Hindawi Publishing Corporation
Journal of Computer Systems, Networks, and Communications
Volume 2009, Article ID 682813, 13 pages
doi:10.1155/2009/682813

Research Article

A Cross-Layer Framework for Efficient Streaming of H.264 Video over IEEE 802.11 Networks

Azfar Moid and Abraham O. Fapojuwo

Department of Electrical and Computer Engineering, University of Calgary, AB, Canada T2N 1N4

Correspondence should be addressed to Azfar Moid, amoid@ucalgary.ca

Received 19 November 2008; Revised 1 March 2009; Accepted 20 April 2009

Recommended by Bechir Hamdaoui

This paper presents a framework for reliable and efficient streaming of H.264 video over an IEEE 802.11-based wireless network. The framework relies on a cross-layer mechanism that jointly adapts the video transcoding parameters at the application layer and the video transmission parameters at the data-link layer to the network conditions defined by buffer length and wireless propagation channel. The effectiveness of the proposed framework is demonstrated through the transmission of three test video sequences (*Akiyo, Container*, and *Foreman*) having different degrees of motion over an IEEE802.11 wireless network. Simulation results show that the proposed cross-layer-based framework provides an enhancement of up to 3 dB in the video quality with a negligible increase (<5%) in the packet processing time. Hence, the proposed framework achieves a good balance in the tradeoff between video quality and packet processing time. The proposed framework, along with its performance results, provides valuable insights on the selection of network parameter values for efficient and reliable transmission of video applications in wireless networks.

1. Introduction

In recent years, the advances in efficient video compression technologies have made it possible for the transmission of video applications over bandwidth constrained wireless channels. The H.264 video format, which is a latest state-of-the-art international video coding standard developed by the joint video team (JVT) of ITU-T and ISO/IEC [1], is recently adopted as a dominant video coding standard in mobile broadcasting and in other advanced video streaming networks. Due to its excellent compression efficiency and ability to adapt to different mobile devices [2], service providers, such as online video storage and telecommunications companies, are also beginning to adopt H.264 to their system model. As far as real-time video streaming over wireless channel is concerned, researchers have already started working toward adapting the H.264 standard for video streaming applications [3], thereby making real-time transmission of the time-sensitive video information possible over bandwidth-constrained wireless channels.

Currently, video is pre-encoded at the content provider's server at different bit-rates, and the clients select the video stream based on their requirements. This method is not an intelligent way of streaming the video content because the network resources can sometime be overused or under-utilized. The use of real-time transcoder, also called transcoder-on-fly, is suggested in literature [4] to adapt the video stream to the network conditions, defined by buffer length and wireless propagation channel. Video transcoding is a process of recompressing a video stream according to the end-user's requirement. The essential part of the process is to closely meet the constraints of the target applications. For example, in the wireless domain, constraints include the varying channel conditions, available transmission bandwidth, current traffic load, desired spatial or temporal resolution, delay allowance, error resilience, and so forth. Homogeneous and heterogeneous transcoding are possible where, in the former, the conversion of bit-stream is done within one video standard and, in the latter, bit-stream conversion is done using multiple video standards. As the H.264 standard is efficient in both the storage and transmission, in this work, we have focused only on the homogeneous transcoding method.

Among the existing techniques of transcoding, the bit-rate reduction techniques are the most efficient ones [5], which provide the dynamic adjustment of bit-rate to meet the conditions of required output video stream. Usually, the bit-rate of the compressed video can be adjusted by changing the quantization parameter (QP) at the re-encoding process, where larger quantization steps (QSs) are used for generating lower bit-rate video stream. When the channel condition is bad, the transmitted video is encoded at a lower bit-rate, to avoid any retransmission delay and packet-loss. This adaptation does not only provide a smooth video quality at the end device but also minimizes the load on the network.

Two different types of problems are typically coexisting in wireless video streaming networks [6, 7]: (1) stabilizing the video buffers at the application layer, and (2) providing efficient error-resilience functionality at the data-link layer. Conventionally, the error-resiliency is provided by injecting the redundant packets in the video stream, but this injection destabilizes the video buffer at the decoder side. Hence, both the aforementioned problems need a joint treatment. As channel adaptive video transcoding is proven to be the most efficient way for video streaming over the wireless channel [8], there is a requirement of redefining the transcoder and decoder buffer dynamics for variable bit-rate encoded videos. At the same time, care must be taken to avoid any packet dropping at the decoder side due to exceeding the deadline time limit. In this paper, both the buffer constraint at the application layer and the time constraint at the data-link layer are jointly treated using a cross-layer mechanism, to achieve an efficient video streaming solution.

Based on the specific applications and requirements, different bit-rate reduction transcoding algorithms are proposed in literature. Recently, a significant improvement in transcoding efficiency is reported in [9], where the authors propose a rate-distortion- (RD-) based model, using different Lagrangian multipliers in the pixel and transform domains to obtain the optimum results. The RD cost is minimized in both pixel and transform domains and experimental results show that the proposed transcoding model provides a good balance in the tradeoff between high performance and transcoding speed. An error-resilient transcoding scheme is presented in [10], where RD-optimized intra- and intermode decisions are made, based on the impact of channel errors propagated to the next frame. The proposed scheme in [10] enhances the performance of the error-resilient transcoder and improves the robustness of the generated bit-stream against packet loss. High peak-signal-to-noise-ratio (PSNR) improvement is also achieved due to the error-resilience property of the scheme proposed in [10]. The authors in [11] have shown that for constant bit-rate video coding, the encoder buffer size can solely be maintained by changing the decoder buffer size according to the bit-rate conversion ratio and transcoder buffer size. Although the constant bit-rate videos are more sensitive to varying channel errors, still the finding that the transcoder buffer can be controlled by the decoder buffer is phenomenal [11] and this has been exploited since then. For example, a fuzzy-logic-based congestion control algorithm has been developed in [12], which changes the sending rate of a video transcoder based on the packet description, instead of using the feedback information of packet loss. A cross-layer packetization and retransmission technique for delay sensitive applications over wireless networks is presented in [13], where the proposed greedy algorithm takes advantage of the available information on retransmission attempts at the medium access control (MAC) layer for improving the streaming video quality. Although the approach presented in [13] might be useful for wavelet coders, where some sub-bands are more important than the others, this scheme cannot be generalized to H.264 video encoders because of the equal priority of all P-frames (i.e., predicted frames). However, the idea of using the retransmission information available at the MAC layer is attractive and can be utilized to assess the network conditions.

The motivation for this work comes from the fact that to the best of our knowledge, a framework for video streaming over the wireless network, considering both the application and data-link layer constraints, is missing in literature. In this paper, we present a cross-layer-based framework for efficient transcoding of the incoming video stream, where a joint treatment of the application layer buffer stabilization and data-link layer error resiliency is considered. This serves as the paper's main contribution.

The paper is organized as follows. Section 2 is the core of the paper where the proposed cross-layer-based video streaming framework is described in detail along with analysis of the relevant parameters. The performance of the proposed framework and simulation results are presented and discussed in Section 3. Section 4 concludes the paper.

