Streaming Stored Continuous Media over Fair-Share Bandwidth

Despina Saparilla
Dept. of Systems Engineering
University of Pennsylvania
Philadelphia, PA 19104
saparill@eurecom.fr

Keith W. Ross
Institut EURECOM
2229, route des Crêtes
Sophia Antipolis, France
ross@eurecom.fr

Abstract—**We investigate the impact of the long-term behavior of fair-share bandwidth on transmission schemes for streaming stored Continuous Media (CM). To obtain typical fair-share bandwidth conditions, we perform a series of Internet experiments in which we monitor TCP bulk-data transfers between various sites, and collect average TCP throughput traces. The collected traces exhibit high-variability over a broad range of time scales as well as self-similar scaling behavior over longer time scales. Under fair-share bandwidth conditions, we evaluate the performance of a series of data transmission schemes for non-layered CM, and of several bandwidth allocation schemes for streaming layered CM. Our findings demonstrate that prefetching during playback over intervals of several minutes is necessary for achieving best quality. For layered CM encoded into two layers, we propose a threshold-based inter-layer bandwidth allocation scheme, and a measurement-based heuristic for dynamically computing the threshold. Our empirical results show that, using conservative estimates for future average bandwidth, our heuristic is highly reliable.**

I. Introduction

Recent years have seen an explosive increase in the consumption of audio and video programming on the Internet. We expect that traffic from continuous-media (CM) streaming applications, delivering stored video and audio to consumers on demand, will constitute a major fraction of Internet traffic in the upcoming years. Currently, most CM streaming applications run over UDP, using proprietary transport protocols that in most cases do not implement adequate congestion avoidance algorithms. Given the projected growth in CM streaming traffic, this poses a threat to the fair and efficient utilization of network resources. To preserve Internet performance, streaming applications should be designed to incorporate effective congestion avoidance mechanisms such that streaming traffic flows obtain only a fair share of link bandwidth [1].

TCP-conformant traffic flows (HTTP, SMTP, NNTP etc.) dominate today's Internet. Thus, to allow for fair bandwidth sharing, streaming applications should be designed to be cooperative with TCP flows by appropriately reacting to congestion. The development of congestion control mechanisms for streaming applications is an area of ongoing research. Recent research has shown that congestion control can be incorporated in UDP-based streaming applications using TCP-friendly algorithms for determining fair-share bandwidth [2][3][4][5][6].

To cope with the unpredictability and variability of available bandwidth between server and client, efficient transmission schemes for streaming stored CM must be implemented in order to achieve satisfactory performance. In this paper we investigate the impact of the long-term behavior of fair-share bandwidth on data transmission schemes for streaming stored CM. We use TCP traces collected by monitoring several TCP bulk-data transfers in a series of Internet experiments between the U.S. and Europe, and between two European sites, to obtain typical fair-share bandwidth traces. Our analysis of the traces shows that fair-share bandwidth exhibits a wide range of variation over many time scales, and a degree of self-similar scaling behavior over long time scales.

We consider data transmission schemes for streaming stored non-layered CM, and for streaming scalable CM encoded into two layers. We evaluate the performance of the transmission schemes in an empirical study that is based on our collected bandwidth traces. For non-layered CM, we find that a significantly higher level of quality can be achieved when the CM is prefetched *during playback* into client storage. Prefetching CM during playback over short intervals, of a few 10's of seconds, performs better than transmission schemes that employ no prefetching. Near-optimal performance is achieved, however, when the CM is prefetched during playback over intervals that are at least three to four minutes long. One interesting observation is that streaming stored CM directly over TCP often provides near optimal performance.

Our results with non-layered CM also motivate the use of layered encoding. Layered CM allows the streaming application to cope with the wide fluctuations in fair-share bandwidth by adjusting the transmission rate or the quality of the stream according to the available bandwidth conditions. We consider scalable CM encoded into two layers, a base layer and an enhancement layer. In our earlier work we developed a model and theory for the problem of dynamically allocating fair-share bandwidth among the two layers in order to minimize the impact of client buffer starvation on the quality of the decoded stream [7]. In this paper we evaluate the performance of two classes of inter-layer bandwidth allocation schemes in simulations based on fair-share bandwidth traces. We then propose a scheme that streams the enhancement layer only when we are confident that the base-layer buffer at the client will henceforth not be starved. To implement this type of streaming scheme, the server must track client prefetch buffer content, as well as estimate the likelihood of future starvation in the base layer. The estimate of likelihood of base-layer starvation depends on (i) the amount of base-layer data prefetched at the client, (ii) the consumption rate of base-layer data, and (iii) a conservative estimate of the future available bandwidth. We develop a threshold-based heuristic for adding and dropping the enhancement layer that uses conservative estimates of future average bandwidth. Our results with fair-share bandwidth traces show that our heuristic threshold-based scheme is highly reliable, that is, it renders the base layer without loss, and it achieves long and continuous viewing of the enhancement layer with infrequent quality fluctuations.

The rest of the paper is organized as follows. In Section II we describe our methodology for collecting TCP traces and analyze of the long-term scaling behavior of the collected traces. In Section III we quantify the benefits of prefetching when streaming non-layered video. Section IV considers layered-CM streaming over fair-share bandwidth. Finally, Section V concludes the paper.

A. Related Research

A significant amount of research reports on the self-similar nature of aggregate network traffic. Statistical analysis of Ethernet LAN traffic and wavelet-based scaling analysis of Internet WAN traffic have shown that aggregate traffic exhibits self-similar scaling behavior over time scales of a few hundreds of milliseconds and larger [8][9][10]. Previous work on the scaling behavior of network traffic has considered packet-level and TCP connection-level traces. In this paper, we focus on the *long-term* behavior of a *single* TCP connection, initiated by a bulk data transfer. We are interested in measuring TCP throughput at the receiver to gain insight on the impact of fair-share bandwidth behavior on streaming CM.

Rejaie et al present a broad range of architectural considerations for streaming layered encoded video [11][12]. They argue for the need for end-to-end congestion control, quality adaptation and error control in multimedia streaming applications. In [11] they develop mechanisms for adapting the quality of streaming video playback while performing congestion control. Their proposed mechanisms are suitable only for congestion control schemes that employ an additive increase multiplicative decrease (AIMD) algorithm. Finally, their buffer allocation mechanisms are evaluated using bandwidth traces from the Rate Adaptation Protocol (RAP) [12], but their performance is not tested using real Internet traces. Our approach differs in many respects. We evaluate the performance of transmission schemes for single-layer and layer-encoded video based on traces collected from TCP connections. We quantify the impact of prefetching data into client buffers, as well as the required client resources. Finally, we develop a threshold-based policy for streaming scalable CM that is based on estimation of future available bandwidth conditions.

