Performance Evaluation of Application Layer Joint Coding for Video Transmission with Smartphones over Terrestrial/Satellite Emergency Networks

Igor Bisio, *Member, IEEE,* Aldo Grattarola, Fabio Lavagetto, Giulio Luzzati *Student Member, IEEE*, and Mario Marchese, *Senior Member, IEEE*

***Abstract*—The deployment of terrestrial/satellite networks, plays a crucial role for for risk and emergency management. In this context, efficient solutions for heterogeneous and mobile networks, including satellite portions that allows wide coverages, represents a key issue. In the mentioned scenario, transmitting video with portable devices (such as smartphone), over terrestrial/satellite networks, to a Remote Monitoring Host (RMH) may support emergency and rescue operations after crisis situations. Unfortunately heterogeneity often implies impairments such as packet losses, due to errors and congestion, which negatively affect the video quality. We present an application layer joint coding algorithm for video transmission, that adaptively applies video compression and channel coding at the application layer, on the basis of the overall network condition estimated in terms of maximum allowable throughput of the network and quality (packet cancellations or *lossiness*). A deep performance investigation, carried out with real implementation of the algorithm, compares the joint coding against fixed schemes and shows that the joint approach adapt the video transmission to terrestrial/satellite emergency networks so allowing a more efficient resource exploitation.**

I. Introduction

The nature of the modern Internet is heterogeneous. It can be defined as an infrastructure formed by multiple sub-networks composed of different transmission media, such as fibre optic cable, coaxial cable, satellites, radio, and copper wire. Such scenarios need significant effort in the fields of the design of reliable and reconfigurable transmission systems, open source software, interoperability and scalability [1].
The mentioned scenario constitutes the reference for this paper: the considered network is composed of heterogeneous portions, in particular, radio and satellite and mobile devices, Smartphones in the case of this work, are employed to acquire and transmit video streams through dedicated *App*s easy to be downloaded and installed. An applicative example of the considered scenario (depicted in Fig.1) may concern future safety support services: after a critical event (i.e., a road accident, fire), first responders (a rescue team or just a person on site) can shoot video and send it to an experienced operator (or to a center) to manage rescue operations more consciously. Such heterogeneous wireless networks may be exploited also in fields such as: tele-learning [2]; electronic help for elderly people and tele-medicine [3]; applications for vehicles [4] trains [5] and planes; and tactical communications [6]. The mentioned environments can be mentioned as *networks for social support*.

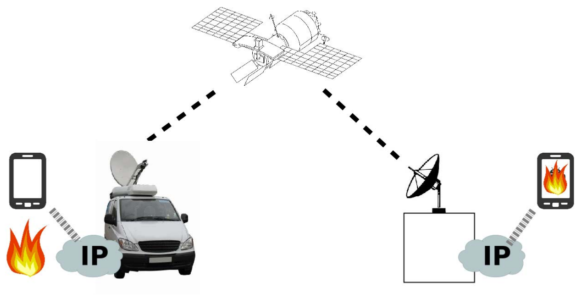

Fig. 1. The reference scenario

Differently from wired channels where loss is mainly imputable to network or resource congestion, the mentioned media may be affected by low Signal-to-Noise Ratios (SNRs), leading to high bit error rates and consequent packet losses (cancellations), ranging from negligible to almost completely impairing. Moreover, the time invariance assumption no longer holds: typical wireless channels may exhibit extremely quick dynamics, due to many factors, such as multipath fading, shadowing, radio interferences, weather conditions. For this reason, a static video compression and protection is a non optimal choice, whereas the flow must be dinamically adapted, jointly considering the impact these tunings have on each other and on the whole system performance.

Coding at the application layer guarantees flexibility and easy-reconfigurability, in addition to the recovering capabilities of the code itself [7]. [8] defines the space of all communications systems and formally demonstrates the existence of two sub-spaces called performance regions. In a region the employment of application layer coding is significantly advantageous while it is detrimental in the second. The first performance region is represented by systems that experience low level of channel errors and, as a consequence, low packet loss probability. On the contrary, the systems with high channel error levels represent the second region in which applying the necessary redundancy causes congestion. In practice, the mentioned coding approach may improve the performance only in systems with low packet loss probability due to channel errors because error prone channels require high levels of redundancy thus causing packet losses due to congestion.
A solution to this limit, suited to video transmissions, is pro-

