Investigation of Link Layer Algorithms and Play–Out Buffer Requirements for Efficient Multicast Services in 3G Cellular Systems

Michele Rossi⋆, Michele Zorzi⋆, Frank H.P. Fitzek†
⋆Department of Engineering, University of Ferrara
Via Saragat 1, I-44100 Ferrara, Italy, e-mail: *[mrossi|mzorzi]@ing.unife.it*
†Department of Communications Technology, Aalborg University
Neils Jernes Vej 12, 9220 Aalborg Øst, Denmark, e-mail: *ff@kom.aau.dk*

***Abstract*— The success of 3G networks depends on the possibility to attract customers to this new technology. Therefore, new services using the spectrum in an efficient manner while satisfying the customers are needed. In this paper we introduce link layer algorithms that are suitable for multicast transmission in 3G cellular systems and we present their impact on video application in terms of play–out buffer requirements. By our approach we show that Hybrid ARQ solutions can be succesfully employed to perform error control in the multicast transmission case. Using these schemes, performance can be increased thereby increasing system capacity and lowering the cost per served user. In the final part of the paper, these solutions are extended for the important multicast video streaming case, where new schemes are devised to avoid the throughput inefficiencies of fully reliable error recovery agorithms.**

I. Introduction

For a financial success of the 3G cellular systems several conditions have to be fulfilled. First of all the highly paid spectrum has to be used in a very efficient way to maximize the possible profit by accepting as many customers in the system as possible. On the other side, to make 3G attractive to the customers new sophisticated services such as video/audio streaming have to be introduced. Furthermore, the price of the mobile has to be low.

To solve the contradiction between bandwidth demanding services such as video and the request for efficient usage of the spectrum, 3GPP [1] introduced multicast/broadcast channels referred to as common channels. Over this type of channels, users with the same interest in a specific content will be served using the same spectrum. As the customers have different channel characteristics, design of link layer (LL) algorithms for error recovery becomes a very challenging task. In case we use well known forward error correction (FEC), it conflicts with the need for efficient usage of the spectrum as FEC is always designed for the worst case scenario. In case Automatic Repeat reQuest (ARQ) mechanisms are used, the delay for data transmission may become large. Streaming services have been identified as the most promising over multicast channels. For such services, a so called play–out buffer is used within the users' terminals to balance the variations in the data transmission. For large delays (as we have to expect in the ARQ case) huge play–out buffers are needed, which, in turn, make the terminal very expensive as long play–starts are the result.

Therefore, in this paper we propose and validate an efficient algorithm for the delivery of multicast flows over the common channel of 3G W-CDMA cellular systems. To investigate the performance of our proposed link layer algorithms, a system simulator has been developed obtaining accurate W-CDMA common channel error traces. In Section III, the error recovery algorithm is presented in detail, whereas its performance evaluation is carried out in Section IV. In this section, further modifications of the proposed scheme are devised to limit some transmission inefficiencies in the video streaming transmission case. Section V concludes the paper.

II. System Under Investigation

We consider here a 3G cellular system, where W-CDMA is used as the radio interface. Moreover, as suggested by [1] we consider that Common CHannels (CCHs) are allocated for the local transmission (at every base station) of multicast data. Obviously, this choice is dictated by the need for an efficient utilization of the channel resources, as motivated above. In order to derive accurate channel traces for this system a W-CDMA cellular system simulator has been developed. The reference scenario together with some details about the simulator are reported in the following.

The service area is composed by $N_c = 9$ hexagonal cells, where a base station is placed at the center of each cell and a given number of users are mobile within the coverage area. Propagation phenomena are modeled through standard techniques, by considering log-normal slow fading, fast fading and path loss [2]. A simple power control algorithm has been implemented following the basic algorithm which can also be found in [3], i.e., the downlink transmitted power is dinamically varied by a constant multiplicative increase/decrease factor ($\Delta = 0.5$ dB) to track a target SIR value (SIR_{th}). For what concerns channel coding and interleaving, we consider here a convolutional half rate Viterbi decoder operating over an interleaving interval $TTI = 80$ ms.