II. Long-term Behavior of Fair-Share Bandwidth

We traced a number of unidirectional TCP bulk data transfers between three pairs of hosts to measure the behavior of the bandwidth that is available to a congestion-aware application. In each data transfer, the server side of the application acted as an *infinite* source, sending bytes as quickly as possible into the TCP connection during one hour. At the receiver, the client application read from the TCP connection as quickly as it could, and recorded measurements on the incoming data stream to determine the throughput of the TCP flow. In particular, for each socket read the receiver recorded the number of received bytes and the interarrival time between successive reads. To ensure that the throughput of the flow was not limited by the receiver's advertised window, the size of the TCP receive buffer was set to the maximum value (64 KB) by modifying the Unix socket options for the connection. Based on these measurements, we collected 1-hour long instantaneous throughput traces for TCP flows during four consecutive days at different times of the day. Three hosts located in different countries participated in the data transfer experiments: host FR located in France was an Ultra-5/10 running SunOS 5.5.1, host US located on the East coast of the United States was an Ultra-1 running SunOS 5.5.1, and host FI located in Finland, was an alpha Workstation running DEC OSF/1. Host FR was connected to a 10 Mbps Ethernet, and hosts US and FI were connected to 100 Mbps Ethernets.

Table I summarizes the collected traces. The average throughput seen by the client application varies considerably depending on the source-destination pair, the direction of the transfer (traces were obtained for both directions of a link), as well as the time of the day. We observe in particular that traces A1-A4 have consistently higher throughput than traces B1-B4, which were collected between the same pair of hosts in the two different directions. Similarly, traces C1-C3 collected from transfers between hosts FI and FR have consistently higher average throughput than traces D1-D3 collected in the opposite direction. (We suspect that these asymmetries in throughput are due to higher link capacity in the direction U.S. to France, or in the direction Finland to France, than in the opposite directions.) Peak throughput is calculated by taking the largest throughput of the trace averaged over all one-second intervals.

We used the instantaneous throughput measurements to study the time-scale behavior of a TCP flow. Figure 1 shows the local averages of traces A1 and A3 computed at two different time scales. The graphs on the left-hand-side illustrate local averages of the traces over 10-second intervals and the graphs on the right show local averages over 100-second intervals. We observe that the throughput traces exhibit a high degree of variability and burstiness over both time scales. Average throughput figures obtained for the remaining traces show similar behavior.

We have taken a simple approach, namely *time-variance plots*, for statistically studying the scaling behavior of the

Trace	Src-Dest.	Throughput (Mbps)			MB
		Peak	Mean	σ	
A1	US-FR	2.41	0.70	0.43	318
A2	US-FR	3.89	1.10	0.82	495
A3	US-FR	3.95	1.93	1.34	869
A4	US-FR	3.99	2.18	1.48	984
B1	FR-US	1.53	0.41	0.23	186
B2	FR-US	1.10	0.28	0.16	125
B3	FR-US	1.13	0.37	0.18	165
B4	FR-US	1.87	0.45	0.23	202
C1	FR-FI	5.91	3.38	1.50	1520
C2	FR-FI	5.98	4.00	1.43	1797
C3	FR-FI	6.07	4.34	1.48	1953
D1	FI-FR	3.01	1.67	1.13	752
D2	FI-FR	3.04	1.79	1.15	807
D3	FI-FR	3.02	1.87	1.19	842

TABLE I
SUMMARY OF 1-HOUR LONG TRACES.

traces. Time-variance plots are useful for determining whether a time series of data is self-similar, and if so, for estimating the self-similarity parameter H (Hurst parameter). For a stationary time series X, the aggregated time series $X^{(m)} = X_k^{(m)}, k = 0, 1, 2, \ldots$ can be obtained by averaging the original series over non-overlapping, adjacent blocks of size m. This aggregating process can be expressed as: $X_k^{(m)} = \frac{1}{m}\sum_{i=km-(m-1)}^{km} X_i$. For a self-similar process, the variance of $X^{(m)}$ obeys the following for large m: $Var(X^{(m)}) \sim \frac{Var X}{m^{\beta}}$, where the self-similarity parameter $H = 1 - (\beta/2), 0 < \beta < 1$. Time-variance plots are obtained by plotting $\log Var(X^{(m)})$ against $\log(m)$. For a self-similar process, the time-variance plot should yield a straight line with slope $-\beta$, from which the Hurst parameter can be estimated. Slope values between -1 and 0 ($0.5 \leq H \leq 1.0$) suggest that the process is self-similar.

Starting with $m = 1$ second, we obtained an initial time average series for each throughput trace. We generated aggregated data series by increasing m by a factor of 2 until a total of ten series were obtained. Our results showed that for all traces the time-variance plots could be closely approximated by a straight line, indicating that the traces exhibit self-similar scaling behavior over longer time scales of seconds to hundreds of seconds. From the time-variance plots we estimated β by fitting a least squares line through the data points. The Hurst parameter values for the 14 traces were in the range $0.60 - 0.87$. From the observations and testing in this section, we conclude that fair-share bandwidth significantly varies on many time scales and exhibits a degree of self-similarity.

III. STREAMING NON-LAYERED CM

Using the traces from the previous section, we now compare the performance of several data transmission schemes for streaming non-layered CM. We consider three types of transmission schemes: full prefetching, no prefetching and partial prefetching.

