Error Control Techniques for Efficient Multicast Streaming in UMTS Networks

Frank Fitzek, Michele Rossi, Michele Zorzi

Department of Engineering, University of Ferrara, via Saragat, 1-44100 Ferrara, Italy.
e-mail: {ffitzek|mrossi|mzorzi}@ing.unife.it

***Abstract*— In this paper we introduce techniques for efficient multicast video streaming in UMTS networks. Multicast streaming is based on the fact that the same video content is conveyed to multiple users in the same cell. Efficient multicast data delivery in UMTS is still an open issue. In particular, suitable solutions have to be found to cope with wireless channel errors, while maintaining both an acceptable channel utilization and a controlled delivery delay over the wireless link between the serving base station and the mobile terminals. We first highlight that standard solutions such as unequal error protection of the video flow are ineffective in the UMTS systems due to the large feedback delay. Then, we propose a local approach to solve errors directly at the UMTS link layer while keeping a reasonably high channel efficiency and saving, as much as possible system resources. The solution we propose in this paper is based on the usage of the common channel to serve all the interested users in a cell. In this way, we can save resources with respect to the case where multiple dedicated channels are allocated. Moreover, we present a hybrid ARQ proactive protocol that, at the cost of some redundancy (added on the link layer flow), is able to greatly improve channel efficiency with respect to the simple ARQ case by making the use of a single common channel for multicast data delivery feasible. Furthermore, we envision the usage of suitably encoded video streams.**

Index Terms – **UMTS, multicast, error control, FEC, hybrid ARQ, common channel, dedicated channel, video quality, PSNR.**

I. Introduction

Currently the UMTS [1] networks become reality in some parts in Europe. Other parts will be deployed within the next years. After the high investment costs of hundreds of billion Euro for the license band for Europe of the 3rd generation technology, further costs of the same amount will come up for the installation of this technology. Obviously, network providers are now interested in a fast return of investment. To convince customers to use UMTS technologies network providers highlight new services such as video services. In contrast to voice services, video services are characterized by large bandwidth requirements, which can be a hundred times higher than a single voice service. The dilemma for the network provider can be described as follows: on one hand new services are needed; speech and data services are already provided cost efficiently with GSM/GPRS network and will not motivate the customer for a new mobile end-system, on the other hand video services can not be billed with the same Cent per bit ratio as the voice calls, because no customer can afford such a service. Of course one may think of decreasing the video quality such that the required bandwidth becomes lower, but this may not satisfy the customers' requirements.

The solution presented in this paper advocates the usage of multicast capability to make video transmissions interesting for the customer improving the Cent per bit ratio maintaining a high video quality. This approach is only valid in case multiple customers per cell are interested in the same content. Multicast can be applied in two different ways to the UMTS network: the content is conveyed to the wireless end system using either the Dedicated (DCH) or the Common CHannel (CCH) of UMTS. The advantage of the dedicated channel is that for each wireless terminal a separate channel is maintained with fast power and error control for each single wireless terminal. On the other side, multiple wireless terminals can be subscribed to the common channel. Here, no fast power control is done and the error control has to be applied in different ways, i.e., a retransmission is needed if at least one among the served users does not correctly receive a given packet. The advantage of the common channel is that less bandwidth is required. In case dedicated channels are used, N times the video bandwidth is need, while the common channels will use the bandwidth only once (with some overhead due to additional error protection).

The aim of the paper is first to clearly state the problem of multicast streaming delivery in a UMTS network in the wireless part. Note, methods to apply multicast in the UMTS backbone is out of the scope of this paper. Afterwards an effective solution to achieve error control while keeping the channel efficiency at a high value is proposed. A small example shall illuminate our approach. In case only one customer is interested in a specific content, a dedicated channel is used. In case multiple customers are interested in such a content, multiple dedicated channels can be used or even a single common channel. It is generally excepted that wireless links are highly error-prone, and therefore the error control is a very important key feature for the transmission. While for each dedicated channel an own error control entity can be initialized, the error control for the common channel has to be done satisfying the wireless terminal with the worse link characteristic. As an example think of a simple ARQ strategy such as Send and Wait: Each time only one wireless terminal out of the multicast group ask for a retransmission, the filling procedure of the play-out buffer is stalled for all users. Moreover, in case only a limited number of packet retransmissions for a given packet are allowed, the performance at the application layer suffers. It depends on the chosen video coder parameter how this deterioration of the underlying link level effects the application quality.

Error control algorithms can be classified into FEC, ARQ and hybrid ARQ techniques. Forward Error Correction (FEC) is based on the transmission of redundant information through error correction codes. In ARQ a feedback link is used to request retransmission of erroneous packets. Hybrid ARQ schemes combine the advantages offered by FEC and ARQ.

The efficient data delivery to a group of multicast receiver has been studied for the first time in [2]. A further contribution can be found in [3]. In both these papers, the authors analytically derive relationships between channel efficiency (expressed as average number of transmission per correctly delivered multicast packet) and the multicast group size, by considering independent and correlated packet losses. In particular, in [3] two key problems are addressed: the efficiency of hybrid ARQ protocols and the comparison between integrated and layered FEC. In layered FEC the packet coding operates independently beneath

This work has been supported by ERICSSON research.

ARQ (or Reliable Multicast, RM) layer. In this case, the role of FEC is to reduce the residual packet loss probability at the link layer, thereby decreasing the number of retransmissions and the network bandwidth requirements. In the integrated model, FEC and ARQ are integrated into the same layer and the FEC module operates as follows: for a given set of data packets it computes and holds a set of parity packets. Then, once the sender has received all the retransmission requests across the feedback channel, it transmits the minimum number of parity packets needed to recover all the losses in the original data set. Integrated FEC gives better performance than layered FEC [3].

A very interesting framework for reliable and efficient multicast delivery can be found in [4]. In this thesis, a large quantity of material can be found. The author also presents some techniques to estimate the number of multicast receivers based on the number of NACKs received at the sender side.

In [6][7][8] the authors focus on the design of reliable multicast protocols using ARQ and FEC. The obtained results show that pro-actively sending a (usually small) number of redundancy packets along with the original data can improve efficiency and could reduce feedback. In these works the NACK implosion problem is also analyzed and it is shown that pro-actively sending a given amount of redundancy packets along with the original data highly reduces the number of retransmission requests from the receivers enabling the multicast group to grow up to thousands of users.

