RAN WG1 meeting #16 **TSGR1#16(00) 1299**
Place : Pusan
Date : October.
Title : Proposal for flexible position BTFD
Source : Mitsubishi Electric (Trium R&D)
Paper for : Discussion

1 Introduction

In order to achieve mapping of services such as speech with a SF of 256 in DL it was considered to use blind transport format detection (BTFD) with flexible positions.
In this paper, in a first section we describe in detail how this can work, in a second section we summarise the impact. Then we conclude on how to proceed.
In the annexe we give RM pattern parameter determination algorithms for the so called method 1 to be introduced in this paper.

2 References

[1] TS25.212 ver 3.4.0 3GPP TSG RAN Multiplexing and channel coding (FDD)

3 Abbreviations

RM	Rate Matching
DTX bit	an indication of DTX
BTFD	Blind Transport Format Detection, that is the R1 official designation of Blind Rate Detection

4 Description of the CCTrCH chain for flexible position BTFD

Main principles of sequential detection

The main principle is that the detection is sequential. TF of TrCH 1 is first detected, then based on the knowedged of the TF of TrCH 1, the TF of TrCH 2 can be detected, and so on.

Exemple of simple sequential detection

This is exemplified on the figure 1 below. On this figure we consider a CCTrCH with 3 TrCHs, namely 1, 2, and 3. For the sake of explanation we assume that the three TrCHs have a TTI of 10ms.
TrCH 1 is in white and has 2 TFs, TrCH 2 is in light grey and has 2 TFs, and TrCH 3 is in dark grey and has 4 TFs.
The amount of data per radio frame for each TrCH and per each corresponding TF is represented in the upper part of the figure. This amount of data for transport channel *i* and transport format *l* is denoted $H_{i,l}$. This is represented by the length of the rectangles on the figure.
On the bottom left part of the figure we represent the 13 possible TFCs of the CCTrCH.
Now on the bottom right part is exemplified the steps of sequential detection. In the example we assume that a radio frame with TFC = 1 was received.

?? At step ✍ a pointer *p* is positioned at the beginning of the received frame. And a set *R* is set to {0, 1, …, 12}, that is to say all the possible TFC are remaining.

?? At step ✍ an amount of data *H* is determined as $H ? \max_{l ? R_1} H_{1,l}$ where $R_1 ? TF_1 ?R?$ is the set of potential TFs for TrCH 1 when the TFC is in R. In other words, *R* containing all the TFCs, we have $R_1 ? ?0,1?$, and so $H ? H_{1,1}$. A block of data *B* is extracted from the frame : it is taken at position *p* and with a size of *H.*

?? At step ✍ the blind rate detection is carried out on block *B*. This allows to detect that transport format of TrCH 1 is TF_1=0,

?? At step ✍ we take into account the detection that gave the TF_1=0. So,

- pointer *p* is moved forward by an amount of $H_{1,\mathrm{TF}_1} ? H_{1,0}$, and
- *R* is updated to $R ? TF_1^{?1} ??0??$, where $TF_1^{?1} ??0??$ denotes the set of TFC for which the TF of TrCH 1 is 0. The new value of *R* is therefore {0,1,…,6} after the update.

?? Step ✍ is the similar to step ✍ but for TrCH 2. *H* is determined as $H ? \max_{l ? R_2} H_{2,l}$, where $R_2 ? TF_2 ?R?$, that is to say $R_2 ? ?0,1?$ because both TF 0 and 1 are taken for TrCH 2 when TFC in {0,1,…,6}, so *H* is determined as $H ? H_{2,1}$, and a bloc *B* is extracted from the received frame at position *p* and with size *H.*

?? Step ✍ is similar to step ✍ but for TrCH 2 : BTFD is carried out on block *B*, and this determines that TF_2 = 0.

?? Step ✍ is similar to step ✍ but for TrCH 2. Having determined that TF_2 = 0 :

- p is moved forward by an amount of $H_{2,\mathrm{TF}_2} ? H_{2,0}$,
- *R* is updated to $R ? TF_2^{?1} ??0??$, that is to say TFC *j* for which $TF_2 ?j? ? 0$ are excluded from *R. R* is therefore {0,1,2,3} after the update.

?? Step ✍ is similar to step ✍ and step ✍ but for TrCH 3. $R_3 ? TF_3 ?R? ? ?0,1,2,3?$, and so *H* is determined as $H ? \max_{l ? R_3} H_{3,l} ? H_{3,3}$. So a block *B* is extracted from the received frame from position *p* and with size *H.*

?? Finally step ✍ is similar to ✍ and step ✍ but for TrCH 3. BTFD is carried out on block B for TrCH 3, which allows to detect that TF_3=1.