2. Video Streaming Framework

2.1. Preliminaries. The proposed cross-layer-based framework comprises a cross-layer module (CLM) that interfaces with the application and data-link layers of the TCP/IP protocol stack, as shown in Figure 1. The CLM consists of four main elements, which are briefly summarized here and their detailed treatment is provided in later subsections. First, the channel estimator is the nucleus of the CLM responsible for estimating the current channel conditions, which are extracted from the information on the packet transmission attempts available at the data-link layer. The estimated channel information is then fed to the buffer controller, transcoding controller, and FEC/ARQ controller, as shown in Figure 1. The second element of the CLM is the buffer controller, which uses the channel information from the channel estimator to control the application layer's buffer overflow/underflow. The third element of the CLM is the transcoding controller, which calculates the video transcoding rate in real-time, based on the information available from the channel estimator and the buffer controller. The final element of the CLM is the FEC/ARQ controller, which optimally calculates the number of redundant FEC packets required for providing the error resilient functionality, based on the estimated channel information. The paper shows that by combining all the four elements of proposed framework, it is possible to achieve an efficient and reliable video streaming

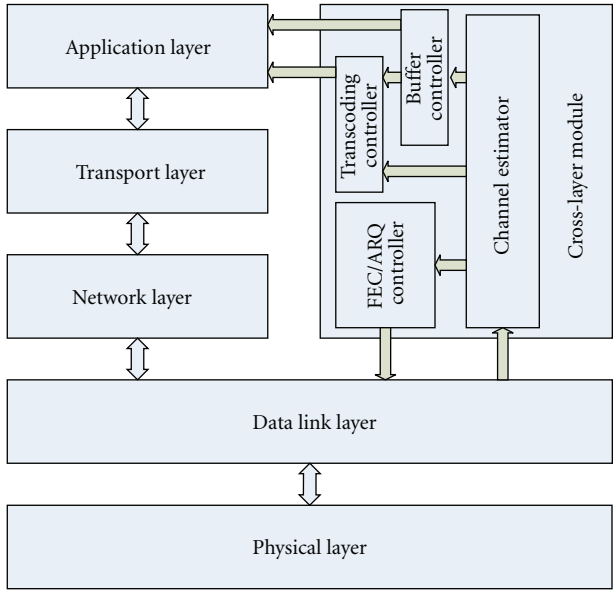

Figure 1: Augmented TCP/IP stack with the proposed cross-layer module (CLM).

over wireless networks. It is important to note here that the application layer buffer management and video rate calculation functionalities can be generalized to any video format and any wireless network. However, the data-link layer error resilient functionality can only be applied to the protocols where FEC and ARQ schemes can be implemented, such as IEEE 802.11 wireless network. Hence, in this work, we have only considered the infrastructure mode IEEE 802.11 wireless network, where a video client is connected to an access point (AP), as illustrated in Figure 2. The incoming video frames are stored at the transcoder buffer before the transcoding operation starts at the AP, where the proposed CLM has also been implemented. In the ensuing analyses and simulations, the transmitter refers to the AP, while the video client is termed as the receiver. The decoder functionality is implemented at the receiver, where video frames are eventually received, buffered, and rendered to the client's terminal. Also, throughout the paper, a packet refers to an IEEE 802.11 data-link layer protocol data unit whereas a frame denotes a video frame at the application layer. In this paper, the terms CLM and cross-layer framework will also be used interchangeably. A slow varying wireless channel is considered in which the channel state does not change during the transmission of one frame. The preceding statement implies that a form of microdiversity is implemented at the AP and wireless client device to combat fast fading.

2.2. Channel Estimation. As the wireless channel varies unpredictably over time and space dimensions, the first step in building the proposed framework is estimating the current wireless channel condition. Conventionally, the channel errors are characterized by the average bit-error rate (BER). According to [14], accurate modeling of the BER requires knowledge of channel coding schemes and the modulation type used. For IEEE 802.11 wireless local area network (WLAN), the modulation types and channel coding schemes vary dynamically based on the data-rate and channel errors [15]. Therefore, instead of using a conventional channel estimator for IEEE 802.11 channel that extracts the information directly from the physical layer, we have used the information of the packet transmission attempts from the data-link layer to estimate the required channel information, which is also consistent with [13].

In a wireless environment, it is important to note that the channel state information (CSI) available at the receiver side cannot directly be used at the transmitter side because of the latency involved in information transfer from the receiver to the transmitter. Therefore, all the real-time video streaming solutions rely on the channel estimation at the transmitter side, where a channel estimation is made, based on different parameters available directly at the transmitter (e.g., the retransmission parameter).

Here is how the channel estimator works. A transmission attempt counter is associated with every outstanding packet at the data-link layer. The counter is initialized to zero for each new packet to be transmitted and incremented by one at every transmission attempt. A maximum number of transmission attempts ($R_{\max}$) is enforced for each packet, in order to prevent excessive packet delay. In essence, further transmission of a packet after $R_{\max}$ unsuccessful transmission attempts is aborted at the data-link layer, this packet is then recovered by higher layer error control mechanisms. Note that each transmission attempt at the data-link layer costs a round-trip time (RTT), which is a measure of the delay in the network. Due to the associated RTT cost, $R_{\max}$ is limited for time-sensitive applications, such as video streaming. If the number of transmission attempts reaches $R_{\max}$, this indicates a bad network condition. The typical $R_{\max}$ value for IEEE 802.11-based wireless network is 4 [16]. In this paper, we assume the threshold of $L_1 = 1$ transmission attempt to indicate a good channel. A second threshold of $L_2 = 2$ indicates a moderate channel condition. The channel condition is considered to be bad if a packet gets transmitted in 3 or 4 attempts, which is also consistent with [13].

Based on the available channel information through the packet transmission attempts and the optimum size of the application layer buffer sizes calculated, the application layer then invokes the best strategy to transcode the incoming video bit-stream. This is a bottom-up approach in which the quality of streaming video is maximized for a given set of network conditions. Similarly, the channel information is used at the data-link layer to calculate the required number of redundant FEC packets to maximize the error-resilience functionality.

2.3. Buffer Management at the Application Layer. Buffer management is another feature of the proposed framework, implemented in the buffer controller, as shown in Figure 1. For real-time video streaming applications, the application layer buffer sizes at both the transcoder and decoder sides play an important role in system performance (e.g., the overall power budget requirement [17]). The key to the

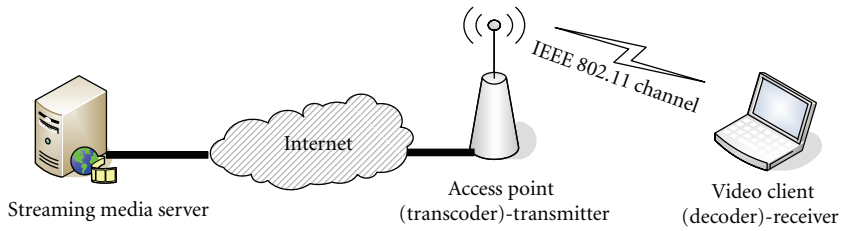

Figure 2: Infrastructure mode IEEE 802.11 wireless network for video streaming.

application layer buffer management is the rate-control scheme employed. A rate control scheme determines the optimum transcoding rate, which is used during the video compression process to adjust the coding parameters, for example, the QP to prevent the application layer buffers from overflow or underflow. Various rate-control schemes have been studied in literature, for example, TM5 for MPEG-2 [18], TMN8 for H.263 [19], VM-18 for MPEG-4 [20], and JVT-I049 for H.264 [21]. To analyze and meet the buffer constraints at the application layer, in this section, we first determine the buffer occupancy at both the transcoder and decoder sides. It is assumed that the maximum size of transcoder and decoder buffers is limited and denoted by $B_t^{\max}$ and $B_d^{\max}$ (in bits), respectively. Moreover, the decoder has a cushion of F video frames in its buffer to provide protection against any blackout periods, in case of buffer underflow. At time t, let the transcoder and decoder buffers be denoted by $B_t(t)$ and $B_d(t)$, respectively. We assume that at the startup time $t = 0$, both the transcoder and decoder buffers are empty, that is, $B_t(0) = 0$ and $B_d(0) = 0$, respectively.

2.3.1. Transcoder Buffer. Let $r(t)$ denote the incoming video bit-rate (in bits/sec) at the transcoder input, $r'(t)$ denotes the bit-rate (bits/sec) of the transcoded video, and $R_c(t)$ is the channel bit-rate (bits/sec). The transcoded video bit-rate can be written as $r'(t) = \beta(t)r(t)$, where $\beta(t)$ is a scaling function. After a video frame y is processed at the transcoder, the total number of bits generated $R_{\text{bg}}^{(y)}(T)$ at the buffer, during a video frame interval time T, is calculated by

$$R_{\text{bg}}^{(y)}(T) = \int_{(y-1)T}^{yT} r'(t)dt, \tag{1}$$

where y (≥ 1) is the video frame index and T is the frame interarrival time.