In all these articles a packet-based Reed Solomon Encoder (RSE) is considered. A software implementation of this encoder is presented in [9][10]. In particular, in this works it is proved that software implementations are feasible in the sense that, even in very slow processors they present very short encoding and decoding delays. This is very important to enable the use of such a packet-based coding scheme directly at the Reliable Multicast (RM) level. For instance, referring to a UMTS scenario, and considering to have a highly reliable fixed network, the problem is the multicast delivery from the Node B to the mobile systems linked with it and subscribed to the multicast group. Hence, it could be very effective to apply such a RS approach directly at the link layer (RLC, Radio Link Control), taking link layer packets (PDUs) as the packet units to be used for coding. In this case most of the work in previously presented papers still applies.

A further approach is the usage of Tornado codes [11]. They are characterized by faster encoding and decoding algorithms.

In [12] and [13], the RDMP protocol is presented. In RDMP packet based FEC is combined with ARQ to transmit multicast messages over the single common forward channel. Also the feedback channel is shared among users. To limit the ACK collision problem the authors rely on the right use of FEC and a back-off algorithm to decide when ACKs have to be sent by users. These strategies are intended to reduce the NACK collision probability over the feedback channel.

The Pragmatic General Multicast (PGM) protocol [14] has been recently proposed and now it is an Internet Engineering Task Force (IETF) experimental Request For Comment (RFC). PGM exploits packet-based coding NACK suppression and NACK elimination to provide reliable data delivery while achieving a high channel efficiency over asymmetric networks.

Most of these contributions refer to the case of multicast transmission over wire-line networks, but similar approaches can be extended to the wireless case as well.

We go further with respect to previous research by proposing algorithms specifically tailored for the UMTS cellular network scenario. We use the incremental redundancy concept for retransmissions and we characterize the proposed schemes considering both delay and channel efficiency. Furthermore, for what concerns delay performance, we consider full delay statistics instead of mean values. The major scope of our ongoing research is the design of algorithms for real environments. Therefore, to highlight the behavior of the proposed schemes in correlated channels, we characterize the link layer error process for each multicast users by means of a two-state Markov chain.

The paper is organized as follows: In Section II a brief overview of the possible ways of handling the multicast delivery problem in a UMTS network is presented. Then, the best suitable way of operating is presented and motivated. In Section III, the system model is presented together with the basic way in which error control can be achieved (simple ARQ). In Section IV packet-based error correcting codes are briefly introduced and their benefits are explained. These codes, are used at the link layer level in addition to plain ARQ techniques in order to improve performance. These hybrid ARQ algorithms are proposed and detailed in Section V. In Section VI their performance is evaluated both in terms of channel efficiency and delay considering independent packet error losses. The effect of correlated losses is investigated in Section VI-B and finally, in Section VII some conclusions are given to outline the main contributions of the paper, while Section VIII gives future directions of our further research.

II. The UMTS Network Architecture

In Figure 1 the involved entities for video streaming in UMTS networks and the related protocol stacks are given ([15][16] refer to the specific case where streaming data arrives from an Internet server to UMTS mobile terminals in a timely fashion. The physical placement of the server, within the IMS (IP Multimedia Subsystem) or the Internet, is out of the scope of this paper.) It is interesting to note that the IP level is not available on all entities along the path. In particular, only the terminal equipment (UE), the SGSN/GGSN (Serving/Gateway GPRS Support Mode) and the Internet server are equipped with an IP layer. For this reason, they are the only terminals able to manage video frames and to eventually take advantage from channel adaptive video coding techniques [17][18][19], where Unequal Error Protection (UEP) is usually employed to improve the video flow robustness against losses. In these algorithms, the channel quality is monitored by means of feedback information and, based on it, an estimate of the video quality perceived at the end user is used to drive both the frame compression process and the amount of redundancy added to each video frame. The effectiveness of this process depends on the time for the feedback to reach the sending entity. If this time is too large with respect to the stationarity period characterizing the underlying error process, an inertial effect will arise in the error control actions. In such cases, the feedback information is ineffective, since the forward flow adapts too slowly to the underlying error sequence and the user quality could be worse than without using adaptation at all. Here, we stress the fact that this last case is the one envisioned in a UMTS network, where End-to-End delays can be up to 0.5 seconds. Hence, standard solutions based on End-to-End adaptive algorithms seem not to be the correct way to improve the video quality is such networks.

On the contrary, noting that the main cause of errors is due to the wireless link one could think to implement some kind of error control mechanism restricted to the Radio Access Network (RAN). In this way, the round trip delay is strongly reduced[1]. The drawback of this solution is that is not possible to operate directly on the video flow, because the UE is the only terminal in the RAN that has access to the IP flow. For this reason, we advocate to perform error control at the Radio Link Control layer (RLC) that is present in both the UE and SRNC

[1]Up to 220 ms at the RLC layer, its actual value depends on the interleaving depth set at the underlying physical layer.

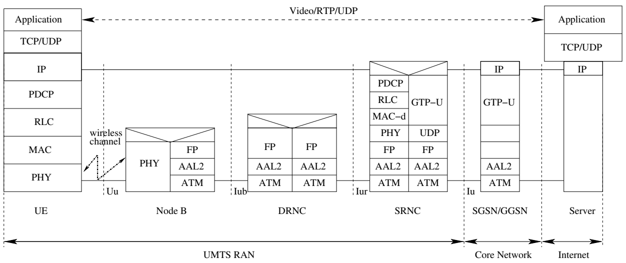

Fig. 1. Entities and Protocol Stack within UMTS network.

(Serving Radio Network Controller). As observed above, we can not operate directly on the video frame flow, but on the other hand we have RLC native feedback information that can be used to drive channel adaptive algorithms. Moreover, in our solution we think to jointly utilize, at the RLC layer, retransmissions and packet-based encoding techniques in order to enhance the channel efficiency while using a single CCH channel to serve all users. The model for the ARQ transmission/retransmission process is presented in the next section, whereas in Section IV we briefly introduce the packet-based FEC technique that will be incorporated in our hybrid ARQ schemes. The hybrid ARQ algorithms will be presented in detail in Section V.