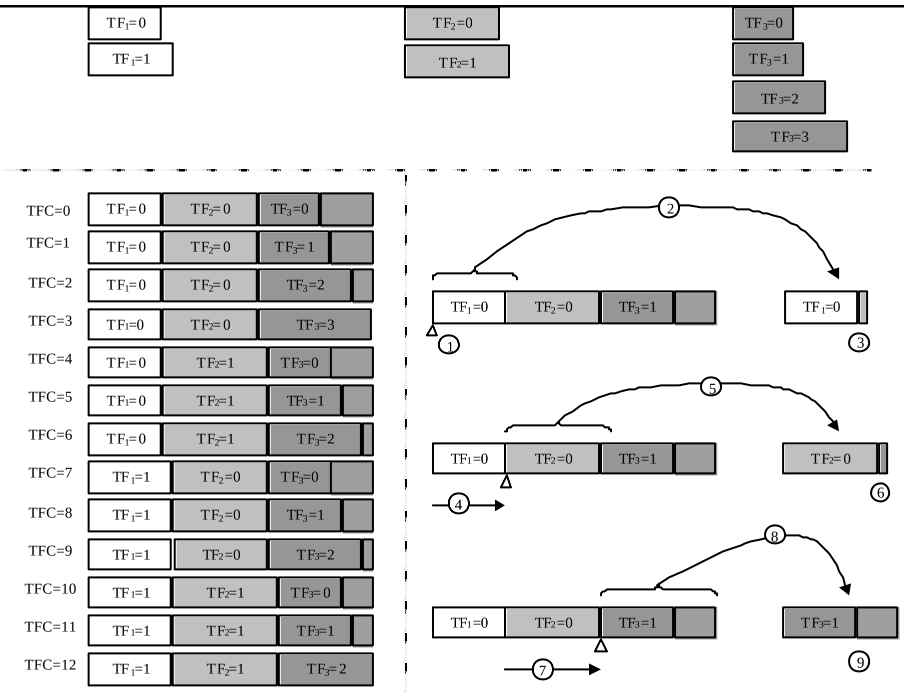


figure 1 sequential detection of TF

In view of this example it can be concluded that the TrCH are ordered in ascending order of TTI duration (F_i). As a matter of fact TF of TrCH *i* can be detected only every $\max\{F_1, F_2, \ldots, F_i\}$ tens of ms because you need to have detected TF of TrCH 1, 2, …,*i*-1 before you can detect TF of TrCH *i*.

Example of double sequential detection

Simple sequential detection has several drawbacks :

- Operations cannot be carried out in parallel. For instance you cannot de-1st-IL or de-rate match TrCH 2 while you channel decode TrCH 1. This means longer processing delays.
- A TF detection error on TrCH *i* affects decoding of TrCH *i*+1, *i*+2, …, *I*.

In order to mitigate these drawbacks, we propose to use double sequential detection as described below :
The set of TrCHs comprised in the CCTrCH is partitioned into two subsets, namely *G* and *D*.
We use the following notations :
$I = I_G + I_D$
$G = \{1, 2, \ldots, I_G\}$ and $D = \{I_G + 1, I_G + 2, \ldots, I\}$
The TrCHs are in ascending order of TTI duration in both *G* and *D*, but there is no order relation between elements of *G* and *D*.
In other words :
$F_1 \leq F_2 \leq \ldots \leq F_{I_G}$ and $F_{I_G+1} \leq F_{I_G+2} \leq \ldots \leq F_I$,
but $F_{I_G} \leq F_{I_G+1}$ is not required.

The multiplexing of TrCH and the 2nd insertion of DTX indication bits is modified as shown on figure 2 below :

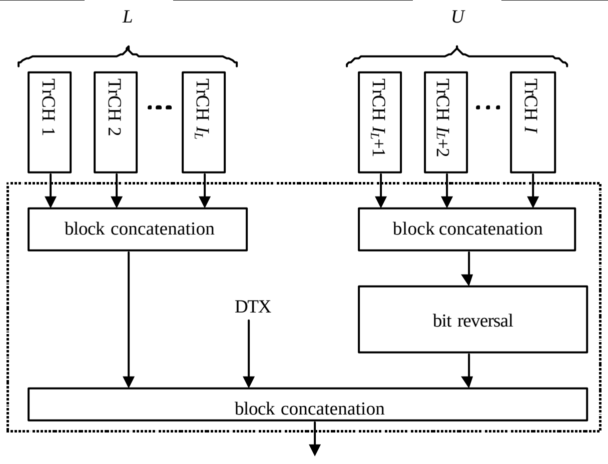


figure 2 TrCH multiplexing and second DTX insertion

TrCH blocks of G are concatenated as usual, and the concatenate block is placed in the beginning of the frame.
TrCH blocks of D are concatenated as usual, then the concatenate block is bit reversed, and then the bit reversed block is placed in the end of the frame.
DTX indication bits are inserted between data from G and data from D. Note that when D is empty there is no change compared to current scheme multiplexing.
Moreover, simple detection is a particular case of double detection for which set D is empty (i.e. $I_D = 0$ and $I_G = I$).

On figure 3 we compare sequential detection versus double sequential detection. This figure is to be compared to figure 1. On the bottom left part we see the frames in case of sequential detection, whereas on the bottom right part we see the frames in case of double sequential detection with G ? ?1,2? and D ? ?3?. We can notice that the border between data from G and D is flexibly moving according to the TFC.