posed in [9, Chapter 1]: increasing protection does not result in an increased offered load because the packet transmission rate is kept constant. For this reason an end-to-end distortion minimization algorithm is devised. The method represents a joint source-channel coding approach. More generally, [10] investigates a joint coding solution at the application layer assuming the traffic to be generated by Gaussian source. This reference individuates the heterogeneous mobile networks as the applicative scenario of the joint coding for video transmissions. It is the same scenario considered in this paper in which the benefit of the coding has been highlighted through real video transmissions with Smartphones over an terrestrial/satellite emergency networks.
Finally, [11] takes into account a joint coding approach driven by a multi-attribute decision making control aimed at optimizing several performance metrics by selecting the coding parameters, its performance are tested through simulations.

A practically implemented solution has been presented in [12] but it is based on a cross-layer approach (i.e., interaction with the physical layer is needed) and it is not suited to be applied when the video source is a smartphone platform.

The contribution of the presented paper is inspired by the cited literature but, to the best of the authors' knowledge, there is no investigation upon real implementations of a joint source-channel coding at the application layer. This paper considers video streams acquired by a smartphone and is aimed at compressing and protecting the video so to guarantee a good perceived quality in case of error prone channels and, simultaneously, to limit the offered load to the network. Differently from the aforementioned approaches, our work employs a method to prevent exceeding the maximum allowable throughput and to estimate the packet loss. The crucial feature of the implemented *App*s is that the overall joint coding is entirely realized at the application layer considering the underlying layers and the overall network as a *black box*.

In practice, the *App*s can be deployed on top of existing infrastructures without any need for rewiring or replacing hardware/software components. The remainder of the paper is organized as follows: in Section II are described the proposed Heuristic Application Layer Joint Coding (ALJC) approach, the simulative scenario developed to validate the proposed algorithm and the obtained numerical results are discussed in Section III. Finally the conclusions are drawn in Section IV.

II. The Proposed Heuristic ALJC Approach

A. Preliminaries

Starting from the traditional source-channel coding approach, the proposed heuristic ALJC is aimed at minimizing the well-known distortion-rate function $D(R)$, defined, for example in [9]. In more detail, a measure of the mentioned quantity is given by the end-to-end distortion

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

where $E[D_{R,k}]$ and $E[D_{L,k}]$ are the expected distortion when the k-th transmitted packet is either correctly receiver or cancelled. ϱ_k is the probability to lose the k-th packet.
Starting from the definitions above described, the ALJC problem can be written similarly to the traditional Joint-Source Channel coding problem [13]. It is worth noticing that in the equation below the problem has been generalized for an arbitrary number of parameters for each kind of coding (i.e., source and channel coding). Analytically:

$$\begin{aligned} & s^{opt},\ c^{opt}: \underset{\{s\in S^{M\times K},\ c\in C^{M\times K}\}}{\arg\min} E[D(s,c)] \\ & s.t.\ R(s,c) \leq R_0 \end{aligned} \tag{2}$$

where $E[D(s,c)]$ is considered in this paper as the expected distortion of the overall transmitted video composed by K packets:

$$E[D(s,c)] = \frac{1}{K}\sum_{k=1}^{K} E[D_k(s,c)] \tag{3}$$

while s is the set of source coding parameters and c the set of Channel coding parameters, respectively

$$\begin{aligned} s = \{ & s_{11},\ldots,s_{1K},\ldots s_{m1},\ldots,s_{mK},\ldots \\ & s_{M1},\ldots,s_{MK} \in S^{M\times K}\} \end{aligned} \tag{4}$$

and

$$\begin{aligned} c = \{ & c_{11},\ldots,c_{1K},\ldots c_{m1},\ldots,c_{mK},\ldots \\ & c_{M1},\ldots,c_{MK} \in C^{N\times K}\} \end{aligned} \tag{5}$$

In practice, for each k-th packet composing a video frame, a set of M (N) source (channel) coding parameters is applied. In other words, the aim is to minimize the expected distortion for a frame given the corresponding bit rate constrain. In fact, $R(s,c)$ is the total number of bit per second transmitted (from the application layer) for a source-channel encoded video frame while R_0 is the throughput constraint imposed by the network (including the protocol layers below the application). It is worth noticing that, even if the above description consider the "packet" as the transmitted unit, in the heuristic method below described, the transmission unit is the "codeword" that will be explicitly defined in the following Section.