Similarly, the transmitted bits $R_{\text{bt}}^{(y)}(T)$ from the transcoder buffer, during the interval $(y-1)T$ to yT, is

$$R_{\text{bt}}^{(y)}(T) = \int_{(y-1)T}^{yT} R_c(t)dt. \tag{2}$$

The instantaneous transcoder buffer occupancy at any time t can be calculated as

$$B_t(t) = \int_0^t (r'(h) - R_c(h))dh. \tag{3}$$

More specifically, the transcoder buffer occupancy after transcoding y frames is given as

$$B_t(yT) = \int_0^{yT} (r'(h) - R_c(h))dh. \tag{4}$$

This can also be written in discrete form as

$$B_t(yT) = \sum_{s=1}^{y} \left[R_{\text{bg}}^{(s)}(T) - R_{\text{bt}}^{(s)}(T)\right], \tag{5}$$

where s (≥ 1) is the frame index.

Equation (5) shows the buffer occupancy after transcoding the yth frame is just the summation of all the accumulated bits at the transcoder buffer during the interval 0 to yT. Equation (5) can also be written in a recursive manner:

$$\begin{aligned} B_t(yT) &= \sum_{s=1}^{y-1} \left[R_{\text{bg}}^{(s)}(T) - R_{\text{bt}}^{(s)}(T)\right] + \left[R_{\text{bg}}^{(y)}(T) - R_{\text{bt}}^{(y)}(T)\right] \\ &= B_t((y-1)T) + \left[R_{\text{bg}}^{(y)}(T) - R_{\text{bt}}^{(y)}(T)\right]. \end{aligned} \tag{6}$$

To avoid transcoder buffer overflow/underflow, the constraint is given as

$$0 < B_t(yT) \leq B_t^{\max}. \tag{7}$$

Equation (7) is interpreted to mean that overflow at the transcoder buffer can be avoided if the instantaneous transcoder buffer occupancy is kept equal to or below the maximum buffer size. Similarly, if the transcoder buffer occupancy exceeds zero, the underflow of video packets can be avoided. Note that only the overflow constraint at the transcoder is of critical nature because its violation would result in packet loss and, consequently, quality loss. The underflow constraint at the transcoder side can be ignored because the decoder might still have the cushion packets to be rendered on the client's device.

By making use of (6) in (7) we have:

$$B_t((y-1)T) + R_{\text{bg}}^{(y)}(T) - R_{\text{bt}}^{(y)}(T) \leq B_t^{\max}. \tag{8}$$

Equation (8) is useful for calculating the upper bound on the transcoder buffer size, from knowledge of the transcoding rate, channel rate, and previous buffer occupancy conditions.

2.3.2. Decoder Buffer. Let $r''(t)$ denote the rate (in bits/sec) of rendering the video sequence to the user terminal. The number of bits rendered $R_{\text{br}}^{(y)}(T)$ to the video terminal during the interval $(y-1)T$ to yT is given as

$$R_{\text{br}}^{(y)}(T) = \int_{(y-1)T}^{yT} r''(t)dt. \tag{9}$$

As the decoder waits for F frames before starting the decoding process, this corresponds to a delay of FT seconds. The initial decoder buffer occupancy at $t = FT$ can be calculated as

$$B_d(FT) = \sum_{s=1}^{F} R_{\text{bt}}^{(s)}(T), \tag{10}$$

which is the accumulation of incoming bits over F frames. In general, the decoder buffer occupancy after decoding the yth frame is given by

$$B_d(yT) = B_d(FT) + \sum_{s=1}^{y}\left[R_{\text{bt}}^{(F+s)}(T) - R_{\text{br}}^{(s)}(T)\right]. \tag{11}$$

The expression given in (11) states that the instantaneous decoder buffer occupancy is a function of the initial buffer occupancy and accumulated bits at the decoder buffer.

On the decoder side, both the buffer overflow and underflow are avoided by maintaining the condition:

$$0 < B_d(yT) \le B_d^{\max}. \tag{12}$$

Equation (12) reveals that the decoder buffer underflow can be avoided by keeping the instantaneous decoder buffer occupancy above zero, and to avoid the buffer overflow, it is required to keep the instantaneous decoder buffer occupancy equal to or below the maximum buffer size. Both the underflow and overflow are of critical nature at the decoder side because the former will lead to the terminal screen blackout due to packet starvation, and the latter would cause packet dropping, eventually leading to video jerks. Hence, proper care should be taken in designing the dynamic buffers at the decoder, to minimize or eliminate the occurrence of buffer overflow and underflow.

Applying (11) in (12), the buffer underflow constraint becomes

$$\sum_{s=1}^{y}\left[R_{\text{br}}^{(s)}(T) - R_{\text{bt}}^{(F+s)}(T)\right] < B_d(FT). \tag{13}$$

The expression given in (13) is the key to finding the threshold number of video frames (F) that the decoder must keep in its buffer to avoid any underflow. For a fixed video rate and, by combining (10) and (13), the minimum number of frames F can be determined.

Similarly, applying (11) in (12), the decoder buffer overflow is avoided by maintaining the condition:

$$B_d(FT) + \sum_{s=1}^{y}\left[R_{\text{bt}}^{(F+s)}(T) - R_{\text{br}}^{(s)}(T)\right] \le B_d^{\max}. \tag{14}$$

Once F is determined from the underflow constraint, the upper bound $B_d^{\max}$ can then be calculated.

2.4. Video Transcoding Rate Calculation. The first step in calculating the transcoding parameters (e.g., target bit-rate and QP) is the bit-budget allocation [1], where an estimate is made to distribute the available bits to each frame. Depending on the scheme chosen for the rate-quantization (RQ) optimization (linear or quadratic), a QP is then assigned to each frame to meet the calculated target encoding rate. This calculation is refined after encoding each frame, where the actual remaining bits are recalculated to be distributed to the remaining frames. The target bit-rate to encode each frame is calculated based on different parameters, such as real-time application layer buffer status, target buffer level of the transcoder, available channel bandwidth, frame rate, actual bits used to encode previous frame, and so forth. The moving vector information in video sequences yields more complex frames and more bits are required to encode such complex frames, thus generating a large number of bits in the transcoder buffer. In case of bad channel condition when packets that cannot be promptly transmitted cause queue buildup, it is required that the transcoding rate be reduced so as to minimize buffer overflow. As far as video coding is concerned, the original MPEG-4 design uses the periodic I-frames (i.e., intraframes) [22], whereas the state-of-the-art H.264 encoder design suggests very few I-frames to be used for refreshing the video quality [1]. Based on the practical H.264 codec design [23], it is assumed that the first frame of a video sequence is encoded as an I-frame, followed by the P-frames in the sequence. The total number of bits required to encode an arbitrary frame y can then be calculated as [1]

$$R_{\text{bg}}^{(y)}(T) = (1-\alpha)\cdot R_1^{(y)}(T) + (\alpha)\cdot R_2^{(y)}(T), \tag{15}$$

where α is the model parameter of which the JVT-G012 standard [24] suggests the value of $\alpha = 0.75$. Moreover, R_1 and R_2 are the transcoding rates (bit/frame) based on the application layer buffer status and amount of bits remaining to encode the frame, respectively, and given by [1]:

$$R_1^{(y)}(T) = \frac{R_c(T)}{f_r} + \eta\left(B_T^{(y)}(T) - B_C^{(y)}(T)\right), \tag{16}$$

$$R_2^{(y)}(T) = (\gamma)\frac{R_{\text{rem}}}{N_{\text{rem}}} + (1-\gamma)R_{\text{act}}^{(y-1)}(T), \tag{17}$$

where, f_r in (16) is the video frame rate (frames/sec), and η is the model parameter (JVT-G012 standard [24] suggests the value of $\eta = 0.5$). The functions $B_T^{(y)}(T)$ and $B_C^{(y)}(T)$ are the application layer target and actual buffer occupancies (bits/frame) for frame y, respectively, and given as [1]:

$$\begin{aligned} B_T^{(y)}(T) &= B_T^{(y-1)}(T) - \frac{B_t(yT)}{y-1}, \\ B_C^{(y)}(T) &= B_C^{(y-1)}(T) + R_{\text{bg}}^{(y-1)}(T) - \frac{R_c(T)}{f_r}. \end{aligned} \tag{18}$$

