ETSI TR 102 582 V1.1.1 (2007-07)

Technical Report

Terrestrial Trunked Radio (TETRA);
Evaluation of low rate (2,4 kbit/s) speech codec

Reference
DTR/TETRA-05131

Keywords
CODEC, radio, TETRA, voice

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Contents

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://webapp.etsi.org/IPR/home.asp).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This Technical Report (TR) has been produced by ETSI Technical Committee Terrestrial Trunked Radio (TETRA).

The present document provides the performance results of an investigation into the suitability of NATO's STANAG 4591 MELP speech codec for use in TETRA.

1 Scope

The present document presents the study carried out to evaluate the feasibility of using the 2,4 kbit/s MELP codec (i.e. STANAG 4591 codec) over TETRA channels.

2 References

For the purposes of this Technical Report (TR), the following references apply:

NOTE: While any hyperlinks included in this clause were valid at the time of publication ETSI cannot guarantee their long term validity.

[1] ITU-T Recommendation P.861: "Objective quality measurement of telephone-band (300-3 400 Hz) speech codecs".

[2] ETSI ETS 300 395-2: "Terrestrial Trunked Radio (TETRA); Speech codec for full-rate traffic channel; Part 2: TETRA codec".

[3] ITU-T Recommendation G.191: "Software tools for speech and audio coding standardization".

[4] Dr Michael Street, CIS Division NATO C3 Agency, The NATO Post-2000 Narrow Band Coder: Test and Selection of STANAG 4591.

[5] North Atlantic Treaty Organization, Standardization Agreement (STANAG).

[6] U.S. Department of Defense, Multi-Excited Linear Predictive Coder (MELP) Bit Stream Study, 15 February 2000.

3 Definitions and abbreviations

3.1 Definitions

For the purpose of the present document, the following terms and definitions apply:

Adaptive Multi-Rate (AMR) codec: speech and channel codec capable of operating at various combinations of speech and channel coding (codec mode) bit-rates

Average Protection Level (APL): metric for assessing the effectiveness of error protection applied to bits within codec frames. APL is dependent on bit distribution within codec frames

codec mode adaptation: control and selection of the codec mode bit-rates

3.2 Abbreviations

For the purpose of the present document, the following abbreviations apply:

ACELP	Algebraic Code Excited Linear Prediction
AMR	Adaptive Multi-Rate
APL	Average Protection Level
CRC	Cyclic Redundancy Check
FEC	Forward Error Correction (Coding)
FS	Frame Stealing
LSF	Line Spectral Frequency
MELP	Minimum Excitation Linear Prediction
MELPe	Minimum Excitation Linear Prediction enhancement
MOS	Mean Opinion Score

MSB	Most Significant Bit
PESQ	Perceptual Speech Quality Measure
RCPC	Rate Compatible Punctured Convolutional (Coding)
SNR	Signal to Noise Ratio
STANAG	Standardisation Agreement
TDMA	Time Division Multiple Access
TETRA	Terrestrial Trunked RAdio

4 General

4.1 Work requirements

It has been decided to use the 2,4 kbit/s mode of the STANAG 4591 codec.

In order to make assessments across the coverage area, rather than in error-free conditions, it is necessary to provide a representative FEC scheme and inject soft channel bit errors with a TETRA modem and radio channel simulation. In order to assess the performance of the codec, the PESQ tool has been used as it reflects the perceived user speech quality of the speech accurately.

4.2 Tasks

As part of this study the following tasks have been carried out:

1) Polynomial search for reducing the mother code rate to ¼.

2) Bit classification.

3) Puncturing investigations for achieving the required code rates.

4) Frame stealing investigations.

5) Performance evaluation using the PESQ tool.

5 Initial study of the TETRA speech Codec

5.1 Introduction

The testbench used is shown in figure 5.1 Note that the highlighted blocks in the figure 5.1 are irrelevant to the measurements mentioned in the present document.