In the full prefetching scheme the server application transmits the CM into the network at the full rate allowed by the available bandwidth. This implies that when the available bandwidth exceeds the CM consumption rate, future portions of the CM are prefetched into client storage, which we suppose to be infinite. (This assumption is justified by the fact that most PCs being sold today have 10-20 Gbytes of disk.) Let $X(t)$ denote the available bandwidth as seen by the client application at time t. Thus, $X(t)$ is the maximum rate at which the client application can read data from the network at time t. Let r denote the consumption rate of the CM in bits per second. (For simplicity, we assume that the CM is CBR encoded). We suppose that the server application only transmits data that will make their deadline for timely consumption. Thus at time t, the server application transmits the CM at rate $X(t)$, and all the transmitted data will eventually be consumed by the client. To smooth out short time scale bandwidth variations and to remove jitter, the scheme allows for a few seconds of CM to build up in the client's prefetch buffer before playback begins. We denote the initial playback delay by Δ seconds. Denoting the time at which the client begins to receive the CM by $t = 0$, at time $t = \Delta$ the client begins to remove the CM from its prefetch buffer at the consumption rate r. If the prefetch buffer becomes empty, the client can partially receive the CM stream by reading directly from the network while incurring some loss of data. (The received data can be used in this case to approximate the stream using an error concealment scheme.) Thus, during the first Δ seconds the client prefetch buffer is fed at rate $X(t)$. For $t \geq \Delta$, the prefetch buffer is fed at rate $X(t)$ and drained at rate r. Let $Y(t)$ denote the content of the client prefetch buffer at time t. When $Y(t) > 0$, there is no loss. When $Y(t) = 0$, data is lost from the stream at rate $[r - X(t)]^{+}$.

Figure 2 shows the fraction of lost data resulting from the full prefetching scheme as the CM consumption rate r varies from 10 Kbps to 3 Mbps. The two graphs correspond to simulations based on traces A1 and A2 in Table III. An initial prefetch delay of 4 seconds was used. There is a rapid increase in the fraction of lost data when the consumption rate of the stream exceeds a certain critical value. Below this critical value the CM can be streamed without any loss. For trace A1, the CM can be streamed without loss at consumption rates below 490 Kbps. For trace A2, shown on the right, there is no loss at consumption rates below 690 Kbps. When the consumption rate exceeds these critical values, the fraction of lost data for both traces increases rapidly to values in the order of 10^{-2}, indicating that the reconstructed CM would suffer significant quality degradation, even when error concealment is used. Similar graphs, not shown here due to space limitations, were obtained for the remaining of the traces. The graphs demonstrate that the fraction of lost data increases with the consumption rate, but the increase can be more or less gradual depending on the trace. For each of the traces, Table II summarizes the maximum CM consumption rate at which the full prefetching scheme has no loss as well as the maximum consumption rate at which there is loss

Trace name	avg. avail. rate	max. rate for no loss	max. rate for 10^{-3} loss
A1	0.70	0.48	0.54
A2	1.10	0.68	0.76
A3	1.93	1.42	1.85
A4	2.18	1.26	2.01
B1	0.41	0.41	0.41
B2	0.28	0.18	0.22
B3	0.37	0.30	0.34
B4	0.45	0.38	0.42
C1	3.38	2.75	2.92
C2	4.00	3.88	4.06
C3	4.34	4.08	4.32
D1	1.67	1.51	1.64
D2	1.79	1.40	1.75
D3	1.87	1.72	1.88

TABLE II
MAXIMUM CONSUMPTION RATE (MBPS) FOR NO LOSS WITH FULL PREFETCHING.

in the order of 10^{-3}. In general, the maximum CM consumption rate at which there is no loss is at least 70% of the average available bandwidth. The maximum rate at which the loss is below 10^{-2} is typically within 15% of the maximum rate for no loss, indicating that there is only a small range of rates that result in loss fractions in the order of 0 and 10^{-3}.

We now consider an alternative transmission scheme in which the server application never prefetches future portions of the CM stream into client storage (but still allowing for an initial playback delay). In the *no prefetching* scheme the server application transmits the CM at the consumption rate, unless the available bandwidth is less than the consumption, at which time the CM is transmitted at the available bandwidth rate. Thus, the transmission rate at time t is $\min\{X(t), r\}$. Data arrives to the client application buffer at rate $\min\{X(t), r\}$, and after a playback delay of Δ seconds, data is removed from the buffer at rate r. As a result, the amount of prefetched data in the client application buffer never increases after time $t = \Delta$.

We evaluated the no prefetching scheme with a playback delay of 4 seconds according the amount of data lost from the CM stream. Figure 2 shows the results for traces A1 and A2. The no prefetching scheme results in significantly higher loss than prefetching. Even at very low consumption rates, the no prefetching scheme generates losses in the order of 10^{-1}. High fractions of lost data occur because this scheme fails to smooth out the short time scale variations in available bandwidth.

We refer to a third transmission scheme that has the effect of partially smoothing the available bandwidth over intervals of short length, as *partial prefetching*. Figure 3 describes the partial prefetching scheme. The server application transfers the CM from storage to the server application buffer. We consider time to be divided into intervals of m seconds and suppose that at the beginning of each interval the application reads from the storage $r \cdot m$ bits of information to the server application buffer. Let $Y_s(t)$ denote the amount of data in the sender application buffer at time t. In each m second interval, the application buffer writes to the network at rate $X(t)$ as long as $Y_s(t) > 0$. If at the end of any interval $Y_s(t) > 0$, the buffer is flushed (i.e., the data remaining in the buffer at the end of the interval is never sent into the network and is lost). Thus, $Y_s(t) = 0$ for $t = 0, m, 2m, \ldots$. We let $R(t)$ denote the rate at which data arrives to the client transport-layer buffer, i.e.,

$$R(t) = \begin{cases} X(t) & \text{when } Y_s(t) > 0 \\ 0 & \text{when } Y_s(t) = 0. \end{cases}$$

As soon as data is received at the client, the application in the client reads the data from the transport-layer buffer into the receiver application buffer. After a playback delay of Δ seconds, the client begins to remove data from the receiver application buffer for decoding and play out. Again denote $Y(t)$ for the amount of data in the client prefetch buffer. The rate at which the client application buffer is drained for $t \geq \Delta$ is

$$D(t) = \begin{cases} r & \text{when } Y(t) > 0 \\ \min\{r, R(t)\} & \text{when } Y(t) = 0. \end{cases}$$

Loss of data occurs if the application buffer is empty, that is $Y(t) = 0$ and $R(t) < r$. The rate at which data is lost from the CM stream at time t can be expressed as:

$$L(t) = [r - R(t)]^{+}\mathbf{1}(Y(t) = 0).$$

We evaluated the partial prefetching scheme while varying the interval size m. We refer to m as the *smoothing parameter*. As m increases, information is written to the sender application buffer at the beginning of each interval in larger blocks. Writing in larger blocks allows smoothing of the transmission rate over the interval's length. A smoothing parameter $m = \infty$ corresponds to the full prefetching scheme whereas $m \rightarrow 0$ corresponds to the no prefetching scheme.