In (17), γ is the model parameter (JVT-G012 standard [24] suggests $\gamma = 0.875$), $R_{\text{act}}^{(y-1)}(T)$ is the amount of actual bits used to encode the $(y-1)$th frame, and R_{rem} is the amount of remaining bits to encode the subsequent remaining frames N_{rem}.

```
Input: number of transmission attempts = L, R_bg^(y)(T)
Output: R_bg^(y)(T)
Begin
    if (L ≤ L_1)
      { /*channel state = Good*/
       R_bg^(y)(T) = 1.2R_bg^(y)(T)
      }
    elseif (L ≤ L_2)
      { /*channel state = Moderate*/
       R_bg^(y)(T) = R_bg^(y)(T)
      }
    else
      { /*channel state = Bad*/
       R_bg^(y)(T) = 0.8R_bg^(y)(T)
      }
End
```

Algorithm 1: Refining the calculated target transcoding rate.

The target bit-rate of a frame, calculated in (15), is required to be further refined by considering the current channel conditions. We propose here a further reduction in the video transcoding rate under bad channel condition. This will not only help reduce the loading on the network but also smooth-out the transcoded video stream. For the moderate channel (i.e., the number of transmission attempts $\leq L_2$), the calculated video bit-rate is used as it is, to take full advantage of the current channel state. Finally, when the channel condition is good (i.e., the number of transmission attempts $\leq L_1$), the target bit-rate is increased to exploit the good channel condition for higher video quality. The proposed algorithm for refining the calculated target transcoding rate is then given by Algorithm 1.

Note that the multiplication factors 1.2, 1, and 0.8 in Algorithm 1 are the rate adjustment factors, which are empirically determined for each channel condition. For the good channel condition, we select a rate adjustment factor of 1.2 because higher values lead to a disruption of the pre-calculated bit-budget allocation in H.264 encoder [1], which should be avoided. Also, in case of bad channel condition, we use a rate adjustment factor of 0.8 because lower values distort the video quality. The rationale for selecting a rate adjustment factor of 1 for moderate channel condition was stated earlier.

Once the refined target bit-rate is obtained, the next step is to calculate the QP for transcoding the incoming video sequence. In H.264/AVC reference software [25], the QP is mapped to QS: when QP increases by a step of size 6, the size of QS doubles. For a given bit-rate, selecting the optimum QP for encoding the video sequence can naturally be posed as an optimization problem, solved by, for example, Lagrangian multiplier methods. Both the linear and quadratic RQ models are proposed for selecting the QP when the target bit-rate is available. Although the quadratic model has a higher accuracy than the linear model in QP selection, the model is unsuitable for real-time transcoding because of its complexity [26]. Consequently, a linear RQ model is selected in this work to achieve a good balance in the tradeoff between complexity and accuracy, for real-time video streaming application, as shown in what follows:

$$R_{\text{bg}}^{(y)}(T) = \text{MAD}^{(y)} \frac{X_1^{(y)}(T)}{\text{QP}^{(y)}} + X_2^{(y)}(T), \tag{19}$$

where MAD is the mean absolute difference of the motion information between a reference frame and a predicted frame, and is used as a measure of frame complexity. The functions X_1 and X_2 are model parameters of the linear RQ model, which are updated after transcoding every frame [24].

```
Input: N_pd, N_FEC, T_tot, T_dl, T_av
Output: N_FEC, T_tot
Begin
    while ((T_tot > T_dl)&(N_FEC > 0))
    {
      N_FEC = N_FEC − 1
      T_tot = Σ_{i=1}^{N_pd+N_FEC} T_av
    }
End
```

Algorithm 2: Enforcing the time constraint at the data-link layer.

2.5. Meeting the Time Constraint at the Data-Link Layer. The fourth key element of the proposed cross-layer-based video streaming framework is the optimum FEC/ARQ controller, which selects the number of redundant FEC packets for a given $R_{\max}$, at the data-link layer. To prevent a video packet from being rejected at the receiver, the total transmission time of a video frame (T_{tot}) must satisfy the time constraint of $T_{\text{tot}} \leq T_{\text{dl}}$, where T_{dl} is the deadline time (i.e, the arrival time of the next video packet at the decoder) of a given video frame. As the deadline time is independent of the existing number of packets in the decoder queue [27], the time constraint $T_{\text{tot}} \leq T_{\text{dl}}$ can only be fulfilled by adjusting T_{tot}. Moreover, T_{tot} further depends on the average transmission time of a packet and total number of data-link layer packets of the given video frame, given by [27] $T_{\text{tot}} = \sum_{i=1}^{N_p} T_{\text{av}}$, where, T_{av} is the average transmission time of a single packet and N_p represents the total number of packets after FEC redundancy is added to the given video frame. If the given video frame is segmented into N_{pd} packets, then N_p is given by $N_p = N_{pd} + N_{\text{FEC}}$, where N_{FEC} is the (initial) number of redundant FEC packets, determined using the cost-throughput ratio- (CTR-) based method proposed in [28]. Based on the calculated average transmission time of a video packet, N_{FEC} needs to be reduced to satisfy the above time constraint. The time constraint is enforced by Algorithm 2.

Algorithm 2 shows that the number of redundant FEC packets is reduced until the time constraint is met or the number of FEC packets reduces to zero, whichever comes first. If the maximum number of transmission attempts ($R_{\max}$) at the data-link layer can be controlled, Algorithm 3

```
Input: N_pd, N_FEC, T_dl, R_max, T_av
Output: N_FEC, T_tot, R_max
Begin
        while ((T_tot > T_dl)&(N_FEC > 0))
        {
            N_FEC = N_FEC − 1
            T_tot = Σ_{i=1}^{N_pd+N_FEC} T_av
        }
        while ((T_tot > T_dl)&(R_max > 1))
        {
            R_max = R_max − 1
            T_tot = Σ_{i=1}^{N_pd+N_FEC} T_av
        }
End
```

Algorithm 3: Enforcing the time constraint at the data-link layer (Algorithm 2 refined).

serves the purpose of meeting the time constraint by reducing $R_{\max}$ in addition to N_{FEC}.

Algorithm 3 shows that if the time constraint is violated, the constraint can be met by reducing the number of redundant FEC packets in a video frame and also limiting the number of transmission attempts.