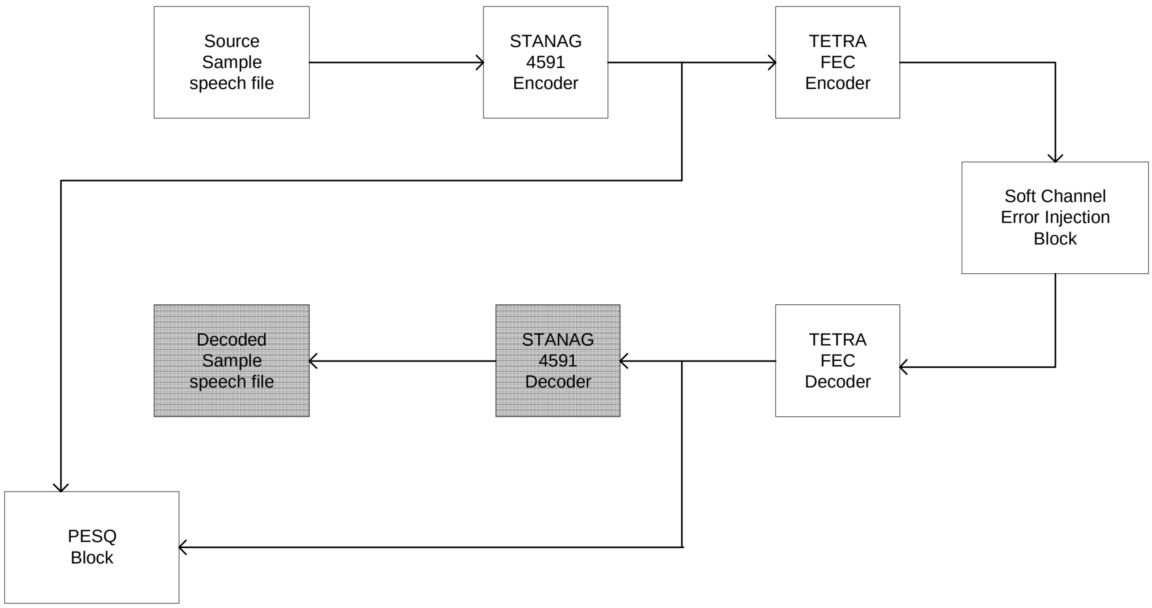

Figure 5.1: Testbed block diagram

5.2 Polynomial search for ¼ mother code rate

From an initial convolutional code used in the original TETRA codec which has a constraint length K=5 and a mother code rate of 1/3, the purpose was to find the best possible fourth polynomial to obtain a new mother code rate of 1/4 with acceptable performance. Indeed, adding a new polynomial will normally increase the error-correction capability of the convolutional code if it is well chosen. In the present clause, the selection criteria used will be explained.

First, let us summarize the properties of the original TETRA convolutional code. It is defined by the following three polynomials, a constraint length K=5 (4 shift registers). As there is no puncturing, its rate is 1/3, which is also known as the "mother code rate".

$G_1(D) = 1 + D + D^2 + D^3 + D^4$

$G_2(D) = 1 + D + D^3 + D^4$

$G_3(D) = 1 + D + D^2 + D^4$

$G_4(D) = ?$ The objective is to find the fourth polynomial.

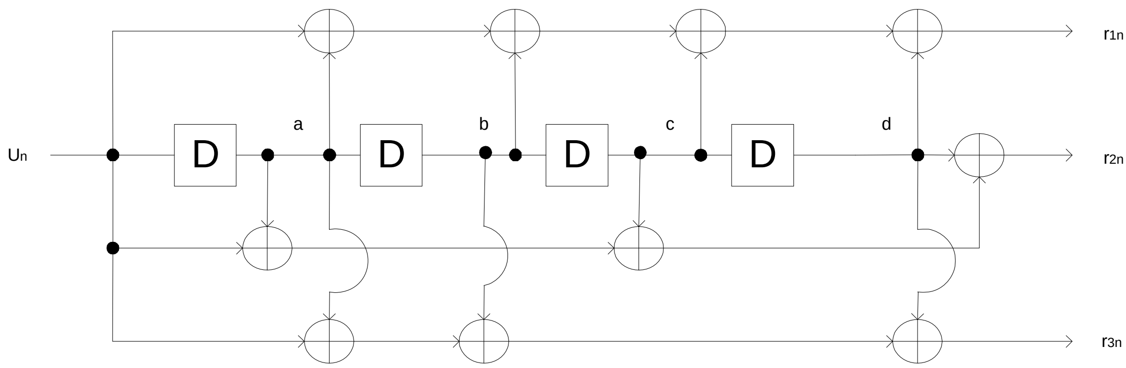

Figure 5.2: Original convolutional encoder structure

Before proceeding further, the notations introduced thus far will be explained first. U_n represents the input bit at time n. S_n is the state represented by "abcd" at time n. In other words, S_n represents the bits U_{n-1},U_{n-2},U_{n-3},U_{n-4}. R_n is the "$r_1 r_2 r_3$" codeword at time n of the branch leading from state S_n to S_{n+1} (represented by output r_{1n}, r_{2n}, r_{3n}). The outputs defined by the generator polynomials are given by the following relationships:

r1n = Un ⊕ Un-1 ⊕ Un-2 ⊕ Un-3 ⊕ Un-4

r2n = Un ⊕ Un-1 ⊕ Un-3 ⊕ Un-4

r3n = Un ⊕ Un-1 ⊕ Un-2 ⊕ Un-4

where ⊕ is the exclusive OR operator.

Table 5.1 shows all the states transitions of the original convolutional code for all possible information bit inputs:

Table 5.1: Original Convolutional Code State Transitions

U_n	S_n	S_{n+1}	R_n
0	0000	0000	000
0	0001	0000	111
0	0010	0001	110
0	0011	0001	001
0	0100	0010	101
0	0101	0010	010
0	0110	0011	011
0	0111	0011	100
0	1000	0100	110
0	1001	0100	001
0	1010	0101	000
0	1011	0101	111
0	1100	0110	011
0	1101	0110	100
0	1110	0111	101
0	1111	0111	010
1	0000	1000	111
1	0001	1000	000
1	0010	1001	001
1	0011	1001	110
1	0100	1010	010
1	0101	1010	101
1	0110	1011	100
1	0111	1011	011
1	1000	1100	001
1	1001	1100	110
1	1010	1101	111
1	1011	1101	000
1	1100	1110	100
1	1101	1110	011
1	1110	1111	010
1	1111	1111	101

The corresponding trellis structure is given in figure 5.2. There are $2^{K-1}=2^4$ =16 states in the trellis. One can see that the minimum free distance (d_{min}) is equal to 12. As we can describe a convolutional code by its trellis diagram, what we call the free distance (or minimum free distance), is the Hamming weight on the branches of the shortest path which diverges from the 0000 state and re-emerges with it. In general, the higher the minimum free distance is for a convolutional code, the better its error performance will be.

Adding a new generator polynomial will not change the number of states, but the mother code rate will drop to ¼. Consequently, the branch values will change with every bit added to each branch. Hence, the minimum free distance will, on average increase, allowing the encoder to have improved error performance.