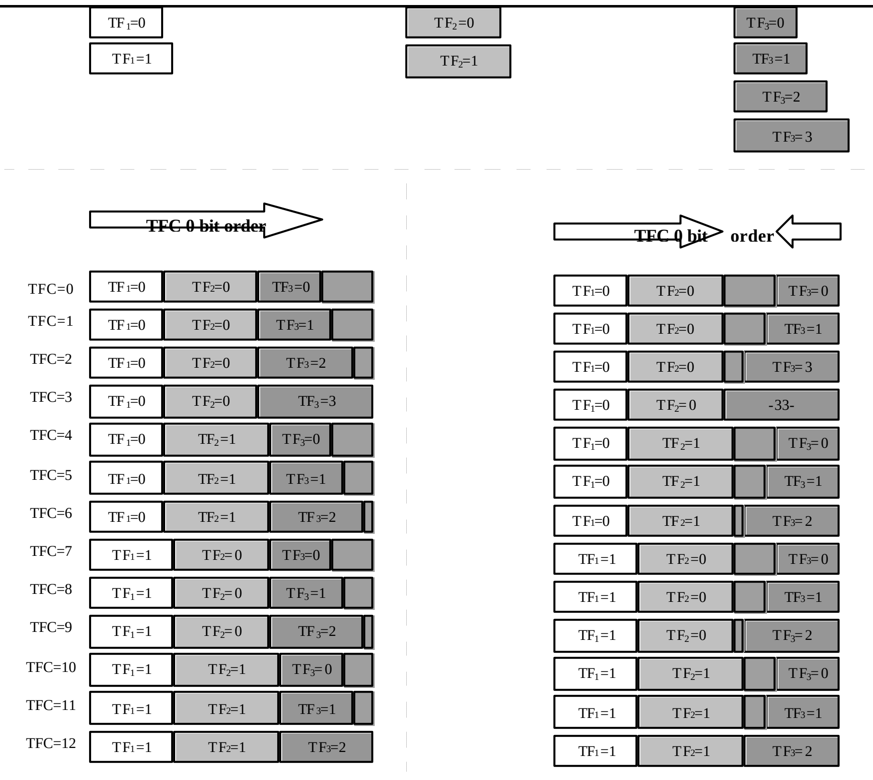


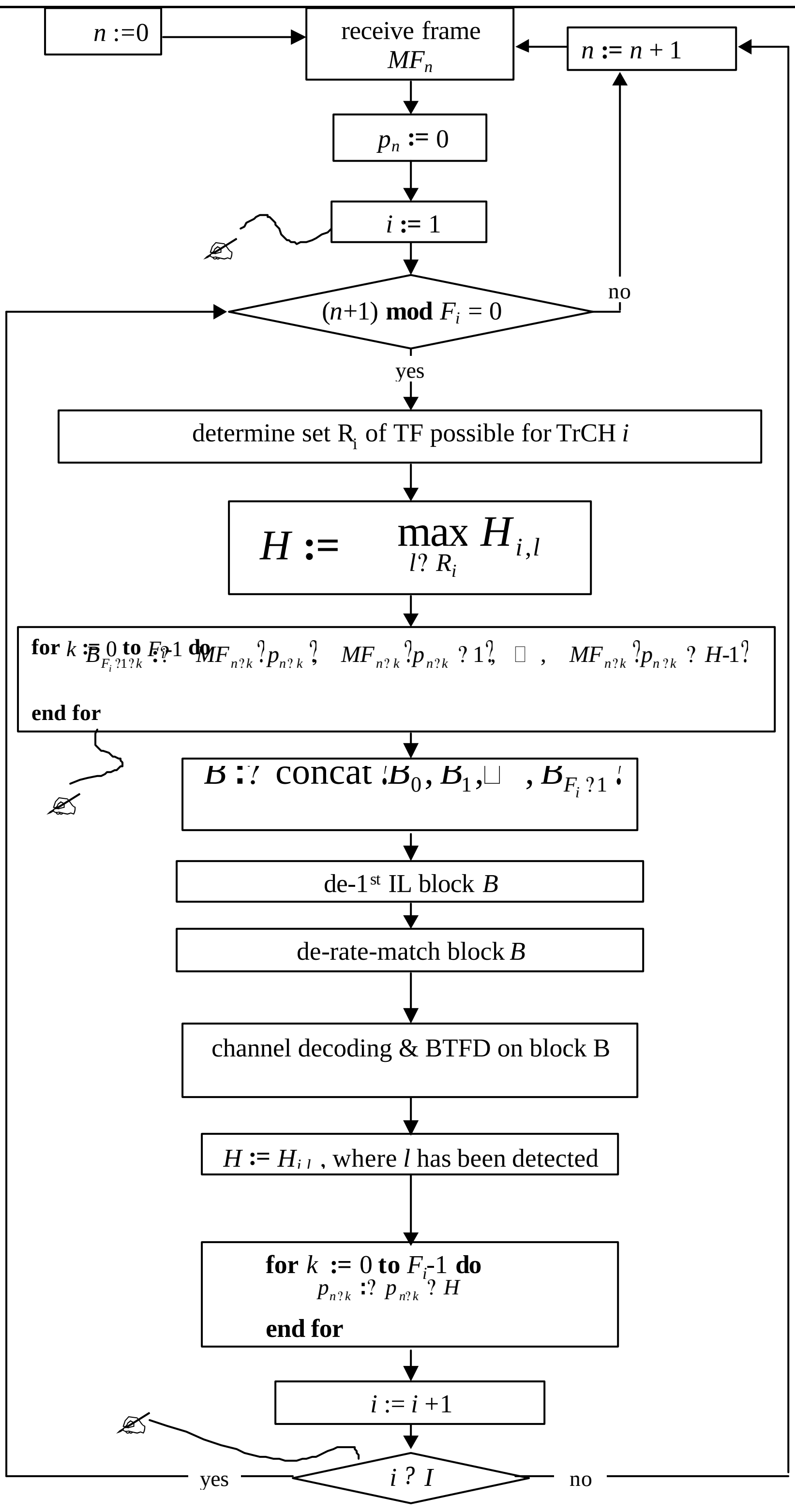
figure 3 Sequential detection versus double sequential detection

General sequential detection algorithm

Below on figure 4, we give the generic algorithm for simple sequential detection.
For double sequential detection the algorithm is run twice in parallel, once for *G* and once for *D.*
When run for *G*, the algorithm is modified such that in step ✍ comparison $i ? I$ is replaced by comparison $i ? I_G$.
When run for *D*, the algorithm is modified such that :

?? in step ✍ initialisation $i :? 1$ is replaced by initialisation $i :? I_G ? 1$, and
?? step ✍ is modified as follows

```
for $k$ := 0 to $F_i$-1 do
    $B_{F_i ? 1 ? k} :? MF_{n ? k}??N_{data} ? 1?? ?p_{n ? k}??, MF_{n ? k}??N_{data} ? 1?? ?p_{n ? k} ? 1??, ,$
    $, MF_{n ? k}??N_{data} ? 1?? ?p_{n ? k} ? H\text{-}1??$
end for
```



$n := 0$
receive frame MF_n
$n := n + 1$
$p_n := 0$
$i := 1$
$(n+1)$ **mod** $F_i = 0$
no
yes
determine set R_i of TF possible for TrCH i
$H := \max_{l ? R_i} H_{i,l}$
for k := 0 **to** F_i-1 **do** $B_{F_i ?1?k}$:? $MF_{n?k}$?$p_{n?k}$?, $MF_{n?k}$?$p_{n?k}$? 1?, , $MF_{n?k}$?$p_{n?k}$? H-1?
end for
B :? concat ?B_0, B_1, , $B_{F_i ?1}$?
de-1st IL block B
de-rate-match block B
channel decoding & BTFD on block B
$H := H_{i\,l}$, where l has been detected
for k := 0 **to** F_i-1 **do**
$p_{n?k}$:? $p_{n?k}$? H
end for
$i := i + 1$
i ? I
yes
no

figure 4

More comments on ordering of TrCHs

Let us assume that there exists some integer U_G (resp. U_D) such that $1 ? U_G ? I_G$ (resp. $1 ? U_D ? I_D$).

