J. Vis. Commun. Image R. 19 (2008) 489–499

Contents lists available at ScienceDirect

J. Vis. Commun. Image R.

journal homepage: www.elsevier.com/locate/jvci

ARMOR – A system for adjusting repair and media scaling for video streaming

Huahui Wu [a,*], Mark Claypool [b], Robert Kinicki [b]

[a] Google, 5 Cambridge Center, Cambridge, MA 02142, USA
[b] Computer Science Department, Worcester Polytechnic Institute, 100 Institute Road, Worcester, MA 01609, USA

ARTICLE INFO

Article history:
Received 1 December 2007
Accepted 9 July 2008
Available online 22 July 2008

Keywords:
Video quality
User study
Streaming MPEG
Temporal scaling
Quality scaling
Forward error correction

ABSTRACT

To optimize scarce network resources and present the highest quality video, streaming video systems need adapt to the video content as well as the network conditions. This paper presents ARMOR, a video streaming system that dynamically adjusts repair and media scaling to meet current video and network conditions. In order to adapt effectively, ARMOR, and any dynamic video adaptation system, needs to predict the video quality as perceived by end users over the range of scaling and repair choices. Thus, this paper first proposes a novel video quality metric called *distorted playable frame rate* that provides estimation of user perceptual quality considering temporal and quality degradations. Comprehensive user studies show distorted playable frame rate is more accurate than other video quality metrics. Analytic experiments with distorted playable frame rate and the ARMOR optimization algorithm illustrate the predictive power of the metric in a dynamic, streaming video system. Lastly, implementation and experiments of a complete, fully-functioning ARMOR system show the effective practicality of the proposed approach.

1. Introduction

The increasing power of today's computers and the availability of high-capacity network connections provide opportunities for streaming video from a variety of sources. Television programs, once broadcast by analog through either the air or cables, are increasingly available for download to end-users both on home and on mobile connections. Video conferencing, once relegated to specialized conference rooms at select corporations, is now possible using mobile phones equipped with digital cameras and Internet capabilities.

The range of network characteristics for streaming video has similarly expanded, with servers and clients connected by networks with variable capacities and loss rates. A video client can be a home-user with a broadband connection with a capacity that is a fraction of that of a typical LAN connection. Increasingly, video applications traverse wireless LANs with the potential for high network capacities, but with the achievable capacities varying with signal strength and other environmental factors. The number of end-hosts with wide-area wireless connections is also growing, providing connections with lower and more varied network capacities than other links. Moreover, the loss characteristics of the last-hop network link can vary substantially with some paths encountering significant congestion at the access point, and with wireless networks tending to have higher bit error rates than wired connections.

An effective streaming video system must adapt to the network conditions to meet the bandwidth constraints and react to packet losses along the end-to-end path [10]. With packet loss, delay-sensitive interactive streaming must provide additional redundancy to recover from the loss, adding extra data to the streaming bitrate. Fortunately, video streaming bitrates are generally quite elastic. Streaming systems can adjust the streamed video bitrate to the available bandwidth; typically either with temporal scaling that reduces the frame rate or with quality scaling that lowers the fidelity of the encoded video frames. Choosing between temporal or quality scaling requires knowledge of the video's characteristics, e.g. high-motion videos need quality scaling to preserve frame rate while low-motion videos need temporal scaling to retain high-quality frames while discarding those frames with only small differences from neighboring frames.

This paper introduces the ARMOR[1] system, depicted in Fig. 1, a video streaming system that dynamically adjusts repair and media scaling to meet video and network conditions. The network protocol modules (UDP sender and UDP receiver) provide information on loss rates, capacities and packet sizes, and the streaming video modules provide details on the video frame sizes and frame types. The media scaler and repair encoder modules reduce streaming bitrates with media scaling and protect against packet loss, respectively. At the core of the ARMOR system is the ARMOR algorithm that uses video

* Corresponding author.
E-mail addresses: hwu@google.com (H. Wu), claypool@cs.wpi.edu (M. Claypool), rek@cs.wpi.edu (R. Kinicki).

[1] ARMOR stands for Adjusting Repair and Media Scaling with Operations Research.

1047-3203/$ - see front matter © 2008 Elsevier Inc. All rights reserved.
doi:10.1016/j.jvcir.2008.07.001

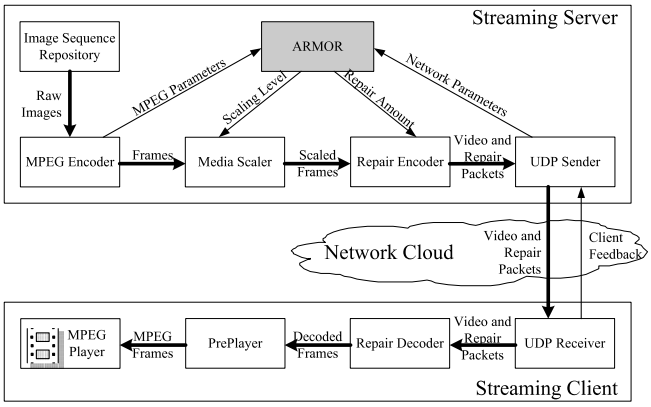

Fig. 1. ARMOR system architecture.

and network characteristics to determine the combination of scaling and repair that yields optimal perceptual quality. Details on the architecture are provided in Section 6.

To effectively improve the quality of streaming video, the ARMOR system needs a performance metric that is a good predictor of the video quality as perceived by the end user. Peak signal-to-noise ratio (PSNR) and playable frame rate, widely used because of their simplicity, are ineffective when videos are both temporally and quality scaled. Furthermore, comprehensive video metrics such as the video quality metric (VQM) [13,18] are not effective under conditions of network loss.

This paper also proposes a new metric for video quality called the distorted playable frame rate (R_D). R_D captures perceptual quality of video that is both temporally degraded through temporal scaling or packet loss, and quality degraded through quality scaling.

Results from a comprehensive user study with over 70 users show R_D is more accurate at predicting perceptual quality than PSNR, frame rate, or VQM. The quality metric plus ARMOR optimization algorithm and the complete ARMOR system implementation provide analytic and realistic experiments that demonstrate the effectiveness of the ARMOR approach. Specific experimental results show that quality scaling is generally more effective than temporal scaling and adding repair to video is most beneficial to high-motion video under conditions of high network loss.

The rest of this paper is organized as follows: Section 2 provides background on media scaling, repair and quality metrics. Section 3 presents the ARMOR quality metric for determining video quality given video and network characteristics. Section 4 provides the ARMOR optimization algorithm that uses the quality metric to determine the best repair and scaling. Section 5 describes the user study that measures the effectiveness of the distorted playable frame rate metric, R_D. Section 6 details implementation and experiments with the ARMOR system and Section 7 summarizes our conclusions and presents possible future work.

2. Background

2.1. Media scaling

To meet a capacity constraint, *temporal scaling* discards lower priority video frames prior to transmission. For instance, with an MPEG Group of Picture (GOP) pattern of '`IBBPBBPBBPBBPBB`', the data rate can be reduced by discarding all the B frames and only sending '`I--P--P--P--P--`'.

While temporal scaling could, in theory, select any of the frames in a GOP to discard, the MPEG frame dependencies limit the practical choice of frames to discard. Table 1 lists some of the scaling levels used in this paper, accounting for the MPEG frame dependencies and minimizing the effect of temporal scaling on the quality of the received video, i.e. frames are discarded evenly.

