Mobile Netw Appl (2014) 19:552–562
DOI 10.1007/s11036-014-0519-6

Smartphones *Apps* Implementing a Heuristic Joint Coding for Video Transmissions over Mobile Networks

Igor Bisio · Fabio Lavagetto · Giulio Luzzati · Mario Marchese

Published online: 21 June 2014

Abstract This paper presents the scheme Heuristic Application Layer Joint Coding (Heuristic-ALJC) for video transmissions aimed at adaptively and jointly varying both applied video compression and source encoding at the application layer used to protect video streams. Heuristic-ALJC includes also a simple acknowledgement based adaptation of the transmission rate and acts on the basis of feedback information about the overall network status estimated in terms of maximum allowable network throughput and link quality (*lossiness*). Heuristic-ALJC is implemented through two smartphone Apps (transmitter and receiver) and is suitable to be employed to transmit video streams over networks based on time varying and possibly lossy channels. A deep performance investigation, carried out through a real implementation of the *App*s over Android smartphones, compares Heuristic-ALJC with static schemes.

Keywords Smartphone Apps · Mobile networks · Heuristic application layer joint coding

I. Bisio (✉) · F. Lavagetto · G. Luzzati · M. Marchese
Università degli Studi di Genova, DITEN, Via all'Opera Pia 13, 16145, Genoa, Italy
e-mail: igor.bisio@unige.it

F. Lavagetto
e-mail: fabio.lavagetto@unige.it

G. Luzzati
e-mail: giulio.luzzati@edu.unige.it

M. Marchese
e-mail: mario.marchese@unige.it

1 Introduction

The nature of the modern Internet is heterogeneous and implies the technical challenges of Quality of Service (QoS) guarantees and the quick deployment of new telecommunications solutions. These challenges need significant effort in the fields of the design of reliable and reconfigurable transmission systems, open source software, interoperability and scalability [6].

The mentioned internet scenario constitutes the reference for this paper: the considered network is characterized by radio and satellite links and includes mobile devices such as smartphones, employed to acquire and transmit video streams through dedicated *App*s. An applicative example of the considered environment concerns future safety support services: after a critical event (e.g., a road accident, a fire), first responders (e.g., a rescue team or just a person on site) can register a video by a smartphone and send it to an experienced operator over wireless/satellite heterogeneous network to allow managing rescue operations more consciously. Such heterogeneous wireless networks may be exploited also for other application scenarios: scheduling of video [16, 18], and of massive multimedia flows [17]; electronic help for elderly people and tele-medicine [4, 13]; applications for vehicles [10], trains [8], and planes. The mentioned environments may be joined under the name *networks for social support*. Another interesting application scenario may be represented by tactical communications [14]. Offering real-time video services with a satisfactory level of Quality of Experience (QoE), over these hybrid systems is very challenging, as it implies the solution of research and development issues due to the peculiarity of the scenario. For example, due to mobility and network wireless nature, links are characterized by low Signal-to-Noise Ratios (SNRs). This may lead to meaningful bit error rates and consequent packet losses (also called cancellations). Moreover, when dealing with wireless channels, time invariance assumption can no longer hold: typical wireless channels may exhibit extremely quick dynamics, due to a number of factors such as multipath fading, shadowing, radio interferences, and weather conditions. In the described framework, static management of video compression and protection is not an optimal choice. Dynamic adaptation of video flow is necessary. It may be acted by opportunely tuning both the amount of data offered to the transmitting device and the amount of redundancy packets to protect the video from losses. A possible improvement may derive by considering the impact that each of these tunings has on the other and by evaluating the joint effect of the two on the whole system performance. Following this scientific line, to guarantee a ready-to-use and satisfactory video fruition, two *App*s, based on the Android OS, have been designed, implemented and tested. As described in detail in the remainder of this paper, the *App*s, a Transmitter *App* and a Receiver *App*, employ an application layer joint coding algorithm for video transmission based on a heuristic approach suited to be applied over smartphone platforms. The algorithm adaptively and jointly varies both video compression and channel coding to protect the video stream. It operates at the application layer and it is based on the overall network conditions estimated in terms of network maximum allowable throughput and quality (packet cancellations or *lossiness*): on the basis of information about packet loss, a given protection level is chosen; in practice, the amount of information and redundancy packets is chosen. Estabilished the amount of available information packets and estimated the maximum allowable network throughput, video compression is consequently adapted to assure the best quality. The proposed solution also includes a simple acknowledgement-based adaptation of the transmission rate at the application layer aimed at not losing information in the application layer buffers. The proposed application layer joint coder considers the underlying funtional layers as a black box. The *App*s do not need any knowledge about implementation details and do not require any intervention regarding the underlying layers. The remainder of the paper is organized as follows: Section 2 surveys the state of the art regarding source-channel joint coding and summarizes the aim of the paper. Section 3 describes the scientific basis of the application layer joint coding and the ALJC problem. Section 4 describes the implemented smartphone *App*s. Section 5 contains the description of the proposed Heuristic Application Layer Joint Coding (Heuristic ALJC) approach. The numerical results are discussed in Section 6. Conclusions are drawn in Section 7.

2 State of the art and aim of the paper

Choi and Momcilovic [5] demonstrates the existence of two sub-spaces called performance regions, and shows that the employment of application layer coding is significantly advantageous in one region, while it is detrimental in the other one. The first performance region contains the systems that experience light channel errors and low packet loss probability. The second region contains the systems characterized by relevant channel errors. Referring to [5], the mentioned coding approach may improve the performance only in the systems with low packet loss probability due to channel errors because error prone channels require so high levels of redundancy that they cause packet losses due to congestion. A solution to this limit is proposed in [2, Chapter 1]: increasing protection does not result in an increased offered load because the packet transmission rate is kept constant or, as done in this paper, adapted to the estimated maximum network throughput. Controlling the overall packet transmission rate, the network load is under control but, increasing protection implies reducing the amount of sent information per time unit (e.g. the size of sent video frames) and, consequently, the quality of sent information. In other words, the impact on the network load is controlled, information is more protected against channel errors, but the information distortion increases and impacts negatively on the QoE. For this motivation, if this type of solutions are applied, an end-to-end distortion minimization algorithm should be devised, to get a joint source-channel coding approach. For instance, as done in this paper, a proper compression level may be selected consequently to the choice of the protection level. Bursalioglu et al. [3] investigates a joint coding solution at the application layer assuming the traffic generated by Gaussian sources. The contribution of this paper is inspired by the cited literature but, to the best of the authors' knowledge, there is no investigation about real implementations of joint source-channel coding at the application layer. This paper considers video streams acquired by a smartphone. The implemented Android *App*s are aimed at jointly compressing and protecting the video dynamically so to guarantee a good QoE of the received video in case of error prone channels, limiting the offered load to the network. To reach the aim, differently from the aforementioned approaches, we employ a method to prevent exceeding the maximum allowable network throughput and to estimate the packet loss. The benefit of the designed coding has been highlighted through real video transmissions with smartphones over an emulated network, similarly as done in [11].