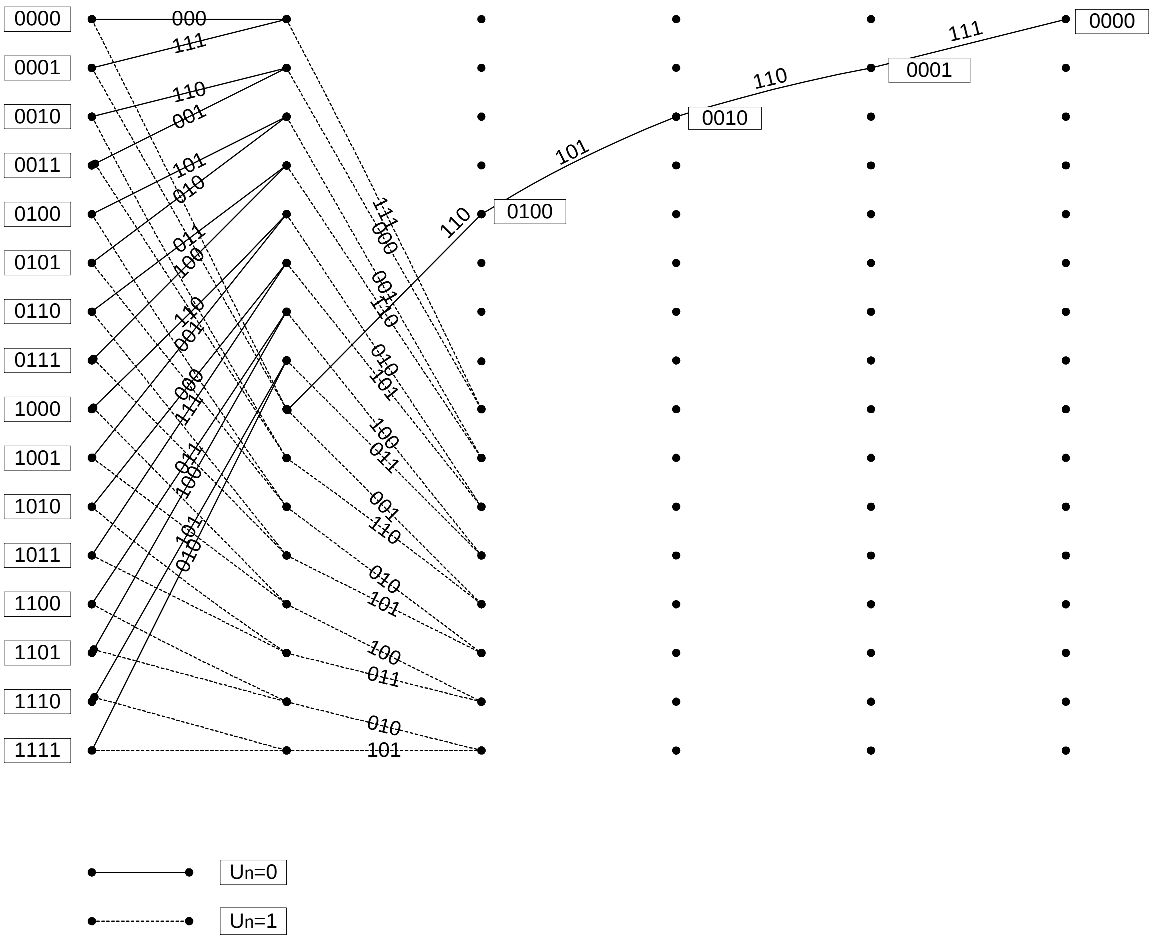

Figure 5.3: Trellis of Original Convolutional Code

In the remaining part of the present clause, the addition of an extra polynomial and the search criteria used to do this will be explained.

As there are 4 shift registers, the degree of polynomials used in this code is 4 or less. Also, we have to consider 31 possibilities (32 minus the all-zero polynomial which is irrelevant). Note that, reuse of any of the existing polynomials is not considered. As a result, there are 28 candidate polynomials to choose from.

Our polynomial suitability criteria is based on maximizing the free distance. A very important property is that the addition of a new generator polynomial will not change the path on which the minimum free distance is calculated. Therefore, to calculate the new free distance, we only need to know the new output values corresponding to the branches of the free distance path indicated on the trellis diagram presented earlier.

For each polynomial tested, we have to calculate the values of the new outputs introduced on the free distance path. This is illustrated in the table 5.2.

Table 5.2: Free distance problem in polynomial addition

U_n	S_{n-1}	S_n	r_{4n}
1	0000	1000	?
0	1000	0100	?
0	0100	0010	?
0	0010	0001	?
0	0001	0000	?

The new minimal free distance will be 12 plus the Hamming weight of the five parity bits.

Table 5.3 shows the results obtained. The second column ($G_4(D)$) lists all the candidate polynomials where a binary codeword 10011 represents $G_4(D) = 1 + D + D^4$ (the MSB of the codeword corresponds to the coefficient of D^4).

The third column lists the r_4 output described earlier.

EXAMPLE: 10100 means that the output value is 1 on the first trellis depth, 0 on the second, etc. Then the last column contains the minimum free distance provided by each polynomial.

Table 5.3: Free distance profile of candidate polynomials

	$G_4(D)$	Outputs $r_{4n}, r_{4n+1}\cdots$	d_{min}
1	00001	10000	13
2	00010	01000	13
3	00011	11000	14
4	00100	00100	13
5	00101	10100	14
6	00110	01100	14
7	00111	11100	15
8	01000	00010	13
9	01001	10010	14
10	01010	01010	14
11	01011	11010	15
12	01100	00110	14
13	01101	10110	15
14	01110	01110	15
15	01111	11110	16
16	10000	00001	13
17	10001	10001	14
18	10010	01001	14
19	10011	11001	15
20	10100	00101	14
21	10101	10101	15
22	10110	01101	15
23	10111	11101	16
24	11000	00011	14
25	11001	10011	15
26	11010	01011	15
27	11011	11011	16
28	11100	00111	15
29	11101	10111	16
30	11110	01111	16
31	11111	11111	17

The arrays 21, 27 and 31 (which are highlighted in grey in the table 5.3) are not considered, as those polynomials are identical to one of the original ones. Therefore, the maximum free distance value that cn be achieved is 16, provided by the following four polynomials:

$G_{4,1}(D)= 1 +D +D^2 +D^3$

$G_{4,2}(D)= 1 +D +D^2 +D^4$

$G_{4,3(}D)= 1 +D^2 +D^3 +D^4$

$G_{4,4}(D)= D +D^2 +D^3 +D^4$

Also, in order to determine which ones provide the best error performance, simulation data are needed.

