3GPP TSG RAN WG1#11 Tdoc R1-00-0254
February 29th - March 3rd, San Diego, USA

Agenda Item: Adhoc4, Rate Matching

Source: Siemens

Title: Improved End Puncturing Scheme for Convolutional Codes

Document for: Decision

---

1. Introduction

For small block sizes different companies pointed out different methods to improve the channel coding scheme. The aim of this proposal now is to present a generally applicable method for short and long blocks which can be very easily implemented as an option immediately after convolutional coding (UL and DL). The additional complexity is very low. The simulations for AMR 4.75 kbps and AMR 7.40 kbps show an improvement between 0.1 and 0.2 dB.

In previous contributions a puncturing scheme for convolutionally coded transport channels has been presented [3]. Following comments received on this proposal we have further optimised it and particularly reduced the complexity. Now we present a variant with negligible complexity increase (merely an offset in the indexing) which still achieves about the same performance improvement. Further more we show the impact of such a scheme on the total capacity for selected realistic transport format combinations.

2. Principle

As mentioned in [2] there is an imbalance in the protection of information bits after channel coding, because of the pre-known start and ending state of the convolutional coder at the beginning and the end of a code block. Characteristic of that imbalance is that the bits near the termination have a lower BER than the bits in between. This effect means that bits transmitted near the beginning or the ending of a code block are protected better against errors than others. But for most applications the bits should be transmitted nearly equally protected (except of course if they are transmitted via different transport channels). To avoid this waste of energy, to achieve a better overall performance and to keep the additional complexity increase low, we propose an improved method by puncturing a fixed number of consecutive bits at both ends of a block. For rate 1/3 codes always the first and last 8 consecutive bits are punctured on both ends and for rate 1/2 the first and last 6 consecutive bits are punctured. That means 16 respectively 12 coded bits are punctured in total. In [3] we have already shown that by pre-puncturing with so called “end puncturing patterns” a gain in performance of about 0.3 dB can be achieved. In this paper we present simulations for the AMR transport format and we achieve a gain in performance of 0.2 dB for the AMR 4.75 kbps. This gain is smaller (0.1 dB) for AMR 7.40 kbps, because of the higher blocksize.

3. Simulation

We present simulations about the performance of the AMR 4.75 kbps codec mode for AWGN and a mutlipath fading channel and we show BER curves for the 7.40 kbps mode (AWGN only). The results with the current multiplexing and channel coding scheme are compared with the new proposal. The difference between the two methods as mentioned above is the additional use of the simplified tail puncturing method. In the following table the simulation assumptions are shown:

AWGN-Simulation	Mutlipath fading channel
• Ideal phase estimation • Convolutional Coding • No overhead for pilot or PC bits considered	• Rake Receiver • Real channel estimation • SF 256, PC (5% Error) • Channel Model Vehicular A • Speed 30 km/h

Table 1: *Link level simulation assumptions*

As described in [1] the AMR 4.75 kbps speech codec delivers 42 Class A and 53 Class B bits and the 7.40 kbps codec mode produces 61 Class A bits and 87 Class B bits per 20 ms. Both modes fit into a SF 256 slot structure. For the calculation of the bit allocation we used exemplary DL slot format 2 with 2 pilot symbols, rate 1/3 coding and no TFCI. In the 4.75 kbps case we used three transport channels: one for Class A, one for Class B and one for a 2 kbps information rate dedicated signalling channel. The physical channel partitioning respectively the semistatic rate matching attributes were chosen so that Class A target BER is 0.01% and Class B target BER is 0.03% [4]. As currently understood by codec experts this meets the requirements for AMR. Table 2 shows the bit allocation for the AWGN simulation. As the bits of Class B have a higher target BER they are protected with a higher code rate needing a lower number of gross bits.