3 Application layer joint coding (ALJC)

The high level block diagram of ALJC ([2, Chapter 1]) is shown in Fig. 1: X and $\widehat{X}$ represent, respectively, source and reconstructed videos (x_i is the i-th frame of X and $\widehat{x}_j$ is the j-th frame of $\widehat{X}$); X_S and $\widehat{X}_S$ are the source encoder output and the source decoder input; X_C and $\widehat{X}_C$ are the channel encoder output and the channel decoder input, respectively. This general scheme is studied on the basis of the Rate-Distortion theory based on the function $R(D)$, defined as the minimum video transmission information needed to represent a given signal (e.g., an image) with a specific average distortion. $R(D)$ is expressed as in (11) [12].

$$R(D) = \min_{\{P(\widehat{x}_j|x_i)\}\in\Gamma} I(X;\widehat{X}) \tag{1}$$

R and D are the source rate and the average source distortion, respectively. $I(X;\widehat{X})$ represents the average mutual information between X and $\widehat{X}$, which measures how much the knowledge of the received video reduces the uncertainty about the transmitted one. Denoting with $P(\widehat{x}_j|x_i)$ the conditional probability to receive $\widehat{x}_j$, given that x_i has been transmitted, the set Γ can be defined as

$$\Gamma = \left\{P(\widehat{x}_j|x_i)\right\} \textit{such that } D(\left\{P(\widehat{x}_j|x_i)\right\}) \le D^* \tag{2}$$

Being D^* the distortion constraint and $D(\left\{P(\widehat{x}_j|x_i)\right\})$ the distortion, analytically written [12] as:

$$D(\left\{P(\widehat{x}_j|x_i)\right\}) = \sum_{i=0}^{N-1}\sum_{j=0}^{M-1} d(x_i,\widehat{x}_j)P(x_i)P(\widehat{x}_j|x_i) \tag{3}$$

$d(\cdot,\cdot)$ is a weight, $P(x_i)$ the probability that the source transmits x_i, and N and M the sizes of source and reconstruction alphabets, respectively. $R(D)$ is usually employed to determine the minimum channel bandwidth necessary for a given video source given a specific distortion constraint or, alternatively, inverting $R(D)$, to find the theoretical bound on the average distortion given a channel capacity constraint, similarly as done in this paper, even if, here, the constraint is not the channel capacity but by the maximum allowable throughput. Analytically, we approximate $D(R)$ in (4), which is the inverse of (1):

$$D(R) = \min_{\{P(\widehat{x}_j|x_i)\}\in\Phi} D(X;\widehat{X}) \tag{4}$$

$$\Phi = \left\{\left\{P(\widehat{x}_j|x_i)\right\} \textit{ such that } R(\left\{P(\widehat{x}_j|x_i)\right\}) \le R_0\right\} \tag{5}$$

R_0 is the throughput constraint and depends on the network conditions. There are two important points to consider: 1) it is often difficult to have closed-form expressions of $R(D)$ and/or $D(R)$: either numerical algorithms or heuristic approximations (as in the case of this work) have to be employed; 2) R_0 (in (5)), which constraints the video transmission rate, is not the capacity offered by the communication means (i.e., of the physical layer) but is the maximum allowable transmission rate (throughput) assured by the overall protocol stack (i.e., including for each layer, processing time, queuing delay, serving time and, for the physical layer, the employed communication means). In practice, by the application layer, the overall communication system, composed of Transport, Network, Data Link and Physical layers, is supposed to be a *black box* represented by its maximum throughput and its packet loss rate, which will be estimated as described in the following. Being $D(R)$ unknown, we use a measure of the end-to-end distortion to define the ALJC problem. In detail, in a network (i.e., from the sender to the receiver transport layer) affected by error (and consequent packet loss) due to both congestion and noisy transmission channels, the reconstructed video frames at the decoder are different from those at the encoder. End-to-end distortion linked to packet k is defined here as a measure of the mentioned difference:

$$E[D_k] = (1-\varrho_k)E[D_{R,k}] + \varrho_k E[D_{L,k}] \tag{6}$$

where $E[D_{R,k}]$ and $E[D_{L,k}]$ are the expected distortions when the k-th transmitted packet is either correctly received or lost/cancelled, respectively. ϱ_k is the probability to lose the k-th packet. The ALJC problem can be written as in [9] on the basis of the Joint-Source Channel coding problem [2, Chapter 1] expressed in (4) and (5): Eq. (7) generalizes the problem for an arbitrary number of parameters for both source and channel coding. The optimal set of source (s^{opt}) and channel (c^{opt}) coding parameters is computed as:

$$\begin{aligned} &s^{opt},\ c^{opt}: \quad \arg\min_{\{s\in S^{M\times K},\ c\in C^{M\times K}\}} E[D(s,c)] \\ &s.t.\ R(s,c) \le R_0 \end{aligned} \tag{7}$$

$E[D(s,c)]$ is the expected distortion of the overall transmitted video composed of K packets:

$$E[D(s,c)] = \frac{1}{K}\sum_{k=1}^{K} E[D_k(s,c)] \tag{8}$$

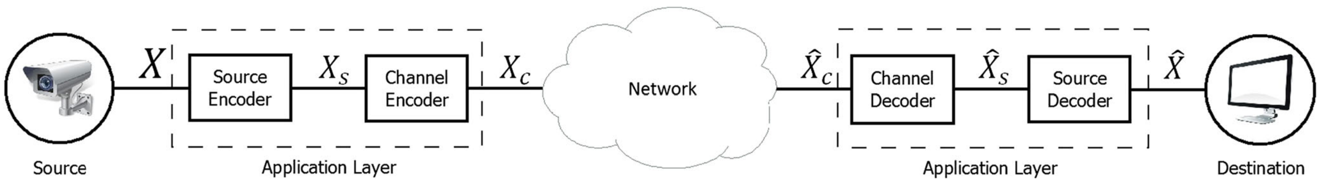

Fig. 1 The general scheme of the joint source-channel coding in [2, Chapter 1] adapted to the ALJC approach

s is the set of source coding parameters and c the set of channel coding parameters, respectively. For each k-th packet composing a video frame, a set of M source and N channel coding parameters is applied.