It is to be noted that in a sequential detection, when the TFs of the $I_G ? U_G$ (resp. $I_D ? U_D$) can be determined with the sole knowledge of the TFs of the U_G (resp. U_D) first TrCH, then the order of the TTI durations of the $I_G ? U_G$ (resp. $I_D ? U_D$) last TrCH needs not to be ascending. We need only that :
$F_1 ? F_2 ? \square ? F_{U_G}$ and $? i ? ?U_G ? 1, \square, I_G ? F_i ? F_{U_G}$
or respectively :
$F_{I_G ? 1} ? F_{I_G ? 2} ? \square ? F_{I_G ? U_D}$ and $? i ? ?I_G ? U_D ? 1, \square, I ? F_i ? F_{I_G ? U_D}$.
but within $?U_G ? 1, \square, I_G ?$ (resp. $?I_G ? U_D ? 1, \square, I ?$) F_i needs not to be monotonic.
When this happens, the algorithms of figure 4 is carried out only on the U_G (resp. U_D) first TrCHs, and the $I_G ? U_G$ (resp. $I_D ? U_D$) last TrCH are de-multiplexed, de-radio-frame-segmented, and de-1st-interleaved as in the classical flexible positions, that is to say in knowledge of the respective TFs. Step ✍ of figure 4 is modified so that the comparison is replaced by $i ? U_G$ (resp. $i ? I_G ? U_D$).

Usage of 1st interleaver

On figure 5 below we show what happens with TrCH 1, where TrCH 1 has a TTI of 20ms, and two TF 0 and 1 such that

- $N^{TTI}_{1,0} ? ?N^{TTI}_{1,0} ? 2 ?H_{1,0} ? 10$
- $N^{TTI}_{1,1} ? ?N^{TTI}_{1,1} ? 2 ?H_{1,1} ? 16$

On the figure, hatching represents DTX bits, or bits from other TrCHs.

We see that 1st interleaving is carried out with a depth of $2 ?H_{1,0} ? 10$, while de-1st interleaving is carried out with a depth of $2 ?H_{1,0} ? 16$. In other words we don't use the same channel interleaver at transmission and at reception!
This can work only thanks to the good properties of the 1st interleaver. If RAN WG1 had selected the complex 1st IL + simple 2nd IL concept instead of the simple 1st IL + complex 2nd IL concept, surely today blind rate detection in flexible position would not be possible at acceptable cost.

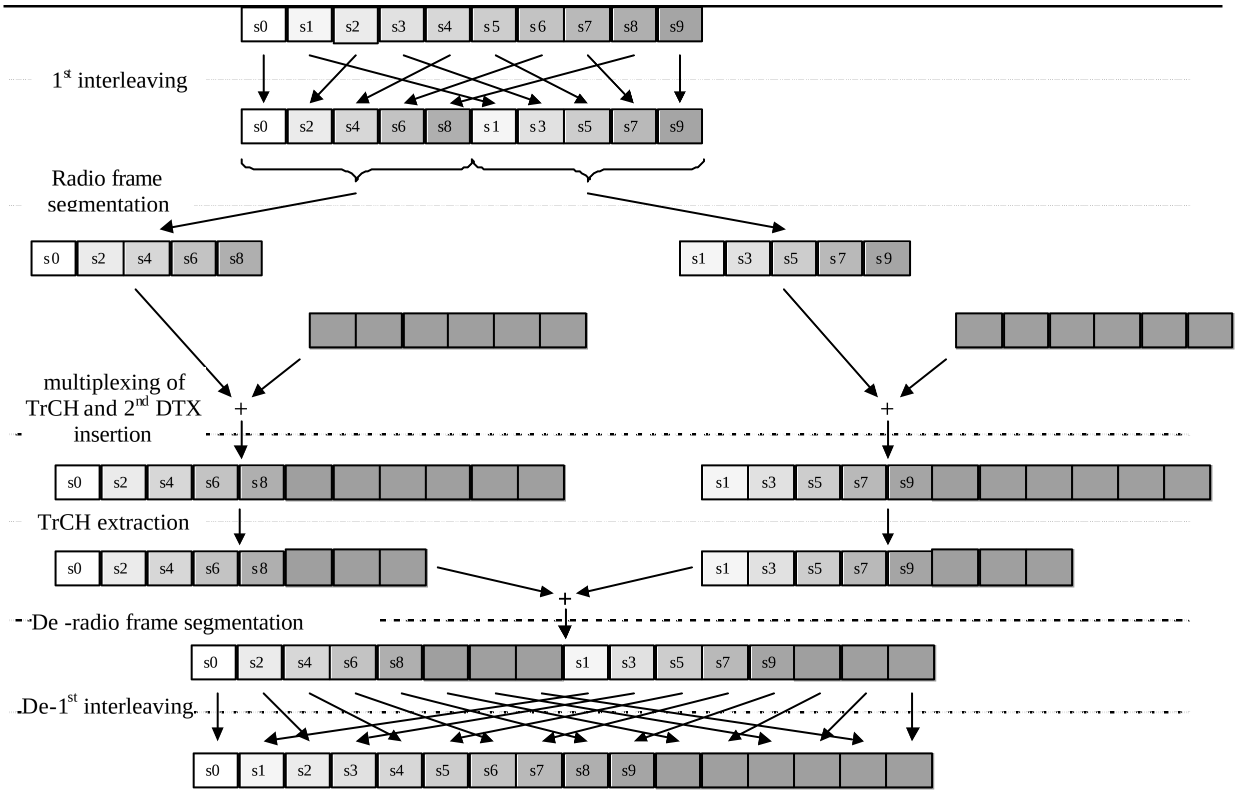

s0 s1 s2 s3 s4 s5 s6 s7 s8 s9
1st interleaving
s0 s2 s4 s6 s8 s1 s3 s5 s7 s9
Radio frame segmentation
s0 s2 s4 s6 s8
s1 s3 s5 s7 s9
multiplexing of TrCH and 2nd DTX insertion
+
TrCH extraction
De -radio frame segmentation
De-1st interleaving

figure 5

Usage of rate matching and of 1st DTX insertion

In order to be able to de-rate match without knowing the TF it is necessary to have a fixed RM pattern. In the following we consider only the RM pattern for the convolutional coding scheme, as BTFD is not considered for other coding schemes.