2.6. Summary. A flowchart that describes the operation of the proposed video transcoding and transmission framework is depicted by Figure 3. The first frame of video sequence is fed to the transcoder, where it is analyzed and, depending on the specified video bit-rate, the initial preset QP is used for transcoding the first frame. As the transcoder buffer was empty before transcoding the first video frame of the group of pictures (GOPs), the dynamic buffer adjustment is not required at this moment. However, if the current video frame is not the first, then the video transcoder calculates the new transcoding parameter, that is, the QP, based on the following information: actual transcoder buffer occupancy, target buffer level, available channel bandwidth, frame rate, bit-rate of the previous frame, and channel information from the channel estimator. If the conditions of application layer buffer constraints are met at both the transcoder and decoder sides, the calculated QP is used for transcoding. Otherwise, in the case of buffer violation due to source coding rate adjustment, a default QP is used to transcode the given frame. Once the video frame is transcoded, it is passed to the data-link layer for processing of the FEC/ARQ functionality. The CTR-based model presented in our previous work [28] is used at this level to generate the initial number of redundant FEC data-link layer packets which serve as an input to Algorithms 2 and 3. For the given ARQ maximum number of transmission attempts, the redundant number of FEC packets is adjusted based on Algorithm 2 or Algorithm 3. Channel estimator then estimates the channel condition based on the number of transmission attempts for a packet. The estimated channel information is then fed back to the application and data-link layers for transcoding and transmitting the next video frame. In summary, the proposed CLM provides the joint functionalities of efficient video transcoding and error-resiliency by considering both the application layer buffer occupancy and data-link layer time constraints.

3. Performance Evaluation

We evaluate, using the simulation approach, the performance of the proposed video streaming framework by streaming three video clips of different motion categories over an IEEE 802.11 wireless network. The three video streams selected for our evaluation are *Akiyo*, *Container*, and *Foreman* categorized as belonging to slow, medium, and fast motion, respectively. The simulation code was developed using the NS2-based platform [29], enhanced by EvalVid framework [30]. The JM reference software (ver. 13.2) [25] has been used for bit rate adjustment during the transcoding of an incoming video sequence. To capture the realistic network operation conditions, three different traffic sources are considered in the simulation environment: (1) an FTP source transmitting packets using TCP protocol, (2) an exponential source transmitting packets using the UDP protocol, and (3) a video streaming source transmitting the test video clips. The FTP source represents a bulk file transfer application over the TCP protocol and, for the simulation, the file is considered big enough such that there is always data to transmit over the length of the simulation. The exponential source represents the bursty traffic, with a maximum packet size of 1500 bytes. Burst time and idle time are each set to 0.5 second and the source rate is set at 256 Kbps. Only the first 100 frames of each video stream considered are encoded for this study, which capture all the motion sequences in each video stream. The first frame of the video sequence is an I-frame, containing only the intracoded macroblocks, while the subsequent frames are P-frames that allow both the intra coded and predicted macroblocks. The video frame rate is set to 30 frames per second, as such T_{dl} is considered to be 1/30 seconds. To avoid any packet dropping at both the transcoder and decoder buffers, the respective maximum buffer sizes $B_t^{\max}$ and $B_d^{\max}$ are kept to 5 times the size of an average I-frame ($5 \times 10,000$ bits), while a cushion size of $F = 3$ frames is chosen at the decoder side. RD optimization [1] was enabled and context adaptive binary arithmetic coding (CABAC) [1] was used for the entropy encoding. Without loss of generality, an IEEE 802.11b link is selected between the AP and the client device, where the maximum data-rate is 11 Mbps. Joint FEC/ARQ [28] mechanism is implemented at the data-link layer to generate the redundant FEC packets, for error correction. The number of FEC packets generated for the good channel condition, when the probability of data-link layer packet error (μ) ranges from 10^{-4} to 10^{-2}, is too few to be used for comparison. Therefore, only moderate to bad channel conditions are considered (when μ ranges

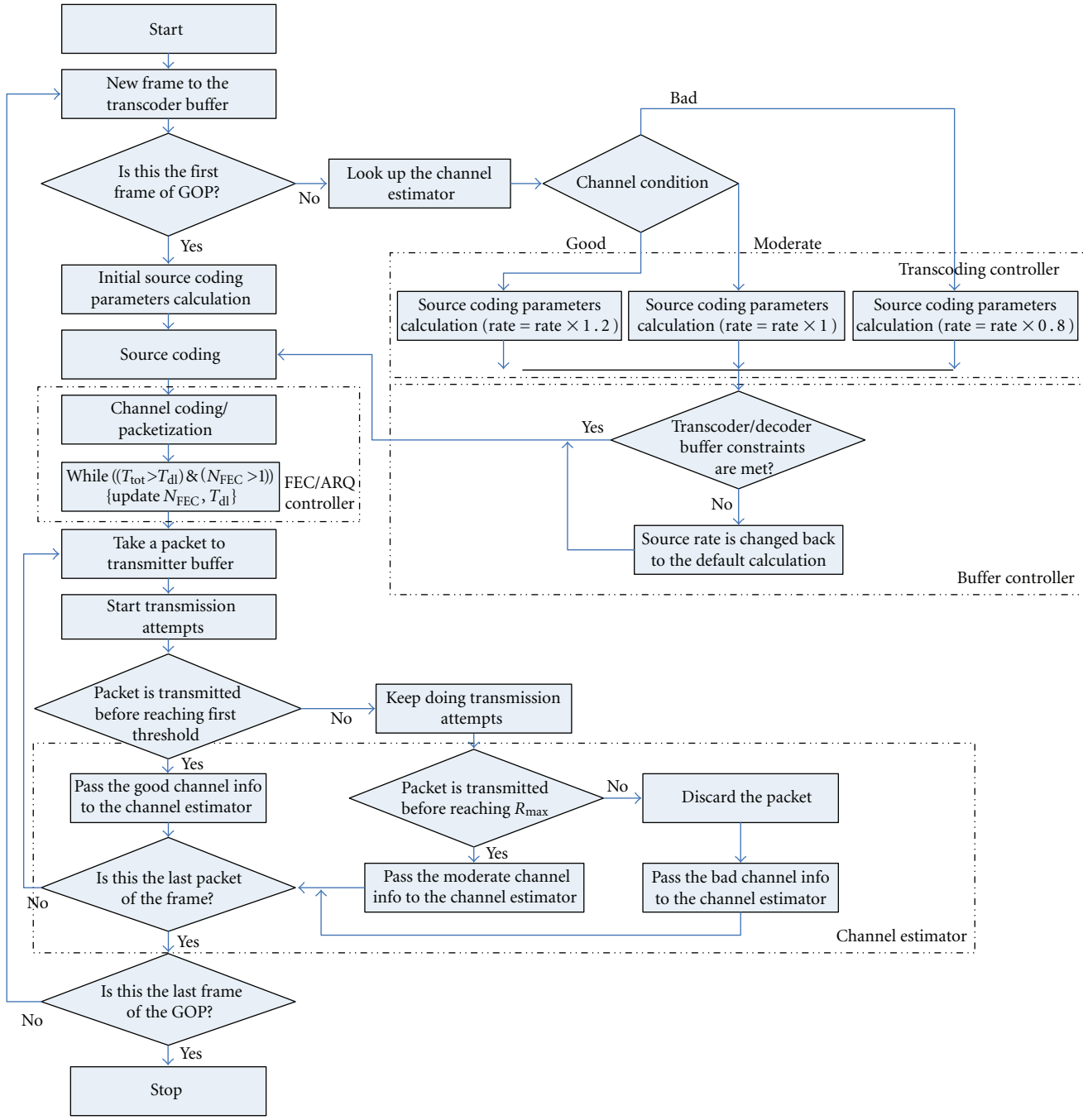

Figure 3: Flow of operations in the proposed framework (GOPs: group of pictures).

from 10^{-2} to 10^{0}). This gives a more realistic count of the redundant FEC packets required for comparison. We assume the Gilbert-Elliot (GE) channel model, which defines the wireless channel to be in either good or bad state and P_{xy} is the probability of going from state x to y, where $x, y \in \{0, 1\}$, 0 and 1 being the bad and good states, respectively. The channel state transition probabilities are set to $P_{00} = 0.5$, $P_{01} = 0.5$, $P_{10} = 0.1$, and $P_{11} = 0.9$, representing the wireless channel has a tendency of being in the good state most of the time.