5.3 Bit classification

The output bits from the STANAG 4591 Encoder are classified into 4 classes according to their sensitivity which is related to the importance of the information they contain. Each speech bit is classified as either Class 0 (minimum protection, code rate=2/3), Class 1 (code rate=4/9), Class 2 (code rate =1/3) and Class 3 (maximum protection, code rate=1/4). In order to make it compatible with the TETRA system, and to use it with the best possible performance, an algorithm was developed to calculate all feasible distribution of these bits.

5.3.1 Bit distribution constraints

The STANAG 4591 speech codec's operation mode is set to 2,4 kbit/s in this study and each speech frame is 22.5 ms long. Therefore, each speech frame contains 54 bits. In the TETRA system, each TETRA TDMA frame lasts for approximately 60 ms and contains 432 bits. It means that we can fit 3 STANAG speech frames into one TDMA frame, with an overflow of 7,5 ms, which is negligible if we assume that a delay less than 180 ms is acceptable. In fact, in order to delete the effects of this delay, we will use a 2 +3 +3 scheme where 2 speech frames are encoded in the first TDMA frame, then 3 speech frames in the second and third TDMA frames.

So in each TDMA frame, there will be 162 information bits when 3 speech frames are encoded and 108 information bits when 2 speech frames are encoded. In addition to the encoded bits, 8 CRC bits and 4 tail bits are added.

The CRC and tail bits are allocated to the most sensitive bits, so they must be encoded with the lowest code rate. Therefore, in order to use all the TDMA bits in a frame, and if we define k_i as the number of speech bits allocated in the Class C_i ($0 \le i \le 3$), we obtain the following relationships:

$$(k_0 \cdot \frac{1}{R_0}) + (k_1 \cdot \frac{1}{R_1}) + (k_2 \cdot \frac{1}{R_2}) + (k_3 \cdot \frac{1}{R_3}) + (12 \cdot \frac{1}{R_3}) = 432 \quad (1)$$

$$k_0 + k_1 + k_2 + k_3 = 162 \quad (2)$$

R_i represents the code rate for Class C_i. When R_0=2/3, R_1=4/9, R_2=1/3 and R_3=1/4 is substituted into the first relationship, the following is obtained.

$$(k_0 \cdot \frac{3}{2}) + (k_1 \cdot \frac{9}{4}) + (k_2 \cdot 3) + (k_3 \cdot 4) = 384 \quad (3)$$

Because the 162 information bits result from 3 speech frames, the number of bits in each class needs to be uniformly distributed between these 3 speech frames. Therefore, for Class i:

$$k_i = F_{i1} + F_{i2} + F_{i3} \quad (4)$$

And

$$F_{i1} = F_{i2} = F_{i3} \quad (5)$$

Here, F_{ij} is the number of bits in the j-th frame belonging to Class i. Consequently, it introduces the condition that the number of bits of each class k_i must be divisible by 3.

The bit distribution of the speech frames are summarized in the figure 5.4.