	Class A	Class B
Speech Bits delivered per 20 ms	42 + 8 (CRC)	53
Addition of Tail Bits and Convolutional Coding R=1/3	174	183
Semistatic Ratematching Parameter	1.0	0.94
Rate Matching	174 → 158	183 → 156

	Dedicated Control Channel
Bits delivered for channel coding per 40 ms	112
Addition of Tail Bits and Convolutional Coding R=1/3	360
Semistatic Ratematching Parameter	1.0
Rate Matching	360 → 332

Table 2: *Bit allocation for the AMR 4.75 kbps AWGN simulation*

Based on the different behaviour of the BER curves in the AWGN and multipath fading simulation we have now a different physical channel partitioning (different semistatic rate matching parameters) to meet the AMR requirements as described above. It can be seen that the difference between the static rate matching attributes increases. The reason for this is that in fading environment it is necessary to spend more energy to obtain the required BER improvement of class A bits versus class B bits.

	Class A	Class B
Speech Bits delivered per 20 ms	42 + 8 (CRC)	53
Addition of Tail Bits and Convolutional Coding R=1/3	174	183
Semistatic Ratematching Parameter	1.0	0.85
Rate Matching	174 → 162	183 → 148

	Dedicated Control Channel
Bits delivered for channel coding per 40 ms	112
Addition of Tail Bits and Convolutional Coding R=1/3	360
Semistatic Ratematching Parameter	1.0
Rate Matching	360 → 340

Table 3: *Bit allocation for the AMR 4.75 kbps multipath fading simulation*

The following results from the AMR 4.75 kbps simulations.

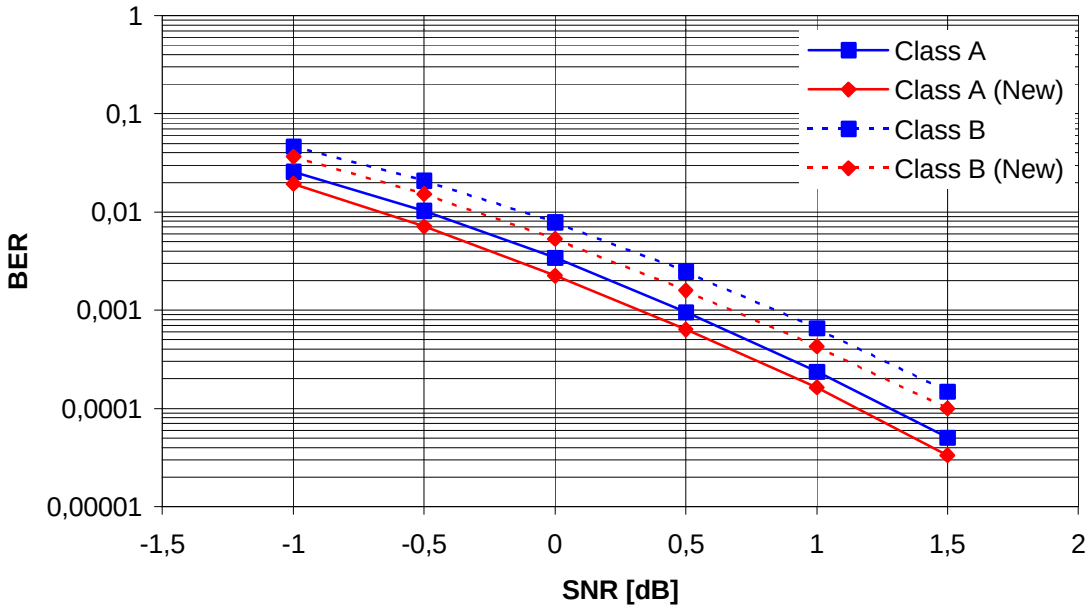

Figure 1: *Performance of normal puncturing and end puncturing for AMR 4.75 kbps (AWGN)*