Figure 2 shows the fraction of lost data resulting from partial prefetching schemes with smoothing parameters $m = 4$ and $m = 60$ seconds as the CM consumption rate varies from 10 Kbps to 3 Mbps. The initial prefetch delay was again 4 seconds. The figure illustrates that the performance of partial prefetching degrades considerably as the smoothing parameter decreases. Furthermore, full prefetching results in significantly lower loss than the partial prefetching schemes considered. For trace A1, the maximum consumption rate for which there is no loss when $m = 60$ seconds is approximately 70% of the maximum rate for no loss achieved with full prefetching. Although the performance of partial prefetching schemes can vary among different traces, our results demonstrate that in order to obtain the performance of the full prefetching scheme, partial prefetching must use smoothing parameters of at least *three* minutes. Table III summarizes some of the results obtained with trace A1 for full prefetching and partial prefetching with $m = 4$ and $m = 60$.

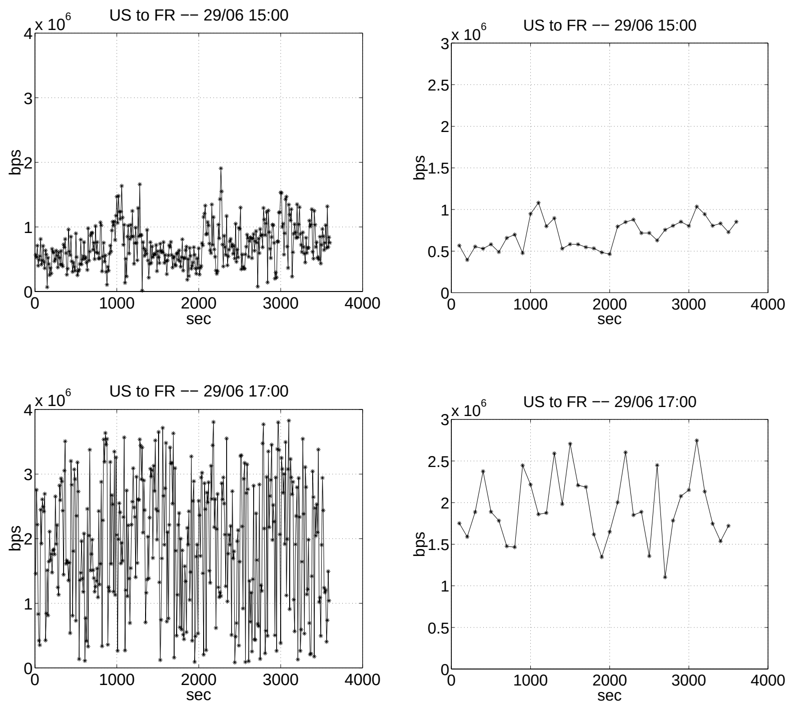


Fig. 1. Average throughput over time scales of 10 and 100 seconds for traces A1 and A3.

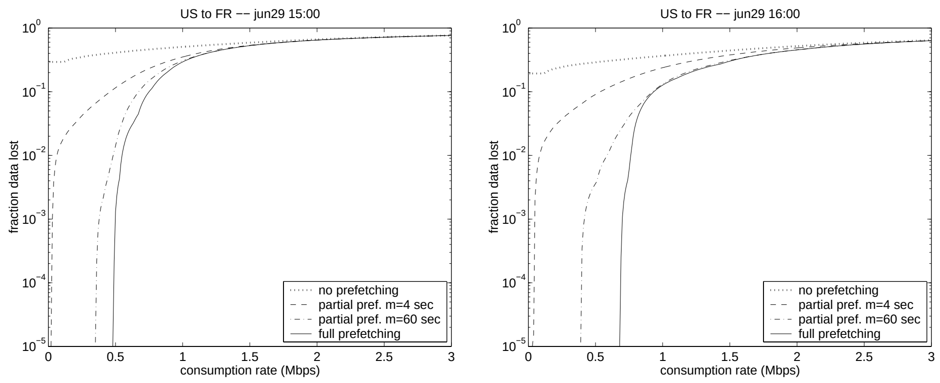


Fig. 2. Fraction of data lost for full prefetching, no prefetching and partial prefetching transmission schemes with traces A1 and A2.

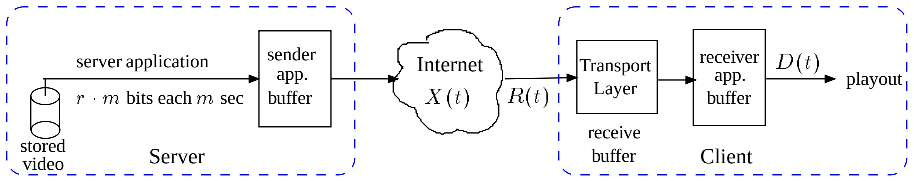


Fig. 3. End-to-end architecture for streaming stored video.

	Full Prefetching		Partial Pref. ($m = 60$)		Partial Pref. ($m = 4$)	
Rate (Mbps)	Loss Fraction	Max Buf. Size (min)	Loss Fraction	Max Buf. Size (min)	Loss Fraction	Max.Buf. Size (min)
0.48	0.00	16.08	9.45e-03	1.00	1.07e-01	0.04
0.49	2.14e-04	15.28	1.16e-02	1.00	1.11e-01	0.04
0.50	1.31e-03	14.38	1.41e-02	1.00	1.15e-01	0.04
0.55	1.04e-02	11.37	3.31e-02	0.98	1.36e-01	0.35

TABLE III
FULL PREFETCHING AND PARTIAL PREFETCHING SCHEMES WITH TRACE A1.

In conclusion, our empirical results for streaming CM under fair-share bandwidth conditions indicate that prefetching is critical. Prefetching over intervals of a few minutes can effectively smooth out short time-scale bandwidth variations as well as long-term droughts in bandwidth. Our results indicate, however, that even the full prefetching scheme can not fully utilize the average available bandwidth. This is clearly seen in Table II where we observe that the maximum consumption rate at which the CM can be streamed with no loss is typically $60 - 70\%$ of the average available rate.