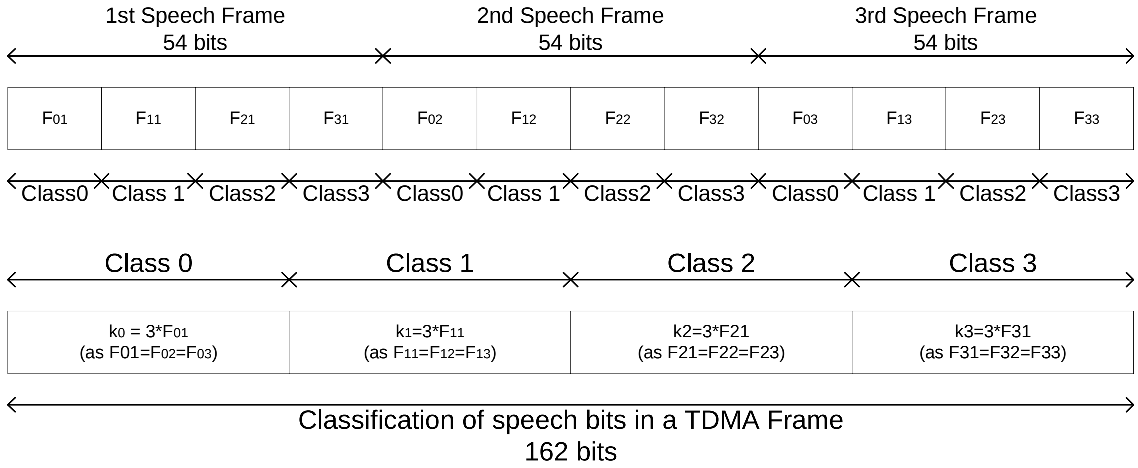

Figure 5.4: Bit distribution for 3 Speech Frames

For the 2 speech frames case, the number of bits in each class is simply equal to 2/3 of previously listed k_i.

5.3.2 Average Protection Level (APL) metric

Having explained the criteria of bit classification, next task is to determine all possible bit distributions in different classes. In order to carry out this task, a metric called the "average protection level" (APL) will be introduced first.

It should be noted that all of the 432 bits of the TDMA frame (only when 3 speech frames are encoded) need to be allocated. Alternatively, a combination of C_i bits could be used which would allocate fewer bits than 432, and then use zero padding. Moreover, a "physically possible" method is required: for a stream of bits belonging to a class, which implies that the number of encoded bits must be an integer.

Hence, a metric must be defined, to characterize the average protection level of the code, given the distribution of bits to different classes. This could give an indication about the protection desired but it will not be sufficient as the highest Average Protection Level (APL) may not necessarily give the best performance in the listening and the PESQ performance tests. In order to measure the contribution of each class of bits, the APL metric is defined as follows:

$$\mathrm{APL}(\%) = \frac{\left(\sum_{i=0}^{3}(1-Ri)\times ki\right)}{\left(\sum_{i=0}^{3} ki\right)}\cdot 100 \qquad (6)$$

As we do not consider the CRC and tail bits in this calculation, they do not appear in the number of bits k_i. Indeed, CRC bits are for error detection rather than correction and hence they are not treated as error correction functions.

According to the above, if the average code rate tends to 1 (no coding), the Average Protection Level (APL) tends to 0 %, and if the code rate tends to 0 (theoretical maximum coding) the APL tends to 100 %.

The APL algorithm has been used taking all the bit distribution constraints into account The results indicate that under the conditions defined above, there are 49 combinations of bit distribution. It should be noted that not all of those distributions may be useful as some protection classes are not used. The average protection level values vary between 48 % and 58 % for all valid distributions.

The results of bit partitioning are listed in the table 5.4:

Table 5.4: APL metric results

Distribution Index	R_0(2/3)	R_1(4/9)	R_2(1/3)	R_3(1/4)	APL (%)
1	0	144	12	6	57,098 766
2	6	132	18	6	56,687 241
3	12	120	24	6	56,275 719
4	12	132	3	15	55,915 638
5	18	108	30	6	55,864 197
6	18	120	9	15	55,504 116
7	24	96	36	6	55,452 675
8	24	108	15	15	55,092 590
9	30	84	42	6	55,041 153
10	30	96	21	15	54,681 068
11	30	108	0	24	54,320 988
12	36	72	48	6	54,629 627
13	36	84	27	15	54,269 547
14	36	96	6	24	53,909 466
15	42	60	54	6	54,218 105
16	42	72	33	15	53,858 025
17	42	84	12	24	53,497 940
18	48	48	60	6	53,806 583
19	48	60	39	15	53,446 503
20	48	72	18	24	53,086 418
21	54	36	66	6	53,395 061
22	54	48	45	15	53,034 977
23	54	60	24	24	52,674 896
24	54	72	3	33	52,314 816
25	60	24	72	6	52,983 540
26	60	36	51	15	52,623 455
27	60	48	30	24	52,263 374
28	60	60	9	33	51,903 290
29	66	12	78	6	52,572 014
30	66	24	57	15	52,211 933
31	66	36	36	24	51,851 852
32	66	48	15	33	51,491 768
33	72	0	84	6	52,160 492
34	72	12	63	15	51,800 411
35	72	24	42	24	51,440 327
36	72	36	21	33	51,080 246
37	72	48	0	42	50,720 165
38	78	0	69	15	51,388 889
39	78	12	48	24	51,028 805
40	78	24	27	33	50,668 724
41	78	36	6	42	50,308 640
42	84	0	54	24	50,617 283
43	84	12	33	33	50,257 202
44	84	24	12	42	49,897 118
45	90	0	39	33	49,845 676
46	90	12	18	42	49,485 596
47	96	0	24	42	49,074 074
48	96	12	3	51	48,713 989
49	102	0	9	51	48,302 467