A first set of users $N_{DCH} \in \{200, 300\}$ communicate through Dedicated CHannels (DCHs) whose bit rate is 30 Kbps (i.e., physical Spreading Factor of $SF_{DCH} = 128$). These users are placed randomly at the beginning of the simulation and move following a pseudo-linear mobility model. The power control procedure is dynamically executed for each user as explained above. This first set of DCH users is regarded here as system interference since our main focus is on the common channel multicast transmission.

A second set of users $N_{CCH} \in \{180, 450, 900\}$ is receiving the multicast flow through a Common downlink CHannel

(CCH) whose bit-rate and spreading factor are $B_r = 120$ Kbps and $SF_{CCH} = 16$, respectively. These users can also be on the move, but their serving base stations remain unchanged. Let us better clarify this point. CCH users are randomly placed at the beginning of the simulation, shadowing and path loss are chosen according to the log-normal and the exponential model, respectively, and are kept constant for the whole simulation time. Subsequently, in order to emulate some degree of mobility, their Doppler frequency f_d is selected, but without changing their spatial coordinates. In this way, we are able to control their fast fading as if they were on the move but without reflecting it into a change of their spatial positions. Therefore, it is possible to investigate multicast delivery algorithms (our main focus here) disregarding the multicast handover management that, by itself, constitutes a problem to be properly handled. The common channel power has been fixed to the constant value $P_{CCH} = P_{max}^{downlink}$.

III. Link Layer Algorithms for Multicast Streaming

When the multicast flow is transmitted by means of a common channel, different users are in general characterized by independent channel error processes. This is very important since it can make traditional link layer retransmission algorithms inefficient as the multicast group size (N_u, where N_u in our scenario is given by $N_u = N_{CCH}/N_c$) increases.

For illustration purpose, a fully reliable multicast service is considered at this point. In that case, a packet needs to be retransmitted if at least one user has not correctly received it. Moreover, an erroneous packet needs to be retransmitted until all the users in the multicast group have received it correctly. However, in this case we have that as N_u increases the probability that at least one user needs a retransmission at a given time increases as well. Therefore, the forward link throughput is heavily degraded by retransmissions, while the available bandwidth for new transmissions becomes very low.

As N_u increases simple ARQ (Automatic Retransmission reQuest) may not be sufficient to cope with this problem, because it simply retransmits the lost packets whenever a retransmission request (Not Acknowledgment, NACK message) arrives at the base station, but it does not account for the possibly different error processes that, at every receiver, are affecting the transmitted packet. To cope with this problem, we propose to exploit packet-based FEC techniques directly at the link layer. A rich literature can be found on the topic [4][5][6][7].

By this approach, some error recovery is performed directly at the receiver side by pro-actively adding some redundancy into the multicast flow. This redundancy can be independently exploited, at each receiver, to recover from losses. If local recovery succeeds, no retransmissions are required and the forward bandwidth can be utilized to allocate new packet transmissions. In more detail, the multicast flow at the link layer of the sending base station is divided in transmission groups (TG) of K packets each. Then, each group is passed to a packet–based encoder where H redundancy packets are generated for every TG. Finally, the whole FEC block composed by $K + H$ packets is sent over the channel. At the receiver side, every user can exploit the redundancy packets by recovering up to H erroneous or lost packets, in any order. Different coding schemes can be used for this purpose, Reed–Solomon and Tornado deserve a particular attention [8], [9]. However, in this work we are mainly interested in discussing the effectiveness of these coding strategies without going into code implementation details. Moreover, in the following, we assume that, in the first encoding phase, it is possible to generate a large (but finite) number of redundancy packets for each FEC block. These packets will be transmitted on-demand as incremental redundancy during the error recovery algorithm. The main benefits of this approach are

- *Improved transmission efficiency*: A single parity packet can be used to repair the loss of any packet in the TG, i.e., it 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 only needs to know the maximum number of lost packets by any receiver but not their sequence number. So, the feedback is reduced from per–packet feedback to per–TG feedback.
- *Improved feedback channel performance*: Thanks to the pro-actively added redundancy, some error recovery is possible at the user terminals, without the need for retransmissions. As a consequence, the number of ACK messages that the users are sending back to the base station can be considerably decreased. This beneficial effect has been largely investigated in previous work (see [10] as an example), where techniques to limit the ACK collision problem have been considered. These results and algorithms still apply in our scenario as well. Results on ACK collision avoidance will not be explicitly considered in this contribution, where the emphasis is mainly put on the proposal and the presentation of schemes for the multicast delivery in 3G networks.