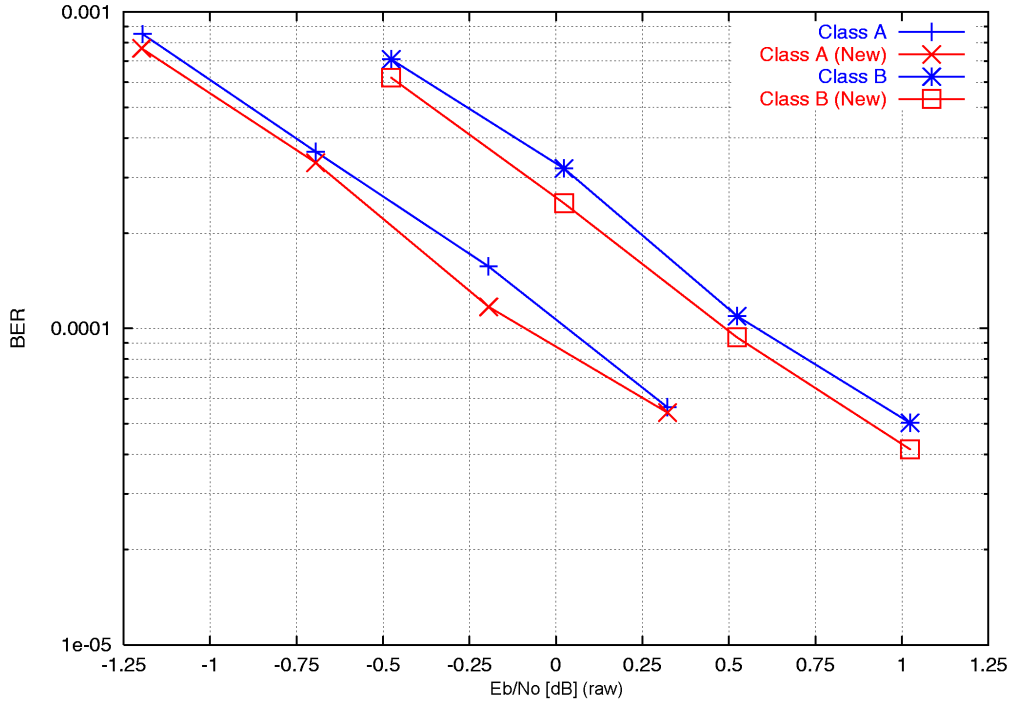

Figure 2: *Performance of normal puncturing and end puncturing for AMR 4.75 kbps (Fading)*

Table 4 shows the bit allocation for the AMR 7.40 kbps AWGN simulation.

	Class A	Class B
Speech Bits delivered per 20 ms	61 + 8 (CRC)	87
Addition of Tail Bits and Convolutional Coding R=1/3	231	183
Semistatic Ratematching Parameter	1.0	0.94
Rate Matching	231 → 222	285 → 258

Table 4: *Bit allocation for the AMR 7.40 kbps AWGN simulation*

Figure 3 shows the performance for AMR 7.40 kbps. The improvement is smaller because of the larger blocksizes, but it is still substantial.