B. Definitions

The proposed ALJC has been implemented by means of two distinct Android Apps, a transmitter and a receiver. The transmitter acquires video image frames through the on-board camera, and the chosen source encoding scheme is the MJPEG, i.e. each individual frame is compressed by a JPEG encoder. The rationale behind the MJPEG encoding lies in its easiness of implementation. For what concerns the channel coding, our choice fell on a packet level LDPC: we employed a already available c++ library, obtained from [14].

For the sake of clarity we define the terms that will be used in this Section to address specific parts of the *App*s.

- a packet level LDPC is used to protects application layer ***video packets***, by generating ***redundancy packets*** all the packes have a size of 1024 bytes;
- each ***video packet*** can be individuated within a ***codeword***, which is the transmission unit (***video packets*** plus ***redundancy packets***), by means of a

- ***sequence number***, and it is assumed that a *video packet* will carry **at most** one video frame;
- there is a feedback channel, which allows the receiver to send ***report packets*** back to the transmitter. It is used to infer the channel status.

C. The Proposed Heuristic Method

The method proposed in this paper is aimed at solving heuristically the problem formally defined in II, and represents the algorithmic core of the implemented Transmitter *App*. The approach has been designed starting from the following assumptions:

1) an analytical formulation of the expected end-to-end distortion is not simply available;
2) the constraint R_0 is usually unknown *a priori*;
3) the packet (codeword) loss probability ϱ_k needs to be determined as well.

The proposed approach is based on three phases: *i)* transmission rate adaptation through the employment of the *report packet* at the application layer; *ii)* selection of the channel coding parameters on the basis of an interpolated packet loss probability function (ϱ_k); *iii)* selection of the source coding parameters.

Starting from the formal definitions of the ALJC problem in Section II-C, a single parameter is employed for both source and channel coding, so (s_{1k}, $\forall k \in [1, K]$, $M = 1$) and (c_{1k}, $\forall k \in [1, K]$, $N = 1$) become $s_{1k} = Q$ and $c_{1k} = R_c$.

The adaptiveness is made possible by means of a *report packet* that carries information about how many packets have been lost, sent each time a new codeword is received. In this way, the transmitter is aware of how fast the mobile network is able to deliver the video and of how vulnerable to losses is the sent video in the process of traversing the entire network. In practice, the transmission rate is regulated on the basis of the reception of the *report packet* whose arrival enables the transmission of further codewords. The adaptive heuristic ALJC approach has the granularity of a codeword and, for each decision stage, it first assess the amount of protection needed to successfully traverse the entire network, then it tries to best exploit the remaining packets to try to fit in frames with the goal of maintaining a usable frame rate, that we devised to be at least 10 frames per second.

1) Channel Coding: As stated in the previous Section, the first step in the joint decision is to assess the network loss proneness by means of a *report packet*. This means estimating ϱ_k introduced in the previous Section, that the channel encoder will use as input to decide the most appropriate amount of protection.

Deriving an analytical formulation of the coded loss probability is a dreadful task, so we tabulated the average loss of video packets as a function of the total amount of lost packets in a codeword (including redundancy packets) and the code rate R_c (i.e., the ratio between the number of video packet and the overall codeword length denoted with W). To do this for a given R_c and a number of lost packets within a codeword (denoted with P_l), we filled the $R_c \times W$ video packets of the codeword with random data, LDPC encoded them and, finally, P_l packets chosen randomly (from an uniform distribution) are cancelled. Then, after LDPC decoding, we compared the reconstructed data vector with the original one, thus obtaining the loss of video packets. This has been iterated one hundred times for each (R_c, P_l) combination, thus giving us an empirical tabulation of the average loss of video packets curves. We depicted the ensemble of these curves as a surface, reported in Fig. 2 a), and we denote it with $L_v(R_c, P_l)$.