Regarding the last item above we observe that, as the number of multicast users increases the number of generated NACKs could be large (e.g., when a large number of users is experiencing bad channel conditions). In these cases, the feedback channel performance is degraded and some countermeasures have to be taken to avoid or limit this fact. Later on, in Section IV-B two algorithms will be presented to cope with this problem.

The hybrid ARQ (HARQ) algorithm that will be considered in this paper is presented in the following.

HARQ algorithm: At the sender side the TG of N PDUs (K data packets plus H redundancy PDUs) is sent first. Each receiver checks for errors in each TG and replies accordingly. In this scheme incremental redundancy is used. To better explain how the algorithm works suppose that, in addition to the N PDUs in the first transmission, R redundancy PDUs have already been sent over the channel for a given TG. In this case, each receiver checks for the number of correctly received PDUs (N_{ok}) among the $N + R$ PDUs sent. Let us refer to a given user $i \in \mathcal{N}$, where $\mathcal{N}$ is the set of multicast users in the cell. If $N_{ok}(i) \geq K$ the original K PDUs can be obtained and the TG is correctly received by user i. Otherwise, if $N_{ok}(i) < K$, user i 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 at that terminal. 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 CCH channel. This procedure is repeated until all users in $\mathcal{N}$ are able to correctly decode the K data PDUs, i.e., when $N_{ok}(i) \geq K \ \forall i \in \mathcal{N}$. This 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 receivers 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. This hybrid scheme and alternatives are described in more detail in [11], where this algorithm has been found to be the best candidate among the considered ones. The aim of this paper is to go further with respect to the results in [11] (obtained considering independent error processes at every user), by testing the algorithm over accurate W-CDMA channel traces, proposing new approaches to increase the effectiveness of FEC and to cope with some throughput inefficiencies in the video streaming case.

IV. Results concerning Buffer Requirements and Channel Efficiency

In this Section, the performance of the algorithm proposed above are addressed in detail considering accurate 3G W-CDMA channel traces. In this case, channel errors tend to be correlated and, as a consequence, the proposed FEC solutions may become less effective. In order to cope with this problem or, at least, to limit its impact on the overall performance, we introduce further interleaving directly at the link layer (Section IV-A). Essentially, we try to break the error bursts, by spreading them over several different FEC blocks. This will improve the coding efficiency at the cost of some additional complexity. The interleaving procedure is illustrated in the next section, whereas the performance evaluation is reported later on in Sections IV-B and IV-C for the fully reliable and streaming video cases, respectively.

A. Link Layer Packet level interleaving

As highlighted in previous research [5], packet-based FEC techniques are very effective to offer low residual PDU error rates to multiple users receiving the same data from a common transmission channel. However, the effectiveness of such techniques decreases as the link layer error burstiness increases. When error bursts are too long, the added redundancy is likely lost and it is useless in recovering from errors. In this case, the redundancy only wastes the available channel resources. To overcome this fact, we apply a matrix interleaving on the TG flow prior to its transmission over the channel. Let us refer to the interleaving buffer size (expressed here in number of LL PDUs) at the LL as B. Then, as reported in Fig. 1, PDUs are first disposed in a $I \times N$ matrix[1], where $I = B/N$ is the interleaving depth. Thereafter, link layer PDUs are sent

[1]Matrix indexes are expressed in units of PDUs.

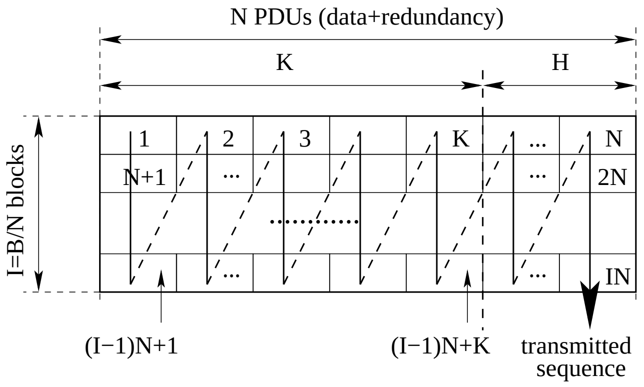

Fig. 1. Link layer interleaving on the link layer TG flow.