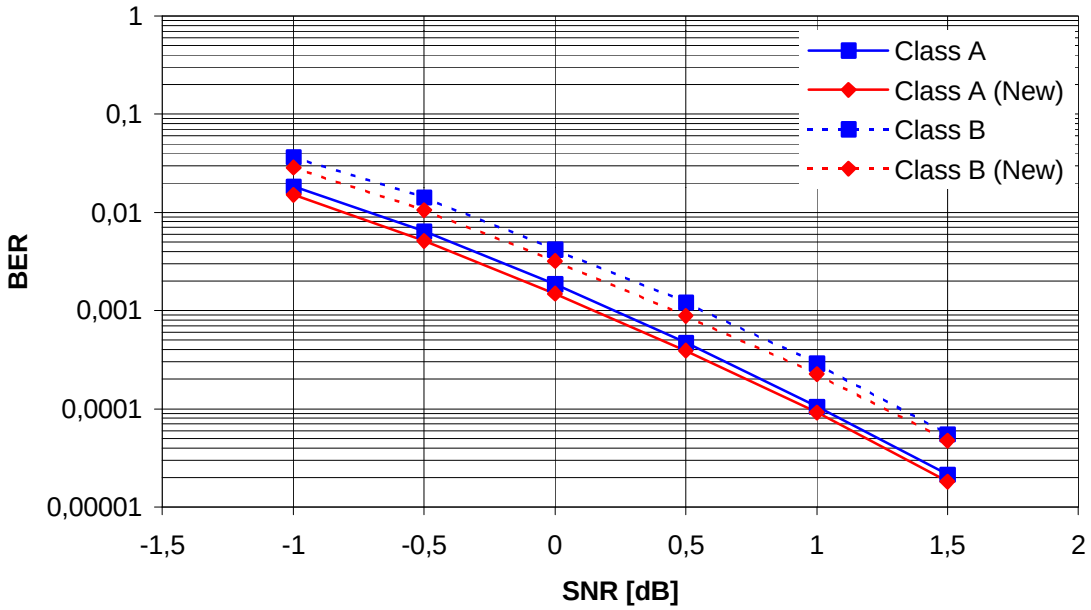

Figure 3: *Performance of normal puncturing and end puncturing for AMR 7.40 kbps (AWGN)*

Figure 4 shows the AWGN performance of a rate 1/2 code. Exemplary we analysed besides other simulations the case of puncturing 100 bits before rate matching to 88 bits after rate matching.

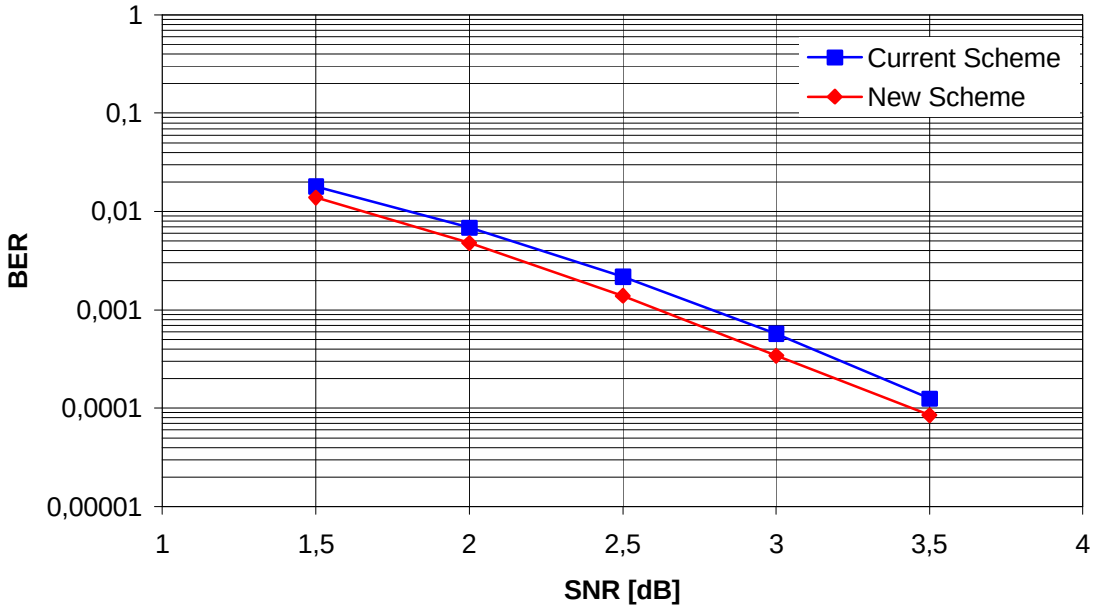

Figure 4: *Performance of normal puncturing and end puncturing for rate 1/2 (AWGN)*

4. Conclusion

The simulations show a performance enhancement compared to the evenly distributed rate matching approach assuming the same effective coding rate but with pre-puncturing the first and last 6 or 8 bits of every coded block depending on the channel coding scheme.

For very short codes we achieved a performance improvement of about 0.3 dB as presented in [3]. The gain declines for middle and especially long codes. For the AMR 4.75 kbps mode with block length of 174/183 bits (before rate matching) we achieve a gain of 0.2 dB and for AMR 7.4 kbps we achieve a gain of 0.1 dB. The performance improvement in case of the presented AMR simulations translates into a reduction of the necessary power per user of 2-4% and a corresponding capacity increase.