An interesting secondary observation is that TCP with prefetching can provide near optimal performance for streaming over fair-share bandwidth for a wide range of consumption rates. For a given trace, we observe two thresholds, t_{good} and t_{bad}. If the consumption rate is below t_{good}, then TCP with prefetching has no loss, and is therefore better than any TCP-friendly UDP scheme. If the consumption rate exceeds t_{bad}, then neither TCP nor UDP schemes will give satisfactory quality. If the consumption rate is between these thresholds, then a TCP-friendly UDP scheme might provide superior performance. However, the gap between these two thresholds is typically very small.

IV. STREAMING LAYERED VIDEO

For the remaining of this paper we consider layer-encoded CM. Layered streaming is an appealing streaming solution for CM applications that need to cope with the wide and random variations of the available bandwidth between server and client. In general, the CM stream is encoded into a base-layer stream and one or multiple enhancement-layer streams. We suppose that the CM is encoded into two layers, a base layer and an enhancement layer. There are advantages to using only a single enhancement layer. A single enhancement layer requires only a single enhancement-layer decoder at the receiver, and adds little coding overhead [13]. When CM is encoded into two layers, a decoding constraint requires the base layer to be available at the client in order to decode the enhancement layer. If only the base layer is delivered, a CM signal can still be reconstructed, resulting in a low but acceptable quality at the receiver. The streaming application should make the enhancement layer stream available for decoding whenever possible, to achieve high quality during playback. At the same time, the application should maintain a relatively constant quality by avoiding frequent and short rendering periods of the enhancement layer.

A. The Layered Streaming Model

We now provide a description of the layered streaming model. For simplicity, we suppose that the base and enhancement layers have been encoded at constant rates, denoted by r_b and r_e, respectively. We let T denote the length of the CM in seconds. Let $X(t)$ denote the available bandwidth at time t. We suppose that the server always transmits the CM at the rate allowed by the available bandwidth. We let $\pi_b(t)$ and $\pi_e(t)$ denote the fraction of $X(t)$ that the server allocates at time t to the base and enhancement layers, respectively. Let $Y_b(t)$ and $Y_e(t)$ be the contents of the base and enhancement layer prefetch buffers (at the client) at time t, respectively. Data is lost from the base layer when the base-layer prefetch buffer is starved, i.e., when $Y_b(t) = 0$, and $X(t) < r_b$. The loss rate in the base layer at time t can be expressed as $L_b(t) = [r_b - \pi_b(t)X(t)]^+ \mathbf{1}(Y_b(t) = 0)$. Loss of enhancement-layer data occurs when there is loss of base-layer data, or when the enhancement-layer buffer at the client is starved. Multiple models can be used for determining loss in the enhancement layer. [7]. In the numerical work that follows, we use a model in which all enhancement-layer data that reach the receiver buffer can be decoded as long as the corresponding base-layer data is also available at the receiver. The loss rate in the enhancement layer at time t can be expressed as $L_e(t) = \max\{\frac{r_e}{r_b} L_b(t),\ [r_e - \pi_e(t)X(t)]^+ \mathbf{1}(Y_e(t) = 0)\}$.

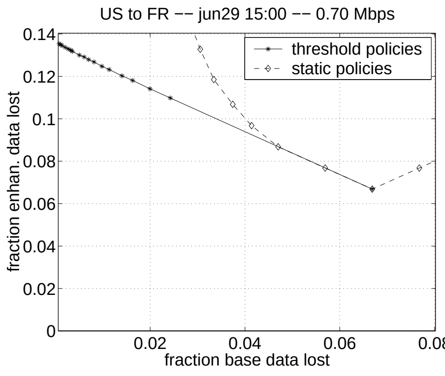
We consider two classes of inter-layer bandwidth allocation schemes, namely, *static* and *threshold* policies. With static policies, the allocation $\pi_b(t)$ is constant until one of the layers is fully prefetched. Under a static policy, denoted by α_b, $0 \leq \alpha_b \leq 1$, the base-layer prefetch buffer is fed at rate $\alpha_b X(t)$ and the enhancement-layer prefetch buffer is fed at rate $\alpha_e X(t)$, where $\alpha_b + \alpha_e = 1$. Under a threshold policy, denoted by $\hat{\pi}$, the fraction of bandwidth allocated to each layer varies according to the buffer contents. A threshold policy is defined as $\hat{\pi} = (\pi_b(t), t \geq 0)$, where $\pi_b(t)$ at time t is given by

$$\pi_b(t) = \begin{cases} 1 & \text{when } Y_b(t) < q_{\text{thres}} \\ \hat{\alpha} & \text{when } Y_b(t) \geq q_{\text{thres}} \\ 0 & \text{when } Y_b(t) > r_b(T-t). \end{cases}$$

The constant q_{thres} represents a threshold of buffered base-layer data. When the amount of base-layer data buffered at the receiver is below q_{thres}, policy $\hat{\pi}$ allocates all of the available bandwidth to the base layer. When the amount of buffered base-layer data exceeds q_{thres}, then a fraction equal to $\hat{\alpha}$ is given to the base layer. Once the base layer has been prefetched, all of the bandwidth is allocated to the enhancement layer. We evaluate the performance of threshold policies for which $\hat{\alpha} = \frac{r_b}{r_b+r_e}$. Intuitively, allocation $\hat{\alpha}$ allocates bandwidth among the layers in proportion to each layer's consumption rate.

B. Trace-driven Simulation

In our numerical work, we suppose for simplicity, that both layers have equal encoded consumption rates. We evaluate the performance of static and threshold bandwidth allocation policies using the collected TCP traces.



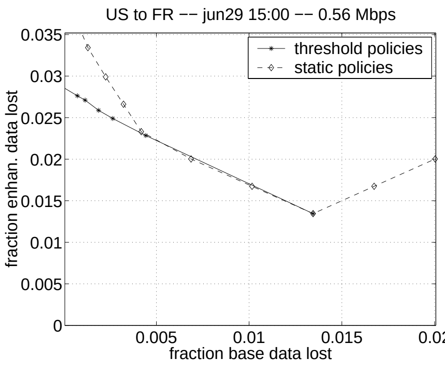


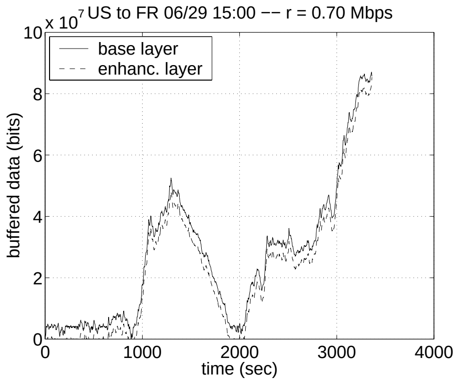
Fig. 4. Fraction of data lost with static and threshold policies, with r equal to 100 and 80% of the average available bandwidth.