Table 1
Some temporal scaling levels

Level (l_{TS})	(N_{PD})	(N_{BD})	Scaling pattern
0	4	10	`IBBPBBPBBPBBPBB`
5	4	5	`IB-PB-PB-PB-PB-`
10	4	0	`I--P--P--P--P--`
14	0	0	`I--------------`

In the table, N_{PD} and N_{BD} are defined as the number of P or B frames which are transmitted in one GOP, respectively, with the scaling patterns provided for each scaling level, l_{TS}. Since typical decoders detect, and accommodate, lost frames, the frames selected for discarding can be removed at the sender with effectively no additional overhead.

To meet a capacity constraint, *quality scaling* encodes each MPEG frame with lower precision and fewer bits by removing low order bits from each DCT coefficient.

This paper assumes SPEG-like (scalable MPEG) [8] quality scaling, with every DCT coefficient divided into four layers: one base layer and three advanced layers. A DCT coefficient, C, is partitioned into the layers using the following equations:

$$\begin{aligned} &\text{Base layer L0}: && C_0 = C \gg 3 \\ &\text{1st Advanced layer L1}: && C_1 = (C \gg 2) \text{ and } 1 \\ &\text{2nd Advanced layer L2}: && C_2 = (C \gg 1) \text{ and } 1 \\ &\text{3rd Advanced layer L3}: && C_3 = C \text{ and } 1 \end{aligned} \tag{1}$$

At the receiver/player, the above steps are reversed to reconstruct the original video. Zero is used instead when some advanced layer(s) is (are) absent, analogous to using a high quantization value during MPEG encoding. Assuming the highest quantization value used is 3 (this yields a high fidelity quality and reasonable bitrate), the relationship of the scaling level, transmitting SPEG layers and equivalent quantization value can be defined as in Table 2. Since SPEG needs to use extra header information to indicate layer information (a 15–25% overhead is indicated in [8]), a 20% overhead is assumed in this paper.

Notice, the model presented in Section 3 is independent of the scaling technique and only requires the relationship among the scaling level, encoding bitrate and video quality. Adaptation of the model to other forms of temporal scaling or quality scaling is reasonable future work.

2.2. Forward error correction (FEC)

Forward error correction (FEC) is a commonly used repair approach to recover video frames damaged by packet loss. Reed–Solomon (R–S) [14] is a typical FEC technique that can be applied at the packet level. With R–S, an application level video frame can be modeled as being transmitted in K packets, and R–S adds $(N - K)$ redundant packets to the K original packets resulting in N packets sent over the network. Although some packets may be lost, the frame can be completely reconstructed if any K or more packets are successfully received.

Table 2
Quality scaling levels

Level (l_{QS})	SPEG layers	Equivalent quantitative v_Q
0	L0 + L1 + L2 + L3	3
1	L0 + L1 + L2	6
2	L0 + L1	12
3	L0	24

To analyze the effects of FEC on video frames for the analytic experiments, the sending of packets is modeled as a series of independent Bernoulli trials. Thus, the probability $q(N,K,p)$ that a K-packet video frame is successfully transmitted with $N-K$ redundant FEC packets along a network path with packet loss probability p is:

$$q(N,K,p) = \sum_{i=K}^{N} \left[\binom{N}{i} (1-p)^i * p^{N-i} \right] \quad (2)$$

Given I, P, and B frame sizes, and the distribution of redundant FEC packets added to each frame type, Eq. (2) provides the probability of successful transmission for each frame type.

2.3. Objective video quality metrics

Since determining the quality of streaming video as perceived by an end-user through subjective user studies is often impractical, many objective video quality metrics have been developed to estimate the quality of streaming video.

The *playable frame rate* is the number of frames that can be decoded and played at the receiver in 1 s. Playable frame rate is simple to calculate and effective with temporal scaling, but is not as effective with quality scaling since two different video clips can have the same frame rate but different picture qualities.

Peak signal-to-noise ratio (PSNR) compares the difference between the original frame and the decoded frame pixel by pixel. Eq. (3) gives the equation for PSNR, where $D(x,y)$ is the pixel in the decoded frame and $O(x,y)$ is the original frame:

$$\begin{aligned} \mathrm{PSNR} &= 20\log(255/\mathrm{MSE}) \\ \mathrm{MSE} &= \sqrt{\frac{1}{N}\sum (D(x,y)-O(x,y))^2} \end{aligned} \quad (3)$$

While easy to compute, PSNR does not take into account human vision, and thus is not always a good predictor of perceived video quality. Besides, when applied to video, PSNR must be averaged across all frames, making it even less accurate when there is frame loss. In this paper, a lost frame's PSNR is computed using the previous played frame as a replacement, as in [11].

Video Quality Metric (VQM) is an objective video quality measurement developed by the Institute for Telecommunication Sciences (ITS) [13]. VQM uses video information such as spatial information, edge energy, temporal information and motion energy as well as PSNR. The VQM tools provided take the original video and the processed video as inputs and produce a distortion value between 0 and 1. A value of 0 means the quality of the processed video is as good as the original video and a value of 1 means the processed video has really poor quality compared to the original video. VQM shows a high correlation with subjective video quality assessments for various video codecs and has been adopted by ANSI as an objective video quality standard.

2.4. Subjective video quality measurements

The most accurate quality measurements are those of the opinions of real users, determined by subjective quality measurements. Typically, subjective measurement studies invite groups of people to watch videos and evaluate their quality. However, subjective measurement can be a time consuming and costly, at best, and impractical or impossible at worst. Moreover, different users can have different opinions on the same streamed video, and even the same user can have different opinions on the same clip under different viewing conditions, making subjective assessment difficult.

The International Telecommunication Union-Radiocommunication (ITU-R) developed a set of standards [5] to perform subjective assessments and these standards are widely accepted for determining the perceptual video quality [1,9,17]. One of the standards, the double-stimulus impairment scale (DSIS) method, is designed to evaluate either a new system or the effect of transmission path impairment. A DSIS assessment includes a series of videos in random order and with various impairments covering all required combinations. Users are first presented with an unimpaired reference, then with the same video degraded and are asked to assess the quality of the second video, comparing it to the first. At the end of the series of sessions, the mean score for each test condition and test video is calculated. A five-point degradation scale is recommended: imperceptible; perceptible, but not annoying; slightly annoying; annoying; and very annoying.

3. ARMOR quality metric

3.1. Parameters and variables

The ARMOR system layers and parameters are indicated in Table 3, where the parameters are:

G: the GOP rate (in GOPs per second) during encoding.

S_I, S_P, S_B: the size (in fixed-sized packets) of I, P or B frames, respectively, depending on quantization value v_Q.

l_{TS}: the temporal scaling level, as in Table 1.

(N_{PD}), (N_{BD}): the number of P or B frames, respectively, sent per GOP after temporal scaling, depending on l_{TS} as in Table 1.

l_{QS}: the quality scaling level, as in Table 2.

v_Q: the quantization value, depending on l_{QS} as in Table 2.

S_{IF}, S_{PF}, S_{BF}: the number of FEC packets added to each I, P or B frame, respectively.

s: the packet size (in bytes).

p: the packet loss probability.

t_{RTT}: the round-trip time (in milliseconds).

T: the capacity constraint (in packets per second), limited by the network capacity or by a TCP-Friendly rate [12].

3.2. Distortion of streaming MPEG

When a video is streamed over an unreliable network under a capacity constraint, its perceptual quality can be degraded by two factors: quantization and frame loss. Degradation from quantization is caused by low accuracy of the DCT coefficients and appears visually as coarse granularity in every frame. Degradation from frame loss, caused by temporal scaling and network packet loss, appears visually as jerkiness in the video playout.

This section uses VQM (see Section 2.3) to measure the distortion from quantization. Section 3.3 uses the playable frame rate to capture the distortion from frame loss. Section 3.4 presents a new overall measurement, namely, distorted playable frame rate, to combine these two factors.