III. MODEL FOR ARQ ERROR CONTROL PROCESS

We consider an Internet server sending a single video stream to a multicast group located in the UMTS network. We consider a single UMTS cell and that a given number of users N_u within the cell is interested in the reception of the flow. We refer as $\mathcal{N}$ to the set of multicast user in the cell, where the number of users in $\mathcal{N}$ is referred to as $|\mathcal{N}| = N_u$. Without loosing generality, we consider that only one multicast group exists in the cell and we investigate how bandwidth requirements and residual error rates at the application level are related to N_u and to the error processes characterizing the link layer at each user. In our model, the time axis at the RLC layer is slotted, where the slot time corresponds to the single link layer packet (PDU) transmission duration. Moreover, we use a Discrete Time two-state Markov Channel (DTMC) to describe the PDU error process affecting the PDU error process for each user in $\mathcal{N}$. In more detail, the channel is characterized by two states, a *"good state"* (state 0), where PDUs are transmitted correctly and the *"bad state"* (state 1), where PDUs are corrupted with probability one. This simple model evolves slot by slot according to its transition probabilities; we refer to as $p_{ij}(u)$, $i, j \in \{0, 1\}$ to the probability that the channel of user u transits in state j in the next slot given that, in the current one it is in state i. We refer as $\mathbf{P}(u)$ to the channel transition probability matrix relative to user u. The plain[2] RLC transmission/retransmission process is modeled as follows:

- At the sender side, higher level packets (SDUs) are segmented into smaller packets, called PDUs. These packets are processed adding a header and a CRC (Cyclic Redundancy Check) field. These fields are needed, at the receiver, to check for the correctness of each packet (CRC) and to perform the reassembly of link layer PDUs.
- At the receiver side, every PDU is checked using its CRC field and, based on the result of this operation, a retransmission for that packet is required sending over the reverse link a negative acknowledgment (NACK) message.
- At the receiver side, link layer PDUs are used to reconstruct higher layer SDUs. SDUs can be passed to higher levels using either *in-order* or *out-of-order* delivery [20].
- The link layer experiences a round trip time that for simplicity is considered to be an integer number of channel slots, m, i.e., a integer number of packet transmission times.
- A given packet is considered to be error-prone in a given slot if at least one multicast user does not correctly receive it.
- If a given packet is transmitted erroneously in a given slot, say slot i, then it must be retransmitted m slots apart, i.e., in slot $i + m$. This operation is equivalent to inserting that packet into the link layer retransmission queue and to re-send it when the relative NACK arrives[3].
- If a packet is correctly delivered[4] in a given slot, say slot i, then a new link layer packet is taken from the link layer queue and is scheduled to be transmitted m slots apart, i.e., in slot $i + m$.
- We consider a *Heavy Traffic* assumption, i.e., a new packet to be sent is always present into the link layer queue. Note that this assumption is a good approximation of the incoming flow at the RLC layer when video/audio streaming is considered.

It is important to note that this model, despite of the numerous simplifications with respect to the real RLC level [20] leads to an upper bound for what concerns the forward link channel utilization because link layer PDUs are always retransmitted with the lowest latency (exactly after one round trip time from their previous transmission). Note also that multiple copies of the same PDU are never transmitted during a single round trip time. In real UMTS RLC implementations, depending on the specific RLC timer profile, situations where the same PDU is transmitted more than once in a round trip time might occur. Note that these configurations are sometimes forced to achieve short RLC SDU delivery delays at the cost of the forward channel throughput. In these particular cases the presented model is not a lower bound for the delay but it still be an upper bound for what concerns the throughput performance. Furthermore, in the scheme presented above one ACK is received for each PDU, i.e., we have one ACK for every slot in which a transmission occurs. This assumption is not true in real systems, where ACK messages of different PDUs are often grouped into single messages called *status reports*. However, the main aim of our work is on the characterization of delay and efficiency for what concerns the forward data flow, i.e., the one flowing from the SRNC to the UE.

IV. PACKET-BASED FEC TECHNIQUES

Forward Error Correction (FEC) techniques involve the transmission of original data along with additional redundant data which are used to reconstruct the original data is some of it is lost. One possible family of codes for packet-based encoding is the Reed-Solomon one. These are block-based error correcting codes with a wide range of applications in digital communications and storage. The Reed-Solomon encoder takes a block of digital data and adds redundant bits. The Reed-Solomon decoder processes each block and attempts to correct errors and recover the original data. The number and type of errors that can be corrected depends on the characteristics of the Reed-Solomon code.

[2] This is the simple ARQ mechanism, further extensions taking into account packet-based coding techniques will be presented later in Section V.

[3] Without loss of generality NACK and ACK messages are considered to be error free.

[4] All users have correctly received the packet.

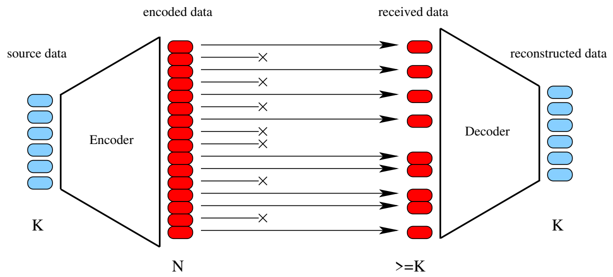

Fig. 2. Packet-based encoding/decoding process. The original K PDUs can be reconstructed if a number of PDUs equal to or greater than K is correctly received out of the N sent PDUs (FEC block).

In more detail, Reed Solomon codes are a subset of BCH codes and are linear block codes. A Reed-Solomon code is specified as $RS(k, n)$. This means that the encoder takes k data symbols of s bits each and adds parity symbols to make an n symbol codeword. There are $n-k$ parity symbols of s bits each (where s is the size of each symbol). The correcting capability of an RS code depends on $n-k$. A RS decoder can correct up to and including t symbols that contain errors in a codeword, where $t = \lfloor (n-k)/2 \rfloor$. However, the decoder can correct more than t symbols only if it knows where the errors are. These known error locations are called erasures. The key idea behind erasures is that K blocks (packets) of source data are encoded at the sender to produce N blocks of encoded data, in such a way that any subset of K encoded blocks suffices to reconstruct the source data. Such a code is called a (K, N) erasure code (RSE code) and allows the receiver to recover from up to $N-K$ losses in a group of N encoded blocks (packets) [2][3]. In each packet a Cyclic Redundancy Check (CRC) field must be present to detect errors (erasures). Note that this feature comes for free at the UMTS RLC layer. In Figure 2 a graphical representation of the encoding/decoding process is given.