General on current RM pattern.

Generally speaking the RM pattern is parametrised by a pattern parameter vector $\vec{e}$. The set J in which $\vec{e}$ is to be found depends on the RM pattern determination algorithm. For instance, with the well established Siemens RM pattern determination algorithm we have

$$\vec{e} = (e_{ini}, e_{plus}, e_{minus}, s)$$

e_{ini}, e_{plus} and e_{minus} are the well known parameters, whereas we denotes by s the implicit tri-state parameter such that :

- s = 1 when repetition is used
- s = 0 when no RM is done (same output as input)
- s = - 1 when puncturing is done

J can therefore be defined as the set of quartet $\vec{e} = (e_{ini}, e_{plus}, e_{minus}, s)$ of integers such that :

$$s = 0, \text{ and } e_{ini} = e_{plus} = e_{minus} = 1,$$

or

$$s = -1,\ e_{plus} \geq 1,\ e_{minus} \in \{1,2,\ldots,e_{plus}-1\}, \text{ and } e_{ini} \in \{1,2,\ldots,e_{minus}\}$$

or

$$s ? 1,\ e_{plus} ? 1,\ e_{minus} ? 1, \text{ and } e_{ini} ? ?1,2,\ldots,e_{minus}?.$$

Given the size N of the input block to the rate matching, the size variation $?N^{\text{got}}$ that is got is a function f of N and of the pattern parameter vector $\overset{\sqcup}{e}$:

$$?N^{\text{got}} ? \text{f}\,?N,\overset{\sqcup}{e}?$$

A study of the Siemens RM pattern determination algorithm can show that in this case a simple analytical formula exists as we have :

$$\text{f}\,?N,\overset{\sqcup}{e}?? s ??1 ? ?\frac{e_{minus} ?N ? e_{ini}}{e_{plus}}???$$

We define the function sgn(x) that will be needed later as follows :

$$\text{sgn}\,?x?? \begin{cases} ?1 & \text{if } x ? 0 \\ 0 & \text{if } x ? 0 \\ 1 & \text{if } x ? 0 \end{cases}$$

In the classical flexible position, a pattern parameter vector $\overset{\sqcup}{e}$ is defined for each transport format as follows :

$$\begin{cases} s ? \text{sgn}\,??N_{i,l}^{\text{TTI}}? \\ e_{ini} ? 1 \\ e_{plus} ? 2?N_{i,l}, & \text{or 1 if } N_{i,l} ? 0 \\ e_{minus} ? 2?\left|?N_{i,l}^{\text{TTI}}\right|, & \text{or 1 if } N_{i,l} ? 0 \end{cases}$$

We these parameters we have exactly $?N^{\text{got}} ? ?N_{i,l}^{\text{TTI}}$

In the flexible position for BRD we cannot do so, because this could lead to slight difference of the pattern according to the TF. So we have two solutions, hereinafter called method 1 and method 2

RM and DTX insertion with method 1.

In method 1, the solution is to find the a set E of parameters that don't overstep the wanted variation $?N_{i,l}^{\text{TTI}}$. If we don't overstep the wanted variation $?N_{i,l}^{\text{TTI}}$ we can complete the variation by inserting a number $?N_{i,l}^{\text{TTI}} ? ?N_{i,l}^{\text{got}}$ of DTX bits in the 1st DTX insertion module. These DTX bits don't help at all the Eb/I balancing, however they are good for the radio frame size equalisation function, and they won't bother too much the Eb/I balancing if they are few enough.

The determination of the RM pattern is therefore all about selecting the element of E that minimise a criteria that is representative of the number of inserted DTX bits.

E is defined as follows :

$$E ? ?\overset{\sqcup}{e} ? J \text{ such that } ?l ? TFS?i? \ \ \text{f}\,?N,\overset{\sqcup}{e}?? ?N_{i,l}^{\text{TTI}}?$$

Once $\overset{\sqcup}{e}$ selected, it is still possible that for some TF l the got variation $?N_{i,l}^{\text{got}} ? f\,?N_{i,l},\overset{\sqcup}{e}?$ slightly differ from the wanted one $?N_{i,l}^{\text{TTI}}$. So the 1st DTX insertion step is used to insert $?N_{i,l}^{\text{TTI}} ? f\,?N_{i,l},\overset{\sqcup}{e}?$.

The exact algorithm to determined $\overset{\sqcup}{e}$ in method 1 is given in annexe.

The big advantage of method 1 is that we keep exactly the same RM pattern determination and we insert no new step, as the 1st DTX insertion already exists.
The drawback is that these DTX bits are useless. The upper bound on the number of inserted DTX bits is roughly $2?F_i ? 1$. So this is not very harmful when $N_{i,l}^{\text{TTI}}$ is in the order of magnitude of 100. However, for very small bit rate, method 1 could be somewhat risky.

RM and DTX insertion with method 2
In method 2 we change the RM pattern determination algorithm. The new algorithm however calls the Siemens algorithm as a sub-routine.

Let us assume that we have a TrCH i with a TFC set $TFS?i?? ?1,2,\ldots,L?$ such that :

$$0 ? N_{i,1}^{\text{TTI}} ? N_{i,2}^{\text{TTI}} ? \ldots ? N_{i,L}^{\text{TTI}}$$

By convention, though TF $l = 0$ is not defined, we set that $?N_{i,0}^{\text{TTI}} ? 0$.
Alternatively we can consider a TrCH i with a TFC set $TFS?i?? ?0,1,2,\ldots,L?$ such that :

$$0 ? N_{i,0}^{\text{TTI}} ? N_{i,1}^{\text{TTI}} ? N_{i,2}^{\text{TTI}} ? \ldots ? N_{i,L}^{\text{TTI}}$$

Let us now assume that we have a data block B of size N>0. It is possible to find the greatest integer k in $?0,1,\ldots,L?1?$ such that $N ? N_{i,k}^{\text{TTI}}$. So we segment block B into k+1 blocks B_1, B_2, …, B_k, B_{k+1} of respective sizes $N_{i,1}^{\text{TTI}}$, $?N_{i,2}^{\text{TTI}} ? N_{i,1}^{\text{TTI}}?$, $?N_{i,3}^{\text{TTI}} ? N_{i,2}^{\text{TTI}}?$, …, $?N_{i,k?1}^{\text{TTI}} ? N_{i,k?2}^{\text{TTI}}?$, $?N_{i,k}^{\text{TTI}} ? N_{i,k?1}^{\text{TTI}}?$, $?N ? N_{i,k}^{\text{TTI}}?$. The segmentation is a plain serial segmention, that is to say concatenation is the inverse operation.