Table 3
System layers and parameters

Layer	Parameters
MPEG	G, S_I, S_P, S_B
ARMOR: scaling and repair (FEC)	l_{TS}, N_{PD}, N_{BD}, l_{QS}, v_Q S_{IF}, S_{PF}, S_{BF}
Network	p, t_{RTT}, s, T

VQM takes an original video and a distorted video as input and returns a distortion value D between 0 (no distortion) and 1 (highest distortion). Previous research shows the perceptual video distortion varies exponentially with the quantization value [4]. Using this idea, our preliminary studies [19] measuring videos encoded with different quantization values with VQM show the distortion, D, can be approximated by an exponential function of the quantization value v_Q as:

$$D = \widehat{D} \cdot v_Q^{\lambda_D} \tag{4}$$

where v_Q is the quantization value decided by l_{QS} as in Table 2, $\widehat{D}$ is the VQM distortion when $v_Q = 1$, and λ_D is the exponential coefficient. Note, the value of λ_D varies with video sequence. However, since the value is correlated to the video's characteristics such as motion and scene complexity, it can be predicted when the video is encoded.

3.3. Playable frame rate

3.3.1. Frame size

When the quantization values change, the frame sizes, and hence, streaming bitrates, change as well. Previous research shows the bitrate of a MPEG stream can be approximated by an exponential function of the quantization value v_Q [4,16]. Our preliminary experiments [19] suggest frame size can be estimated by an exponential function of quantization value v_Q, given as:

$$\begin{cases} S_I = \widehat{S_I} \cdot v_Q^{\lambda_I} \\ S_P = \widehat{S_P} \cdot v_Q^{\lambda_P} \\ S_B = \widehat{S_B} \cdot v_Q^{\lambda_B} \end{cases} \tag{5}$$

where v_Q is the quantization value, $\widehat{S}_*$ is the frame size when $v_Q = 1$, and λ^* is the exponential coefficient. Note, all the results, S^*, need to be rounded up to the nearest integer, $\lceil S_* \rceil$, since each video frame must be divided into a whole number of packets when sent on the network.

3.3.2. Playable frame rate

Our previous work [20] derived a model to estimate total playable frame rate for streaming MPEG with temporal scaling. With the model, the total playable frame rate R can be estimated as a function of frame sizes, packet loss probability, temporal scaling pattern and FEC amounts. Essentially, with lower packet loss, smaller video frames, fewer frame dependencies, and greater FEC protection, the total playable frame rate is higher:

$$R = R(p, (N_{PD}, N_{BD}), (S_I, S_P, S_B), (S_{IF}, S_{PF}, S_{BF})) \tag{6}$$

Since N_{PD} and N_{BD} are decided by l_{TS} as in Table 1, and S_I, S_P, and S_B are decided by l_{QS} as in Eq. (5) and Table 2, this equation can be written as:

$$R = R(p, l_{TS}, l_{QS}, (S_{IF}, S_{PF}, S_{BF})) \tag{7}$$

The estimated frame rate is then used to measure the quality distortion from frame loss, as indicated in Section 3.4. In our model, only correctly decoded frames are shown to the user and corrupted (or partially decodable) frames are not shown. Our ARMOR implementation follows our model and does not show corrupted frames, either. In the event that a system wanted to show corrupted frames, our model and ARMOR implementation could be adjusted appropriately.

3.4. Distorted playable frame rate

When a video is streamed over a network with quality scaling and temporal scaling, the video quality distortion is determined by both the quantization and the frame loss. Encoding the video with a high quantization value makes every frame have a visually coarse granularity and yields intra-frame quality distortion. Any frame loss from temporal scaling and during transmission reduces the playable frame rate and results in visual jerkiness in the playout, yielding inter-frame quality distortion.

To produce the best quality video, the sender needs to use the best quantization level *and* the receiver needs to receive all the frames. So these two factors are combined in a multiplicative fashion, which is referred to as the distorted frame rate, R_D:

$$R_D = (1 - D) \cdot R \tag{8}$$

where D is the quality distortion from Eq. (4) and R is the playable frame rate from Eq. (7).

The motivation behind R_D is as follows: if a video is streamed with the best quantization value, its intra-frame quality distortion is 0 and the video quality is decided only by the playable frame rate R. With any other quantization value, every frame carries less visual detail and its contribution to the video quality (measured by the frame rate) is reduced by the quality distortion D. A previous preliminary user study in [19] indicates a high correlation between user perceptual quality and distorted playable frame rate R_D, which suggests that R_D is effective in representing the overall video quality. In Section 5, a more comprehensive user study evaluates the correlation between user perceptual quality and distorted playable frame rate, R_D.

While the discussion of R_D has been specific for MPEG-1, the analysis and results should hold for other compression techniques, as well. MPEG-4, for example, with built-in repair and local concealment will likely increase the playable frame rate R and reduce the quantization distortion D. But modification of the R model and D model, while left as future work, can still be used to predict overall video quality.

4. Optimization algorithm

For given network conditions and MPEG video parameters, the total distorted playable frame rate R_D varies with the quality scaling level, the temporal scaling level, and the amount of FEC for each type of frame as a function $R_D(p, l_{TS}, l_{QS}, (S_{IF}, S_{PF}, S_{BF}))$ where the streaming bitrate is limited by the capacity constraint, T. Thus, this model can be used to optimize R_D, the distorted playable frame rate, using the following operation research equation:

$$\begin{cases} \text{Maximize}: & R_D = (1 - D(l_{QS})) \cdot R(p, l_{TS}, l_{QS}, (S_{IF}, S_{PF}, S_{BF})) \\ \text{Subject to}: & G \cdot ((S_I(l_{QS}) + S_{IF}) + N_{PD}(l_{TS}) \cdot (S_P(l_{QS}) + S_{PF}) \\ & \quad + N_{BD}(l_{TS}) \cdot (S_B(l_{QS}) + S_{BF})) \leqslant T \end{cases} \tag{9}$$

Unfortunately, finding a closed form solution for the non-linear function R_D is difficult since there are many saddle points. However, given that the optimization problem is expressed in terms of integer variables over a restricted domain, a complete search of the discrete space is feasible. With fixed input values for (p, RTT, s), G and functions of $(N_{PD}(l_{TS}), N_{BD}(l_{TS}))$ and $(S_I(l_{QS}), S_P(l_{QS}), S_B(l_{QS}))$, each set of values of l_{TS}, l_{QS}, and (S_{IF}, S_{PF}, S_{BF}) can determine the distorted playable frame rate R_D with the following steps:

(1) l_{QS} is used to obtain a quantization value v_Q from Table 2. The video frame sizes (S_I, S_P and S_B) are then approximated using Eq. (5).
(2) The video streaming bitrate is estimated using the video frame sizes, the FEC frame sizes and $((N_{PD}), (N_{BD}))$. If the estimated bitrate is larger than the capacity constraint T, the set of values of l_{QS}, l_{TS} and (S_{IF}, S_{PF}, S_{BF}) are invalid and R_D is returned as 0.
(3) Otherwise, the playable frame rate R is estimated by inputting $(p, l_{TS}, l_{QS}, S_{IF}, S_{PF}, S_{BF})$ into Eq. (6).

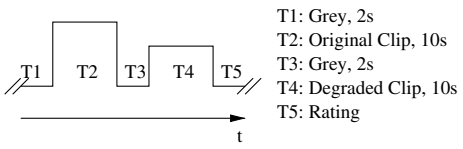

Fig. 2. Timeline for playout of one pair of clips.

(4) Using v_Q, the distortion from quality scaling D is approximated using Eq. (4).
(5) Knowing R and D, the distorted playable frame rate R_D is estimated using Eq. (8).