Suppose to have K source information packets, $\{d_1, d_2, \ldots, d_K\}$ each of which is P bits long. The RSE encoder takes these K packets and produces $N-K$ **parity** packets $\{p_1, p_2, \ldots, p_{N-K}\}$. Then the original K data packets plus the $N-K$ parity packets are sent along over the channel. Each FEC block of N packets is commonly referred to as *transmission group* (TG). At the receiver side it is sufficient that K of these N packets are received correctly in any order to reconstruct the original K packets (see Figure 2). Most RSE implementations send the first K original data as the first K packets to simplify decoding (*systematic code*). In this way, if all the K data packets are correctly received, no decoding at all is required at the receiver. If, due to channel error, at the receiver side $l \leq \min(K, N-K) < N-K$ packets have to be reconstructed the decoding overhead is proportional to l (see [10]). Especially in the cases of multicast there are additional benefits using the parity packets for loss recovery instead of transmitting the lost packets:

- **Improved transmission efficiency**: A single parity packet can be used to repair the loss of any packet in the TG. This means that *a single parity packet* can repair the loss *of different data packets at different receivers*. This fact is extremely useful since different receivers are in general affected by independent error processes.

- **Improved scalability in terms of group size**: In ARQ schemes the sender needs to know the sequence number of each lost packet. Instead, using parity packets for loss repair, the sender needs to only know the maximum number of packets lost by any receiver but not their sequence number. So, the feedback is reduced from per-packet feedback to per-TG feedback. In fact, depending on the number of packet lost, a given amount of new redundancy packets, obtained by the original K packets, can be transmitted over the channel (incremental redundancy) so that the original data is recovered if at least K packets are correctly received among all the received packets for the TG (N FEC block packets plus incremental redundancy packets).

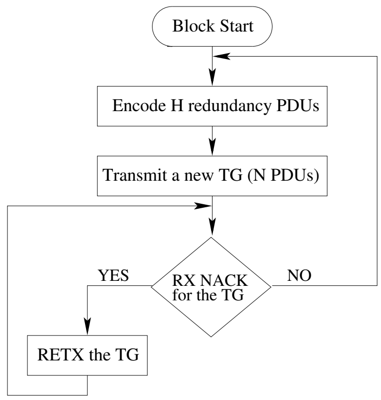

Fig. 3. Sender protocol for the algorithm A1.

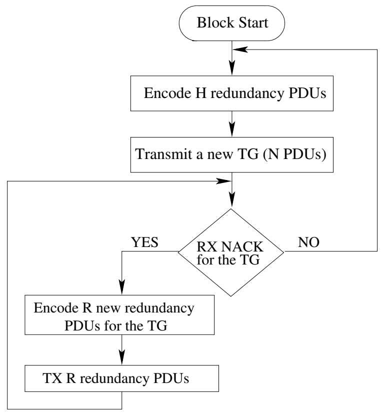

Fig. 4. Sender protocol for algorithms A2 and A3.

V. ALGORITHMS FOR MULTICAST DELIVERY

In this section we propose three hybrid ARQ algorithms that can be used at the RLC level to improve both efficiency and delay performance with respect to the simple Selective Repeat ARQ scheme described in Section III. The hybrid schemes are described in detail in the following:

A1: The sender collects groups of K packets and then feed them to the packet-based encoder by obtaining H redundancy PDUs so that a total number of $N = K + H$ PDUs is available for each TG, as discussed in Section IV. The entire TG (N PDUs) is sent to all receivers over the CCH channel. Note that at the sender side some amount of redundancy (H PDUs) is proactively added to the forward multicast flow. As said above, this redundancy can be independently exploited by each multicast receiver to recover the original K packets when the number of lost PDUs a TG is less than or equal to H. When a multicast receiver is able to correctly decode less than K PDUs in a TG it sends back to the transmitter a NACK message for that TG including the TG identifier. Finally, at the sender side, all NACKs for the same TG are collected and the entire TG is retransmitted if at least one multicast user required its retransmission, i.e., if the number of collected NACKs is greater than 0. A flow graph for this scheme is depicted in Figure 3. As will be shown in the next this solution gives good performance in term of delay, but it is not very efficient in term of throughput. At least one drawback is present in that scheme. In fact, a priori retransmitting all the PDUs in a TG may be not a good choice, because not all the receivers should need the retransmission of all the N PDUs and this may decrease the channel efficiency.

A2: As in the previous scheme, at the sender side TGs of N PDUs (K data packets plus H redundancy PDUs) each are sent first. Then, each receiver checks for errors in each TG and replies accordingly. In more detail, if a receiver detects less than K correct PDUs for a TG it sends back to the sender a NACK including the TG identifier. The sender collects incoming NACKs and, if the number of collected NACKs for a given TG is greater than 0 the following procedure is activated:

- The K original PDUs included in the erroneously received TG are feed again to the packet-based encoder by obtaining a new redundancy packet, i.e., a redundancy packet for that TG that is however different from all redundancy packet previously transmitted.
- The new redundancy PDU (*incremental redundancy*) is sent over the CCH channel.

The flow graph for this scheme is plotted in Fig. 4. Referring to that figure we have that $R = 1$, i.e., the number of redundancy packets transmitted for each received NACK is always equal to one. In this scheme, each receiver collects all the received packets for a TG, i.e., the N PDUs sent in the first TG transmission plus the, say R, redundancy PDUs sent in the following retransmissions (triggered by NACKs). The original K PDUs in a TG can be recovered if the number of correct PDUs, N_{ok}, over the $N + R$ PDUs is greater than or equal to K. It is interesting to note that this scheme tries to maximize the channel efficiency. In fact, at each retransmission request the minimum amount of redundancy is retransmitted to all users (only one PDU). So, if at least one user needs more than one new PDU to obtain the K original data packets he will require a further retransmission. For this reason this scheme is also characterized by the largest delay for the correct delivery of a TG.