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: in this case the decision would always fall at the end of the range (maximum protection). For this reason we define a cost function $C(R_c, P_l)$ composed of $L_v(R_c, P_l)$ and a coefficient inversely proportional to the amount of actual video packets carried by the codeword. The obtained function is convex and its minima give us the best decision for each (R_c, P_l) combination. $C(R_c, P_l)$ is represented in Fig. 2 b).

$$C(R_c, P_l) = L_v(R_c, P_l) + \frac{1}{R_c \times W} \tag{6}$$

The cost in equation (6) is computed offline only once, and provides a static decision rule for what concerns the channel coder, which is highlighted with the red line, and can be denoted $R_c(P_l)$ and defined as

$$R_c(P_l) = \operatorname*{arg\,min}_{R_c} C(R_c, P_l) \; \forall P_l \in [0, W] \tag{7}$$

2) Source Coding: Once the channel coder sets the available payload, the source coder produces video packets with the goal of minimizing the expected distortion. The employed parameter is the quality index denoted with Q, an integer ranging from 0 (worst quality, smallest image size) to 100 (best quality, biggest image size). There is no explicit relationship between Q and the output frame size denoted with F_s, and in general such relationship is not deterministic, since the DCT-based approach of the JPEG compression that tends to compress more efficiently frames with weak high frequency components. Nonetheless, since we need to know, at least statistically, what is the expected frame size for a given quality index, the $F_s(Q)$ behaviour 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 $n \in [1, N]$, the mentioned average is

$$E[F_s(Q)] = \frac{1}{N}\sum_{n=1}^{N} F_s^n(Q) \tag{8}$$

This average can be interpreted as the ideal continuous rate-compression curve sampled at regular intervals, (i.e., the frame size as a discrete function of Q) depicted in Fig. 3. To obtain $Q(F_s)$ we inverted the obtained $E[F_s(Q)]$ and we empirically fitted it with a negative exponential function as represented in Fig. 3.

It is worth noticing that the mentioned source and channel encoders represent only an implementation choice for the first release of the *App*s. Other encoders can be easily applied

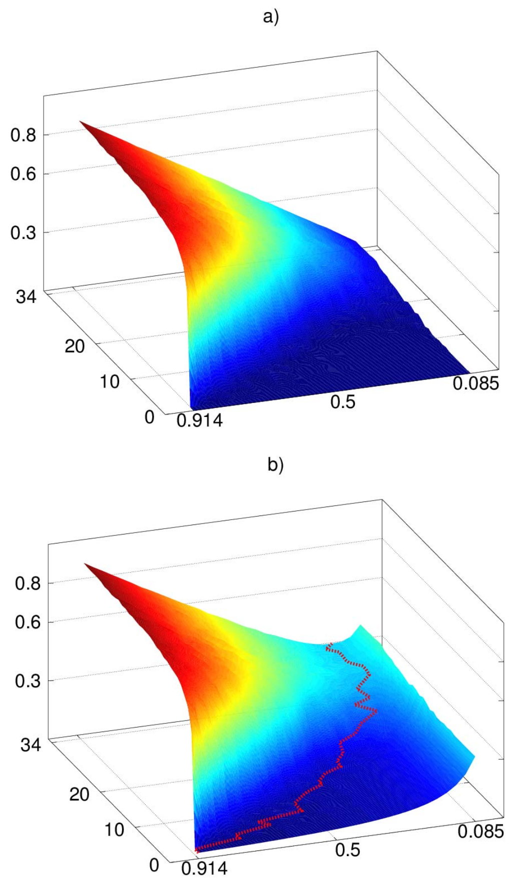

Fig. 2. a) Average codeword data packet loss, and b) cost $C(R_c, P_l)$, both as a function of the code rate and the number of lost packets. The cost's minimum represents the optimum choice.

and the performance of alternative approaches is object of ongoing investigation. In particular, MPEG compression and Reed-Solomon codes are currently considered.

Formally, referring to (7), the source coder choses the least distorting compression allowed. Given the constraints, coming from the available payload left from the channel coder, and the goal of keeping a fluid video reproduction, it assess the desired frame size, and choses Q by means of the tabulated $Q(F_s)$.

III. Performance Investigation

A. Testbed, Scenarios and Performance Metrics

We realized a testbed to simulate the scenario depicted in the introduction. In this configuration, two separate Android devices communicate through a WiFi local network connected to a machine which emulates the effect of a mobile network. On the receiving side, another WiFi network is used to interconnect the second device.

The core of the emulation consists of a regular PC equipped with two network interface cards, each connected to one access point. This by using the *netem* tool it is possible to tune the bandwidth, the packet loss and bit error rate, the delay and the packet buffer queuing policy.

