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

Igor Bisio(✉), Fabio Lavagetto, Giulio Luzzati, and Mario Marchese

DITEN, University of Genoa, Genoa, Italy
{igor.bisio,fabio.lavagetto,mario.marchese}@unige.it
giulio.luzzati@edu.unige.it

Abstract. This paper presents the Heuristic Application Layer Joint Coding (Heuristic-ALJC) scheme 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 performance investigation, carried out through a real implementation of the *App*s over Android smartphones, compares Heuristic-ALJC with static schemes.

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 [1].

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. 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

© ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2016
I. Bisio (Ed.): PSATS 2014, LNICST 148, pp. 123–131, 2016.
DOI: 10.1007/978-3-319-47081-8_12

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 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. Established 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 functional 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 framework behind this work has been preliminarily presented in [2] and described in detail in [3].

2 State of the Art and Aim of the Paper

[4] 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 [4], 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 [5, Chap. 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. [6] 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 [7].

3 Implemented ***App*s**

3.1 Preliminarily Definitions

The implemented applications put into operation video streaming between two distinct smartphones based on the Android OS. We describe the two *App*s (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 [8] 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 [9] 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. 1, which shows also Heuristic-ALJC actions described in Sect. 4, 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 Subsect. 3.2. 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.

3.2 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.

3.3 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 Sect. 4.

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.

4 Heuristic-ALJC

Heuristic ALJC method proposed in this paper is aimed at solving heuristically the problem formally defined in literature 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, a single parameter is employed in the implemented *App*s for both source and channel coding. 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 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 and passes it to the JPEG coder. The main actions performed by ALJC are evidenced in the left part of Fig. 1.

5 Performance Investigation

5.1 Testbed

We have implemented a testbed to emulate the reference scenario described in the introduction. Two separate Android devices, implementing the transmitter

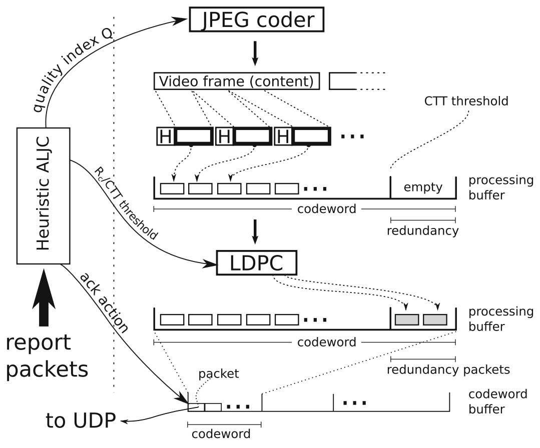

Fig. 1. Heuristic-ALJC actions

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).

5.2 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 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,

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 %

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 [10]. $SSIM(f_i, \widehat{f}_i)$ 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* 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 (1) satisfies such requirements

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

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

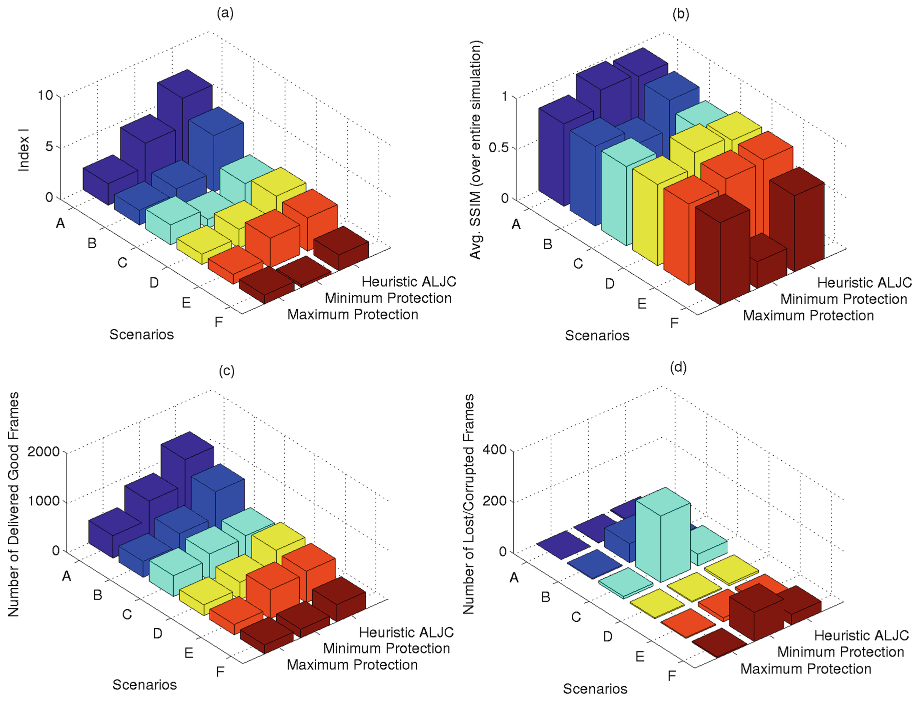

Fig. 2. Simulation of static channel behaviour

5.3 Performance Results

(1) Static Channel Scenarios: In this Section we show how our Heuristic-ALJC behaves when channel characteristics do not vary over time. Figure 2 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 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. Fouda, M.M., Nishiyama, H., Miura, R., Kato, N.: On efficient traffic distribution for disaster area communication using wireless mesh networks. Springer Wirel. Personal Commun. (WPC) **74**, 1311–1327 (2014)
2. Bisio, I., Grattarola, A., Lavagetto, F., Luzzati, G., Marchese, M.: Performance evaluation of application layer joint coding for video transmission with smartphones over terrestrial/satellite emergency networks. In: 2014 International Conference on Communications (2014, to appear)

3. Bisio, I., Lavagetto, F., Luzzati, G., Marchese, M.: Smartphones apps implementing a heuristic joint coding for video transmissions over mobile networks. Mob. Netw. Appl. **19**, 552–562 (2014)
4. Choi, Y., Momcilovic, P.: On effectiveness of application-layer coding. IEEE Trans. Inf. Theory **57**(10), 6673–6691 (2011)
5. Bovik, A.C.: Handbook of Image and Video Processing (Communications, Networking and Multimedia). Academic Press Inc., Orlando (2005)
6. Bursalioglu, O., Fresia, M., Caire, G., Poor, H.: Joint source-channel coding at the application layer. In: Data Compression Conference, 2009, DCC 2009, pp. 93–102, March 2009
7. Martini, M., Mazzotti, M., Lamy-Bergot, C., Huusko, J., Amon, P.: Content adaptive network aware joint optimization of wireless video transmission. IEEE Commun. Mag. **45**(1), 84–90 (2007)
8. Gallager, R.: Low-density parity-check codes. IRE Trans. Inf. Theory **8**(1), 21–28 (1962)
9. Planete-bcast, inria, ldpc codes download page. http://planete-bcast.inrialpes.fr/article.php3?id_article=16
10. Wang, Z., Bovik, A., Sheikh, H., Simoncelli, E.: Image quality assessment: from error visibility to structural similarity. IEEE Trans. Image Process. **13**(4), 600–612 (2004)