $\alpha_{2,2}$ varied over time, we did not observe much change in the error statistics. However, it is still not conclusive whether a window size of 20 is suitable for all cases. We are currently investigating this issue and its impact on the computational complexity of the RTO estimation.

5. CONCLUSIONS AND FUTURE WORK

In this study, we developed an adaptive delay predictor and a redundancy-controllable timeout estimator. The proposed two-step RTO estimation method optimally manages the trade-off between the amount of overwaiting and redundant retransmission rate, which allows the applications to maximize their error-recovery capabilities under a given redundant rate budget. We summarize our main findings as follows:

- The RTO estimators that are developed for TCP severely suffer from long loss detection times. We were able to reduce the detection times by tweaking the estimation parameters, *e.g.*, k in (3) and $\mathbf{W}$ in (4). However, the resulting rate of spurious timeouts was unacceptably high.
- Provided that the packets are transmitted at sufficiently short intervals, consecutive delay samples show a strong correlation. Wiener predictors can be used to fully exploit this correlation and produce uncorrelated prediction-error samples. We showed that these uncorrelated error samples could be modeled by a Gaussian distribution, implying that the error samples were indeed independent. Thus, Wiener prediction models can completely characterize the packet delay process. We also showed that AR(2) predictors could be safely used in practice instead of their Wiener counterparts.
- Adaptivity to time-varying network conditions is the key in a successful RTO estimation. Slow adaptation potentially leads to a significant performance degradation in terms of redundant/late retransmissions.

In this study, to keep the discussion focused and concrete, we ignored the interdependency relations among the video frames and packets. However, a *media-aware* RTO estimation method may prioritize the packets that carry a more important payload and the packets whose decoding deadlines are sooner, over the less important and non-urgent packets. A promising direction in our future work is to investigate the potential quality improvements that can be gained through media-aware RTO estimation.

Another direction for future work is to investigate whether a hybrid ARMA (autoregressive/moving average) model could be used to improve Jacobson's algorithm. While data-oriented applications do not explicitly impose delivery deadlines to TCP packets, a quick loss detection may be beneficial to achieve a higher TCP throughput, particularly in the short-lived flows.

Acknowledgments– The authors would like to thank Mehmet A. Begen of University of British Columbia for his helpful discussions. This work is supported by NSF under NSF award CCF-0430907.

References

[1] A. C. Begen and Y. Altunbasak, "Timely inference of late/lost packets in real-time streaming applications," in *Picture Coding Symp. (PCS)*, 2004.

[2] P. A. Chou and Z. Miao, "Rate-distortion optimized streaming of packetized media," *Microsoft Research Technical Report MSR-TR-2001-35*, 2001.

[3] M. Kalman, E. Steinbach, and B. Girod, "Adaptive media playout for low delay video streaming over error-prone channels," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 6, pp. 841–851, June 2004.

[4] V. Jacobson, "Congestion avoidance and control," in *ACM SIGCOMM*, 1988.

[5] Computing TCP's Retransmission Timer. [Online]. Available: http://www.ietf.org/rfc/rfc2988.txt

[6] D. Loguinov and H. Radha, "On retransmission schemes for real-time streaming in the internet," in *IEEE Int. Conf. Computer Communications (INFOCOM)*, 2001.

[7] M. Allman and V. Paxson, "On estimating end-to-end network parameters," in *ACM SIGCOMM*, 1999.

[8] R. Ludwig and K. Sklower, "The eifel retransmission timer," *SIGCOMM Comput. Commun. Rev.*, vol. 30, no. 3, pp. 17–27, 2000.

[9] Q. Li and D. L. Mills, "Jitter-based delay-boundary prediction of wide-area networks," *IEEE/ACM Trans. Networking*, vol. 9, no. 5, pp. 578–590, Oct. 2001.

[10] L. Ma, G. R. Arce, and K. E. Barner, "TCP retransmission timeout algorithm using weighted medians," *IEEE Signal Processing Lett.*, vol. 11, no. 6, pp. 569–572, June 2004.

[11] R. Sinha and C. Papadopoulos, "An adaptive multiple retransmission technique for continuous media streams," in *ACM NOSSDAV*, 2004.

[12] C. Papadopoulos and G. M. Parulkar, "Retransmission-based error control for continuous media applications," *ACM NOSSDAV*, 1996.

[13] I. Rhee, "Error control techniques for interactive low bitrate video transmission over the internet," in *ACM SIGCOMM*, 1998.

[14] P. Karn and C. Partridge, "Improving round-trip time estimates in reliable transport protocols," in *ACM SIGCOMM*, 1987.

[15] A. Mukherjee, "On the dynamics and significance of low frequency components of internet load," University of Pennsylvania, Tech. Rep. MS-CIS-92-83, 1992.

[16] J. Wenyu and H. Schulzrinne, "Modeling of packet loss and delay and their effects on real-time multimedia service quality," in *ACM NOSSDAV*, 2000.

[17] D. Loguinov and H. Radha, "End-to-end internet video traffic dynamics: Statistical study and analysis," in *IEEE Int. Conf. Computer Communications (INFOCOM)*, 2002.

[18] RTP: A Transport Protocol for Real-Time Applications. [Online]. Available: http://www.ietf.org/rfc/rfc1889.txt

[19] P. J. Brockwell and R. A. Davis, *Introduction to Time Series and Forecasting*. Springer, 2003.

[20] E. W. Zegura, K. L. Calvert, and M. J. Donahoo, "A quantitative comparison of graph-based models for Internet topology," *IEEE/ACM Trans. Networking*, vol. 5, no. 6, pp. 770–783, 1997.

[21] S. McCanne and S. Floyd. Network simulator. [Online]. Available: http://www.isi.edu/nsnam/ns

[22] Simple Network Time Protocol (SNTP) Version 4. [Online]. Available: http://www.ietf.org/rfc/rfc2030.txt

[23] H.264 AVC reference software. [Online]. Available: http://iphome.hhi.de/suehring/tml/download