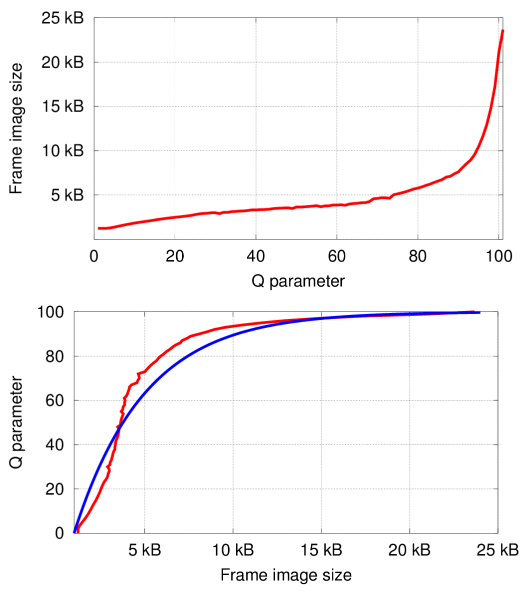

Fig. 3. Above, image size as a function of the input quality factor parameter Q. Below, In red, Q values as a function of the desired compressed image size, and, in blue, the fitted curve

TABLE I. The 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%

TABLE II. The transmitter application source and channel coding settings

	R_c	Q
ALJC	*dynamic*	*dynamic*
Minimum Protection	30/35	60
Maximum Protection	4/35	20

In order to evaluate the behaviour of the implemented *App*s we compared with two antipodal and static policies (i.e., minimal protection and maximum protection (TABLE II). The test runs evaluate each coding choice behaviour during three minutes long sessions, exploiting the aforementioned network emulating machine that simulates different bandwidth and loss conditions experienced by the network. A second run of tests deals with the system's adaptation capabilities in time varying conditions. Each run of test assumes a 600 ms latency, which is typical of geostationary earth orbit (GEO) systems.

In order to measure the quality of individual frames of the MJPEG sequence, we utilize a well known metric, the Structural SIMilarity (SSIM) index. It was first introduced in

[15], and it provides a quality measure of one of the images being compared, provided the other frame is regarded as of perfect quality. The SSIM represents a good choice since it follows the MOS more closely than other indexes such as the Peak Signal to Noise Ratio (PSNR) or Mean Square Error (MSE). The SSIM index is computed over small patches of frame, and the whole frame index is obtained by averaging the individual patches' values. The individual SSIM value for a patch is given by

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1) + (2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)} \tag{9}$$

where 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

$$\sigma_{ij} = \frac{1}{U-1}\sum_{k=1}^{V}(i_k - \mu_i)(j_k - \mu_j) \tag{10}$$

where U is the number of pixel contained in the patch and V is the number of patch.

The SSIM index ranges from 0 (completely uncorrelated frames) to 1 (identical frames), and this is useful since this can be considered a degradation factor. In order to evaluate the performance of a given test run we devised a performance index with the following requirements

- it must reward high quality frames
- it has to reward a fluent video stream, i.e. a high frame throughput
- it must penalize corrupted or lost frames

We found that the following index I satisfies such requirements

$$I = \frac{\sum_{i=1}^{U} SSIM(f_i, \widehat{f_i}) \cdot f_{received}^{TOT}}{T_{sim}} \tag{11}$$

and can be interpreted as a *quality-weighted average frame rate*. In order to collect the information needed to compute such metrics, we arranged the implemented smartphone *App*s to write down each frame: the transmitter writes a full quality version, while the receiver application records the source encoded version (i.e., including degradation due to compression). The frame written out are temporally referenced, so that it is possible to also infer information about the frame rate.
A test run is structured in the following steps:

1) The transmitter starts streaming to the receiver, which is individuated through its IP address.
2) When the duration limit is reached, the transmitter stops broadcasting and sends to the server the reference file containing the original full quality frames.
3) As soon as it is done sending, the receiver sends its data, containing the received (and possibly corrupted) video frames.