$$s = \{s_{11}, \ldots, s_{1K}, \ldots s_{m1}, \ldots, s_{mK}, \ldots s_{M1}, \ldots, s_{MK} \in S^{M \times K}\} \quad (9)$$

$$c = \{c_{11}, \ldots, c_{1K}, \ldots c_{n1}, \ldots, c_{nK}, \ldots c_{N1}, \ldots, c_{NK} \in C^{N \times K}\} \quad (10)$$

$R(s, c)$ is the total number of transmitted bits per second (at the application layer) for an encoded video frame. R_0, as said, is the throughput constraint imposed by the network and estimated in this paper. The practical aim of ALJC is to minimize the expected distortion for each frame given the corresponding bit rate constraint. Being an analytical expression of (6) hardly available, this paper proposes a heuristic solution of the problem in (7), as detailed in the remainder of the paper.

4 Implemented *Apps*

4.1 Preliminarily definitions

The implemented applications put into operation video streaming between two distinct smartphones based on the Android OS. We describe the two *Apps* (Transmitter and Receiver), the software architecture, and the related structures in the following.

The chosen source encoder for video frames is MJPEG. From the practical viewpoint, an MJPEG video flow is a series of individual JPEG coded pictures representing the video frames. Concerning channel coding, LDPC [7] has been chosen for its computational feasibility. The resolution for video frames is QCIF (Quarter Common Intermediate Format, 176×144 pixels). The source coder is implemented by Android's API through a Java object to compresses a raw image through JPEG by quality index (decided by the heuristic algorithm proposed in this paper) as an input. The LDPC codec has been taken from an existing implementation [1] by adapting the source code as a library of the Android Native Development Kit (NDK).

The sequence of information processing actions of MJPEG video frames may be described as follows. A single video frame (i.e. a JPEG coded picture) is a **content** that is identified by a unique **content id**. The video stream is composed of a sequence of video frames. As shown in the right part of Fig. 2, which shows also Heuristic-ALJC actions described in Section 5, each video frame is divided into **video packets** (each **video packet** contains, at most, one video frame) also adding a proper header H, described in detail in Section 4.2. Video packets are stored in a **processing buffer** of fixed length (35 packets in this paper). Once the number of **video packets** in the buffer reaches a certain threshold (called channel coding threshold - CCT, dinamically managed by the heuristic algorithm introduced in this paper), the video packets contained in the buffer enter the LDPC coder that generates a number of **redundancy packets** suitable to fill the rest of the buffer. In practice, the threshold CTT decides the amount of packets dedicated to transmit video information and, consequently, the amount of redundancy packets. Both video and redundancy packets have a length of 1024 [byte]. The sequence of video packets and the related **redundancy packets** compose a **codeword** (of 35 packets, as said), identified by a **sequence number**. The stream of packets composing codewords is stored into a **codeword buffer** from where the UDP transport protocol picks up and transmits the packets. A single packet is the transmission unit handled by UDP. A feedback channel allows the receiver to send ***report packets*** back to the transmitter. It is used to obtain information about the channel status.

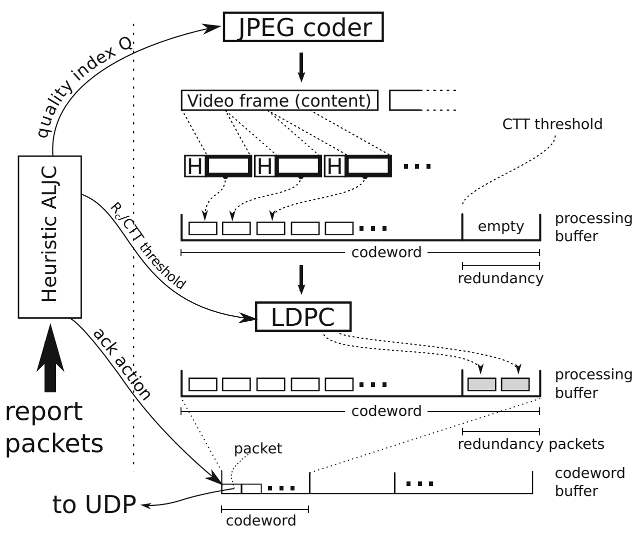

Fig. 2 Heuristic-ALJC actions

4.2 Application layer packet header

A small amount of control data (i.e. a header) in order to allow decoding operations and rebuilding individual *contents* from the transport layer data flow has been added to video packets. It is composed of 24 [byte], six Java integers, and contains the following fields: **FEC**, the number of redundancy packets; **Content ID**, a progressive number that identifies to which *content* (i.e., frame) the payload data belongs to; **Codeword Number**, a progressive number identifying the *codeword* which the *packet* belongs to; **Sequence Number**, a progressive number that individuates the *packet* position within the *codeword*; **Content Size**, which specifies the number of bytes composing the *content*; and **Offset**, measured in bytes, which indicates the distance from the beginning of the *content* (i.e., the JPEG image) where the *packet*'s payload must be written when the *content* is rebuilt.

4.3 Transmitter and receiver *App*

The transmitter *App* has the tasks: to acquire frames from the smartphone camera; to compress them by using JPEG; to perform LDPC-encoding; to queue codewords in the codeword buffer employed to regulate the transmission rate; and to deliver them to the UDP transport protocol. The transmitter app is composed of: ***streamer***, performing data processing and transmission, and ***listener***, managing the feedback information received by report packets. The *listener* enables the adaptive capabilities of the transmission, and exploits the feedback information to compute source-channel coding parameters and to adapt the transmission rate to the maximum allowable throughput, as explained in Section 5.

The receiver *App* has a similar structure: a ***listener*** is bound to a particular UDP port and stores the received packets. LDPC decoder acts when either *i)* the reception of a *codeword* is complete or *ii)* a packet belonging to a more recent *codeword* (i.e., a codeword with a higher *Codeword Number*) unexpectedly arrives. Once the content of the LDPC protected stream has been recovered, JPEG frames are rebuilt and sequentially displayed on screen. Whenever a decoding session is completed, a ***responder*** fills the associated *report packet* and sends it to the transmitter.