Figure 4 shows our results with trace A1. The average throughput of this one-hour long trace is approximately 700 Kbps. We performed a series of simulations in which we varied the total consumption rate of the encoded CM between 100% and 70% of the average trace bandwidth. At each consumption rate, we evaluated the performance of several static policies, obtained by varying the constant allocation of bandwidth to each layer, and the performance of threshold policies, obtained by varying the threshold for buffered base-layer data. The graphs in Figure 4 show the fraction of data lost from the two layers at consumption rates equal to 100% and 80% of the average available bandwidth. Both types of policies can render the base layer without loss, but in all cases a significant fraction of data is lost from the enhancement layer. This is not a surprising result, given our observations in the previous section, indicating that with non-layered video, no loss is achieved at consumption rates that are typically below 70% of the average available bandwidth.

Our results show that threshold policies attain significantly better perfromance than static policies. We compare the performance of two specific policies: the threshold policy that renders the base layer with no loss, while minimizing the fraction of data lost from the enhancement layer, and that of the static policy that achieves similar performance. At a rate equal to the average connection bandwidth ($r = 0.7$ Mbps), a static policy ($\alpha_b = 0.72$) results in 35% of the enhancement data lost, when there is no loss in the base layer. A threshold policy ($q_{\text{thres}} = 4$ Mbits) results in 14% of the enhancement layer lost without any base layer loss. At a consumption rate equal to 80% of the bandwidth ($r = 0.56$ Mbps), a threshold policy results in no base layer loss and 2.87% of the enhancement data lost; a static policy results in no base-layer loss and 4.3% of the enhancement layer lost.

Rapid fluctuations in playback quality are undesirable. We focus on the threshold policy discussed in the previous paragraph, which renders the base layer with no loss while minimizing the fraction of data lost from the enhancement layer. We will refer to this policy as the "optimal" threshold policy, and examine its performance in terms of the resulting fluctuations in quality. Figure 5 shows the amount of data buffered in each layer as a function of time under the optimal threshold policy, for consumption rates equal to 100% and 80% of the average available bandwidth. We remark that there is an initial period during which the enhancement layer buffer is frequently drained and starved after very short filling periods. Following this initial period, the enhancement layer can be fully supported for longer periods of time. For example, at a consumption rate equal to the average available bandwidth ($r = 0.7$ Mbps), a constant high quality can be achieved with both layers shortly after the 15th minute into the CM. After the 34th minute into the video, the enhancement layer stream can be played back without further loss. At 80% of the average available bandwidth ($r = 0.56$ Mbps), the enhancement layer can be continuously supported after the 10th minute into the video.

We performed a similar experiment for all of the fair-share bandwidth traces. Table IV summarizes some of our results In all cases, the threshold policy implemented could stream the base layer with no loss. Each entry in the table indicates the time after which the enhancement layer can be played back continuously without loss In most cases, when the consumption rate equals the average available bandwidth, the enhancement layer can not be played back continuously from the beginning of the video. In many cases, continuous playback of the enhancement layer is not achieved until halfway through the video, as is for instance the case with traces A1, A2, A4 or D2. We observe that for traces B2 and B4, the enhancement layer is rendered con-



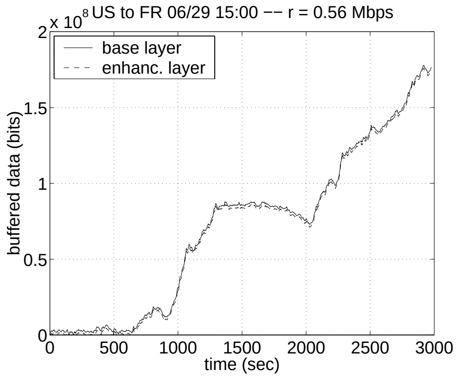


Fig. 5. Buffered data for threshold policies resulting in no base layer loss ($r = 100$ and 80% of average available bandwidth).

tinuously for an even shorter period of time (i.e. the final $15 - 20$ minutes of the video). At 85% of the average available bandwidth, the enhancement layer can be played back continuously without loss nearly from the beginning of the video for most traces. At 75% of the average available bandwidth the threshold policy can in almost all cases deliver both layers without loss from the beginning of the video.

We remark that the results obtained with traces A2 and B1 differ from those obtained with the remaining traces. The average throughput evolution for trace A2 (Fig. 1) exhibits an increasing trend. The average connection bandwidth is considerably higher than the bandwidth available during the first half of the connection. Consequently, the enhancement layer can not be supported early on into the video, even at 75% of the average trace bandwidth. For trace B1, the average throughput evolution exhibits a decreasing trend resulting in unusually good performance at all consumption rates considered.

In summary, layered streaming requires an inter-layer bandwidth allocation scheme that adapts to the long-term fluctuations in the available bandwidth in order to deliver the highest quality stream possible. At the same time, the scheme should avoid rapid fluctuations in quality. Our results in this section demonstrate that a threshold-based allocation scheme that varies the prioritization of the two layers according to the amount of data buffered at the client can achieve good peformance. Threshold policies perform significantly better that static policies when the available bandwidth exhibits long-lived fluctuations. On the other hand, when available bandwidth does not fluctuate over longer time scales (such bandwidth conditions were not considered in this paper), then static policies would perhaps provide good performance as well. We observe, however, that threshold policies can result in rapid fluctuations in playback quality. Since future bandwidth conditions are not known, the threshold policy may result in frequently adding and dropping the enhancement layer. In the following subsection we develop a threhold-based heuristic scheme for adding and dropping the enhancement, which also aims at reducing quality fluctuations.