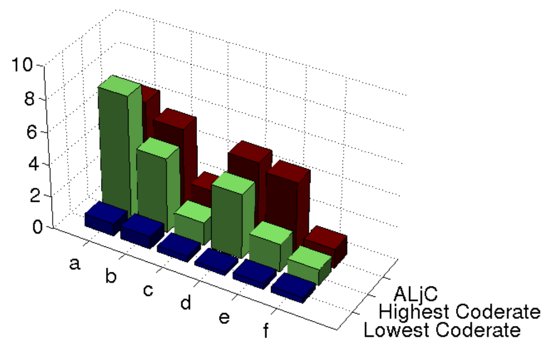

Fig. 4. Performance index I in various simulations (static channel behaviour).

B. Emulation Campaigns Results

1) Static Channel Scenarios: In this Section we show how our joint coding framework behaves in a stationary network condition (i.e. channels whose characteristics do not vary over time). In Fig.4 the referenced scenarios are those introduced in the previous Section. Scenario A simulates a wideband, error free channel: in this particular case the adaptive ALJC policy seems to perform slightly worse than a minimum protection one. This is reasonable, since the latter consistently employs high quality compression and waste a little quantity of bandwidth for protection, while ALJC more cautiously because the compression tuning "oscillates" due to a non-perfect network condition estimation in terms of both maximum allowable throughput of the network and quality. There is a minimal amount of lost packets to be imputable to kernel drops due to busy CPU. Obviously, the maximum protection approach here is the worst scoring, due to the unnecessarily poor quality of the frames and the low throughput due to high bandwidth waste. Scenario B is much more interesting: a significant BER (10%), thus high packet loss, causes the minimum protection policy to lose a substantial amount of packets, thus bringing down the overall sequence SSIM.

A very high BER (35%) represents a highly challenging channel: here the performance index cannot be as high as in previous scenarios, because the ALJC policy needs to cut down on the offered load by using more aggressive compression in order to let the frames through the smaller payload left by the channel encoder. In the meantime, the minimal protection approach now loses almost half the information fed into the network, while the conservative maximum protection policy is not able to transport a sufficient amount of information.

Scenarios D, E and F simulates narrowband channels instead. The behaviour is roughly the same as in the previous scenarios, given a lower band that allows for a lower throughput.

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. The first part (Fig.5, a), b), and c)) is aimed at showing what is the reaction to a network condition change due to a sudden narrowing of the available bandwidth. The graphs shows the average SSIM

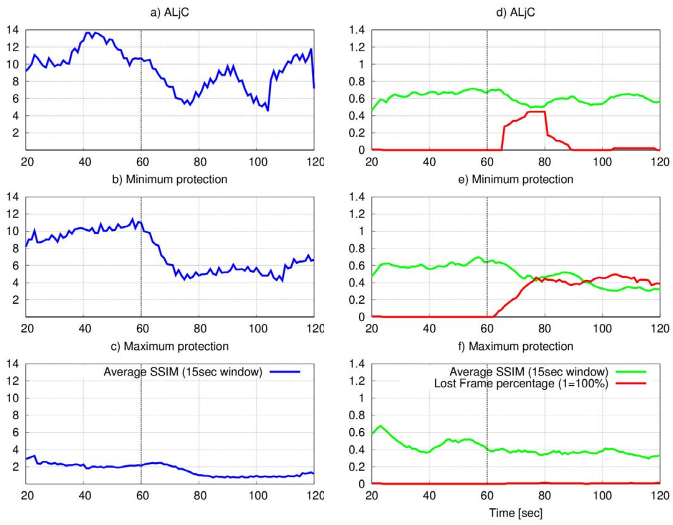

Fig. 5. Dynamic behaviour of the ALJC algorithm, in case of bandwidth drop (a), b) and c)) and BER increase (d), e) and f))

as a function of the time, as the channel bandwidth drop from 400 to 180 [Kbps]. The ALJC is able to recover after an initial drop by tuning down the offered load by adopting a more aggressive compression. The second part (Fig.5, d), e), and f)) considers a scenario where the networkterrestrial/satellite emergency network suddenly enters a "bad" state due to the high bit error rate. Again, the ALJC algorithm is able to recover after a transitory phase were it experiences a significant loss, by lowering the code rate through the R_c parameter. As a trade-off, the average SSIM tends to be lower, due to the narrower window left by a more conservative protection. The minimum protection approach here is helpless as soon as the channel begins losing packets, and the average SSIM is inevitably brought down by the packet loss. On the other hand, the maximum protection policy here is able to let the frames through the noisy channel, the frame loss is nearly unnoticeable, and the average SSIM is almost unaffected, although poor.