5 Heuristic-ALJC

Heuristic ALJC method proposed in this paper is aimed at solving heuristically the problem formally defined in (7) and represents the algorithmic core of the implemented Transmitter *App*. The constraint R_0 and the packet loss probability ϱ_k are usually unknown *a priori* and need to be determined. Our heuristic ALJC solution is based on three phases: *i)* transmission rate adaptation through the employment of the *report packets* at the application layer; *ii)* selection of the channel coding parameters; *iii)* selection of the source coding parameters.

Each *report packet* carries information about the number of lost packets for each codeword and is sent each time a codeword is received. In this way, the transmitter is aware of how fast the mobile network can deliver the video, i.e., the transmitter derives an estimation of the maximum network throughput currently available, and of how vulnerable to losses is the sent video in the process of traversing the entire network. Concerning transmission rate adaptation, the regulation is acted on the basis of the *report packet* reception that enables the transmission of further codewords. Once the report packet for a given codeword is received, the corresponding codeword is acknowledged and removed from the codeword buffer. In case report packets are missing or delayed, the consequence is that the codeword buffer may saturate. In this case the transmission of codeword packets stops until a new report packet arrives so avoiding losing packets in the codeword buffer but affecting the average transmission rate. The rationale on the basis of this rate adaptation scheme is that, assuming the return channel reliable, the missing/delayed reception of report packets is interpreted as errored/narrowband forward channel. In the case the transmission rate adapter should not take any action, the transmission rate is limited to one codeword each 10 [ms]. Being a codeword composed of W [byte] (35 packets of 1024 + 24 [byte], for payload and header respectively, the maximum possible transmission rate is limited by the ratio $\frac{W}{10}$ [byte/ms]. Concerning the selection of source and channel parameters, referring to the formal definitions of the ALJC problem in Section 3, a single parameter is employed in the implemented *App*s for both source (s_{1k}, $\forall k \in [1, K]$, $M = 1$) and channel coding (c_{1k}, $\forall k \in [1, K]$, $N = 1$). In the following, the mentioned parameters will be denoted as $s_{1k} = Q$ and $c_{1k} = R_c$. R_c is the ratio between the overall number of video packets (i.e. the CTT threshold) and the fixed codeword length W (R_c =CTT threshold / W). R_c is computed by Heuristic-ALJC as specified in Section 5.1 and passed to the LDPC channel coder. Established the CTT threshold value and estimated through the arrival frequency of report packets the maximum network throughput currently available, Heuristic-ALJC choses the best value of the JPEG coder quality index Q as specified in Section 5.2 and passes it to the JPEG coder. The main actions performed by ALJC are evidenced in the left part of Fig. 2.

5.1 Channel coding

The first step in the joint channel/source decision is to assess the packet loss by means of a *report packet*. With respect to the notation introduced in Section 3, this means estimating ϱ_k, which the Heuristic-ALJC uses as input to decide the most appropriate amount of protection.

Deriving an analytical formulation of the loss probability represents a non-trivial task. For this reason, we have measured the average video packet loss as a function of the total amount of lost packets in a codeword (including redundancy packets) and of the code rate R_c. To do this for a given R_c and number of lost packets within a codeword (denoted with P_l), we have filled the CTT threshold $= R_c \times W$ video packets of each codeword with random data, and we have LDPC-encoded them so generating [$W-$CTT threshold] redundancy packets. Then, P_l packets chosen randomly (from an uniform distribution) have been cancelled. After these actions, we have performed LDPC decoding and compared the reconstructed data vector with the original one, thus obtaining the video packet loss. This process has been iterated one hundred times for each (R_c, P_l) combination, so providing an empirical computation of the average loss of video packets $L_v(R_c, P_l)$ versus the: number of lost packets in a codeword (P_l) and code rate (R_c). The goal of a channel coder is the minimization of the decoded packet loss probability, which has a monotonically decreasing behaviour with respect to R_c. Indeed, the loss curve approaches zero as the code rate R_c decreases, so the decision would always fall at the end of the range, which assures the maximum protection. To avoid, if possible, such channel coder behaviour, we have defined a cost function $C(R_c, P_l)$, in (11), composed of $L_v(R_c, P_l)$ and of a coefficient inversely proportional to the amount of video packets $R_c \times W$ carried by the codeword. The obtained function is convex and its minimum over R_c, identified as $R_c(P_l)$ and shown in (12), guarantees the best decision for each (R_c, P_l) combination. $C(R_c, P_l)$ values versus the: number of lost packet in a codeword P_l and code rate R_c are shown in Fig. 3, where the minimum $R_c(P_l)$ is evidenced through the dotted line.

$$C(R_c, P_l) = L_v(R_c, P_l) + \frac{1}{R_c \times W} \tag{11}$$

The minimum operation in (12) is computed offline only once and provides a decision rule for what concerns the channel coder when a specific value of P_l has been recovered from the feedback information contained in the report packet of a given codeword.

$$R_c(P_l) = \arg\min_{R_c} C(R_c, P_l)\ \forall P_l \in [0, W] \tag{12}$$

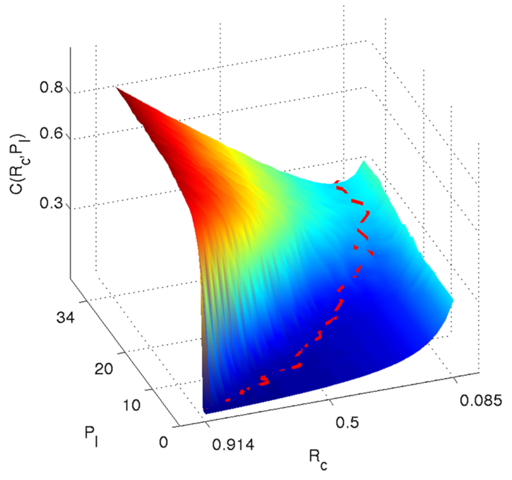

Fig. 3 $C(R_c, P_l)$ vs the number of lost packets in a codeword (P_l) and code rate (R_c). The minimum of $C(R_c, P_l)$ shown by the dotted line is the optimal choice

Due to implementation reasons, used R_c values are limited to the range $[\frac{4}{35}, \frac{30}{35}]$, identifying a maximum and minimum protection configuration, respectively.

