Mueller Exhibit 22

TSG-RAN Working Group 1 meeting #6
Hannover, Germany
August 30 – September 3, 1999

TSGR1#7(99)b32

Agenda item:

Source: Ericsson

Title: Transport block concatenation and code block segmentation

Document for: Decision

---

1 Introduction

It is currently very unclear in [1], what unit the channel coding is performed on. When the transport format attributes of fixed bit rate TrCHs are identical, they can be multiplexed in the 1st multiplexing. If they are multiplexed, it is clear from Section 4.2.2 in [1], that all transport blocks (TrBks) of the multiplexed TrCHs are concatenated before coding. Further, if the resulting number of bits including tail exceeds 512 for convolutional coding or 5120 for turbo coding, segmentation is performed.

For convolutional coding, the output of the segmentation is referred to as code blocks and it is clear from Section 4.2.3.1.1 that this is the unit that tail bits are attached to (assuming that coding block is the same as code block). For turbo coding the output of the segmentation is referred to as encoder input segments. The input to the segmentation is referred to as data block. When first multiplexing has been performed, it is assumed that the data block corresponds to the output of the 1st multiplexing, i.e. concatenated TrBks. Unfortunately, it is not clear what happens when there is no 1st multiplexing. The data block could then correspond to either a TrBk or several concatenated TrBks.

In this paper it is proposed that all TrBks on a TrCH always are concatenated before coding. If the number of bits after concatenation exceeds (512-Tail) for convolutional coding or (5120-Tail) for turbo coding, segmentation is performed. It is proposed that the unit that coding is performed over is referred to as code block.

2 Consequences of TrBk concatenation

If TrBks are concatenated before coding, the number of tail bits per transmission time interval (TTI) can be reduced. Further, the gain from turbo coding will be larger.

The disadvantage with concatenating TrBks before coding is that Hybrid type II/III ARQ becomes less efficient. CRC is always added on TrBk level. With Hybrid type II/III ARQ, the CRC is used to check if the TrBk was received correctly. If not, the redundancy is increased. Hence, even if only one TrBk was in error, the extra redundancy will be transmitted for all TrBks encoded together. Consequently, the potential gain with Hybrid type II/III ARQ will decrease.

It is proposed that concatenation of TrBks always is performed. If the number of bits after concatenation exceeds (512-Tail) for convolutional coding or (5120-Tail) for turbo coding, segmentation is performed. It is also proposed that the unit that coding is performed over is referred to as code block. Further, it is proposed that this functionality is separated from the channel coding block. A new block called TrBk concatenation / Code block segmentation should then be inserted in the multiplexing chain and section 4.2.3.1.2 and 4.2.3.2.4 can be merged. If Hybrid type II/III ARQ is included in release 00, then the possibility to bypass this concatenation/segmenation also needs to be included (note that only segmentation will not be needed since WG2 will set upper limits on TrBk sizes). However, we do not see any reason to include the possibility to avoid concatenation before coding in release 99 since Hybrid type II/III ARQ will not be part of it.

3 References

[1] TSG RAN WG1, "TS 25.212 Multiplexing and channel coding (FDD)".

4 Text proposal for 25.212

[A new block should be inserted into Figure 1 and Figure 2 of 25.212, as illustrated below.]

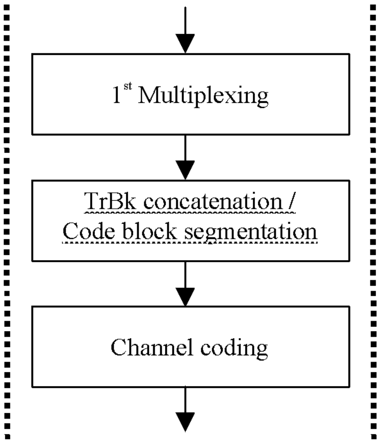

Figure 1: Changes in Figure 1 and Figure 2 of 25.212

-- snip --

4.2.3 Transport block concatenation and code block segmentation

All transport blocks in a TTI are serially concatenated. If the number of bits in a TTI is larger than Z, then code block segmentation is performed after the concatenation of the transport blocks. The maximum size of the code blocks depend on if convolutional or turbo coding is used for the TrCH.