C. *Dynamic Threshold Policies based on Bandwidth Estimation*

In this subsection we develop a heuristic for adding and dropping the enhancement layer according to conservative estimates of average bandwidth conditions, and the amount of data buffered at the client. Recall that threshold policy $\hat{\pi}$ allocates bandwidth to the enhancement layer according to the static policy $\hat{\alpha}$ only when the amount of buffered base-layer data is above threshold q_{thres}. Otherwise, policy $\hat{\pi}$ allocates all of the available bandwidth to the base layer. Our approach is to allow the server-side of the application to dynamically determine q_{thres}. To this end, we suppose that it is desirable to render the entire base layer without loss. At all times, the server should estimate whether it is likely that there will be no starvation of the base-layer prefetch buffer at the client, assuming that the scheme will henceforth allocate a fraction of the available bandwidth equal to $\hat{\alpha}$ to the base layer. The estimate of the likelihood of no starvation depends on (i) the amount of base-layer data buffered at the receiver, (ii) the drain rate of base-layer data, and (iii) a conservative estimate of future available bandwidth.

To estimate future available bandwidth, we use a weighted exponential moving average (WEMA) of the past and current bandwidth observations. We denote the estimate of the average available bandwidth from time s onward by $X_{\text{avg}}(s)$. Given this estimate we determine the threshold at time s as follows:

$$q_{\text{thres}}(s) = (T - s)r_b - \hat{\alpha}(T - s) \cdot X_{\text{avg}}(s),$$

The above expression represents a conservative choice of the threshold. It “guarantees” (based on the estimated future available bandwidth) that the base-layer buffer will not be starved during the remaining viewing duration if at any time s, policy $\hat{\pi}$ begins to allocate bandwidth to the enhancement layer according to policy $\hat{\alpha}$. A less conservative choice for the threshold at time s maintains the no-starvation condition during the next C seconds: $q_{\text{thres}}(s) = C \cdot (r_b - \hat{\alpha} \cdot X_{\text{avg}}(s))$. As the value of C decreases in the above equation, the threshold choice becomes less conservative, allowing the server to begin streaming the enhancement layer sooner. In order to reduce rapid fluctua-

Trace	Rate (% avg. bandwidth)			Trace	Rate (% avg. bandwidth)		
	100 %	85 %	75 %		100 %	85 %	75 %
A1	33.90	10.78	3.41	B1	0.07	0.07	0.07
A2	37.08	30.72	30.63	B2	37.36	21.65	3.71
A3	13.78	1.12	1.05	B3	46.95	2.47	0.07
A4	28.87	0.33	0.32	B4	45.85	0.07	0.07
C1	8.18	1.27	0.07	D1	20.00	0.07	0.07
C2	10.10	0.07	0.07	D2	32.98	0.20	0.07
C3	10.90	0.07	0.07	D3	4.78	0.07	0.07

TABLE IV

TIME IN MINUTES (INCLUDING A PLAYBACK DELAY OF 0.07 MINUTES) AFTER WHICH THE ENHANCEMENT LAYER CAN BE PLAYED BACK WITHOUT LOSS. THE RESULTS ARE FOR THRESHOLD POLICIES WITH NO BASE LAYER LOSS.

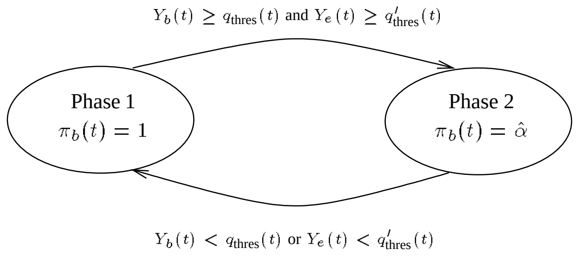

$Y_b(t) \geq q_{\text{thres}}(t)$ and $Y_e(t) \geq q'_{\text{thres}}(t)$
Phase 1
$\pi_b(t) = 1$
Phase 2
$\pi_b(t) = \hat{\alpha}$
$Y_b(t) < q_{\text{thres}}(t)$ or $Y_e(t) < q'_{\text{thres}}(t)$

Fig. 6. State transition diagram for the dynamic threshold heuristic.

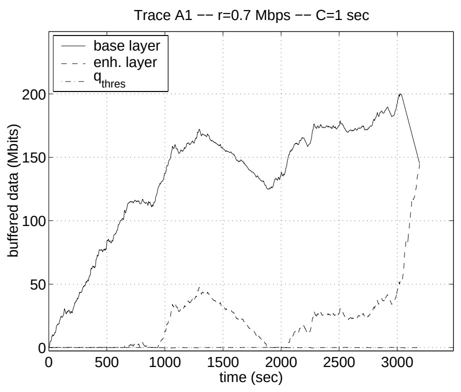
tions in quality caused by frequently adding and dropping the enhancement layer, our heuristic uses an additional condition for adding the enhancement layer. The second condition entails adding the enhancement layer only if it is likely, given the estimated future available bandwidth, that the enhancement layer can be supported for at least the next C' seconds. We express the latter condition by defining a threshold for buffered enhancement-layer data, denoted by q'_{thres}, and determined at time s as follows: $q'_{\text{thres}}(s) = C' \cdot (r_e - (1 - \hat{\alpha}) \cdot X_{\text{avg}}(s))$.

The server streams the enhancement layer at time s if the amount of buffered enhancement layer data $Y_e(s)$ is greater that $q'_{\text{thres}}(s)$. The add and drop mechanisms utilized by our dynamic threshold heuristic are illustrated in Figure 6. In phase 1, the streaming policy allocates all of the available bandwidth to the base layer. In phase 2, bandwidth is allocated among the layers in proportion to the consumption rates of the layers.

To implement the above heuristic, the server side of the application must determine a reasonable prediction interval for future average bandwidth conditions. The length of the prediction interval determines how often to update the bandwidth estimate and how often to recompute the threshold levels. In addition, the application must determine suitable values for C and C'.