5.2 Source coding

Once the channel coder sets the number of video packets [CTT threshold] within a codeword, and estimated the available maximum network throughput, i.e. the constraint R_0 in (7), the maximum transmission rate available for information video packets is fixed. More technically: being the constraint R_0 in [byte/s] known, the maximum number of bytes which may be dedicated to video information for each codeword is limited by the quantity $\frac{R_0}{W} \times R_C$ [byte]. The quality index denoted with Q is used by JPEG and is an integer ranging from 0 (worst quality, smallest image size) to 100 (best quality, biggest image size). In general, the relation $F_s(Q)$ between Q and the output frame size (assumed here as an inverse measure of the distortion) denoted with F_s is not deterministic, due to the DCT-based approach of the JPEG compression that tends to compress more efficiently frames with weak high frequency components. Nevertheless, for the sake of this paper it is important to know, at least statistically, what is the expected frame size for a given quality index: $E[F_s(Q)]$. For this reason $E[F_s(Q)]$ has been derived heuristically by averaging the size of $N = 100$ frames, making sure not to acquire blank scenes, for each $Q \in [0, 100]$. Denoting with $F_s^n(Q)$ the size of the n-th frame, where $n \in [1, N]$, $E[F_s(Q)]$ is computed as in (13).

$$E[F_s(Q)] = \frac{1}{N} \sum_{n=1}^{N} F_s^n(Q) \tag{13}$$

$E[F_s(Q)]$ is shown in Fig. 4(a). The function $\frac{1}{E[F_s(Q)]}$ is assumed in this paper as a measure of the average distortion (in (8)).

To obtain the Q index starting from the frame size, we have inverted numerically $E[F_s(Q)]$ and we have empirically approximated it by the negative exponential function in Fig. 3(b), so that $0 \le Q < 100$ (14).

$$Q = \begin{cases} 100(1 - e^{\frac{100-E[F_s]}{3000}}) \ , & E[F_s] \ge 100 \text{ [byte]} \\ 0 & , \ 0 < E[F_s] < 100 \text{ [byte]} \end{cases} \tag{14}$$

Given the information in Fig. 4(b), Heuristic-ALJC choses Q as follows: the maximum number of bytes dedicated to video packets is known $(\frac{R_0}{W} \times R_C)$; fixing also the video frame rate (to 10 frames per second in this paper) so to assure a given fluidity of the video, the frame size is also fixed to $\frac{R_0 \times R_c}{W \times 10}$ [byte]. The obtained frame size value is used as $E[F_s(Q)]$ value in (14) to get the Q value. Operatively the minimum utilized Q value is 20 because values below it provide unacceptable quality images.

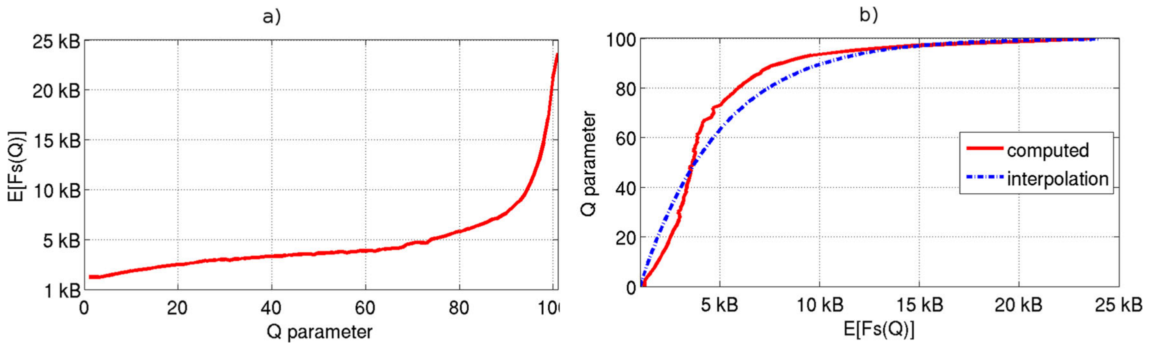

Fig. 4 **a** $E[F_s(Q)]$ as a function of the input quality factor parameter Q, **b** Q values as a function of $E[F_s(Q)]$: computed numerical values shown through the continuous line, interpolated analytical approximation through the dotted line

6 Performance investigation

6.1 Testbed

We have implemented a testbed to emulate the reference scenario described in the introduction. Two separate Android devices, implementing the transmitter and receiver Apps, communicate through a WiFi local network connected to a machine that emulates the effect of a mobile network. On the receiving side, another WiFi network is used to interconnect the second device. The emulation machine is a regular PC running a Linux-based operating system, and implementing the *netem* tool to manage the outgoing traffic of each WIFI interface by tuning available channel bandwidth, packet loss, bit error rate (BER), and delay (fixed to 100 [mS] in all shown test).

6.2 Scenarios and performance metrics

Table 1 contains bandwidth and BER values for each emulated scenario.

In order to evaluate the performance, we have compared Heuristic-ALJC, implemented through the two designed Apps, with two opposite static policies assuring minimum

Table 1 Test scenarios

	Bandwidth	BER
A	400 Kbps	0 %
B	400 Kbps	10 %
C	400 Kbps	35 %
D	180 Kbps	0 %
E	180 Kbps	10 %
F	180 Kbps	35 %
G	400 Kbps→180 Kbps	0 %
H	400 Kbps	0 %→35 %

protection/maximum quality (R_c=30/35, Q=100), and maximum protection/minimum quality (R_c=4/35, Q=20). The first group of tests evaluates Heuristic-ALJC behaviour during three minutes long sessions, for static channel conditions. A second group of tests investigates the system adaptation capabilities over time by varying network conditions. In order to measure the quality of individual frames of the MJPEG sequence, we utilize the Structural SIMilarity ($SSIM$) index, introduced in [15]. $SSIM(f_i, \widehat{f_i})$ provides a quality measure of one of the frames $(\widehat{f_i})$ supposed the other frame (f_i) of perfect quality. *SSIM* represents a good choice since it follows the Mean Opinion Score - MOS more closely than other indexes such as the Peak Signal to Noise Ratio (PSNR) and the Mean Square Error (MSE). *SSIM* is computed over small portions of a frame, and the whole frame index $SSIM(f_i, \widehat{f_i})$ is obtained by averaging the individual portion values. $SSIM'$ value for a single frame portion is given by (15).

$$SSIM'(x, y) = \frac{(2\mu_x\mu_y + C_1) + (2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{15}$$

x and y are the two small image blocks, μ_i is the i-th block pixel value average, σ_i is the i-th block pixel standard deviation, and σ_{ij} is the quantity in (16),

$$\sigma_{ij} = \frac{1}{U-1}\sum_{k=1}^{V}(i_k - \mu_i)(j_k - \mu_j) \tag{16}$$

where U is the number of pixels contained in the portion and V is the overall number of portions.