A3: At the sender side TG of N PDUs (K data packets plus H redundancy PDUs) are sent first. Each receiver check for errors in each TG and replies accordingly. In this scheme incremental redundancy is also used. To better explain as the algorithm works let us suppose that, in addition to the N PDUs in the first transmission, R redundancy PDUs have been already sent over the channel for a given TG. In this case, each receiver check for the number of correctly received PDUs (N_{ok}) among the $N + R$ PDUs sent. Let us refer to a given user in $\mathcal{N}$, say user i. If $N_{ok}(i) \geq K$ the original K PDUs can be obtained and the TG is correctly received. Otherwise, if $N_{ok}(i) < K$, the receiver sends back a NACK including the TG identifier and $r_i = K - N_{ok}(i)$, i.e., the minimum number of new redundancy PDUs needed for the correct decoding of the K data PDUs. The sender collects incoming NACKs and computes $R_{max} = \max_{i \in \mathcal{N}}(r_i)$. Then, $R = R_{max}$ new redundancy PDUs are encoded for that TG and are transmitted over the common channel. This procedure is repeated until all users in $\mathcal{N}$ are able to correctly decode the K data PDUs, i.e., when $N_{ok}(i) \geq K \ \forall i \in \mathcal{N}$. A flow graph for this scheme is reported in Figure 4, where the number of redundancy PDUs sent upon the reception of a NACK message is equal to R_{max}, as specified above. This last scheme tries to achieve a trade-off between channel efficiency and delay. In fact, for each retransmission request, the minimum number (channel efficiency $\uparrow$) of PDUs needed to ensure that all multicast receiver will be able to recover the original K packets is sent (delay $\downarrow$). Note that R_{max} new PDUs guarantee the successful decoding of a TG by all receivers only if no channel errors occur, or if channel errors are such that each receiver is able to decode at least r_i PDUs out of the R_{max} transmitted.

VI. Results

In this section, the performance of the plain Selective Repeat ARQ presented in Section III is compared to the one achieved for the three algorithms described in the previous section. Three kinds of performance metrics are considered. The first one is the common channel mean throughput, that is defined as the number of data PDUs sent in a given time interval $[0, t]$, $N_{data}(t)$, with respect to the total number $N_{tot}(t)$. The mean throughput can be obtained by $throughput = \lim_{t \to +\infty} N_{data}(t)/N_{tot}(t)$. This metric accounts for the channel resources consumed by redundancy and retransmissions. The last two metrics are relative to the delay needed to correctly receive a TG. Here, we first define the *RLC SDU delivery delay* as the number of slot elapsed between the one where the first PDU of a given RLC SDU is transmitted for the first time over the channel and the slot where all the multicast users in $\mathcal{N}$ correctly received the last PDU composing that SDU, i.e., all users in that "final" slot correctly received the entire SDU. Hence, the SDU delay computed in this way can be interpreted as a "worst case" SDU delay.

We refer to the probability that a SDU is delivered in $x \geq 0$ slots as $P_d[x]$, whereas the complementary delivery delay distribution, i.e., the probability that a SDU is delivered in more than x slots is labeled as $ccdf[x]$. In the next sections, both the SDU mean delivery delay and its $ccdf$ are reported. Note that, in the video streaming case this metric is directly related to the delay performance perceived at the UE IP level. In fact, the UDP protocol is typically used for streaming application. Hence, the delays experienced by higher layer IP packets in their transmission between the SRNC and the UE are entirely due to the RLC error control process and to all the additional delays involved in the physical layer processing[5].

In the results discussed in the following sections we consider a link layer logical bit rate of 240 Kbps (spreading factor equal to 16) a RLC round trip time (RTT) of 80 ms (this is a lower bound for the RLC RTT in a UMTS network) and a link layer PDU length of 352 bits. Then, we have about 56 PDUs in a link layer round trip time. Moreover we consider a fixed SDU packet length of 500 bytes[6].

A. Performance in independent error channels

In this section, we consider that all multicast users are affected by an independent (*iid*) PDU error process and that the average PDU error probability for user $i \in \mathcal{N}$ is equal to ε_i.[7]

As a first result, in Figure 5, we report the mean SDU delivery delay, i.e., the mean time needed to correctly transmit a full SDU to all users in the multicast group as a function of the multicast group size, N_u, considering that all users are characterized by an *iid* channel with the same PDU error probability $\varepsilon = 0.1$, i.e., $\varepsilon_i = \varepsilon = 0.1$, $\forall i \in \mathcal{N}$. From that

[5] In this paper we consider physical layer processing delays as constant and we include them in the link layer round trip time m.

[6] This is a typical value for the frame packet length in a video streaming flow [21].

[7] This can be achieved by setting $p_{11}(i) = p_{01}(i) = \varepsilon_i$, $\forall i \in \mathcal{N}$.

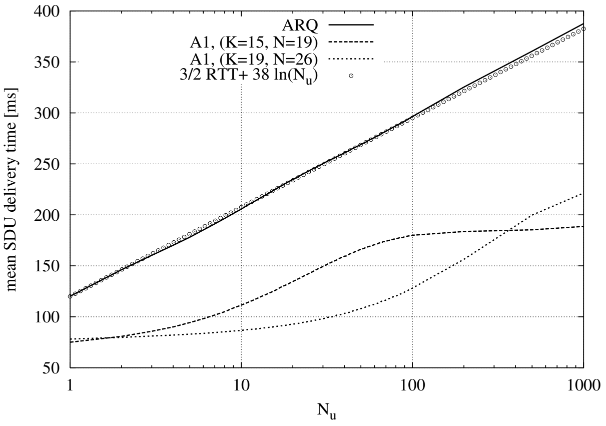

Fig. 5. Mean SDU transmission time as a function of the number of multicast users, N_u with $\varepsilon = 0.1$ and *iid* channel. Comparison between A1 scheme and selective repeat ARQ.

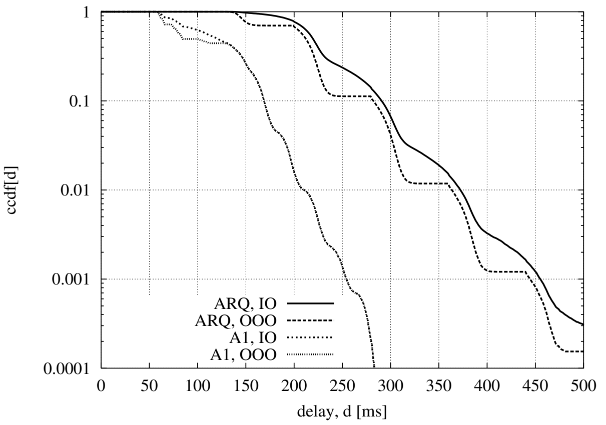

Fig. 6. SDU complementary cumulative delivery delay statistics, $N_u = 10$, $\varepsilon = 0.1$, *iid* channel. Comparison between A1 (K=15, N=19) scheme and selective repeat ARQ.

figure it is very interesting to note that in the simple ARQ case, the behavior of the mean SDU transmission time is logarithmic in N_u. In other words, the SDU delivery delay tends to infinity as N_u increases. Also in the hybrid ARQ case the SDU delivery delay tends to infinity when $N_u \to \infty$ but here the delay increase very slowly. This is a very important fact because it considerably improves the system scalability in term of multicast group size. Moreover, we can note that a more powerful code is more insensitive to the multicast group size and it is able to keep the maximum SDU delay to a smaller value.