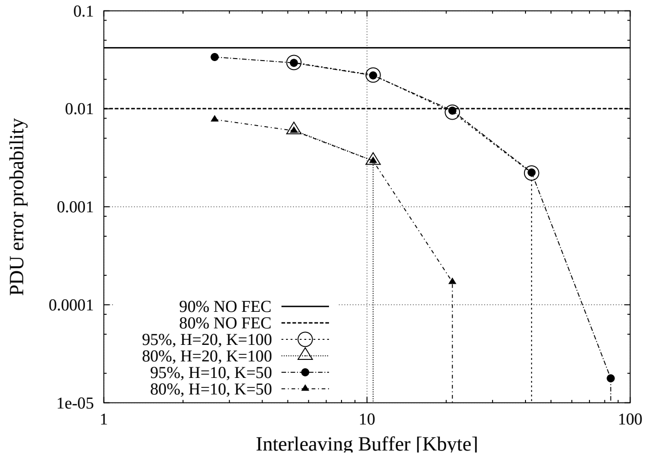

Fig. 2. Residual link layer packet error rate: effectiveness of the interleaving approach on the link layer packet flow as a function of the interleaving buffer size (B).

reading the matrix by columns, i.e., the transmitted sequence will be: $\{1, N+1, 2N+1, \ldots, (I-1)N+1, 2, 2N+2, (I-1)N+2, \ldots, N, 2N, \ldots, IN\}$. Note that all PDUs belonging to the same block are transmitted over the channel spaced by $I-1$ packets.

As an example, in Fig. 2, the LL packet residual error rate is reported for the 80-th and the 90-th percentile of the CCH users as a function of the interleaving buffer size. The graph has been obtained considering a first set of DCH users ($N_{DCH} = 200$) that are on the move as explained above, while CCH ($N_{CCH} = 1000$) users are static[2]. It is worth noting that, if the interleaving buffer is large enough, the FEC can completely avoid losses in 80% ($B = 10$ Kbyte) and 90% ($B \approx 30$ Kbyte) of the cases.

B. Throughput and Delay Performance of a Fully Reliable Multicast Service

In the results discussed in the following sections, we consider a LL logical bit rate of $B_r = 120$ Kbps ($SF_{CCH} = 16$) a LL round trip time (RTT) of 220 ms (this is a typical maximum value for the LL RTT in a 3G network [1] and it is due to the

[2]A Doppler frequency of $f_d = 2$ Hz has been considered in such a case. Due to the low f_d value, long LL bursts are experienced by CCH users.

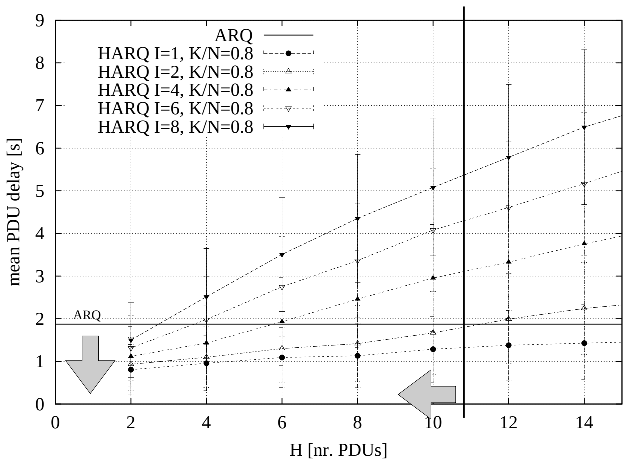

Fig. 3. Mean link layer packet delay as a function of the number of FEC redundancy packets (H).