With these steps for each set of values, the space of possible values for l_{TS}, l_{QS} and (S_{IF}, S_{PF}, S_{BF}) can be completely searched to determine the scaling levels and the amount of FEC packets for each frame type to maximize the distorted playable frame rate under the capacity constraint. In fact, the computation required by the search can be done in real-time,[2] making the determination of optimal choices for adaptive FEC feasible for most streaming MPEG connections.

Notice, some of the searching variables can be fixed. For example, if l_{TS} is fixed at 0, then the algorithm uses quality scaling only. If (S_{IF}, S_{PF}, S_{BF}) are fixed at to 0, the video is transmitted without FEC.

5. User study

The distorted playable frame rate (R_D) derived in Section 3 provides an analytic estimate of streaming video quality when repair and media scaling are employed. This section provides the details of a comprehensive user study designed to evaluate R_D. Individual volunteer users were asked to carefully view selected video clips that were chosen as being representative of streaming videos with FEC, scaling and packet loss. Users were asked to compare the original clip against a degraded clip and rate the difference in perceived quality. The user results are analyzed and used to determine whether distorted playable frame rate, R_D, accurately reflects perceived user quality for a variety of videos – with low or high motion, temporal or quality distortion, with or without repair and varying amounts of simulated packet loss.

5.1. Methodology

To compare the degradation of the impaired video to the original video, the DSIS method was used (see Section 2.4) with minor modifications. Fig. 2 shows the timeline of one pair of testing clips. Specifically, for each pair of video clips, the original 10-s clip is presented first, followed by one 10-s version of the same clip, degraded by scaling and frame loss. After viewing the pair of clips, the user rates the second clip relative to the first clip on a five-point degradation scale from "Same" to "Much worse" without additional labels to avoid numeric biasing. The display is grayed out for 2 s between clips and between subsequent clip pairs.

A Visual C++ test harness application was created for the user study. Each user downloads the application bundled with the videos to a Windows PC and executes the test locally. Initially, the application gathers user demographic information and provides user directions. Seventeen pairs of 10-s video clips (see Section 5.3) are presented. After each clip pair, the user rates the difference in perceptual quality. Normally, each user takes about 8–10 min to complete the study. The application includes an option whereby the user can exit the application in the middle of the experiment.

[2] It takes about 20 ms to find the best FEC and scaling pattern using our approach on a Pentium-4 1.7 GHz PC for a GOP of 'IBBPBBPBBPBBPBB'. Optimizations of the code and a faster machine can make searching even faster.

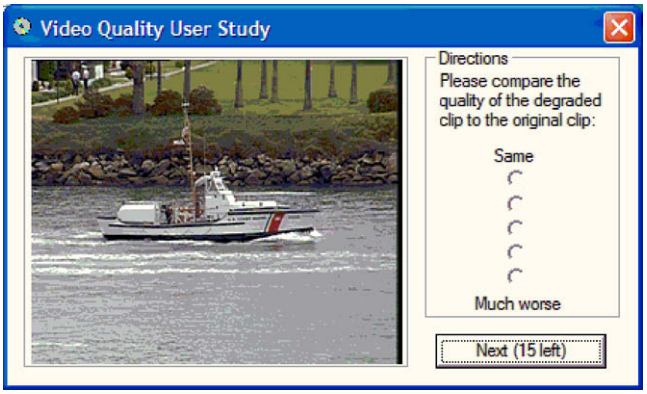

Fig. 3. Screenshot of the main dialog.

To encourage participation, all participants were eligible for a raffle of one $50, two $25 and five $10 BestBuy gift certificates, and there was also extra credit given for participants from one academic course.

5.2. User study application

The application begins by displaying a dialog box to collect demographic data that includes gender, age, major, status, computer video viewing frequency and computer monitor type. A background system call automatically retrieves each user's screen resolution. After presenting another dialog box that provides user directions, the application opens the main dialog box, shown in Fig. 3, that displays the video clips and asks the user to compare the quality of the degraded clip to the original clip. After the user proceeds through each pair of video clips, *Notepad* is opened with the results displayed and the user is asked to copy all the results and send them via email to a central repository.

5.3. Video clips

Seventeen video clip pairs were used in the study.[3] The first video pair was a training exercise designed to provide the user with an understanding of the evaluation system and to serve as a baseline for the "worst" quality video presented. For training, the original clip was played followed by a severely degraded version with only the "much worse" option available for selection. The next 16 pairs were evaluated by the user. For each clip pair, the first clip was the original with the best quality and the second clip was degraded, using all combinations of the following four independent factors:

(1) Video content:[4] low motion (*News*) with a "talking head" of a person reading the news or high motion (Coastguard, *Coast*) showing a moving ship with a panning background.
(2) Packet loss rate: low loss rate (*1%*) or high loss rate (*4%*).
(3) Repair: adjusted FEC (*Adjusted*) or no repair (*None*).
(4) Scaling method: temporal scaling (*TS*) or quality scaling (*QS*).

In relation to the ARMOR layers as in Table 3 in the ARMOR architecture, the first two factors are provided by the application and network layers, respectively, while the levels for the last two factors are optimally determined by the ARMOR algorithm.

[3] Our previous experience with such users studies shows that 10–15 min is the time limit for participation for most users. Hence, the limit of about seventeen 30-s clips.
[4] Given the limited number of video clips each user would view, one video from each end of the motion/scene complexity spectrum was selected.

Table 4
Clip factor combination

Video	Loss (%)	Repair	Scaling	R	D	R_D
News	1	None	TS	20.0	0.09	18.2
News	1	None	QS	23.7	0.13	20.5
News	1	Adjusted	TS	30.0	0.09	27.2
News	1	Adjusted	QS	30.0	0.09	27.2
News	4	None	TS	2.4	0.09	2.2
News	4	None	QS	20.1	0.37	12.6
News	4	Adjusted	TS	7.2	0.09	6.5
News	4	Adjusted	QS	29.6	0.25	22.3
Coast	1	None	TS	4.4	0.06	4.2
Coast	1	None	QS	22.9	0.27	16.8
Coast	1	Adjusted	TS	8.0	0.06	7.5
Coast	1	Adjusted	QS	30.0	0.27	21.9
Coast	4	None	TS	0.6	0.06	0.6
Coast	4	None	QS	17.8	0.41	10.5
Coast	4	Adjusted	TS	2.0	0.06	1.9
Coast	4	Adjusted	QS	28.3	0.41	16.7

Table 4 identifies the factor combinations for the 16 video clips, the playable frame rate R, quantization distortion D, and the distorted playable frame rate R_D derived from the model. For the degraded clips, the video data plus repair data are temporally or quality scaled to meet a TCP-Friendly constraint such that 1.76 Mbps is available in the 1% loss case and 0.69 Mbps is available in the 4% loss case.

5.4. Results

All 74 users who voluntarily took part in the study finished the experiment completely. Sixty-five of the users were male with 53 in their late teens and early twenties, 15 users were between 23 and 31 and only 6 users were older than 31. Over half of the users were computer science majors. Over 60% of the users were undergraduate students and about one-third of the users were graduate students. About half of the users indicated they watched computer videos daily and about 90% of the users watched computer videos at least once a week. Two-thirds of the users had an LCD monitor. More than one-third of the users had a screen resolution of 1280×1024 and another one-third had a resolution of 1024×768.

5.4.1. Video quality

Fig. 4 correlates distorted playable frame rate (R_D) and the perceptual quality expressed by the users' scores. Each data point in the figure represents the comparison of the original and degraded video clips. The x-axis is the distorted playable frame rate R_D derived using the model in Section 3 and the y-axis is the mean rating score for all users from "Much worse" (1) to "Same" (5), shown with a 95% confidence interval. Visually, the relationship between R_D and user score is almost linear. The error on a least squares line fit of the data points confirms this, having an r-square value[5] of 0.962.