In Figure 6 the *ccdf* is plotted comparing the A1 scheme against the simple ARQ. The multicast group size is $N_u = 10$. The labels IO and OOO are used to indicate the statistics regarding in-order and out-of-order delivery of RLC SDU [20], respectively. The delay has been re-scaled in ms in order to allow a better understanding of the delays involved. Form this figure we can note that the SDU delivery delay is largely reduced using hybrid ARQ. For example, for a delay $d = 250$ ms in the in-order delivery case a simple ARQ presents a probability of 0.3 (3 PDUs in 10) that a SDU is transmitted in a time longer than d, whereas in the hybrid ARQ case the probability for the same event is of about 0.0001. Moreover, using the hybrid scheme, in-order and out-of-order delivery cases are almost equivalent, the only differences are for d up to $3RLC_{RTT}/2$. So, the out-of-order delivery feature does not appear to give a significant

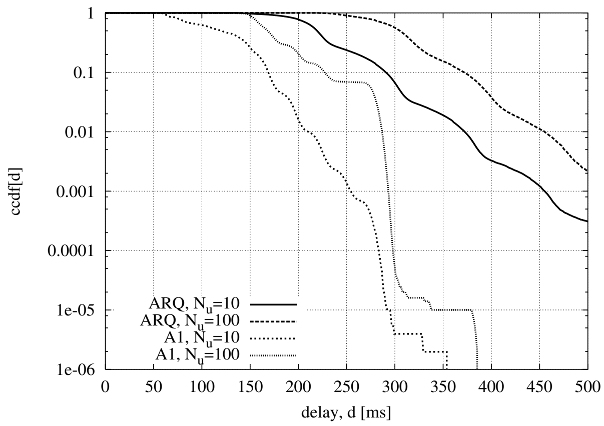

Fig. 7. SDU complementary cumulative delivery delay statistics by varying N_u, $\varepsilon = 0.1$, *iid* channel, in-order delivery case. Comparison between A1 (K=15, N=19) scheme and selective repeat ARQ.

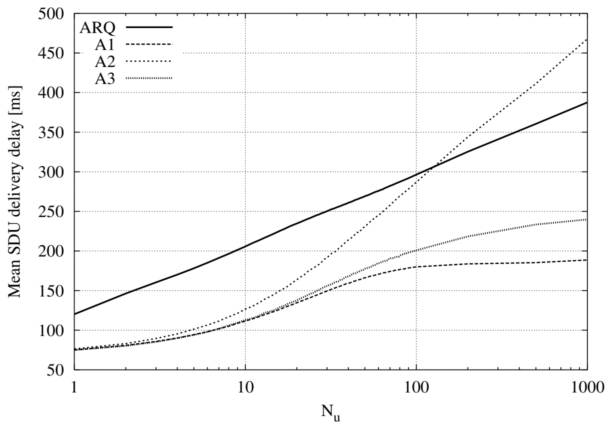

Fig. 8. Mean SDU delivery delay time as a function of N_u, $\varepsilon = 0.1$, *iid* channel. K=15, N=19.

gain when hybrid ARQ solutions are considered. The fact that in-order and in out-of-order cases are equal means that the delivery of a SDU in the in-order case for $d \geq 3RLC_{RTT}/2$ does not depend on erroneous PDUs contained in previously transmitted SDUs. In other words, considering a tagged SDU and considering that the first PDU composing it is transmitted for the first time at time $t_1 = 0$, the probability that a PDU transmitted before than the tagged SDU (time $t_2 < t_1$) is still erroneous at time $t_3 \geq 3RLC_{RTT}/2$ is negligible. This effect is due to the use of FEC.

In Figure 7 we can observe that increasing the group size (N_u) the hybrid ARQ largely limits the SDU delays. In particular, the SDU delivery statistics, after a certain point (in $d \approx 300$ ms), starts decreasing very rapidly. The statistics regarding the simple ARQ, instead is shifted to the right without any shape change. This means that hybrid ARQ techniques are more robust with respect to an increasing number of multicast users in the system.

The comparison between mean delivery delays between all schemes is plotted in Figure 8. All hybrid schemes when N_u is small are characterized by a lower delay with respect to the simple ARQ. However, as N_u increases, scheme A2 quickly reach the simple ARQ performance. This is due to the fact that in A2 only one packet retransmission is allowed at a time. In this way channel resources are saved, but at the cost of a poor delay performance. On the other side, scheme A1 is the one

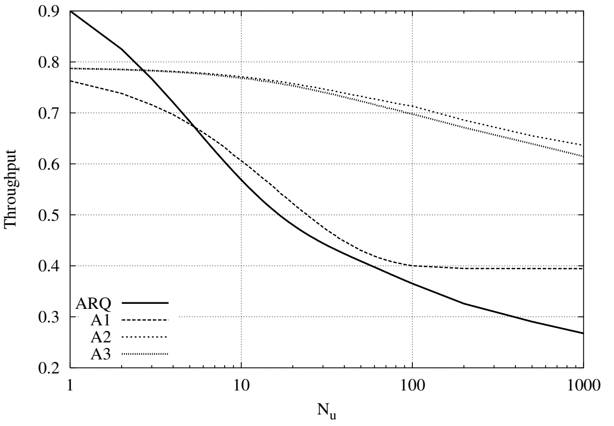

Fig. 9. Throughput as a function of N_u, $\varepsilon = 0.1$, *iid* channel. K=15, N=19.

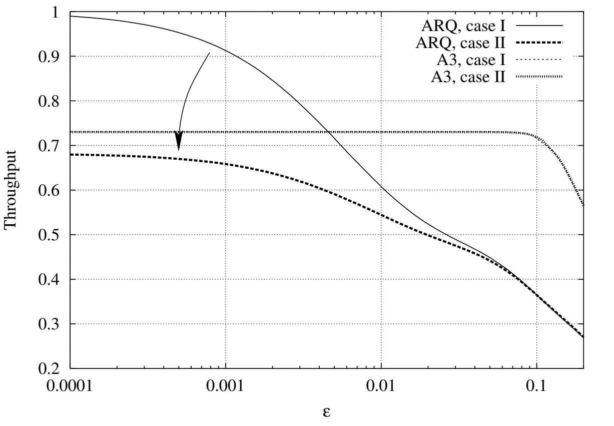

Fig. 10. Robustness of hybrid ARQ schemes against a small number of users suffering from a bad channel condition. $N_u = 100$, A3 (K=15, N=19).

characterized by the shortest delay. It is interesting to note that also scheme A3, where the minimum amount of redundancy is retransmitted so that all users can recover from errors, is able to keep the SDU delay to an acceptably low value.