Each block B_k is then separately rate-matched with the well known Siemens algorithm thus configured to obtain a size variation of $??N_{i,k}^{\text{TTI}} ? ?N_{i,k?1}^{\text{TTI}}?$.
Finally the resulting blocks are concatenated.

Note : the parametrisation of Siemens algorithm for RM of block B_k is the following :

?? $s ? \operatorname{sgn}??N_{i,k}^{\text{TTI}} ? ?N_{i,k?1}^{\text{TTI}}?$

?? $e_{ini} = 1$

?? $\begin{cases} e_{plus} ? 1 \\ e_{minus} ? 1 \end{cases}$ if $s ? 0$; $\begin{cases} e_{plus} ? N_{i,k}^{\text{TTI}} ? N_{i,k?1}^{\text{TTI}} \\ e_{minus} ? \left|?N_{i,k}^{\text{TTI}} ? ?N_{i,k?1}^{\text{TTI}}\right| \end{cases}$ otherwise

5 Summary of impacts

for the transmitter

items	impact	comment
CRC attachment, channel coding 1st IL, radio frame segmentation, 2nd IL, physical channel segmentation, mapping to physical channel	**none**	
RM (RM method 1)	The pattern parameters $\overset{\square}{e}$? $(e_{ini}, e_{plus}, e_{minus}, s)$ are determined in a new way	e_{ini} can be different from 1. e_{plus} and e_{minus} can be odd, $\overset{\square}{e}$ does not depend on TF l
1st DTX insertion (RM method 1)	Can be used to insert few DTX bits, this has mainly the merits of maintaining the radio frame equalisation.	? $N_{i,l}^{TTI}$? f $(N_{i,l}, \overset{\square}{e})$ DTX indication bits are inserted.
RM (RM method 2)	new pattern determination algorithm, different from Siemens algorithm. The new algorithm makes an RM pattern by concatenating several RM patterns that are determined by Siemens algorithm.	
1st DTX insertion (RM method 2)	**none**	
TrCH multiplexing and 2nd DTX insertion	when the double sequential detection is used, then an additional bit reversal function is necessary for the TrCHs from set D.	We believe that additional complexity is null. However the change is too significant and unforeseen to be accepted at this stage of the project.

for the receiver

items	impact	comment
inverses of CRC attachment, 2nd IL, physical channel	**none**	

segmentation, mapping to physical channel		
channel decoding	the same scheme as in fixed position BTFD can be used	
inverse RM (method 1)	see transmitter about pattern parameters $\vec{e} ? \{e_{ini}, e_{plus}, e_{minus}, s\}$. de rate matching is carried out assuming TF l maximising $N_{i,l}^{TTI}$, like in fixed positions.	
inverse RM (method 2)	see transmitter part.	
inverse 1st DTX insertion	no such thing = this is done during channel decoding, as in fixed positions	
inverse 1st IL	the depth can be different from that of transmission	we believe that this is no problem, as the current IL fortunately copes with that.
inverse TrCH multiplexing, radio frame segmentation, 2nd DTX insertion	TrCH are extracted sequentially, instead of being demultiplexed	We believe that this is a significant impact and that flexible positions BTFD cannot be accepted for release ‘99
parallelism	Decoding TrCh in parallel is seriously impaired.	

6 Conclusion

In this paper we have made a study of the BTFD in flexible position. We propose this scheme for release 4 or 5.

7 Annexe : determination of RM pattern parameters in method 1

Determination of s

This is simply done as follows :

$$s ? \min_{\substack{l?TFS?i?\\ N^{\mathrm{TTI}}_{i,l}?0}} ?\mathrm{sgn}??N^{\mathrm{TTI}}_{i,l}??$$

The rational of this formula is that if at least for one TF we need to puncture then we need to puncture for all.
When the determined s is null, e_{ini}, e_{plus} and e_{minus} are simply set to 1. Otherwise we proceed as follows

Determination of e$_{plus}$ *when* s ? *0*

$$e_{plus} ? 1 ? 2 ? \max_{l?TFS?i?} ?N^{\mathrm{TTI}}_{i,l}?$$

Note that this is only the least value that can do the job. Any greater or equal value could also do the job. There might be some advantage for instance in taking e_{plus} as a power of 2.
The rationale for this formula is that, according to the expression of $\mathrm{f}?N,\overset{\sqcup}{e}?$, when e_{minus} is incremented by one, then $?N^{\mathrm{got}}$ is incremented by roughly N/e_{plus}. In order to get a granularity of one on $?N^{\mathrm{got}}$ we need therefore that $e_{plus} ? 2?N$, the factor 2 is because of the rounding function $x\square\ ?x?$ in the expression of $\mathrm{f}?N,\overset{\sqcup}{e}?$.

Determination of e$_{ini}$ *and* e$_{minus}$ *when* s ? *0*

In the following, for the sake of simplicity, in the definition of the set J of rate matching pattern parameters vectors, we have substituted the condition $e_{ini} ? ?1,2,...,e_{minus}?$, by the condition $e_{ini} ? ?1,2,...,e_{plus} ? 1?$. This has no impact on the solution because among the point minimising the number of DTX inserted, we use a selection criteria that minimises e_{ini}.