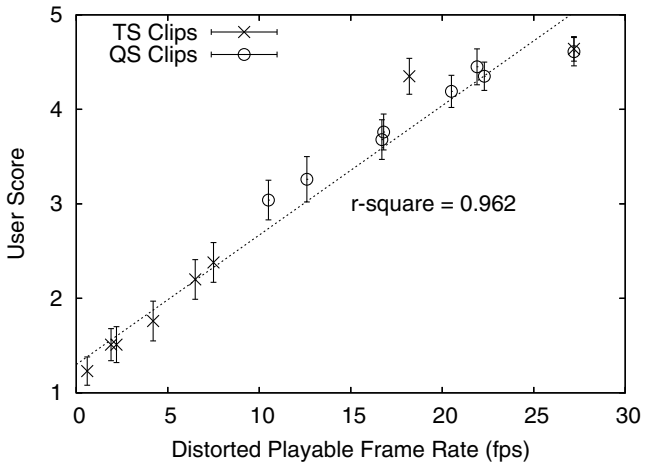

Fig. 4. User score versus distorted playable frame rate (R_D).

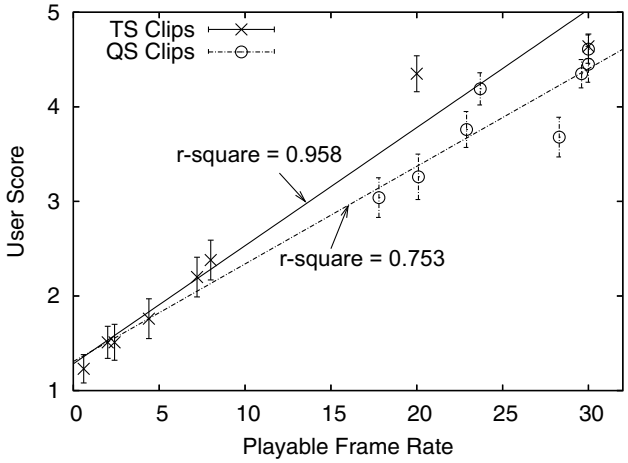

Fig. 5. User score versus playable frame rate (R).

To provide a reference for the accuracy of R_D, the user ratings are compared against two commonly used video quality metrics, playable frame rate (R) and PSNR, and against the more sophisticated, but less widely used, VQM metric.

When using only temporal scaling, the playable frame rate R can be used as a predictor of perceptual quality. Fig. 5 depicts the correlation of R and user perceptual quality. Each data point represents a video pair comparison, where the '×'s are temporally scaled clips and the 'o's are quality scaled clips. The x-axis is the playable frame rate R derived by Eq. (6) and y-axis is the mean rating score for all users, shown with a 95% confidence interval. The solid line with an r-square of 0.958 is a least squares fit with the data points from the temporally scaled clips and the dashed line with an r-square of 0.753 is a least squares fit with the data points from the quality scaled clips. While the correlation of R and user score for temporal scaling is nearly linear, the correlation for R and user score is significantly lower for quality scaling. This implies playable frame rate is an ineffective measure of perceptual quality for streaming clips when quality scaling is used.

VQM score (1 – the distortion value provided by VQM) is compared to user perceptual quality in Fig. 6. Each data point represents a video pair comparison where the y-axis is the mean user rating score with a 95% confidence interval and the x-axis is one minus the distortion on the degraded clip compared to the original clip, measured by the VQM tool. The visual correlation between VQM score and user score is not as strong as for R_D and is represented by the least squares line fit r-square of 0.884.

Fig. 7 depicts the correlation for PSNR and user perceptual quality. Each data point represents a video pair and the y-axis is the mean user rating score shown with a 95% confidence interval. The x-axis is the PSNR (in decibels) computed with the original clip and the degraded clip. The correlation between PSNR and user score is even lower than that for VQM, and the least squares line fit has an r-square of 0.821.

[5] r-square, the coefficient of determination, is a measure of how the line explains the variation in the data, and an r-square of 1 means a perfect line fit.

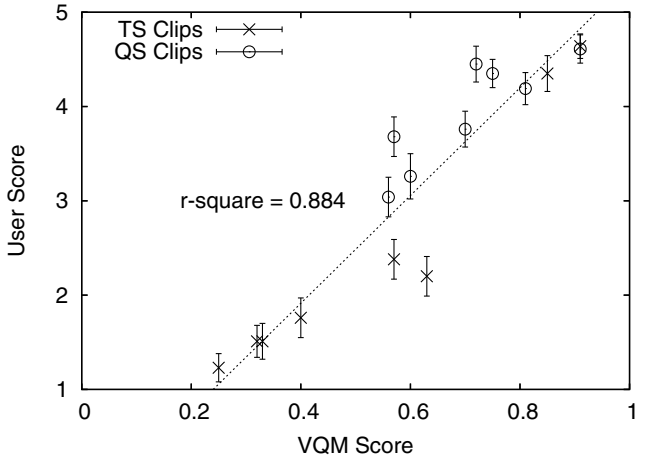

Fig. 6. User score versus VQM score.

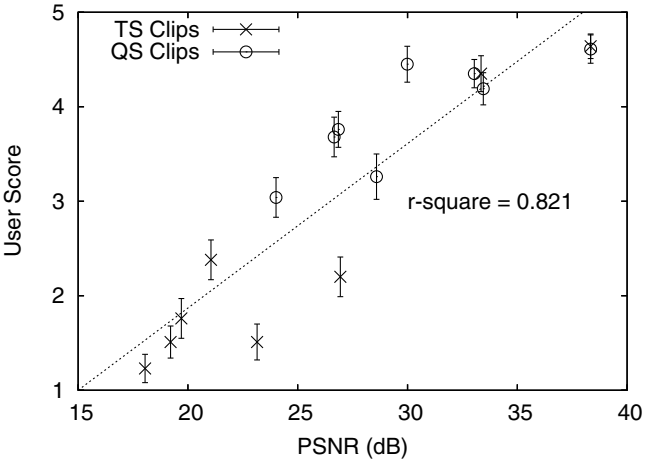

Fig. 7. User score versus peak signal-to-noise ratio (PSNR).

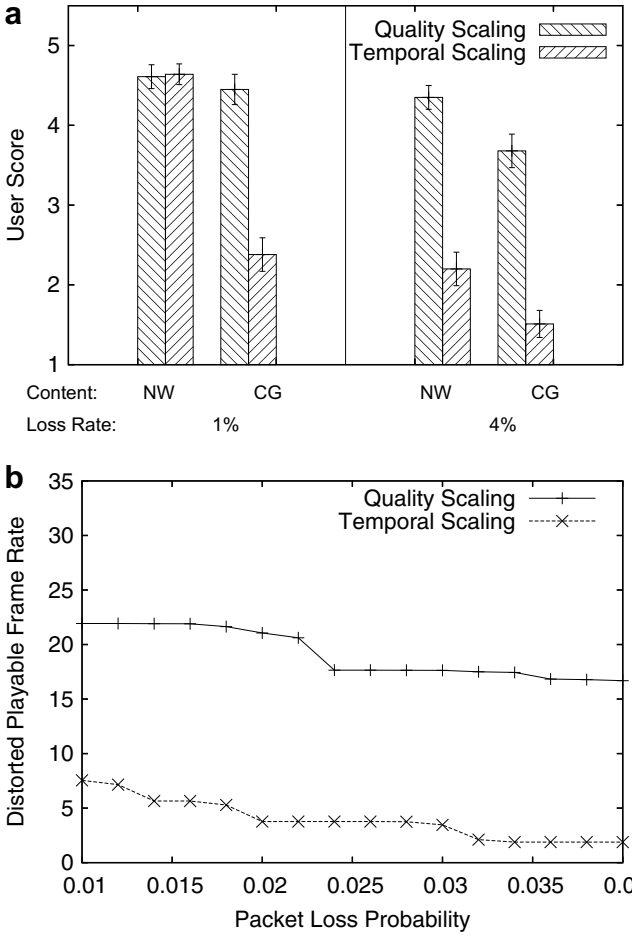

Fig. 8. Temporal scaling versus quality scaling (all with adjusted repair). (a) Measured user scores, (b) ARMOR predicted scores (for coastguard).