4.2.3.1 Concatenation of transport blocks

The bits input to the transport block concatenation are denoted by $b_{im1}, b_{im2}, b_{im3}, \ldots, b_{imB_i}$ where i is the TrCH number, m is the transport block number, and B_i is the number of bits in each block (including CRC). The number of transport blocks on TrCH i is denoted by M_i. The bits after concatenation are denoted by $x_{i1}, x_{i2}, x_{i3}, \ldots, x_{iX_i}$, where i is the TrCH number and $X_i = M_i B_i$. They are defined by the following relations:

$x_{ik} = b_{i1k}$ $\quad k = 1, 2, \ldots, B_i$

$x_{ik} = b_{i,2,(k-B_i)}$ $\quad k = B_i + 1, B_i + 2, \ldots, 2B_i$

$x_{ik} = b_{i,3,(k-2B_i)}$ $\quad k = 2B_i + 1, 2B_i + 2, \ldots, 3B_i$

…

$x_{ik} = b_{i,M_i,(k-(M_i-1)B_i)}$ $k = (M_i - 1)B_i + 1, (M_i - 1)B_i + 2, ..., M_iB_i$

4.2.3.2 Code block segmentation

<Ericsson's note: It is proposed that filler bits are set to 0.>

Segmentation of the bit sequence from transport block concatenation is performed if $X_i>Z$. The code blocks after segmentation are of the same size. The number of code blocks on TrCH *i* is denoted by C_i. If the number of bits input to the segmentation, X_i, is not a multiple of C_i, filler bits are added to the last block. The filler bits are transmitted and they are always set to 0. The maximum code block sizes are:

convolutional coding: $Z = 512 - K_{tail}$

turbo coding: $Z = 5120 - K_{tail}$

The bits output from code block segmentation 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.

Number of code blocks: $C_i = \lceil X_i / Z \rceil$

Number of bits in each code block: $K_i = \lceil X_i / C_i \rceil$

Number of filler bits: $Y_i = C_iK_i - X_i$

If $X_i \leq Z$, then $o_{i1k} = x_{ik}$, and $K_i = X_i$.

If $X_i \geq Z$, then

$o_{i1k} = x_{ik}$ $k = 1, 2, \ldots, K_i$

$o_{i2k} = x_{i,(k+K_i)}$ $k = 1, 2, \ldots, K_i$

$o_{i3k} = x_{i,(k+2K_i)}$ $k = 1, 2, \ldots, K_i$

…

$o_{iC_ik} = x_{i(k+(C_i-1)K_i)}$ $k = 1, 2, \ldots, K_i - Y_i$

$o_{iC_ik} = 0$ $k = (K_i - Y_i) + 1, (K_i - Y_i) + 2, \ldots, K_i$

4.2.4 ~~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 $x_{ir1}, x_{ir2}, x_{ir3}, \ldots, x_{irX_i}$. 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 $E_i = C_iX_i$. The output bits are defined by the following relations:

$c_{ik} = x_{i1k}$ $k = 1, 2, ..., X_i$

$c_{ik} = x_{i,2,(k-X_i)}$ $k = X_i + 1, X_i + 2, \ldots, 2X_i$

$c_{ik} = x_{i,3,(k-2X_i)}$ $k = 2X_i + 1, 2X_i + 2, \ldots, 3X_i$

…

$c_{ik} = x_{i,C_i,(k-(C_i-1)X_i)}$ $k = (C_i - 1)X_i + 1, (C_i - 1)X_i + 2, \ldots, C_iX_i$

The relation between o_{irk} and x_{irk} and between K_i and X_i is dependent on the channel coding scheme.

The following channel coding schemes can be applied to TrCHs.

- Convolutional coding
- Turbo coding
- No channel coding

Table 1: Error Correction Coding Parameters

Transport channel type	Coding scheme	Coding rate
BCH	Convolutional code	1/2
PCH		
FACH		
RACH		
DCH		1/3, 1/2, or no coding
DCH	Turbo code	

Note1: The exact physical layer encoding/decoding capabilities for different code types are FFS.
Note2: In the UE the channel coding capability should be linked to the terminal class.

4.2.4.1 ~~4.2.3.1~~ Convolutional coding

4.2.4.1.1 ~~4.2.3.1.1~~ Convolutional coder

- Constraint length K=9. Coding rate 1/3 and ½.
- The configuration of the convolutional coder is presented in Figure 3.
- The output from the convolutional coder shall be done in the order starting from output0, output1 and output2. (When coding rate is 1/2, output is done up to output 1).
- K-1 tail bits (value 0) shall be added to the end of the cod~~ing~~e block before encoding.
- The initial value of the shift register of the coder shall be "all 0".