Algorithm 1

Given the values of e_{plus} and s already determined e_{ini} and e_{minus} are determined as follows:

We denote E_l the set of couple $?e_{ini},e_{minus}?$ such that $?N^{\mathrm{TTI}}_{i,l}$ is not overstepped, in other words :

$$E_l ? ??e_{ini},e_{minus}?\text{such that } ?e_{ini},e_{minus},e_{plus},s?? \ J \text{ and } \mathrm{f}?N^{\mathrm{TTI}}_{i,l},?e_{ini},e_{plus},e_{minus},s??? ?N^{\mathrm{TTI}}_{i,l}?$$

And E is here the intersection of the sets E_l :

$$E ? \bigcap_{l?TFS?i?} E_l$$

So generally speaking E is can be determined as follows :

```
01:  $first\_time$ := **true**
02:  **for all** $l$ **in** $TFS ?i?$ **such that** $N^{TTI}_{i,l}$ ? 0 **do**
03:      détermine $E_l$
04:      **if** $first\_time$ **then**
05:          $E$ := $E_l$
06:          $first\_time$ := **false**
07:      **else**
08:          $E$ := $E$ ? $E_l$
09:      **end if**
10:  **end for**
```

It can be noted that at line 2 TF l such that $N^{TTI}_{i,l}$? 0 are excluded. This has no impact on the result, because if $N^{TTI}_{i,l}$? 0 then E_l contains all the other sets $E_{l?}$.

Because the function x ?x? is monotonic it is possible to show that E, E_l and any intersection of E_l's can be defined as a region limited by a graph. This region as the look given by the figure 6 below. The region is the greyed surface, whereas the graph is the bold staircase curve.

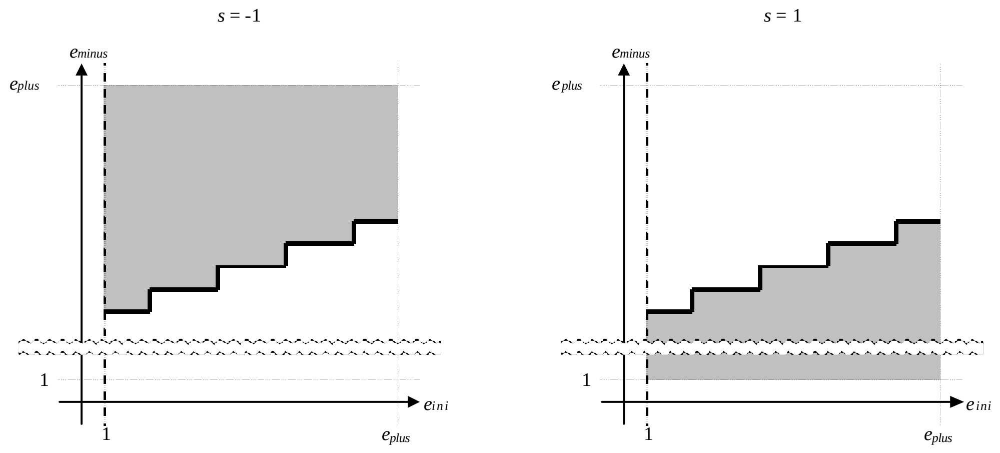


figure 6

So, henceforth we use, instead of sets E and E_l, arrays G and GL of e_{plus} integers, indexed from 1 to e_{plus}, and storing the graph.
So the step of determining E_l in line 3 can be done as follows:

```
10:  $A$ := $s$ ?? $N^{TTI}_{i,l}$ ?$e_{plus}$
11:  $Q$ := $A$ **div** $N^{TTI}_{i,l}$        -- *Euclidian division,* $A$ div $N^{TTI}_{il}$ ? ?$A/N^{TTI}_{il}$ ?
12:  $R$ := $A$ **mod** $N^{TTI}_{i,l}$        -- *Remainder,* $A$ mod $N^{TTI}_{il}$ ? $A$ ? $N^{TTI}_{il}$ ??$A/N^{TTI}_{il}$ ?
13:  $e_{minus}$ := $Q$
14:  $e_{ini}$ := -R
15:  $stop$ := **false**
16:  $e^{c}_{ini}$ := 1
17:  **repeat**
18:      $e_{minus}$ := $e_{minus}$ + **1**
```

```
19:         $e_{ini}$ := $e_{ini}$+ $N_{i,l}^{TTI}$
20:         if $e_{ini}$ ? $e_{plus}$ then
21:             stop := true
22:             $e_{ini}$ := $e_{plus}$
23:         end if
24:
25:         while $e'_{ini}$ ? $e_{ini}$ do
26:             $GL[e'_{ini}]$ :? $e_{minus}$
27:             $e'_{ini}$ :? $e'_{ini}$ ? 1
28:         end while
29:     until stop
```

The algorithm above can be explained as follows: graph *GL* is indeed the curve in the (e_{ini} , e_{minus}) plan defined by the following equation :

$$f\left(N_{il}^{TTI},\left(e_{ini},e_{minus},e_{plus,i},s_i\right)\right) ? ?N_{i,l}^{TTI}$$

This graph has the shape of a staircase of constant stair-step size such that each stair-step in the stair corresponds to incrementing e_{minus} by one, and e_{ini} by N_{il}^{TTI} . Therefore in lines 13 and 14 a point (e_{ini},e_{minus}) is determined on the edge of a stair-step. At each iteration of the loop lying from line 17 to line 29, a new stair-step is determined. And at each iteration of the loop lying from line 25 to line 28 a point is determined in the stair-step under determination.

Assigning E_l to E at line 5 simply consists in copying graph *GL* into graph *G* as follows :

```
for $e_{ini}$ := 1 to $e_{plus,i}$ do
    $G[e_{ini}]$ := $GL[e_{ini}]$
end for
```

The intersection of E and E_l at line 8 consists for each value of e_{ini} to take the greater value of e_{minus} in case of puncturing, et the less one in case of repetition. This is in more details below :

```
for $e_{ini}$ := 1 to $e_{plus,i}$ do
    if $s?G[e_{ini}] > s?GL[e_{ini}]$ then
        $G[e_{ini}]$ := $GL[e_{ini}]$
    end if
end for
```

Once graph *G* is determined, set *E* is available as illustrated on figure 6. So a point in *E* is chosen by minmising a criteria C that is representative of the number of DTX inserted . For instance this criteria is the maximum over all the TF of the ratio between the number of DTX inserted and the number of bits before rate matching:

$$C(\vec{e}) ? \max_{\substack{l ? TFS(i) \\ N_{i,l}^{TTI} ? 0}} \left(\frac{?N_{i,l}^{TTI} ? f\left(N_{i,l}^{TTI},\vec{e}\right)}{N_{i,l}^{TTI}} \right)$$