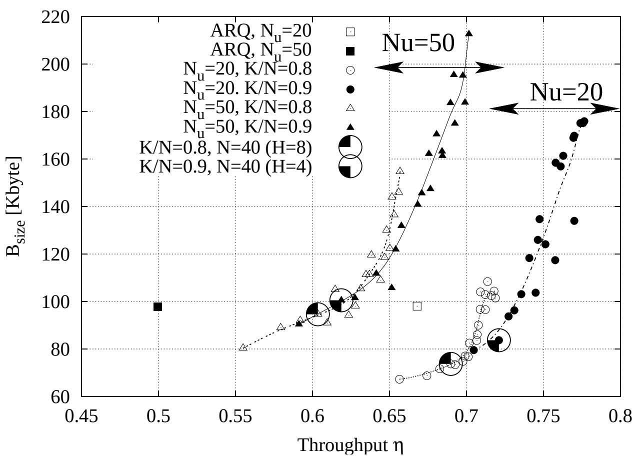

Fig. 4. Tradeoff between throughput (η) and play–out buffer requirements (B_{size}).

large interleaving depth of $TTI = 80$ ms) and a LL PDU length of 360 bits. With these values, we have that about 77 PDUs are transmitted in a LL RTT. Moreover, with the term ARQ, we refer here to the standard Selective Repeat ARQ algorithm, where a retransmission for a packet is scheduled if at least one user in the multicast set is requiring for its retransmission. The 3G system simulator has been configured as explained in Section II considering $N_{DCH} = 200$ and $N_{CCH} \in \{180, 450, 900\}$. By its execution, a set of CCH channel traces have been obtained for every user in the system. Further, these traces have been used (off-line) as the input for a HARQ simulator, from which performance measures have been derived. In the first part of this section performance results will be given for a fully reliable service, i.e., erroneous data is always retransmitted. Later on (Section IV-C), the focus will be put on the video streaming case and this assumption will be relaxed.

As a first set of results, we focus on the channel efficiency (η). In Table I, η values are reported considering $N_u \in \{20, 50, 100\}$, $f_d = 40$ Hz, $H = 8$, $I = 2$ for two K/N values ($K/N \in \{0.8, 0.9\}$). In the same table, the throughput for plain ARQ and the throughput gains (HARQ *Vs* ARQ) are also reported. The advantage offered by HARQ solutions becomes clear as N_u increases.

N_u	SCHEMES →	ARQ	HARQ(K/N=0.8)	HARQ(K/N=0.9)
20	η	0.667	0.69	0.73
20	η Gain [Kbps]	——	2.8 Kbps	7.6Kbps
50	η	0.5	0.6	0.64
50	η Gain [Kbps]	——	12 Kbps	16.8 Kbps
100	η	0.374	0.5	0.54
100	η Gain [Kbps]	——	15.2 Kbps	19.92 Kbps

TABLE I
HARQ THROUGHPUT GAINS CONSIDERING $I = 2$, $H = 8$, $f_d = 40$ HZ AND $B_r = 120$ KBPS.

The motivation behind the choice of the parameters $I = 2$ and $H = 8$ is illustrated in Fig. 3, where the mean LL packet delivery delay is reported. From this figure it is clear that, in addition to the good achievable throughput performance, the selected case is also a good compromise for the delay, which is still of the same order of magnitude as the one introduced by Selective Repeat ARQ. Note that as I increases, the delivery delay also increases since, due to the interleaving process, a longer time has to be waited for to send new FEC blocks. This long delay also impacts the outstanding retransmissions since the packets contained in the matrix are always sent in a row.

In the sequel we look at the play-out buffer requirements when a video streaming flow is being transmitted over the CCH channel. The trade-off between buffer requirements and channel efficiency is reported in Fig. 4. What is referred here as *maximum buffer size* (B_{size}) is the buffer dimension which is needed to avoid that *buffer starvation occurs* at the application play-out buffer. To track the play-out buffer occupancy during the simulation, we consider a variable input flow λ that corresponds to the link layer outgoing flow (considering *in-order* delivery of LL packets), whereas we account for a constant output flow μ which is set at the value $\mu = B_r \times \eta$. The evaluation of the required buffer size B_{size} is therefore executed off-line, after finding η. As can be clearly observed in Fig. 4, by tuning the coding redundancy H, it is possible to trade channel efficiency for buffer requirements (B_{size}). The curves in Fig. 4 have a first part where the slope $\mathcal{S} = \mathrm{d}B_{size}/\mathrm{d}\eta$ is still limited, and a second part, in which $\mathcal{S}$ suddenly increases. Obviously, this second part should be avoided, since only marginal throughput advantages can be achieved at the expense of a substantial increase of B_{size}. Moreover, for illustration purpose, two points have been marked in this figure ($N = 40, H = 8$ and $N = 40, H = 4$) to highlight how the same choice for the FEC block parameters translates in terms of B_{size} and η for different N_u values.