-- snip --

~~4.2.3.1.2 Segmentation into code blocks for convolutional coding~~

~~*<Note: It is for further study if the maximum code block size is 504 or shorter.>*~~

~~If the transport blocks or multiplexed transport blocks are longer than [504] bits (including CRC bits), they are segmented before convolutional encoding. Denote the number of transport blocks before coding by P and the number of bits in each transport block or the sum of the number of bits in the multiplexed blocks by M. Note that if first multiplexing is performed, all transport blocks of a transport channel in the same transmission time interval are multiplexed together, i.e. P=1. The bits before segmentation can then be described as follows:~~

~~Bits in transport block 1before segmentation: $d_{1,1}, d_{1,2}, d_{1,3}, \ldots d_{1,M}$~~

~~Bits in transport block 2 before segmentation: $d_{2,1}, d_{2,2}, d_{2,3}, \ldots d_{2,M}$~~

~~…~~

Bits in transport block P before segmentation: $d_{P,1}, d_{P,2}, d_{P,3}, \ldots d_{P,M}$

If M ≤ [504], no segmentation is performed. If M > [504] the following parameters are calculated:

Number of code blocks: S = round_up(PM / [504])

Length of coded blocks: C = round_up(PM / S)

Remainder: R = PM – S round_down(PM / S)

Number of filler bits: F = S – R, if R≠0
F = 0, if R=0

round_up(x) means the smallest integer number larger or equal to x.

round_down(x) means the largest integer number smaller or equal to x.

The F filler bits are appended to the end of the last code block before tail insertion and channel encoding. They are denoted $f_1, f_2, f_3, \ldots f_F$. The bits after segmentation are denoted by $u_{1,1}, u_{1,2}, u_{1,3}, \ldots u_{1,C}, u_{2,1}, u_{2,2}, u_{2,3}, \ldots, u_{2,C}, \ldots u_{S,1}, u_{S,2}, u_{S,3}, \ldots, u_{S,C}$, and defined by the following relations:

$u_{1,k} = d_{1,k}$ k = 1, 2, 3, ... C

$u_{2,(k-C)} = d_{1,k}$ k = C + 1, C + 2, C + 3, ... 2C

...

$u_{j,(k-(j-1)C)} = d_{1,k}$ k = (j-1)C + 1, (j-1)C + 2, (j-1)C + 3, ... M

$u_{j,(k-(j-1)C)} = d_{2,(k-M)}$ k = M + 1, M + 2, M + 3, ... jC

$u_{j+1,(k-jC)} = d_{2,(k-M)}$ k = jC + 1, jC + 2, jC + 3, ... (j+1)C

...

$u_{S,(k-(S-1)C)} = d_{P,(M-C+F+k-(S-1)C)}$ k = (S – 1)C + 1, (S – 1)C + 2, (S – 1)C + 3, ... SC – F

$u_{S,(k-(S-1)C)} = f_{k-SC+F}$ k = SC – F + 1, SC - F+ 2, SC – F +3, ... SC

<Note: Above it is assumed that all transport blocks have the same size. There are cases when the total number of bits that are sent during a transmission time interval is not a multiple of the number of transport blocks. A few padding bits are then needed but the exact insertion point (in the multiplexing chain) of these bits is for further study.>

-- snip --

4.2.3.2.4 Encoding blocks for Turbo code

Input data blocks for a turbo encoder consist of the user data and possible extra data being appended to the user data before turbo encoding. The encoding segments for a turbo encoder are defined in terms of systematic bits. The segment includes the user data, a possible error detection field (CRC), possible filler bits, and the termination. The maximum encoding segment length for turbo coding is 5120 bits. The Algorithm for combining and segmentation is as follows:

Inputs:

N_{DATA} size of input data block to turbo encoder

N_{TAIL} number of tail bits to be appended to the encoding segments (termination)

Outputs:

N_S number of segments

N_{TB} number of bits in the turbo encoder input segments

N_{FILL} number of filler (zero) bits in the last turbo encoder input segment

Do:

1. Let N_S = round_up(N_{DATA} / (5120 – N_{TAIL}))

2. Let N_{TB} = round_up (N_{DATA} / N_S) + N_{TAIL};

3. Let N_{REM} = remainder of N_{DATA} / N_S;

4. If N_{REM} not equal to 0 then insert N_{FILL} = (N_S – N_{REM}) zero bits to the end of the input data else N_{FILL} = 0.

5. End.

Here *round_up(x)* stands for an smallest interger number being larger or equal to *x*.

All turbo encoder input segments are of equal size and therefore the same turbo interleaver can be used for all turbo segments. A number of systematic bits over an entire channel interleaving block at output of the encoder is

N_S * (round_up(N_{DATA} / N_S) + N_{TAIL}).

The N_{FILL} filler bits are padded to the end of the last encoding segment in order to make the last segment equal size to the precedent ones. The filler bits are encoded.