SSIM index ranges from 0 (completely uncorrelated frames) to 1 (identical frames) and can be considered as a degradation factor. In order to evaluate the performance we have devised a performance index with the following requirements. It must reward high quality frames, a fluent video stream, and penalize corrupted or lost frames. Index I in (17) satisfies such requirements

$$I = \frac{\sum_{i=1}^{U} SSIM(f_i, \widehat{f_i}) \cdot f_{received}^{TOT}}{T_{sim}} \tag{17}$$

and can be interpreted as a *quality-weighted average frame rate*.

6.3 Performance results

6.3.1 Static channel scenarios

In this Section we show how our Heuristic-ALJC behaves when channel characteristics do not vary over time. Figure 5 shows the values of: Index I (a); average SSIM over the entire test (b); number of delivered good (decodable) frames (c); and number of lost/corrupted frames (d), for Heuristic-ALJC, Minimum and Maximum Protection schemes, for scenarios from A to F. The Maximum Protection scheme assures no loss (d) in all scenarios, even in 10 % BER (B and E) and 35 % BER (C and F) scenarios, but it dedicates so many packets to redundancy that the transmission rate of video frames is too reduced. This implies a limited number of delivered frames (c). Index I is low for any scenario. The Minimum Protection scheme behaviour may be satisfying for no loss scenarios, even if the large Q value imposed implies large frame size and consequent limited number of delivered frames (c), but it is highly inefficient for loss scenarios, where the large number of corrupted frames (d) heavily affects the quality (b) and consequently, Index I value (a). Heuristic-ALJC, by estimating the network available throughput over time, by tuning the protection level and adapting the source coding, always outperforms static solutions concerning Index I. It assures the highest number of successfully delivered frames (c) for all scenarios, and keeps the number of lost/corrupted frames low enough so not to affect the quality (b).

6.3.2 Dynamic channel scenarios

In this Section we evaluate the system's ability to adapt the ALJC parameters (i.e., Q and R_c) to network conditions that change over time.

Figure 6 shows the values (averaged over a period of 5 [s]) of: Index I (a); SSIM (b); Frame Rate (c); Number of Lost/Corrupted Frames (d); for Heuristic-ALJC, Minimum and Maximum Protection schemes in case of a sudden narrowing of the available channel bandwidth from 400 to 180 [Kbps] (scenario G). Bandwidth changes at time 60 [s]. No loss is imposed. Dotted lines show the average values over the entire time periods before and after bandwidth variation. Heuristic-ALJC can keep the frame rate (c) much higher than the two static schemes after bandwidth