Since the criteria to be used is a monotonic function increasing with the number of DTX to be inserted, and as $x ? ?x?$ is also monotonic, it is not necessary to compute it on all the points in *E*, but it suffices to compute it on the graph *G* edging set *E*, as we are sure that the intersection between the set of minima of $C(\vec{e})$ and the graph is not empty. This minimising of $C(\vec{e})$ is done below:

Table 1

```
01:  $best\_criteria$ := +?
02:  for $e_{ini}$ := 1 à $e_{plus}$ do
03:      $e_{minus}$ := $G[e_{ini}]$
04:      $new\_criteria$ := C($e_{ini}$, $e_{minus}$, $e_{plusi}$, $s$)
05:      if $new\_criteria < best\_criteria$ then
06:          $(best\_criteria, best\_e_{ini}) := (new\_criteria, e_{ini})$
07:      end if
08:  end for
09:  $e_{ini}$ := $best\_e_{ini}$
10:  $e_{minus}$ := $G[best\_e_{ini}]$
```

Note that in case there are several minima on the graph G, table 1 ensures that the one chosen is that for which e_{ini} is minimum.

Complexity of algorithm 1

Generally speaking, $\max_{l\in TFS(i)} N_{i,l}$ is in the order of magnitude of hundred, and therefore so is the determined e_{plus}. So, it seems at a first glance that algorithm 1 is rather complex as you have to handle an array G of e_{plus} elements. In fact, it is possible to implement algorithm 1 with less complexity: as a matter of fact, the ratio $\max_{l\in TFS(i)} N_{i,l} \Big/ \min_{\substack{l\in TFS(i)\\ N_{i,l}\neq 0}} N_{i,l}$ is generally in the order of magnitude of several units, or tens. So the graphs G and GL comprise a number of stair-steps in the order of magnitude of $\#TFS(i)\cdot\left(\max_{l\in TFS(i)} N_{i,l} \Big/ \min_{\substack{l\in TFS(i)\\ N_{i,l}\neq 0}} N_{i,l}\right)$ that can be substantially less than e_{plus}. So instead of storing all the points with $e_{ini}\in\{1,\ldots,e_{plus}\}$, one can store only those point that are at edges of stair steps.

The same reasoning can be applied to the algorithm of table 1. One can determine for each stair step of graph G those values of of e_{ini} for which the number of DTX bits can vary for at least one TF. The criteria can then be computed only for these points.

Algorithm 2

Finally, one can use the simpler sub-optimal algorithm of table 2 hereinafter. We have not yet completely evaluated the impact of the simplification, but on the few example we have considered we could not find any degradation in terms of greater number of DTX bits inserted.

The simplification is illustrated on figure 7 below.

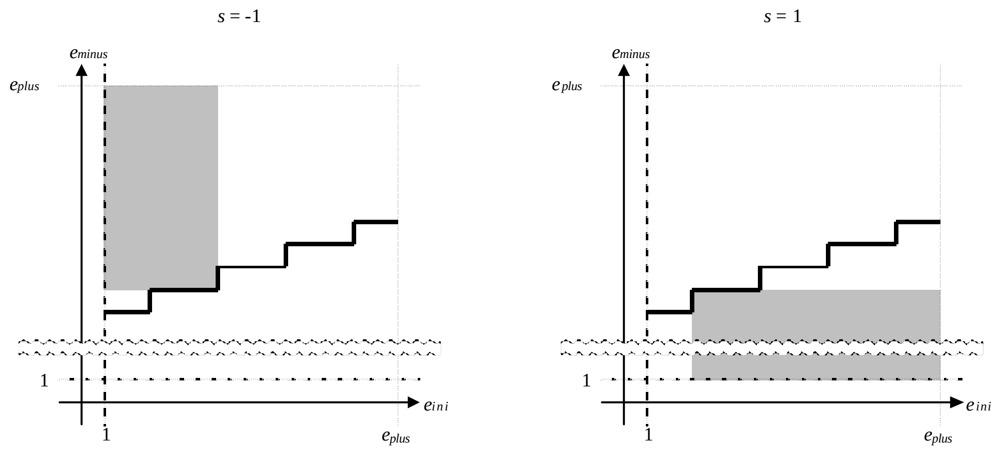


figure 7

Instead of considering the complete set E_l (resp. E) edged by a graph, we consider only a rectangular region that is a subset of E_l (resp. E). This retangular region is greyed on figure 7. The simplification is that to store the region instead of storing all the graph, you need just to store one corner of the rectangle, that is to say only one point.

On table 2, the consider subset of E_l est defined the point (new_e_{ini}, new_e_{minus}), whereas the subset of E is defined by the point (e_{ini}, e_{minus}). So, lines 6 and 7 are analogous to the determination of E_l, lines 9 and 10 analogous to the assignment of E_l to E, and lines 13 to 18 analogous to the intersecting de E et E_l.
The final value of (e_{ini}, e_{minus}), once all intersecting have been done is the found solution.

Table 2

```
01:  *first_time* :=**true**;
02:  **for all** $l$ **in** $TFS ? i ?$ **such that** $N^{TTI}_{i,l}$ ? 0 **do**
03:      $A := s ?? N^{TTI}_{i,l} ?e_{plus}$ ;
04:      $Q$ := $A$ **div** $N_{il}$
05:      $R$ := $A$ **mod** $N_{il}$
06:      $new\_e_{minus}$ := $Q$+1
07:      $new\_e_{ini}$ := $N_{i,l}$-R
08:      **if** *first_time* **then**
09:          $e_{minus}$ := $new\_e_{minus}$
10:          $e_{ini}$ := $new\_e_{ini}$
11:          *first_time* := **false**;
12:      **else**
13:          **if** $s ?new\_e_{minus}$ ? $s ?e_{minus}$ **then**
14:              $e_{minus}$ := $e_{minus}$
15:          **end if**
16:          **if** $s ?new\_e_{ini}$ ? $s ?e_{ini}$ **then**
17:              $e_{ini}$ := new_$e_{ini}$
18:          **end if**
19:      **end if**
```

```
20:     end for
```