The paper shows that end puncturing has very interesting properties regarding performance especially when supporting channel coding with short information blocks (e.g. when used with AMR). The proposed scheme is generally applicable for short, middle and long blocks, but the performance improvement decreases with increasing blocksize. In our view it would be advantageous to implement the proposed method after convolutional coding in UL and DL as an option.

References

[1] TS26.101 V 3.0.0 Mandatory Speech Codec speech processing functions – AMR Speech Codec Frame Structure

[2] TSG R1-99G51, “New puncturing scheme for convolutional codes”, October 99, Siemens

[3] TSG R1-99J08, “End puncturing for short convolutional codes”, November 99, Siemens

[4] TSG R1-99B85, “Effect of EEP and UEP on channel coding for AMR”, August 99, Nokia

Appendix

Textproposal for TS25.212

In the following we describe changes of TS25.212 V 3.1.1 that are needed to introduce this scheme. As can be seen we need only some minor changes of chapter 4.2.3.

4.2.3 Channel coding

Code blocks are delivered to the channel coding block. They are denoted by $o_{ir1}, o_{ir2}, o_{ir3}, \ldots, o_{irK_i}$, where *i* is the TrCH number, *r* is the code block number, and K_i is the number of bits in each code block. The number of code blocks on TrCH *i* is denoted by C_i. After encoding the bits are denoted by $y_{ir1}, y_{ir2}, y_{ir3}, \ldots, y_{irY_i}$. The encoded blocks are serially multiplexed so that the block with lowest index *r* is output first from the channel coding block. The bits output are denoted by $c_{i1}, c_{i2}, c_{i3}, \ldots, c_{iE_i}$, where *i* is the TrCH number and $E_i = C_i Y_i$. The output bits are defined by the following relations:

~~$c_{ik} = y_{i1(k)}$ $k = 1, 2, \ldots, Y_i$~~

$c_{ik} = y_{i1(k+t)}$ $k = 1, 2, \ldots, Y_i$

~~$c_{ik} = y_{i,2,(k-Y_i)}$ $k = Y_i + 1, Y_i + 2, \ldots, 2Y_i$~~

$c_{ik} = y_{i,2,(k+t-Y_i)}$ $k = Y_i + 1, Y_i + 2, \ldots, 2Y_i$

~~$c_{ik} = y_{i,3,(k-2Y_i)}$ $k = 2Y_i + 1, 2Y_i + 2, \ldots, 3Y_i$~~

$c_{ik} = y_{i,3,(k+t-2Y_i)}$ $k = 2Y_i + 1, 2Y_i + 2, \ldots, 3Y_i$

…

~~$c_{ik} = y_{i,C_i,(k-(C_i-1)Y_i)}$ $k = (C_i - 1)Y_i + 1, (C_i - 1)Y_i + 2, \ldots, C_iY_i$~~

$c_{ik} = y_{i,C_i,(k+t-(C_i-1)Y_i)}$ $k = (C_i - 1)Y_i + 1, (C_i - 1)Y_i + 2, \ldots, C_iY_i$

The relation between o_{irk} and y_{irk} and between K_i and Y_i and the parameter t ~~is~~ are dependent on the channel coding scheme.

The following channel coding schemes can be applied to TrCHs:

- Convolutional coding
- Turbo coding
- No channel coding

The values of t in connection with each coding scheme:

- Convolutional coding, tail puncturing, 1/2 rate: t=6; 1/3 rate: t=8
- Convolutional coding, no tail puncturing: t=0
- Turbo coding t=0
- No channel coding t=0

The values of Y_i in connection with each coding scheme:

- Convolutional coding, ½ rate: $Y_i = 2*K_i + 16$-*2*t; 1/3 rate:* $Y_i = 3*K_i + 24$-*2*t*
- Turbo coding, 1/3 rate: $Y_i = 3*K_i + 12$
- No channel coding, $Y_i = K_i$