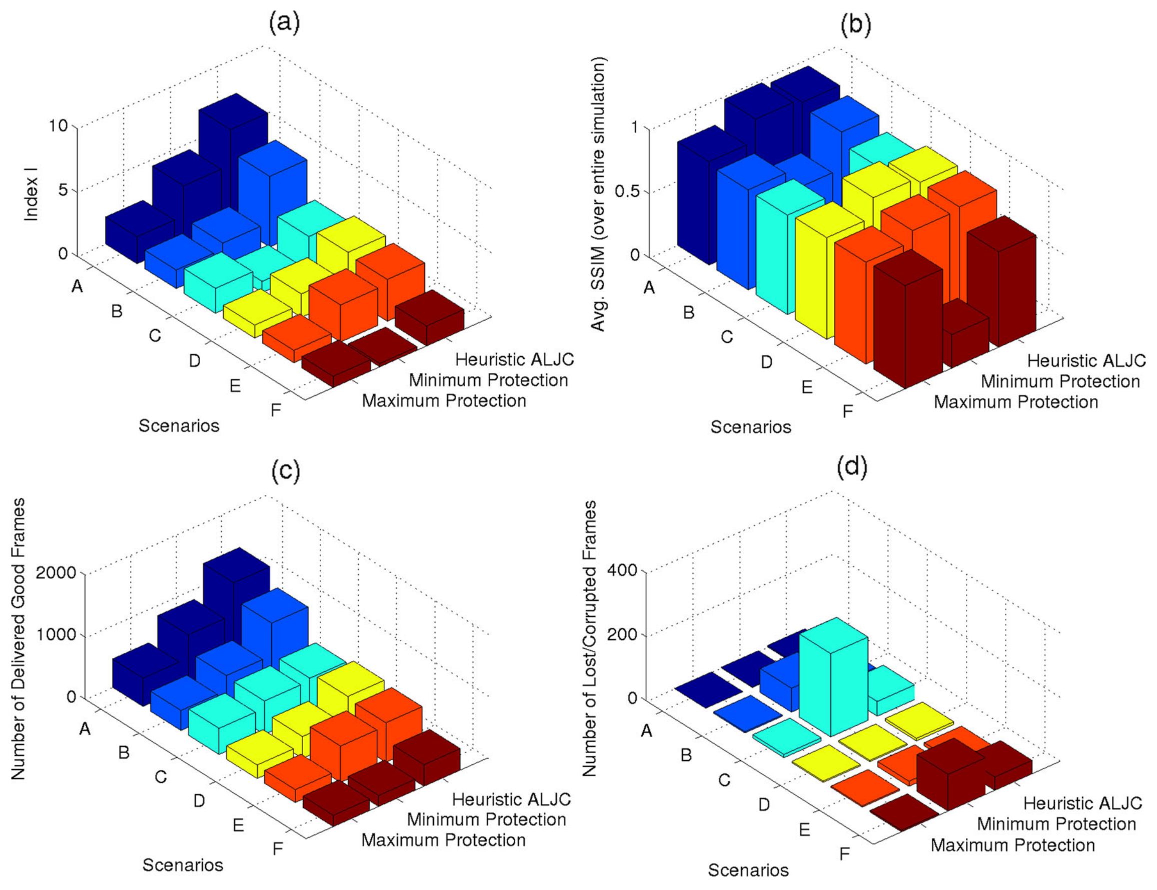

Fig. 5 Simulation of static channel behaviour

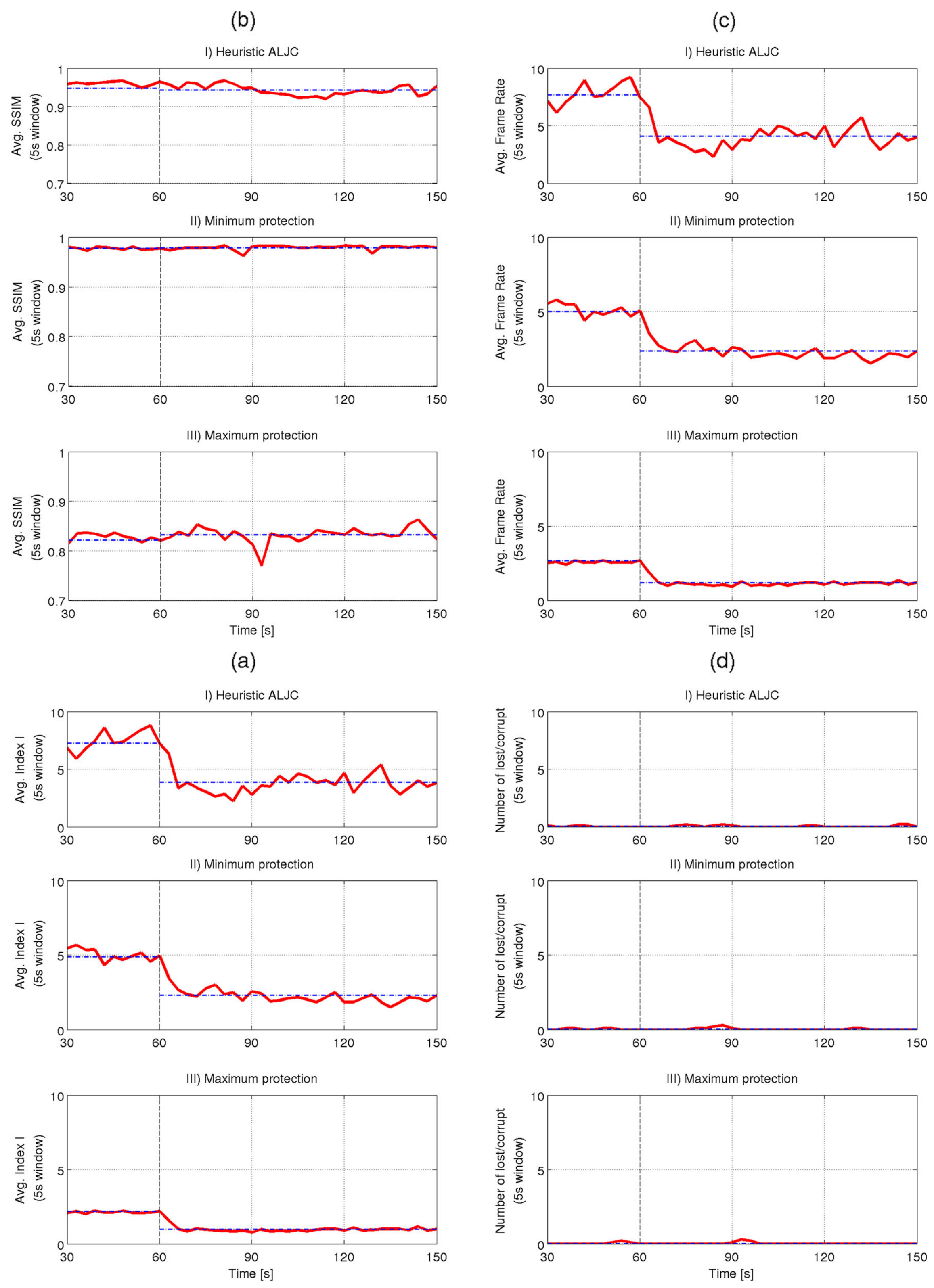

Fig. 6 Simulation of a sudden bandwidth drop

restriction and the SSIM value (b) almost constant over the entire test. The overall effect is that the Index I (a) keeps a satisfying level also after bandwidth narrowing. Loss (d) has no effect in this test. Figure 7 shows the same quantities of Fig. 6 by using scenario H: 400 [kpbs] bandwidth and BER variation from 0 % to 35 % at time 60 [s]. Dotted lines