In order to gain some insights in the low mobility scenario ($f_d = 2$ Hz), in Fig. 5 we plot the FEC error correction probability by varying H, f_d and I. The FEC error correction probability is defined here as the probability that the pro–actively added redundancy PDUs can successfully correct the errors occurring during the first transmission of a FEC block. In such a case we do not need further packets to be retransmitted. It is clear that the added redundancy is effective and that its convenience is higher over heavily correlated channels ($f_d = 2$ Hz). Over such channels, also the interleaving depth has a positive impact. The throughput/buffer relationships, in

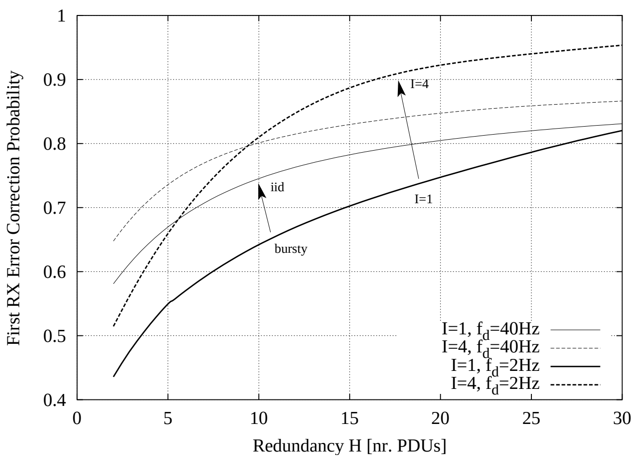

Fig. 5. Link layer FEC error correction probability as a function of H.

such a case, are very similar to what reported in Fig. 4.

C. Efficient Delivery of Multicast Video Flows

As clearly highlighted by the results in the previous section, when a fully reliable service is provided, η is decreasing with the number of multicast users in the system N_u (Table I). Moreover, the throughput also shows a strong dependence on the system interference (in our case on N_{DCH}). For instance, by adding 100 DCH users in the system ($N_{DCH} = 300$) and considering the parameters as in Table I with $N_u = 50$, we have that $\eta \to 0.32$, i.e., it has almost halved with respect to the case where $N_{CCH} = 200$. This is to reflect that, due to the increased interference, more CCH users experience bad channel conditions and frequently ask for retransmissions. As a consequence, they heavily impact the forward channel throughput, since in the recovery algorithm every retransmission request is satisfied. In some cases this behavior should be avoided. Consider for example the transmission of a *video streaming flow*. In that case some residual errors can be tolerated (depending on the minimum acceptable video quality), but the flow has to be transmitted in a timely manner and respecting the constraints imposed by the users play-out buffers (to avoid buffer starvation). In such a scenario, it is unacceptable to have a small portion of the users congesting the forward channel transmission, since they will waste the system resources (common channel bandwidth) leading to a poor video quality for the remaining users in the system. Instead, it would be better to keep the transmitted flow to a reasonable throughput value, by accepting some degradation especially at the users experiencing a bad channel state. In such a situation the best policy is no longer to retransmit the lost packets for every user, instead, it would be better to control the number of retransmissions to maintain the forward channel throughput to an acceptable value.

As a first solution to this problem, one could use the simple but effective strategy given in the following. A limit on the number of satisfied retransmission requests is imposed. In particular, after the first reception of a FEC block, every user communicates its reception status to the base station, by sending a NACK message when the FEC block is undecodable. At the sender side, the incoming NACKs are collected and the retransmission process for the corresponding block is initiated only if the number of retransmission requests is large enough, i.e., if the number of received NACKs for that block exceeds a given threshold τ (that can be expressed as a percentage of the number of multicast users in the cell, N_u). However, even if this mechanism is very simple, it presents the following major drawbacks

- It does not limit the number of NACKs flowing on the backward channel. In this case, in fact, the sender still needs to acquire the NACK messages from every multicast user in the cell to evaluate the percentage of incoming requests (τ). As a consequence, error prone channels will lead to a highly loaded uplink channel which in turn causes non trivial problems such as collisions and the need for uplink management procedures.
- The retransmission process can be denied for those users with good channel condition. Such users rarely ask for retransmissions, but their requests could be dropped if τ is large enough. In these cases, the system is unfair, since a *good user*, which is rarely asking for the channel resource is penalized in the same manner as the *bad users* that continuously ask for retransmissions.