3.1. Processing Time at the Access Point Buffer. The increase in processing time of a video packet at the AP is an important metric to quantify the cost associated with the proposed cross-layer framework versus the improvement in video quality. When the cross-layer framework for video transmission is not implemented, the AP puts video packet in the transmission queue without any preprocessing of the packet. However, when the proposed framework is used, there is an associated delay at the AP, where the video packets are processed (at both the application and data-link

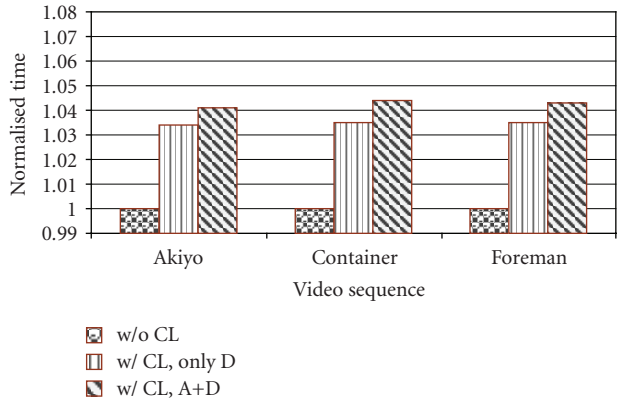

Figure 4: Video packet processing time at the AP buffer.

layers) for delivery. Three different scenarios are considered to compare the processing times at the AP buffer: (1) the parameters of both the application and data-link layers are adapted using the proposed framework (denoted by w/ CL A+D), (2) when only the parameters at data-link layer are adapted (denoted by w/ CL, only D), and (3) without any adaptation of parameters at the application and data-link layers (denoted by w/o CL). In Figure 4, the processing times at the AP are compared for the three test video sequences, under the moderate channel condition ($\mu = 10^{-2}$). All these processing times are normalized with respect to the packet processing time when no cross-layer signaling mechanism is used (w/o CL). Two observations can be made from Figure 4. First, the processing times are independent of the video motion category, and second, the highest processing delay is incurred when the proposed framework is used, due to the extra processing required at both the application and data-link layers. As the maximum size of a video packet at data-link layer is predefined and the packet sizes belonging to the video clips of different motion categories are identical, there is no effect of video motion category in the packet processing time. Comparing the results of cases (1) and (2), it is seen from Figure 4 that when parameters of both the application and data-link layers are jointly adapted (case 1), there is a marginal delay increase of less than one percent compared to that of case 2. This indicates that adapting the transcoding parameters at the application layer to the current channel conditions does not produce significant processing delays. This is attributed to the fact that the application layer merely limits the calculated value of the QP using the information available from cross-layer signaling mechanism. On the other hand, when the parameters at the data-link layer are adapted for a given maximum value of the transmission attempts (e.g., $R_{\max} = 2$ in this case), the calculation and generation of the number of FEC packets take much of the processing time, that is, approximately 3.5 percent above the processing time when the cross-layer signaling mechanism is not used. It is concluded from Figure 4 that there exists a small (3–5 percent) processing time cost associated with the implementation of the proposed cross-layer framework.

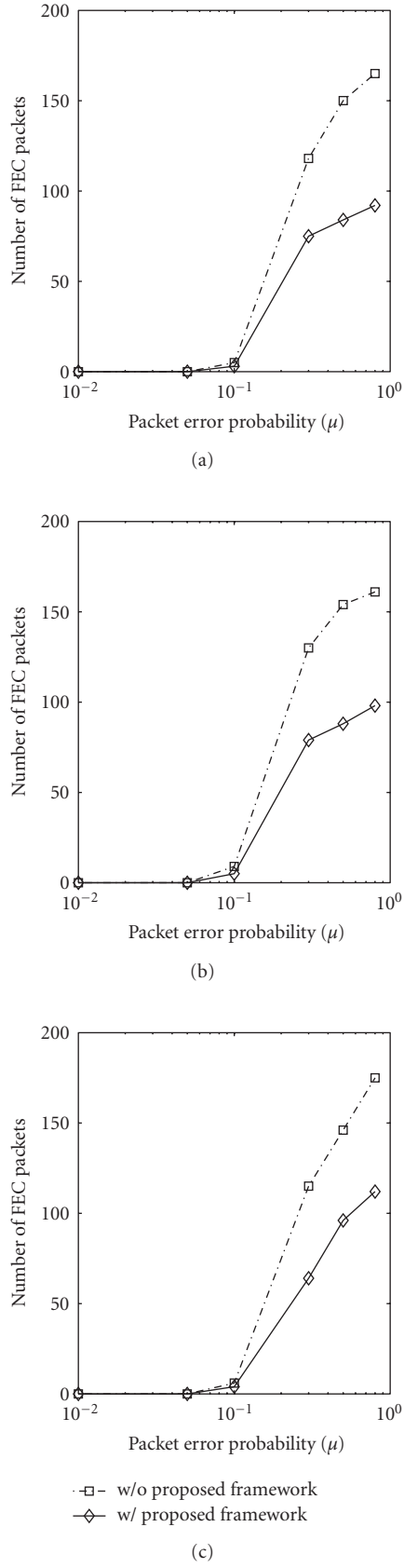

Figure 5: Number of redundant FEC packets (for 100 frames) for the three test sequences: (a) *Akiyo*, (b) *Container*, and (c) *Foreman*.

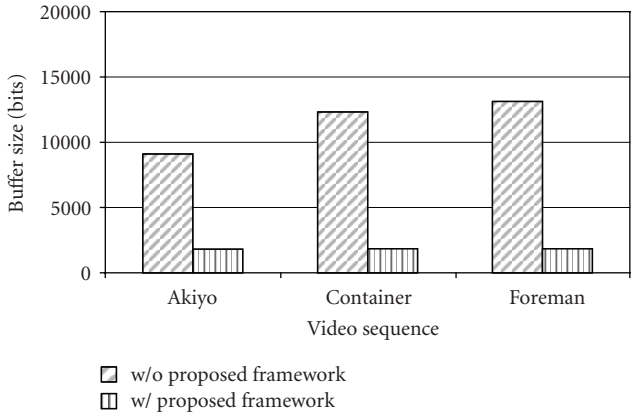

Figure 6: Comparison of the application layer buffer sizes for three test video sequences under moderate channel condition ($\mu = 10^{-2}$).

3.2. Number of Redundant FEC Packets. For the bad channel condition (i.e., μ ranges from 10^{-2} to 10^{0}), there is a need for an increased number of redundant FEC packets to provide error resiliency against the channel. If the deadline time is fixed, for a given number of transmission attempts, the time constraint is satisfied by limiting the number of redundant FEC packets. This puts an upper bound on the number of redundant FEC packets that must be generated, to avoid any packet dropping at the decoder buffer caused by violation of the deadline time constraint. The total number of FEC packets injected in the video stream, when the probability of packet error is varied from 10^{-2} to 10^{0}, is shown in Figures 5(a), 5(b), and 5(c), for the three video sequences *Akiyo*, *Container*, and *Foreman*, respectively. It is seen from Figure 5 that the number of redundant FEC packets increases when the channel condition worsens because these redundant FEC packets are required to provide error-resiliency against the bad channel. Clearly, an increase in the number of redundant FEC packets translates to an increase in the network load. It is concluded from Figure 5 that the proposed framework imposes less packet loading on the network than when it is not used, the advantage becoming significant under bad channel conditions. The reduced loading is attributed to the limit placed on the number of transmission attempts for each packet.

3.3. Dynamic Buffer Optimization. To satisfy the buffer constraints at the application layer, the proposed video streaming framework adapts the video bit-rate according to the current network conditions. For example, under the moderate channel condition ($\mu = 10^{-2}$) when the buffer requirements of the three test video sequences are compared, it is seen in Figure 6 that the average buffer requirement for the dynamically stabilized buffers (labeled as w/ proposed framework) drops by almost an order of magnitude as compared to the fixed buffers (termed as w/o proposed framework). The reduction in buffer size is attributed to the fact that the buffer sizes are now calculated in real-time for each video frame instead of being fixed at a worst-case value when the proposed framework is not enabled.