Table 5
t-Test of raw user scores, comparing quality scaling and temporal scaling

Video	Loss (%)	Mean difference	*p*-Value
News	1	0.03	0.748
News	4	2.15	0.001
Coast	1	2.07	0.001
Coast	4	2.16	0.001

In summary, the figures presented show that distorted playable frame rate R_D more accurately reflects user perceptual quality than playable frame rate R, VQM or PSNR. This fact is leveraged in the ARMOR system where R_D is utilized to optimize the repair and scaling for a video stream.

5.4.2. Scaling methods

Given the network conditions, ARMOR can optimize MPEG streaming video quality by adjusting the temporal and quality scaling. This section focuses on analyzing the user video perception scoring in relation to adaptive scaling techniques. ARMOR analytic modeling is also provided to extend the measured behavior to include predicted behavior in additional system situations.

Fig. 8a provides bar graphs of the mean user rating with a 95% confidence interval, and compares temporal and quality scaling where all the clips are protected by adjusted repair to overcome network packet loss. The left region includes clips subject to a 1% loss rate and a TCP-Friendly constraint of 1.76 Mbps while the right region represents clips subject to 4% loss and a TCP-Friendly constraint of 0.69 Mbps. For both regions the two left-most bars are scores from the low motion *News* clips and the right-most bars are scores from the high motion *Coastguard* clips.

In the relatively unconstrained-capacity region on the left, when loss rate is low and the video clip has little motion, temporal and quality scaling are equally effective in providing high quality video. However, in the other three scenarios, quality scaling produces much better perceptual quality than temporal scaling. Table 5 provides *t*-test scores that compare the users' ratings between quality and temporal scaling. The table shows the benefits from quality scaling over temporal scaling are statistically significant except for the aforementioned first case, where the low-motion clip is relatively unconstrained by the available bandwidth.

As confirmed by our user study, the distorted playable frame rate R_D can be used in analytic experiments to predict the impact of the scaling method on the streaming quality for a range of video and network parameters. However, for clarity, only the Coastguard video results over a packet loss range are shown. In Fig. 8b where the *x*-axis is the packet loss probability and the *y*-axis is the predicted distorted playable frame rate the two graphs provide data sets for the Coastguard video under quality and temporal scaling. Clearly, the model shows that for this high motion clip quality scaling should provide substantial benefits to distorted playable frame rate over temporal scaling over the entire loss rate range. The quality scaling improvement on average is approximately 10 frames/s.

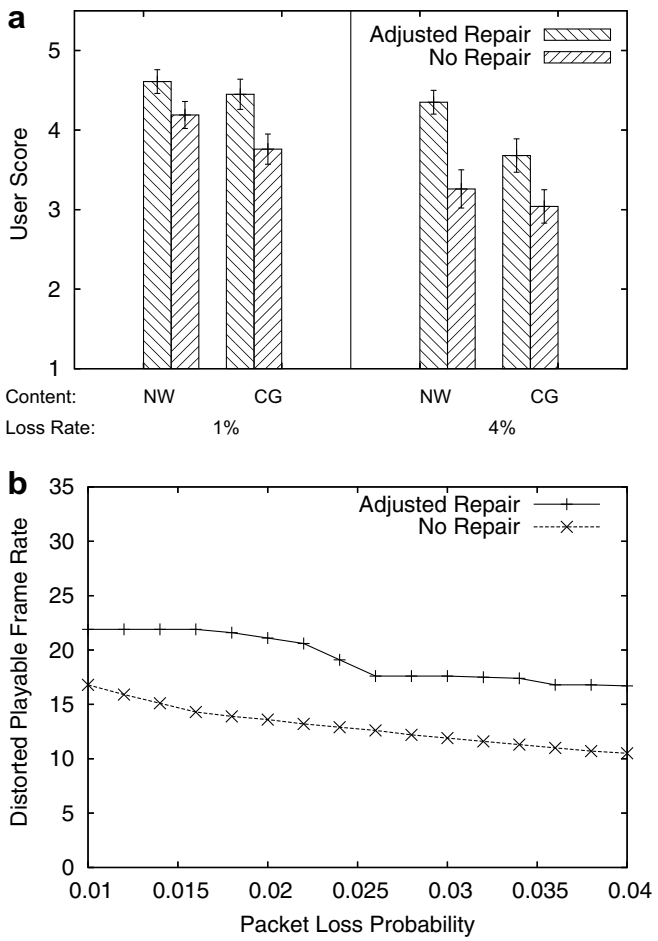

Fig. 9. Adjusted repair versus no repair (all with quality scaling). (a) Measured user scores, (b) ARMOR predicted scores (for coastguard).

5.4.3. Repair methods

Given the network conditions, ARMOR can optimize MPEG streaming video quality by adjusting the repair. This section analyzes the user scores to gauge the impact of adjusting repair on video perceptual quality and reviews the results of analytic experiments run to predict perceptual quality for additional system conditions.

Fig. 9a partitions the user study scores to consider the improved video quality due to adjusted repair. The axes and regions are the same as in Fig. 8a, and the data sets are for the clips with adjusted repair and the clips without repair. In this figure, the *x*-axis is the video clip instance and the *y*-axis is the mean rating score for all users shown with 95% confidence intervals. The left region has a loss rate of 1% and a TCP-Friendly constraint of 1.76 Mbps while the right region has a loss rate of 4% and a TCP-Friendly constraint of 0.69 Mbps. In each region, the left side bars are the low motion *News* clips and the right side bars are the high motion *Coastguard* clips. The '\' bars represent the clips with adjusted repair and the '/' bars represent the clips without repair. For all the clips, the total bitrate used by the video and the repair is scaled to meet a TCP-Friendly capacity constraint using quality scaling.

The figure shows that adjusted repair provides better perceptual quality than no repair in all cases. Table 6 gives *t*-test results between user ratings for videos with adjusted repair and user ratings for videos with no repair. The table shows the benefits from adjusted repair are statistically significant in all cases.

Next, R_D is used to analytically compare the impact of repair methods on streaming quality over a range of network loss conditions. Fig. 9b depicts the analytic experiment results with the same axes as in Fig. 8b and the two data sets being Coastguard with adjusted repair and Coastguard with no repair. Fig. 9b shows adjusted repair provides substantial benefits to distorted playable frame rate versus no repair over the entire range of loss rates. The average improvement is approximately 5 frames/s.

Table 6
t-Test of raw user score, comparing adjusted repair and no repair

Video	Loss (%)	Mean difference	*p*-Value
News	1	0.42	0.002
News	4	1.09	0.001
Coast	1	0.69	0.001
Coast	4	0.64	0.001

6. Implementation

This section describes the implementation of a fully-functioning implementation of the ARMOR streaming video system and shows results from preliminary experiments evaluating ARMOR on a network testbed. In particular, the ARMOR system incorporates: (1) full-featured MPEG encoding, (2) dynamic temporal scaling and FEC repair that adapt every GOP, (3) a network protocol that measure current loss rates and round-trip times and streams at a TCP-Friendly bitrate, and (4) the ARMOR quality metric and optimization algorithm that adjust repair and scaling to maximize the video quality. The effect of this implementation lends credibility to using the ARMOR model and optimization algorithm to predict video quality, shows that dynamic adjustment of repair and media scaling can be effectively done in practice, and provides an architecture for use by other streaming video systems.