To solve these drawbacks, we devise a second solution that is based on a probabilistic approach rather than on the selection of the hard threshold τ. The task of the base station controller is to keep the forward channel efficiency to a reasonable level η^*. In order to maintain such level, some retransmission requests have to be denied, as above. Let us define an overall retransmission acceptance probability P as the probability that the retransmission process is initiated for a given FEC block. This probability comes from the superposition of the acceptance probabilities at every user. Let us better explain this point. The generic user i fails to decode a FEC block with probability p_{ei} and subsequently decides whether or not a retransmission for that block should be initiated with probability p_{ri}. Then, user i sends a NACK for the generic FEC block with probability $p_i = p_{ei} \times p_{ri}$. In some sense, we are adding a probabilistic filter after the decoding process in order to decide whether the retransmission for an undecodable block has to be requested. It is important to observe that the mechanism operates directly at the user terminal, by denying (in a probabilistic manner) the transmission of NACK messages. This is very useful in order to obtain a simple and distributed algorithm and to limit the number of NACKs flowing over the backward channel. At the sender side, the retransmission for a block is initiated if at least one NACK is received for that block, i.e., using the HARQ algorithm presented in the previous sections. The modifications are performed at the receiver side only adding the probabilistic decision in the NACKs sending process. Hence, the overall retransmission probability can be obtained as

$$P = 1 - \prod_{i=1}^{N_u} (1 - p_i) \qquad (1)$$

where p_i is the block retransmission request probability for user i. Let us pose the following constraint: $p_i = p^* \ \forall i \in \{1, 2, \ldots, N_u\}$, which means that every user has the same probability to send a NACK over the backward channel, i.e., the same resource request capability is assigned to every user.

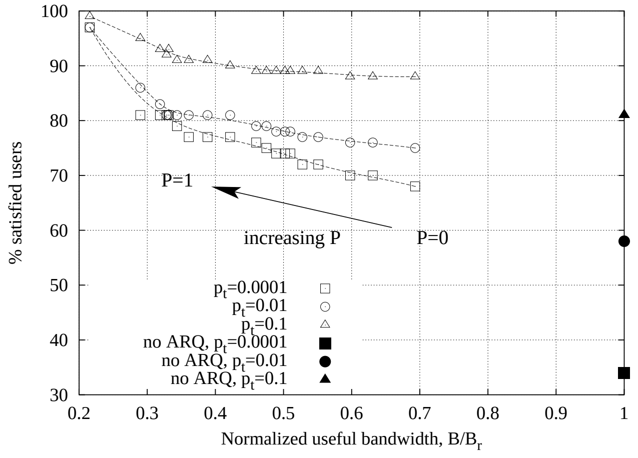

Fig. 6. Percentage of satisfied users as a function of the normalized useful bandwidth.

Thanks to this assumption p_{ri} can be found as

$$p_{ri} = \min\left(1, \frac{p^*}{p_{ei}}\right) \tag{2}$$

where

$$p^* = 1 - (1 - P)^{1/N_u} \tag{3}$$

To run the algorithm the sending base station only needs to communicate P and N_u to every multicast user in the cell. The users will then independently estimate p_{ei} and obtain p_{ri} according to Eq. (2). The target channel efficiency η^* can then be selected by appropriately tuning the P parameter. The algorithm has the following advantages

- It can be simply implemented at every multicast user (in a distributed manner).
- Only two global parameters (P and N_u) are needed, at every user, to run the algorithm. These parameters can be disseminated through an initial setup phase and subsequently updated by the base station controller according to various needs such as a change in N_u and/or in η^*.
- Since NACKs are inhibited directly at their generation point, i.e., at the users' terminals, the mechanism can highly reduce the backward channel utilization.