In Figure 9 we report the channel efficiency as a function of N_u. As observed above, scheme A2 is the one characterized by the highest efficiency, whereas A1 is characterized by the worst one. All the hybrid algorithms outperform simple ARQ as N_u become greater than 3, i.e., when the multicast group is very small the overhead due to the proactively added redundancy in schemes A1, A2 and A3 (a priori data protection) is more channel consuming than performing retransmissions only (simple ARQ). However, it is important to note that hybrid ARQ schemes lead to a higher throughput as N_u increases over 3 and that for large N_u values they are still achieving a acceptably high channel efficiency, whereas the simple ARQ in such a case has very poor throughput performance. Another interesting result is that performance of scheme A3 is very close to the upper bound (that, as observed above is given by scheme A2). This scheme may be a good candidate to be effectively used for multicast data delivery because it is characterized by a good channel efficiency and also its delay performance are not too worse with respect to scheme A1.

Finally, in Figure 10 we compare the simple ARQ with scheme A3 considering $N_u = 100$ in the following two cases. **Case I**: all the 100 users are characterized by $\varepsilon_i = \varepsilon$, $\forall i \in \mathcal{N}$,

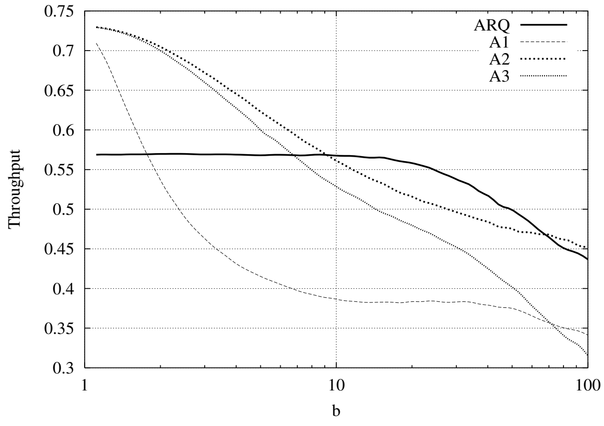

Fig. 11. Effect of the channel burstiness. $\varepsilon = 0.1$, $N_u = 10$, K=19, N=26.

where ε is the PDU error probability reported in the abscissa of Figure 10. **Case II**: 95 users are characterized by the error probability ε reported in the abscissa in Figure 9, whereas the remaining 5 users are affected by a fixed error probability of 0.1. It is very interesting to note that the hybrid scheme in case II presents almost the same performance than in Case I. In other words, the hybrid ARQ is almost insensitive to a small number of users in a bad channel condition. On the contrary, simple ARQ performance is heavily degraded also when a small fraction of the users is affected by a bad channel state. This is important to achieve a scheme that is robust against dynamic changes in the number of multicast user joining a group and to unpredictable changes in the users' channel state.

B. Effect of burst errors

In this section we report some results on the effect of the channel burstiness on hybrid ARQ performance. In particular, we refer to $b(i) = 1/p_{10}(i)$ as the average burst length of user $i \in \mathcal{N}$, where $b(i)$ is the average number of slots (PDU transmissions) that the channel of user i remains in the bad state in a row. In Figure 10 we plot the channel efficiency as a function of b, where we consider $N_u = 10$ multicast users all affected by the same burst length $b(i) = b$, $\forall i \in \mathcal{N}$. We also consider that ε is fixed at the value 0.1.

The effect of the error burstiness must be surely taken into account since, depending on the cell load and on the distance between the Node B and the user, the CCH channel can be likely characterized by larger burst than the dedicated one. Longer bursts are often measured near to cell borders, where the CCH downlink power should not suffices to counteract for path loss and user interference. In the dedicated channel case this phenomenon is less influent due to soft handover mechanisms.

It is interesting to observe as the performance of hybrid ARQ schemes quickly degradates as b increases. Simple ARQ, instead, is more robust in this case. This results is quite obvious because when errors occur in burst, all the redundancy PDUs in a TG are likely lost as well as the first K data packets, i.e., in such cases, redundancy PDUs can not be used for recovery purpose. Among hybrid schemes, A1 is the worst one, A2 is the best, whereas A3 performs in the middle.

Several solutions are possible to improve performance when channel errors occur in burst. First of all both K and N can be increased, by taking (for example) their ratio $\rho = K/N = 1 - H/N$ constant. In this way it is more probable that the burst will affect only a fraction of the TG and that the remaining packets are sufficient ($\geq K$) to correctly decode it. However,

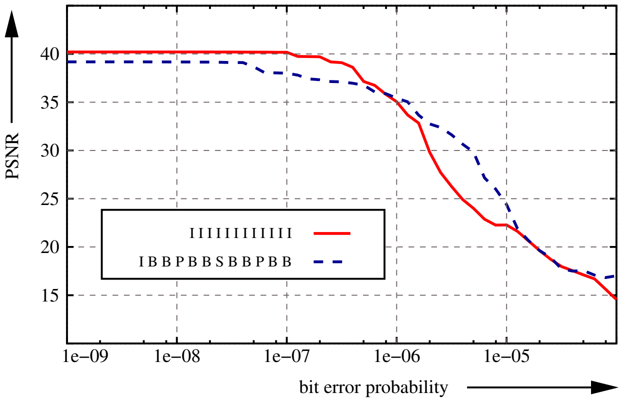

Fig. 12. Video quality versus bit error probability.

this way of proceeding improves the performance at the cost of a higher memory utilization at the packet-based decoder. Moreover, larger block sizes also implies a longer TG decoding time [10]. Another possible solution is to increase the PDU size by keeping constant K and N. The positive aspect of this solution is that the TG decoding speed increases [10], whereas the drawback is that the decoding process becomes more vulnerable to independent PDU errors.