IV. Conclusions

In this paper we present the implementation of Smartphone *App*s suited to be employed to transmit video streams on networks based on time varying and lossy channels, such as those entailed by mobile terrestrial/satellite emergency networks. The *App*s are based on a Heuristic Application Layer Joint Coding (ALJC) approach, that adaptively applies compression and information to the sent video stream. It is shown that using only information available at the application layer we can implement a system that outperforms oblivious static coding under nearly every network condition. We then presented an investigation, carried out with real implementation of the ALJC over Android smartphones, comparing the ALJC against fixed schemes and showed that the joint approach allows a more efficient resource exploitation.

References

[1] M. M. Fouda, H. Nishiyama, R. Miura, and N. Kato, "On efficient traffic distribution for disaster area communication using wireless mesh networks," *Springer Wireless Personal Communications (WPC)*, 2013, to appear.

[2] L. F. Motiwalla, "Mobile learning: A framework and evaluation," *Computers and Education*, vol. 49, no. 3, pp. 581 – 596, 2007.

[3] J. Caldeira, J. Rodrigues, and P. Lorenz, "Toward ubiquitous mobility solutions for body sensor networks on healthcare," *Communications Magazine, IEEE*, vol. 50, no. 5, pp. 108–115, May 2012.

[4] J. Lloret, K. Z. Ghafoor, D. B. Rawat, and F. Xia, "Advances on network protocols and algorithms for vehicular ad hoc networks," *Mob. Netw. Appl.*, vol. 18, no. 6, pp. 749–754, Dec. 2013. [Online]. Available: http://dx.doi.org/10.1007/s11036-013-0490-7

[5] O. Karimi, J. Liu, and C. Wang, "Seamless wireless connectivity for multimedia services in high speed trains," *Selected Areas in Communications, IEEE Journal on*, vol. 30, no. 4, pp. 729–739, May 2012.

[6] N. Suri, G. Benincasa, M. Tortonesi, C. Stefanelli, J. Kovach, R. Winkler, R. Kohler, J. Hanna, L. Pochet, and S. Watson, "Peer-to-peer communications for tactical environments: Observations, requirements, and experiences," *Communications Magazine, IEEE*, vol. 48, no. 10, pp. 60–69, October 2010.

[7] T. de Cola, H. Ernst, and M. Marchese, "Performance analysis of ccsds file delivery protocol and erasure coding techniques in deep space environments," *Comput. Netw.*, vol. 51, no. 14, pp. 4032–4049, Oct. 2007. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2007.04.015

[8] Y. Choi and P. Momcilovic, "On effectiveness of application-layer coding," *Information Theory, IEEE Transactions on*, vol. 57, no. 10, pp. 6673–6691, 2011.

[9] A. C. Bovik, *Handbook of Image and Video Processing (Communications, Networking and Multimedia)*. Orlando, FL, USA: Academic Press, Inc., 2005.

[10] O. Bursalioglu, M. Fresia, G. Caire, and H. Poor, "Joint source-channel coding at the application layer," in *Data Compression Conference, 2009. DCC '09.*, March 2009, pp. 93–102.

[11] I. Bisio, F. Lavagetto, and M. Marchese, "Application layer joint coding for image transmission over deep space channels," in *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, 2011, pp. 1–6.

[12] M. Martini, M. Mazzotti, C. Lamy-Bergot, J. Huusko, and P. Amon, "Content adaptive network aware joint optimization of wireless video transmission," *Communications Magazine, IEEE*, vol. 45, no. 1, pp. 84–90, Jan 2007.

[13] A. Katsaggelos, Y. Eisenberg, F. Zhai, R. Berry, and T. Pappas, "Advances in efficient resource allocation for packet-based real-time video transmission," *Proceedings of the IEEE*, vol. 93, no. 1, pp. 135–147, 2005.

[14] R. Gallager, "Low-density parity-check codes," *Information Theory, IRE Transactions on*, vol. 8, no. 1, pp. 21–28, 1962.

[15] Z. Wang, A. Bovik, H. Sheikh, and E. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *Image Processing, IEEE Transactions on*, vol. 13, no. 4, pp. 600–612, April 2004.