C.1 Trace-driven simulations

We evaluated the performance of the dynamic threshold heuristic in simulations using the fair-share bandwidth traces. Fig. 7 shows results obtained with trace A1, when bandwidth estimates and threshold levels are updated each



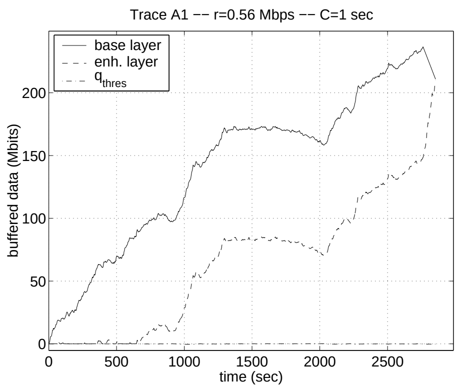


Fig. 7. Buffered data for dynamic threshold policies with $C = C' = 1$ second.

second. C and C' were both set to one second. Each of the graphs shows the amount of buffered data in each layer, as well as the computed threshold level, for different total consumption rates (in this case $q_{\text{thres}}(s) = q'_{\text{thres}}(s)$). We observe that at both consumption rates the base layer is rendered without loss. When the total consumption rate is equal to 100% of the average available bandwidth ($r = 0.7$ Mbps), approximately 25% of the data is lost from the enhancement layer. The fraction of enhancement-layer loss is higher in this case than in the case of the "optimal" threshold policy when 14% of the data was lost.(Figure 5). We remark from Fig. 7, however, that the heuristic renders the enhancement layer for two long and continuous periods, and avoids the initial rapid fluctuations in quality that are observed under the threshold policy. This is clearly seen in Fig. 8, which compares the buffer content under the threshold policy and under the heuristic during the first 16 minutes of the video, when the consumption rate equals the average available bandwdith. Furthermore, we observe from Fig. 7 that the heuristic results in approximately the same high-quality viewing time as the threshold policy, delivering the enhancement layer without further loss after the 34th minute into the video.

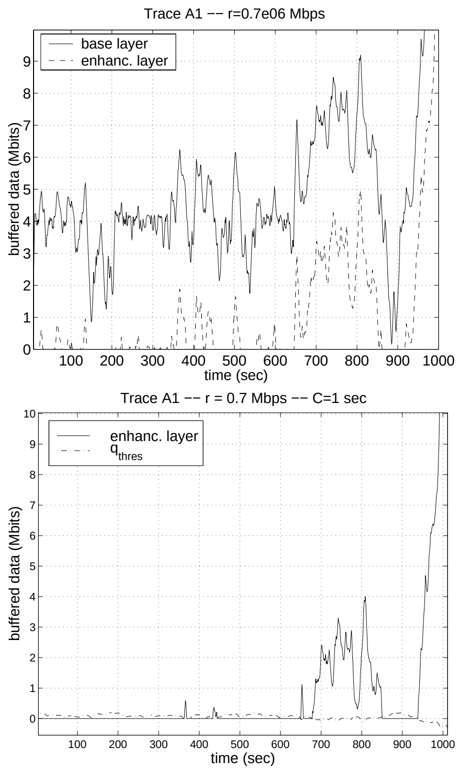


Fig. 8. Quality flutctuations using the threshold policy and using the dynamic threshold heuristic (bottom).

We performed similar simulations for the remaining traces using $C = C' = 1$ second. Again bandwidth estimates and threshold levels were updated every second. Table V summarizes some of our results when the total consumption rate is equal to 85% of the average available bandwidth. Each entry in the table indicates the time after which the enhancement layer can be played back continuously without loss. The results are only marginally different from those in Table IV corrsponding to the "optimal" threshold policy. Thus our heuristic inter-layer allocation scheme based on conservative bandwidth estimates is reliable in avoiding loss of base-layer data, while successfully adapting to the higher bandwidth conditions.

V. Conclusion

We studied schemes for streaming stored CM over fair-share bandwidth traces. We observed that fair-share bandwidth fluctuates significantly over a broad range of time scales, and that to achieve good performance streaming schemes must prefetch CM during playback over intervals that are several minutes long. Additionally, we showed that layered encoding is desirable for streaming CM over fair-share bandwidth and that a simple threshold allocation scheme provides good performance. We developed a heuristic threshold-based bandwidth allocation scheme for streaming two layers that is based on conservative estimates of average bandwidth conditions. Our empirical results indicated that such a measurement-based heuristic attains good performance.

Trace	Time (min)	Trace	Time (min)
A1	10.80	B1	0.48
A2	30.72	B2	21.78
A3	1.32	B3	6.23
A4	0.98	B4	5.83
C1	2.60	D1	0.10
C2	0.30	D2	0.38
C3	0.40	D3	0.13

TABLE V

High-quality viewing with dynamic threshold heuristic.

References

[1] S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in the Internet," *IEEE/ACM Trans. on Networking*, vol. 7, no. 4, pp. 458–472, Aug. 1999.

[2] J. Mahdavi and S. Floyd, "TCP-Friendly Unicast Rate-Based Flow Control," Tech. Rep., Jan. 1997.

[3] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The macroscopic behavior of TCP congestion advoidance algorithm," *Computer Communications Review*, vol. 27, July 1997.

[4] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP Throughput: A Simple Model and its Empirical Validation," in *Proc. of ACM SIGCOMM*, Sept. 1998.

[5] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven Layered Multicast," in *Proc. ACM SIGCOMM*, Stanford, CA, Aug. 1996.

[6] J. Padhye, J. Kurose, D. Towsley, and R. Koodli, "A model based tcp-friendly rate control protocol," in *IEEE NOSSDAV'99*, Basking Ridge, NJ, June 1999.

[7] D. Saparilla and K. W. Ross, "Optimal Streaming of Layered Video," in *Proc. of IEEE Infocom*, Tel Aviv, Israel, March 2000.

[8] A. Feldman, A. C. Gilbert, W. Willinger, and T. G. Kurtz, "The changing nature of network traffic: Scaling phenomena," *Computer Communication Review*, vol. 28, no. 2, Apr. 1998.

[9] M. E. Crovella and A. Bestavros, "Self-similarity in World Wide Web Traffic - Evidence and Possible Causes," in *Proc. of ACM Sigmetrics*, 1996, pp. 160–169.

[10] V. Paxson and S. Floyd, "Wide Area Traffic: the Failure of Poisson Modeling," *IEEE/ACM Transactions on Networking*, vol. 3, pp. 226–244, 1995.

[11] R. Rejaie, M. Handley, and D. Estrin, "Quality Adaptation for Congestion Controlled Video Playback over the Internet," in *Proc. of ACM SIGCOMM*, Camb., MA, Sept. 1999, number 4, pp. 189–200.

[12] R. Rejaie, M. Handley, and D. Estrin, "RAP: An End-to-End Rate-Based Congestion Control Mechanism for Realtime Streams in the Internet," in *Proc. of IEEE INFOCOM*, New York, Mar. 1999.

[13] H. Radha, Y. Chen, K. Parthasarathy, and R. Cohen, "Scalable Internet Video Using MPEG-4," *Signal Processing: Image Communication*, vol. 15, pp. 95–126, 1999.