Another possible technique to improve robustness against burst is to perform interleaving over several TGs before sending them over the channel. The drawbacks of this last method is the need for more memory and the presence of additional delays. From preliminary results, obtained by accurate simulation of the UMTS cellular system, this solution seems the more promising. The interleaving depth has to be carefully chosen based on the characteristics of the RLC PDU error process in order to use the lowest amount of memory while still achieving good error correction performance and acceptable delays.

VII. Conclusions

In this paper, we report some preliminary results and suggestions, which address the multicast data delivery problem in a UMTS system, where a multicast flow is transmitted over a common channel to different (multicast) users. Multicast user mobility is not considered, i.e., we do not address the problem of user handover between UMTS cells. Instead, both ARQ and packet-based FEC techniques are investigated considering both channel efficiency and delay performance. Hybrid ARQ techniques (HARQ) are considered and compared against simple ARQ. The following main conclusions can be drawn: 1.) Simple ARQ solutions do not suffice to obtain both good channel utilization and delivery delay performance as the number of multicast users increases. 2.) Hybrid ARQ techniques are more robust against an increasing number of multicast users. They provide both a high forward channel throughput and short delivery delays. However, packet-based FEC in order to be effective *MUST be properly dimensioned*, i.e., depending on the link layer error process observed at the receiver side. 3.) For this reason the accurate characterization of such a link layer process is a pivotal point for the correct design of hybrid ARQ techniques.

VIII. Outlook

In this paper we focused on the error-free transmission of the multicast content. As video codecs are able to deal with error prone payload, the bandwidth used can reduced furthermore by stopping the retransmission process in case a sufficient quality can be guaranteed at the mobile side. In [22] we have shown the vulnerability of H.26L video encoded bit streams in presence of bit errors. Figure 12 gives the picture signal noise ratio (PSNR) versus the bit error probability. In case the remaining bit error probability is below 10^{-6}, the video content can still be decoded with high quality. This knowledge can be used to disable retransmission if the number of requesting mobiles is low. Further studies, to characterize the common channel error process and the effectiveness of the RLC layer interleaving approach discussed in Section VI-B are on the way and are the topic for current and future research.

References

[1] Web site: `http://www.3gpp.org`

[2] C. Huitema, *"The case for packet level FEC,"* Proceedings of IFIP 5th International Workshop on Protocols for High Speed Networks (PsHSN 1996), INRIA, Sophia Antipolis, FRANCE, Chapman & Hall. October 1996.

[3] J. Nonnenmacher, E.W. Biersack, D. Towsley, *"Parity-based loss recovery for reliable multicast transmission,"* IEEE/ACM Transactions on Networking, vol. 6, no. 4, pp. 349-361. August 1998.

[4] J. Nonnenmacher, *"Reliable multicast transport to large groups,"* Ph. D. Thesis. Ecole Polytechnique Federale de Lausanne. 1998.

[5] A. J. McAuley, *"Reliable broadband communications Using a Burst Erasure Correcting Code,"* Proc. ACM SIGCOMM 90, pp. 287-306, Philadelphia, PA. September 1990.

[6] D. Rubenstein, J. Kurose, D. Towsley, *"Real-Time Reliable Multicast Using Proactive Forward Error Correction,"* IEEE NOSSDAV 1998, Cambridge, United Kingdom. July 1998.

[7] D. Rubenstein, J. Kurose, D. Towsley, *"Real-Time Reliable Multicast Using Proactive Forward Error Correction,"* CMPSCI Tech. Report 98-19, University of Massachusetts at Amherst. March 1998.

[8] R. Kumar, *"Design and implementation of a Proactive Real-Time Reliable Multicast Using Proactive Forward Error Correction (FEC),"* Master Thesis, University of Massachusetts at Amherst. May 1998.

[9] L. Rizzo, *"On the feasibility of software FEC,"* DEIT Technical Report LR-970131. Available as: `http://www.iet.unipi.it/ luigi/softfec.ps`

[10] L. Rizzo, *"Effective erasure codes for reliable computer communications protocols,"* Computer Communication Review, April 1997.

[11] J. W. Byers, M. Luby, M. Mitzenmacher, *"Accessing multiple mirror sites in parallel: Using tornado codes to speed up downloads,"* IEEE INFOCOM '99, pp. 275-283, New York, NY. March 1999.

[12] L. Rizzo, L. Vicisano, *"RMDP: an FEC-based Reliable Multicast protocol for Wireless environments,"* ACM Mobile Computer and Communication Review, vol.2, no.2. April 1998.

[13] L. Rizzo, L. Vicisano, *"A Reliable Multicast data Distribution Protocol based on software FEC techniques (RMDP),"* Proc. of the Fourth IEEE HPCS 1997 Workshop, Chalkidiki, Grece. June 1997.

[14] J. Gemmel, T. Montgomery, T. Speakman, N. Bhaskar, J. Crowcroft, *"The PGM Reliable Multicast Protocol,"* IEEE Network, January/February 2003.

[15] J. Laiho, A. Wacker, T. Novosad, *"Radio Network Planning and Optimization for UMTS,"* John Wiley & Sons, Ldt. October 2001.

[16] H. Holma, A. Toskala, *"WCDMA for UMTS: Radio Access for Third Generation Mobile Communications,"* John Wiley & Sons, Ldt. 2000.

[17] Bernd Girod, Jacob Chakareski, Mark Kalman, Yi J. Liang, Eric Setton, Rui Zhang, *"Advances in Network-adaptive Video Streaming,"* Wireless Communications and Mobile Computing, vol. 2, no. 6, pp. 549-552. September 2002.

[18] U. Horn, K. Stuhlmuller, M. Link and B. Girod, *"Robust Internet Video Transmission Based on Scalable Coding and Unequal Error Protection,"* Image Communications, Special Issue on Real-time Video over the Internet, pp. 77-94, vol. 15. September 1999.

[19] T. Stockhammer, H. Jenkac, C. Weiss, *"Feedback and error protection strategies for wireless progressive video transmission,"* IEEE Transactions on Circuits and Systems for Video Technology, vol. 12, no. 6, pp. 465-482. June 2002.

[20] 3GPP TS 25.322, *"Third Generation Partnership Project: Technical Specification Group Radio Acess Network; Radio Link Control (RLC) Specification (Release 5)."*

[21] Web site: `http://www.acticom.info`

[22] F. H. P. Fitzek, P. Seeling, M. Reisslein, M. Rossi and M. Zorzi, *"Investigation of the GoP Structure for H.26L Video Streams,"* Technical Report: acticom–02–004, acticom – mobile networks. December 2002.