6.1. System overview

The system architecture is depicted in Fig. 1 in Section 1. There are two types of modules in the figure: (1) our ARMOR module, which is the quality model and optimization algorithm and is denoted by a gray box, and (2) all other data processing modules denoted by transparent boxes. The information flows can also be divided into two types: video data flows denoted by wide dark arrows, and ARMOR data and control flows denoted by thin arrows.

The text below describes the ARMOR system, considering first the video data flow, then the ARMOR flows:

(1) A data flow starts in the *Image Sequence Repository* as a sequence of raw images stored on disk or captured live and is encoded into video frames. In our current implementation, the raw images are stored on disk and encoded into MPEG-1 using FFMPEG.[6] A simple ARMOR header is added before each MPEG frame, containing the frame size (in bytes), frame type (I, P or B), and frame sequence number.
(2) The video frames, including their ARMOR headers, are then scaled to meet bitrate constraints by the *Media Scaler*. Currently, only temporal scaling is implemented, allowing some low priority frames to be discarded using the scaling levels in Table 1. However, since evaluation of temporal scaling covers a larger range of perceptual quality degradation than quality scaling, the results should hold for quality scaling, as well. Also, adding quality scaling to FFMPEG would require a substantial engineering effort, so implementing quality scaling is left as future work.

[6] http://ffmpeg.sourceforge.net/index.php.

(3) The *Repair Encoder* takes the video frames, including their ARMOR headers, from the Media Scaler, splits each frame into packets, and adds repair data as appropriate. The current implementation uses a wrapper class over Rizzo's software FEC [15] to provide packet-level FEC.
(4) The *UDP Sender* takes the video and repair packets from the Repair Encoder and sends them over the wide-area network (WAN, shown by a cloud in Fig. 1)[7] to meet the network's bitrate constraint. The current implementation sends at a TCP-Friendly data rate, computed as in Padhye et al. [12], based on loss and round-trip time feedback from the UDP receiver.
(5) The *UDP Receiver* at the client receives the video and repair packets and provides network feedback to the UDP Sender. Currently, the receiver computes the loss rate and round-trip time for a 5-s sliding window and reports to the sender every 200 ms.
(6) The *Repair Decoder* uses the video and repair packets to try to recover each frame. When the video frame cannot be recovered, in our case when the number of lost packets is higher than the number of redundancy packets, the frame is discarded as unplayable. Otherwise, it is playable and sent to the PrePlayer.
(7) The *PrePlayer* takes the decoded frames from FEC Decoder and removes the ARMOR headers. It also records some basic statistics for performance evaluation, such as number of sent frames, number of received frames, and playable frame rates.
(8) The *MPEG Player*, in our case an FFMPEG player, takes the playable MPEG frames from the PrePlayer and plays the video out on the screen.
(9) The *ARMOR module*, shown in grey, takes MPEG parameters from the MPEG Encoder and network parameters from the UDP Sender and determines the scaling and repair that provide the best distorted playable frame rate. The current implementation computes the temporal scaling level and amount of FEC every GOP that maximizes the playable frame rate.[8] The GOP pattern and MPEG frame sizes from the previous GOP are used for the current GOP, along with the latest network loss rate and bitrate constraint provided by the UDP sender.

As a rough estimate of the size of the implementation, the approximate number of lines of code and computation complexity for each component[9] is provided in Table 7. The ARMOR system is about 14k lines of code total, with about 75% of that original code from the full-featured FFMPEG. The original FEC code was modified somewhat, and in total provides about 5% of the code base. The UDP networking code and the ARMOR module are about 6% of the total code, respectively. The computation time is mostly (82%) spent on video encoding and decoding. The FEC component takes up about 9% of the computation time. The ARMOR module is very light-weight and takes up only 1% of the computation time.

6.2. Experimental settings

Preliminary experiments with the ARMOR implementation are run to validate analytic experiments of predicted performance and demonstrate the practicality of the working ARMOR system. The packet size, round-trip time and packet loss probability are chosen based on the characteristics of many network connections [3,6]. For all experiments, the round-trip time t_{RTT} is fixed at 50 ms in the NistNet router, and packet loss probability p is varied from 0.01 to 0.04 in steps of 0.005 with a packet size s of 1 Kbyte.

A video clip named *Paris* is used to provide a test-case with a medium amount of motion and detail. Paris shows two people sitting at a table and talking while making high-motion gestures, and has 1200 raw images of size 352 × 288 pixels (CIF). A commonly used MPEG GOP pattern, '`IBBPBBPBBPBBPBB`', and a typical full motion encoding frame rate R_F of 30 frames/s (fps) are used, providing 40 s of encoded MPEG video. These settings yield the average I, P and B frame sizes to be 24.24 Kbytes, 5.20 Kbytes and 1.18 Kbytes, respectively, and are rounded up to the next integer for the analytic experiments.

Table 7
Lines of code and computation complexity for ARMOR system components

Component	Approximate lines of code			Computation complexity
	Original	New	Total	
FFMPEG	10,000	50	10,050	0.82
Rizzo's FEC	650	330	980	0.07
FEC Wrapper	–	1010	1010	0.02
UDP	–	850	850	0.02
ARMOR	–	840	840	0.01
PrePlayer	–	290	290	0.04
Scaler	–	50	50	0.02
Total	10,650	3420	14,070	1

6.3. Results

Using the implemented ARMOR system and the experimental settings, the playable frame rates with different repair methods over a range of network loss rates are explored. For each loss rate, the playable frame rates are compared for MPEG streaming without repair and MPEG streaming with adjusted repair.

Fig. 10 depicts the playable frame rates for adjusted repair and no repair for the *Paris* video. Fig. 10a depicts the result measured in the implemented ARMOR system while Fig. 10b shows the result from the analytic experiments that use only the ARMOR model and algorithm. The x-axes are the packet loss probabilities, and the y-axes are the playable frame rates. From the data in these figures, adjusted repair provides better video quality over the range of network loss rates. The benefits of adjusted repair over no repair are substantial, with adjusted repair providing 3–10 more frames per second for all rates.

The figures also demonstrate that the experiment results from the implemented system are quite similar to those from the analytic models, suggesting that the ARMOR optimizations are robust in face of system inaccuracies that are introduced in real-world implementations. For some loss rates, our analytical estimates do differ from the actual frame rates achieved by the real system, mostly stemming from inaccurate loss and frame size predictions, resulting in a slightly sub-optimal use of FEC. Future work includes replacing the UDP modules with alternate streaming protocols, such as DCCP [7], for more accurate network adaptation. Still the difference is only about 1.5 frames/s on average and for most cases, the ARMOR system yields better quality than the quality predicted by analytical experiment.

7. Conclusions

This paper presents ARMOR, a novel system for adjusting repair and media scaling for streaming video. The contributions include: (1) a new metric, distorted playable frame rate, for predicting video

[7] In our testbed, the WAN is emulated by NistNet [2].
[8] The playable frame rate is equivalent to the distorted playable frame rate when there is only temporal scaling.
[9] A *component* in this context may include several modules, such as the MPEG Encoder and Player.