In the following we report some performance results regarding these mechanisms. Here we say that a user is satisfied if its packet error probability is less than a given threshold p_l. For instance, this threshold could be easily related to a minimum video quality. In Fig. 6, we report the percentage of satisfied users as a function of the normalized useful bandwidth B/B_r for the second scheme above, by varying the parameter P. The case where no ARQ is considered at the link layer is labeled as "no ARQ". When $P = 1$, all the retransmission requests are satisfied with a consequent throughput degradation (this is just the fully reliable HARQ scheme presented in Section III). On the other hand, as $P \to 0$ some retransmission requests can be controlled according to the distributed algorithm presented above. In this case, the useful bandwidth, i.e., the bandwidth available to transmit new data is increased at the expense of some residual errors. The bad users are the most penalized, since these users are the ones introducing the highest degradation to the common channel resource. Hence, using the probabilistic scheme above, the useful bandwidth can be adaptively selected by means of the parameter P and an appropriate trade–off (bandwidth/throughput) can be chosen. This can be useful as new interference is added to the system or when users are experiencing bad channel conditions. The aim of the approach is to keep the useful bandwidth to a reasonable level to avoid the buffer starvation at the good users' terminals.

We have verified that the performance of the first scheme above (fixed threshold τ) is always worse than the one given by the probabilistic approach. As an example, for p_l up to 0.001 the number of satisfied users in the second approach is increased of 15 %, whereas it is increased of 10 % for $p_l = 0.001$.

V. Conclusions

In this paper error control algorithms for multicast data transmission in 3G cellular systems have been proposed. Their performance has been tested by simulation over accurate W-CDMA channel traces. A fully reliable service has been investigated first. Hybrid ARQ has been found to significantly outperform plain ARQ. Throughout the paper, particular attention has been paid to the multicast video streaming case, by giving quantitative measurements for the play–out buffer requirements and the involved trade–offs. Finally, some schemes have been devised for unreliable multicast cases (e.g., video streaming), where some residual errors can be tolerated (e.g., according to a minimum video quality). These schemes are used to control the forward channel useful bandwidth. Further studies on these on–line algorithms, used to handle retransmission requests while controlling the video quality at each user, are the main focus of our future research activity.

References

[1] "3GPP Third Generation Partnership Project." [Online]. Available: http://www.3gpp.org

[2] M. Zorzi, M. Rossi, and G. Mazzini, "Throughput and energy performance of TCP on a wideband CDMA air interface," *Wiley Wireless Commun. and Mobile Computing (WCMC)*, vol. 2, no. 1, pp. 71–84, Feb. 2002.

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

[4] C. Huitema, "The case for packet level FEC," in *International Workshop on Protocols for High Speed Networks (PsHSN 1996)*, INRIA, Sophia Antipolis, France, October 1996, pp. 109–120.

[5] J. Nonnenmacher, E. Biersack, and D. Towsley, "Parity-based loss recovery for reliable multicast transmission," *IEEE/ACM Trans. Networking*, vol. 6, no. 4, pp. 349–361, Aug. 1998.

[6] A. J. McAuley, "Reliable Broadband Communications Using a Burst Erasure Correcting Code," in *Proceedings of ACM SIGCOMM 1990*, Philadelphia, PA, US, Sept. 1990, pp. 287–306.

[7] J. Nonnemacher, "Reliable Multicast Transport to Large Groups," Ph.D Thesis, Ecole Polytechnique Federale de Lausanne, France, 1998.

[8] L. Rizzo, "Effective Erasure Codes for Reliable Computer Communications Protocols," *ACM Computer Communication Review*, vol. 27, no. 2, pp. 24–36, Apr. 1997.

[9] J. W. Byers, M. Luby, and M. Mitzenmacher, "Accessing multiple mirror sites in parallel: Using Tornado Codes to speed up downloads," in *Proceedings of IEEE INFOCOM*, New York, US, Mar. 1999, pp. 275–283.

[10] D. D. Lucia and K. Obraczka, "Multicast feedback suppression using representatives," in *Proceedings of IEEE INFOCOM*, Kobe, Japan, Sept. 1997, pp. 463–470.

[11] F. Fitzek, M. Rossi, and M. Zorzi, "Error Control Techniques for Efficient Multicast Streaming in UMTS Networks," in *Proceedings of SCI 2003*, Orlando, Florida, US, July 2003.