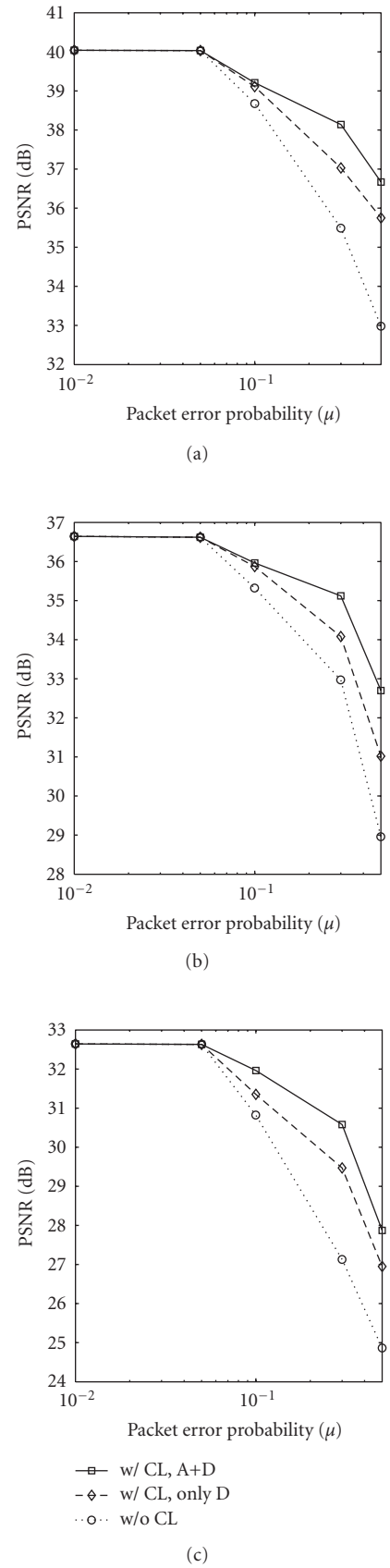

Figure 7: Objective video quality of the three test sequences, (a) *Akiyo*, (b) *Container*, (c) *Foreman*.

Figure 8: Subjective quality test for three test video sequences.

A comparison of the predicted buffer sizes for the three test video sequences reveals that the video sequences with high motion content tend to require a larger buffer space, when the proposed cross-layer optimization is not used. This dependence of buffer sizes on the video motion category makes a fixed allocation of buffer space highly inefficient because the application layer buffers must be provisioned for the worst case (i.e., for the videos with the highest motion content). On the other hand when the proposed framework is used, it is seen in Figure 6 that the average buffer size requirement at the decoder is independent of the video motion category. This is due to the fact that the dynamic buffer allocation scheme takes advantage of RD optimization at the application layer, for example, a large QP (typically 40, in JM software encoder for H.264) is used for the videos with high motion content to encode the video frames (in contrast to QP values of 20–30 for the slow-medium motion categories), when the target bit-rate is specified.

3.4. Objective and Subjective Video Quality. Peak-signal-to-noise-ratio (PSNR) is the most commonly used metric to measure the quality of reconstruction in image compression and is therefore used in this paper as a measure of objective video quality. Figures 7(a), 7(b), and 7(c) show the improvement in PSNR with the proposed framework for the three test video sequences *Akiyo*, *Container*, and *Foreman*, respectively. The three scenarios described in Section 3.1 are considered. When the channel condition gets worse (i.e., μ goes beyond 10^{-1}), the proposed cross-layer framework performs 3-4 dB better in terms of PSNR than the case when no such cross-layer-based approached is used. This improvement is more or less the same in all the three test video sequences. The performance improvement is due to the fact that in the case of bad channel condition, the less number of FEC packets added per frame guarantees the non-violation of the deadline time constraint. The video frames rendered to the client device without being dropped causes the increase in PSNR. Moreover, the application layer-based

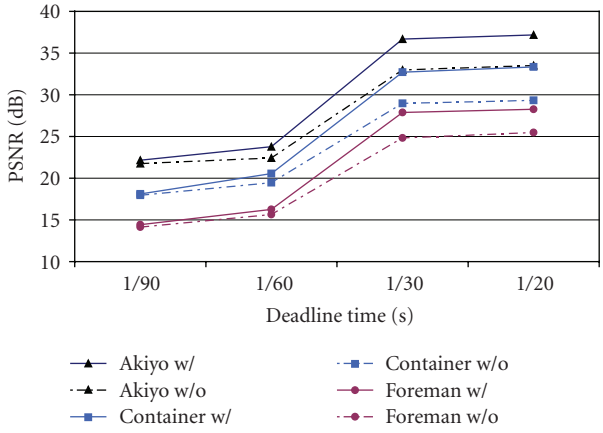

Figure 9: Effect of deadline time on received video quality.

optimization also lowers the video bit-rate for bad channel conditions, thereby reducing the number of video packets and giving high probability of transmission without a packet getting dropped. For comparison, we also consider a partial cross-layer-based framework, where only the data-link layer-based optimization is achieved. It is concluded from the results in Figure 7 that for the bad channel condition (e.g., $\mu = 5 \times 10^{-1}$), the application layer based rate adjustment in the proposed framework contributes up to 1 dB of improvement in PSNR, while the rest of approximately 2 dB PSNR improvement (totaling to approximately 3 dB) comes from the data-link layer optimization. This shows that under worse channel condition, most of the PSNR gains come from the data-link layer optimization. This result is consistent with Figure 4, where the data-link layer optimization exhibits higher packet processing time. The foregoing confirms that the proposed framework demonstrates a tradeoff between the PSNR gain and the increase in packet processing time.

We also assess the subjective quality of the three test video sequences and the results are shown in Figure 8. The source format is *qcif* (quarter common intermediate format), where the pixel values are 176×144. Without loss of generality, frame number 50 of each test video sequence is arbitrarily chosen, which is in the middle of the 100 frame test video sequences. Figures 8(a), 8(b), and 8(c) are the reference video frames of the video sequences *Akiyo*, *Container*, and *Foreman*, respectively, and are given here for comparison purpose. Figures 8(a'), 8(b'), and 8(c') are the screenshots of the three test video sequences when they are reconstructed at the decoder under the channel error of $\mu = 5 \times 10^{-1}$, and in the absence of the proposed framework. The degradation in video quality is visible, meaning that the viewers will have poor quality of experience watching the video. When the cross-layer-based framework is used for video streaming, it is seen in Figures 8(a"), 8(b"), and 8(c") that the errors are very limited as the erroneous bottom parts of Figures 8(a") and 8(b"), and the top left part of Figure 8(c") cannot be easily recognized. The corresponding PSNR values and the gain in PSNR with the proposed framework are listed in Table 1.

Table 1: PSNR values (in dB) for the subjective quality test.

	Akiyo	*Container*	*Foreman*
Original (dB)	40.041	36.644	32.648
$\mu = 0.5$, w/o CL (dB)	32.978	28.963	24.824
$\mu = 0.5$, w/ CL (dB)	36.671	32.703	27.878
w/ versus w/o CL Gain (dB)	3.393	3.740	3.054

3.5. Impact of Deadline Time on Video Quality. When the temporal resolution is controlled at the decoder, such that the rate at which the video frames are rendered at the client terminal be reduced for the bad channel condition, the value of deadline time T_{dl} increases, thereby relaxing the constraint of $T_{tot} \leq T_{dl}$. For the bad channel condition, that is, $\mu = 5 \times 10^{-1}$, the effect of deadline time on the three test video sequences in terms of video quality (PSNR) is given in Figure 9. The label-suffixes w/ and w/o in Figure 9 refer to the scenarios where the simulations are conducted with and without the proposed framework, respectively. Clearly, decreasing the deadline time adversely affects the output video quality due to the increased likelihood that more video frames would violate the time constraint and dropped. For example, from Figure 9, at a deadline time of 1/60 second, the achieved PSNR gain is about 1 dB for the three test sequences. On the other hand, increasing the deadline time to 1/20 seconds leads to better video quality, as a gain of approximately 3.5 dB is observed in all the three test video sequences. An increased deadline time allows more number of FEC packets to be added on the transmitted video stream, for a given R_{max}. However, when temporal resolution is decreased to increase the deadline time, it must be done with caution because the human eye can detect flickering at reduced frame rates.