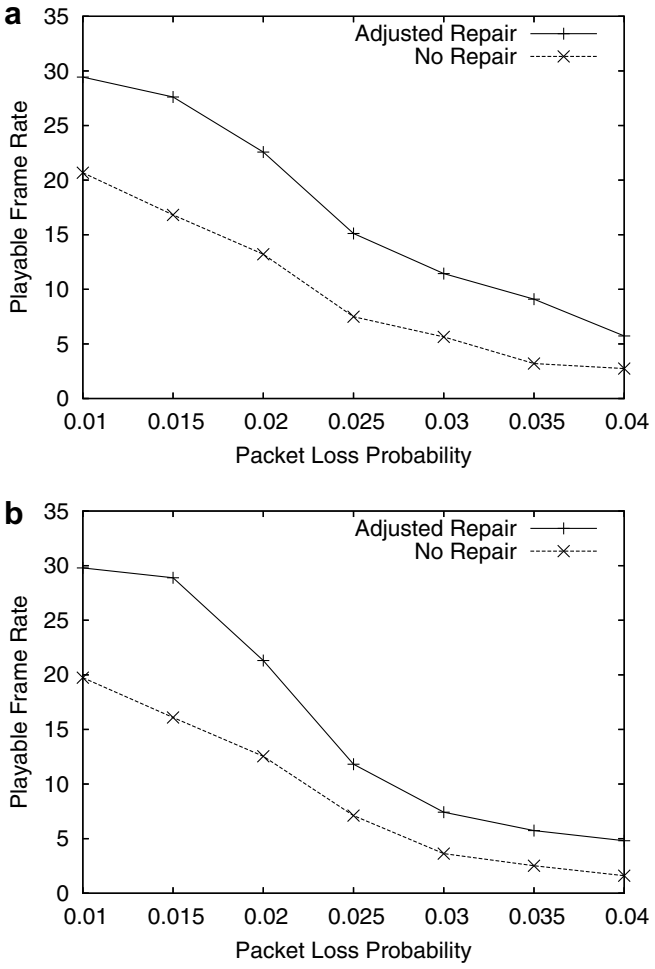

Fig. 10. Comparison of repair methods (paris video with temporal scaling). (a) ARMOR system measurement, (b) analytical experiment.

quality over a range of video and network conditions; (2) a comprehensive user study that demonstrates the accuracy of distorted playable frame rate compared to alternative approaches; (3) the core ARMOR module that includes a quality metric of streaming video with scaling and repair and an optimization algorithm that adjusts the repair and scaling to maximize the distorted playable frame rate; (4) the ARMOR system architecture, which includes video, network and ARMOR modules; and (5) analytic and implementation experiments that show the ability of the ARMOR system to adapt to network and video factors to improve streaming video quality.

Specific findings include:

Analysis of user ratings from a large user study shows that distorted playable frame rate is effective at capturing the perceived quality of video with degradations from both temporal scaling and frame loss as well as from quality scaling. The other widely used metrics of PSNR and playable frame rate, as well as the more sophisticated VQM, are not as effective at predicting perceptual quality.

Analysis of user ratings and analytic experiments with the ARMOR model demonstrate that quality scaling provides better video quality than does temporal scaling under capacity constrained conditions. Quality and temporal scaling provide approximately the same quality for streaming low motion videos over a network with little packet loss. Video streamed with repair, in the form of FEC, is substantially beneficial to video quality compared with video streamed with no repair. However, the benefits to video quality from adding repair are not as significant as the benefits from choosing the right method of media scaling.

The proposed ARMOR system is more than an offline, analytic tool for evaluating scaling and repair choices, but is effective in practice, demonstrated by implementation experiments on a fully-implemented system. The ARMOR model and optimization algorithm can be run in real-time, adjusting the repair and scaling per MPEG GOP, with the experimental results showing the analytic predictions closely match the measured results. The ARMOR system also illustrates the ability of the encoding, scaling, repair and network modules to work in conjunction with the ARMOR module that determines the optimal scaling and repair.

Possible future work includes implementation of quality scaling and alternate network protocols that are specifically designed for streaming media. In addition, bandwidth estimation techniques used by commercial media players at the start of a streaming session, could be used to determine the initial streaming rate. Alternative repair techniques may provide additional merits over the FEC approached used in this paper. ARMOR could be evaluated with time-varying bandwidth and packet loss in conjunction with these system improvements. Future work can use classification of video motion and scene complexity to predict the λ^* exponential coefficients (Section 3). And future user studies could seek to determine if the same distorted playable frame rate is achieved by using different combinations of temporal scaling and quality scaling.

References

[1] W. Ashmawi, R. Guerin, S. Wolf, M. Pinson, On the impact of policing rate guarantees in Diff-Serv networks: a video streaming application perspective, Proceedings of ACM SIGCOMM (2001) 83–95.
[2] M. Carson, D. Santay. NIST Net – A Linux-based Network Emulation Tool. Available from: <http://snad.ncsl.nist.gov/itg/nistnet>.
[3] J. Chung, M. Claypool, Y. Zhu. Measurement of the congestion responsiveness of RealPlayer streaming video over UDP, Proceedings of the Packet Video Workshop (PV), Nantes, France, April 2003.
[4] P. Frossard, O. Verscheure, Joint source/FEC rate selection for quality-optimal MPEG-2 video delivery, IEEE Transactions on Image Processing 10 (12) (2001) 1815–1825.
[5] International Telecommunications Union. Methodology for the subjective assessment of the quality of television pictures. Technical Report ITU-R BT.500-11, ITU Recommendation, 2002.
[6] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, D. Towsley, Inferring TCP connection characteristics through passive measurements, In: Proceedings of IEEE Infocom, Hong Kong, China, April 2004.
[7] E. Kohler, M. Handley, S. Floyd, Datagram congestion control protocol (DCCP), IETF Request for Comments (RFC) 4340 (2006).
[8] C. Krasic, J. Walpole, W.-C. Feng, Quality-adaptive media streaming by priority drop, Proceedings of Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV), Monterey, CA, USA, June 2003, pp. 112–121.
[9] R.R.-F. Liao, A.T. Campbell, A utility-based approach for quantitative adaptation in wireless packet networks, Wireless Networks 7 (5) (2001) 541–557.
[10] K. Mayer-Patel, L. Le, G. Carle, An MPEG performance model and its application to adaptive forward error correction, Proceedings of ACM Multimedia, December 2002.
[11] K. Ngan, T. Meier, Z. Chen, Improved single-video-object rate control for MPEG-4, IEEE Transactions on Circuits and Systems for Video Technology (2003).
[12] J. Padhye, V. Firoiu, D. Towsley, J. Kurose, Modeling TCP throughput: a simple model and its empirical validation, Proceedings of ACM SIGCOMM, Vancouver, Brisish Columbia, Candada, 1998.
[13] M. Pinson, S. Wolf, A new standardized method for objectively measuring video quality, IEEE Transactions on Broadcasting 50 (3) (2004) 312–322.
[14] I.S. Reed, G. Solomon, Polynomial codes over certain finite fields, Journal of the Society of Industrial and Applied Mathematics (SIAM) 8 (2) (1960) 300–304.

[15] L. Rizzo, Effective erasure codes for reliable computer communication protocols, Computer Communication Review (1997).
[16] S. Sakazawa, Y. Takishima, M. Wada, Y. Hatori. Coding control scheme for a multi-encoder system, Proceedings of 7th International Workshop on Packet Video (PV), March 1996.
[17] S. Winkler, C. Faller, Audiovisual quality evaluation of low-bitrate video, in: G. John (Ed.), Proceedings of SPIE Human Vision and Electronic Imaging. San Jose, CA, January 2005.
[18] S. Wolf, M. Pinson. Application of the ntia general video quality metric (VQM) to HDTV quality monitoring, Proceedings of The Third International Workshop on Video Processing and Quality Metrics for Consumer Electronics (VPQM), Scottsdale, AZ, USA, January 2007.
[19] H. Wu, M. Claypool, R. Kinicki. Adjusting forward error correction with quality scaling for streaming MPEG, Proceedings of Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV), Stevenson, WA, USA, June 2005.
[20] H. Wu, M. Claypool, R. Kinicki, Adjusting forward error correction with temporal scaling for TCP-friendly streaming MPEG, ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP) 1 (4) (2005) 315–337.