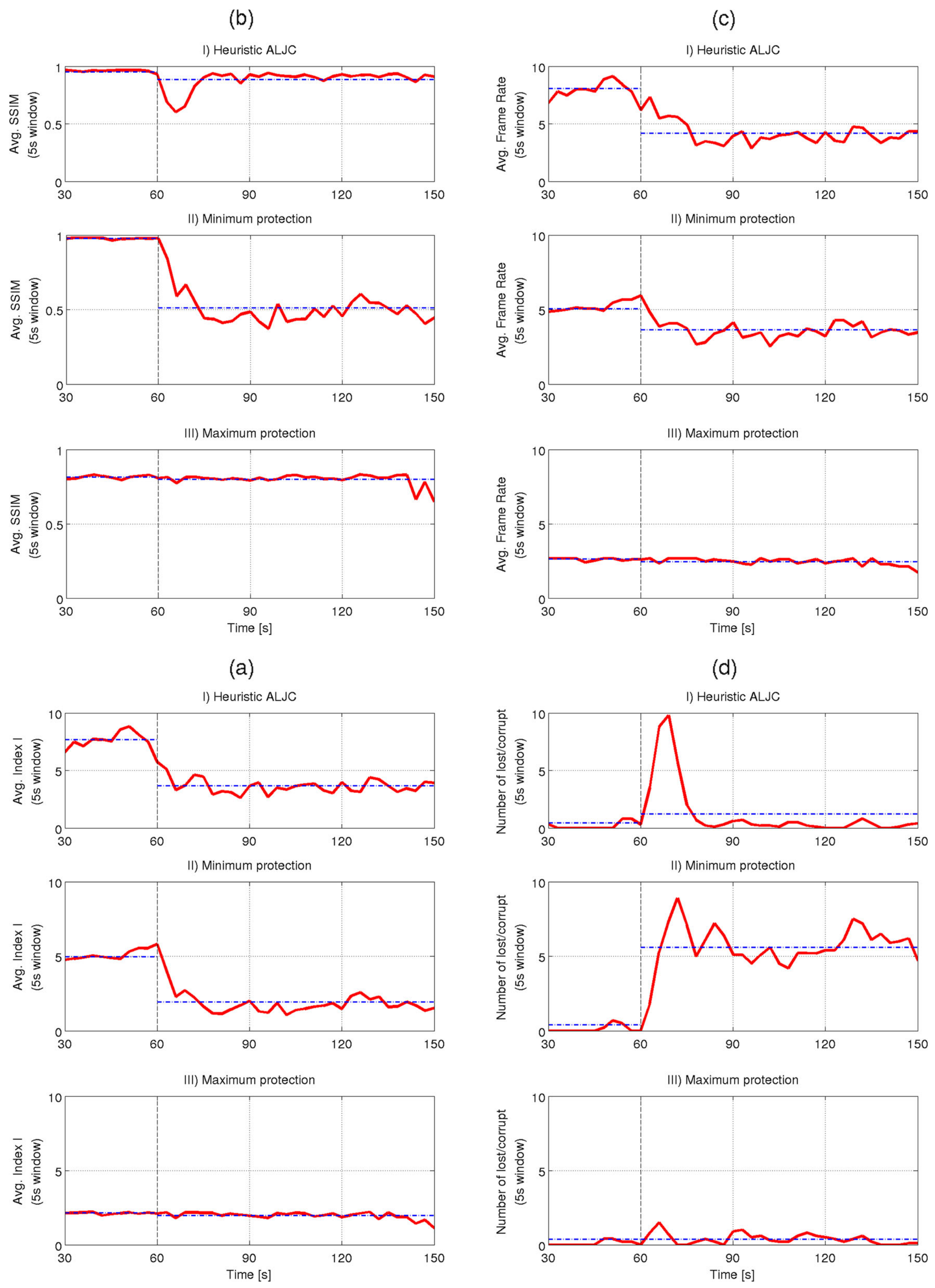

Fig. 7 Simulation of a sudden BER increase

show the average values over the entire time period before and after BER variation. Heuristic-ALJC quickly reacts to BER change. The reaction effect is very clear concerning Loss (d), which drastically increases after time 60 [s], but is quickly reduced by acting dynamically on the LDPC coder. The action is very efficient because the average loss value is almost the same before and after BER change. The effect of Heuristic-ALJC is clear also concerning SSIM (b), which is temporarily reduced during the loss period, but which, on average, is practically kept constant independently of the channel condition variation. Frame rate behaviour is shown in (c) and the consequent I values, kept quite high also after BER change, are shown in (a).

7 Conclusions

In this paper we have presented Heuristic-ALJC to transmit video streams on networks characterized by time varying and possibly lossy channels. From the practical viewpoint, Heuristic-ALJC adaptively applies both video compression and encoding to protect video streams at the application layer on the basis of a feedback about the overall network conditions, measured in terms of both maximum allowable network throughput and link quality (packet cancellations). The performance investigation, carried out through the real implementation of the Heuristic-ALJC over Android smartphones, shows that Heuristic-ALJC adapts the video transmission to network conditions so allowing an efficient resource exploitation and satisfactory performance and outperforming static coding under all tested network conditions.

References

1. Planete-bcast, inria, ldpc codes download page. http://planete-bcast.inrialpes.fr/article.php3?id_article=16
2. Bovik AC (2005) Handbook of Image and Video Processing (Communications, Networking and Multimedia). Academic, Orlando
3. Bursalioglu O, Fresia M, Caire G, Poor H (2009) Joint source-channel coding at the application layer. In: Data Compression Conference, 2009. DCC ’09., pp. 93–102. doi:10.1109/DCC.2009.10
4. Caldeira J, Rodrigues J, Lorenz P (2012) Toward ubiquitous mobility solutions for body sensor networks on healthcare. IEEE Commun Mag 50(5):108–115. doi:10.1109/MCOM.2012.6194390
5. Choi Y, Momcilovic P (2011) On effectiveness of application-layer coding. IEEE Trans Inf Theory 57(10):6673–6691. doi:10.1109/TIT.2011.2165130
6. Fouda MM, Nishiyama H, Miura R, Kato N On efficient traffic distribution for disaster area communication using wireless mesh networks. Springer Wireless Personal Communications (WPC) (2013, to appear)
7. Gallager R (1962) Low-density parity-check codes. IRE Trans Inf Theory 8(1):21–28
8. Karimi O, Liu J, Wang C (2012) Seamless wireless connectivity for multimedia services in high speed trains. IEEE J Sel Areas Commun 30(4):729–739. doi:10.1109/JSAC.2012.120507
9. Katsaggelos A, Eisenberg Y, Zhai F, Berry R, Pappas T (2005) Advances in efficient resource allocation for packet-based real-time video transmission. Proc IEEE 93(1):135–147. doi:10.1109/JPROC.2004.839621
10. Lloret J, Ghafoor KZ, Rawat DB, Xia F (2013) Advances on network protocols and algorithms for vehicular ad hoc networks. Mob Netw Appl 18(6):749–754. doi:10.1007/s11036-013-0490-7
11. Martini M, Mazzotti M, Lamy-Bergot C, Huusko J, Amon P (2007) Content adaptive network aware joint optimization of wireless video transmission. IEEE Commun Mag 45(1):84–90. doi:10.1109/MCOM.2007.284542
12. Shannon CE (1959) Coding theorems for a discrete source with a fidelity criterion. Inst Radio Eng Int Conv Rec 7(part 4):142–163
13. Silva BM, Rodrigues JJ, Lopes IM, Machado TM, Zhou L (2013) A novel cooperation strategy for mobile health applications. IEEE J Sel Areas Commun 31(9):28–36. doi:10.1109/JSAC.2013.SUP.0513003
14. Suri N, Benincasa G, Tortonesi M, Stefanelli C, Kovach J, Winkler R, Kohler R, Hanna J, Pochet L, Watson S (2010) Peer-to-peer communications for tactical environments: Observations, requirements, and experiences. IEEE Commun Mag 48(10):60–69. doi:10.1109/MCOM.2010.5594678
15. Wang Z, Bovik A, Sheikh H, Simoncelli E (2004) Image quality assessment: from error visibility to structural similarity. IEEE Trans Image Process 13(4):600–612. doi:10.1109/TIP.2003.819861
16. Zhou L, Geller B, Zheng B, Wei A, Cui J (2009) System scheduling for multi-description video streaming over wireless multi-hop networks. IEEE Trans Broadcast 55(4):731–741. doi:10.1109/TBC.2009.2032795
17. Zhou L, Yang Z, Rodrigues J, Guizani M (2013) Exploring blind online scheduling for mobile cloud multimedia services. IEEE Wirel Commun 20(3):54–61. doi:10.1109/MWC.2013.6549283
18. Zhou L, Yang Z, Wen Y, Rodrigues J (2013) Distributed wireless video scheduling with delayed control information. doi:10.1109/TCSVT.2013.2291311