4. Conclusion

In this paper, the buffer constraint at application layer is jointly considered with time constraint at data-link layer, leading to an optimized solution of transcoding and transmitting the H.264 video stream over an IEEE 802.11-based wireless network. The four key elements of the proposed framework are channel estimator, buffer controller, transcoding controller, and FEC/ARQ controller. The proposed model is fully adaptive to the changes in the network conditions and size of the video frames. Simulation results clearly establish the validity of the proposed model, as video quality is improved (up to 3 dB, under bad channel condition) with a minimal increase in packet processing time (less than 5 percent). Moreover, the proposed framework imposes less packet loading on the network than when it is not used, as the number of redundant FEC packets are reduced by up to 50 percent under bad channel condition. This shows the effectiveness of the proposed framework in maximizing the available resources for background traffic. The application layer buffer requirements are also dropped by almost an order of magnitude when the proposed framework is used as compared to the case where it is not implemented, thereby

decreasing the memory requirement. It is concluded that the proposed framework will support efficient streaming of video over IEEE 802.11 wireless networks.

Acknowledgments

The authors acknowledge the support of the University of Calgary, TRLabs, and National Sciences and Engineering Research Council (NSERC), Canada, for this research. The authors also thank the anonymous reviewers whose comments and suggestions have enhanced the quality of the paper.

References

[1] ITU-T Rec. H.274 and ISO/IEC 14497-10 (MPEG4-AVC), "Advanced video coding for generic audiovisual services," ver. 1, May 2003; ver. 2, January 2004; ver. 3 (with FREXt), September 2004; ver. 4, July 2005.

[2] Z. Wei, K. L. Tang, and K. N. Ngan, "Implementation of H.274 on mobile device," *IEEE Transactions on Consumer Electronics*, vol. 53, no. 3, pp. 1109–1117, 2007.

[3] A. Argyriou, "Error-resilient video encoding and transmission in multirate wireless LANs," *IEEE Transactions on Multimedia*, vol. 10, no. 5, pp. 691–700, 2008.

[4] Y. Wang, J.-G. Kim, S.-F. Chang, and H.-M. Kim, "Utility-based video adaptation for Universal Multimedia Access (UMA) and content-based utility function prediction for real-time video transcoding," *IEEE Transactions on Multimedia*, vol. 9, no. 2, pp. 213–220, 2007.

[5] J.-W. Kim, G.-R. Kwon, N.-H. Kim, A. Morales, and S.-J. Ko, "Efficient video transcoding technique for QoS-based home gateway service," *IEEE Transactions on Consumer Electronics*, vol. 52, no. 1, pp. 129–137, 2006.

[6] Q. Li and M. van der Schaar, "Providing adaptive QoS to layered video over wireless local area networks through real-time retry limit adaptation," *IEEE Transactions on Multimedia*, vol. 6, no. 2, pp. 278–290, 2004.

[7] M. C. Yuang, P. L. Tien, and S. T. Liang, "Intelligent video smoother for multimedia communications," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 2, pp. 136–146, 1997.

[8] A. Vetro, J. Xin, and H. Sun, "Error resilience video trnscoding for wireless communications," *IEEE Wireless Communications*, vol. 12, no. 4, pp. 14–21, 2005.

[9] H. Shen, X. Sun, and F. Wu, "Fast H.264/MPEG-4 AVC transcoding using power-spectrum based rate-distortion optimization," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 6, pp. 746–755, 2008.

[10] H. Shu and L.-P. Chau, "The realization of arbitrary down-sizing video transcoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 16, no. 4, pp. 540–546, 2006.

[11] P. A. A. Assuncao and M. Ghanbari, "Buffer analysis and control in CBR video transcoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, no. 1, pp. 83–92, 2000.

[12] E. Jammeh, M. Fleury, and M. Ghanbari, "Fuzzy-logic congestion control of transcoded video streaming without packet loss feedback," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 3, pp. 387–393, 2008.

[13] M. van der Schaar and D. S. Turaga, "Cross-layer packetization and retransmission strategies for delay-sensitive wireless multimedia transmission," *IEEE Transactions on Multimedia*, vol. 9, no. 1, pp. 185–197, 2007.

[14] Q. Zhang, W. Zhu, and Y.-Q. Zhang, "Channel-adaptive resource allocation for scalable video transmission over 3G wireless network," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 8, pp. 1049–1063, 2004.

[15] B. Bing, *Emerging Technologies in Wireless LANs: Theory, Design, and Deployment*, Cambridge University Press, Cambridge, UK, 2007.

[16] V. Sgardoni, P. Ferre, A. Doufexi, A. Nix, and D. Bull, "Frame delay and loss analysis for video transmission over time-correlated 802.11A/G channels," in *Proceedings of the IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC '07)*, September 2007.

[17] M. Yokotsuka, "Memory motivates cell-phone growth," *Wireless Systems Design*, vol. 9, no. 3, pp. 27–30, 2004.

[18] ISO/IEC JTCI/SC29/WG11, "Test Model 5," 1993.

[19] ITU-T/SG15 TMN8, "Video Codec Test Model," Portland, Ore, USA, June 1997.

[20] MPEG-4 Video Verification Model V18.0, "Coding of moving pictures and audio," N3908, ISO/IEC, JTC1/SC29/WG11, January 2001.

[21] G. Sullivan, T. Wiegand, and K. P. Lim, "Joint model reference encoding methods and decoding concealment methods," Sec. 2.7: Rate Control, JVT-I049. San Diego, Calif, USA, September 2003.

[22] ISO/IEC JTC 1, "Coding of Audio-Visual Objects—Part 2: Visual," ISO/IEC 14497-2 (MPEG4 Visual Version 1), April 1999; Amendment 1 (Version 2), February 2000; Amendment 4 (streaming profile), January 2001.

[23] D. Marpe, T. Wiegand, and G. J. Sullivan, "The H.264/MPEG4 advanced video coding standard and its applications," *IEEE Communications Magazine*, vol. 44, no. 8, pp. 134–142, 2006.

[24] Z. G. Li, F. Pan, K. P. Lim, G. N. Feng, X. Lin, and S. Rahardaj, "Adaptive basic unit layer rate control for JVT," in *Proceedings of the JVT-G012, 7th Meeting*, Pattaya, Thailand, March 2003.

[25] JM reference software, Maintained by K. Sühring, June 2008, http://iphome.hhi.de/suehring/tml/.

[26] M. Jiang and N. Ling, "Low-delay rate control for real-time H.264/AVC video coding," *IEEE Transactions on Multimedia*, vol. 8, no. 3, pp. 467–477, 2006.

[27] A. Moid and A. O. Fapojuwo, "Three-dimensional absorbing Markov chain model for video streaming over IEEE 802.11 wireless networks," *IEEE Transactions on Consumer Electronics*, vol. 54, no. 4, pp. 1672–1680, 2008.

[28] A. Moid and A. O. Fapojuwo, "An analytical model for optimum byte-level and packet-level FEC assignment using buffer dynamics," *Research Letters in Communications*, vol. 2008, Article ID 547184, 4 pages, 2008.

[29] Network Simulator, http://www.isi.edu/nsnam/ns/.

[30] "EvalVid—A video quality evaluation tool-set," http://www.tkn.tu-berlin.de/research/evalvid/.