The above bit distributions need to be tested in order to determine which ones provide the best error correction performance which will be discussed in the results clause.

5.4 Puncturing Patterns

The present clauseaddresses the puncturing pattern selection, which enables to obtain higher code rates from a Convolutional Code with a mother code rate of 1/4. A low rate 1/n convolutional code (called the mother code) is periodically punctured with period p to obtain a family of codes with rate p/v, where v can be varied between p+1 and np.

As an example, we consider punctured convolutional codes obtained from a rate 1/4 mother code. To generate a p/v punctured convolutional code (p/v > 1/4), we delete (4p-v) bits every 4p code bits corresponding to the encoded output of p information bits by the original rate ¼ code. The resulting rate is then equal to the desired rate r=p/v. For example, if we want to obtain an 8/18 code rate from a ¼ mother code, we have to delete 14 bits ((4x8)-18) every 32 bits.

The deleted bit pattern must be carefully selected to obtain desirable performance. The puncturing pattern is represented by a puncturing matrix. For a chosen puncturing period p, and an original 1/n mother code rate, the size of a puncturing matrix will be (n,p). A puncturing matrix is filled with ones and zeros, a "1" is allocated for a transmitted bit and a "zero" for a deleted bit. If we want to obtain a p/v code rate, the puncturing matrix will be filled with v ones.

To aid the explanations in the present clause, the following example is provided for the user.

EXAMPLE: a puncturing matrix from a ¼ mother code rate, which provides an 8/18 code rate.

$$A1 = \begin{bmatrix} 11111111 \\ 01011010 \\ 10000101 \\ 00111000 \end{bmatrix}$$

In this example, the puncturing period is 8. A convolutional code with a mother code rate of ¼ means that for one input, 4 outputs are generated by 4 different polynomials. Each line of the puncturing matrix concerns the outputs generated by a polynomial, and each column indicates which outputs will be transmitted or deleted for an input. In our example, when the encoder receives the first input, only the outputs generated by the first and third polynomials will be computed and transmitted. For the second input, the outputs of the first and second polynomials are transmitted, etc. For the input eight, the outputs of the first and third polynomial are transmitted, and when the ninth input arrives, it acts like if it was the first one (because the puncturing period is 8).

The concept of Rate Compatible Punctured Convolutional (RCPC) codes has been introduced by Hagenauer in 1988: it adds a rate-compatibility restriction rule to the puncturing rule. It implies that all the code bits of a high rate code of the family are used for the lower rate codes. Let $p(r_1)$ and $p(r_2)$ be the puncturing matrices of two rate-compatible codes (r_1 and r_2 are the code rates, with $r_1 > r_2$). The restriction rule means that if an element of $p(r_1)$ is equal to one ($p_{ij}(r_1)=1$), then the same element in $p(r_2)$ is also equal to one ($p_{ij}(r_2)=1$).

But obviously, deleting outputs of an original code to obtain higher code rates will degrade the error correction performance of the code. Now that we have introduced the puncturing process, we will present how the puncturing patterns have been chosen to minimize the degradation in error correction of the RCPC code.

The search for a good code (leading to low bit error rates) is a complex task. It is not evident that the best codes without puncturing lead to the best codes with puncturing. No constructive method is known for determining the puncturing matrices of a RCPC family. However, the intuitive approach to try to obtain performed punctured codes is to keep the minimum free distance as high as possible while constructing each matrix for each punctured code.

In an earlier clause, search for a good fourth polynomial was performed by trying to maximize the free distance on the path on which it is calculated. Table 5.5 presents the branch values generated by the three existing polynomials and the additional fourth, on the free distance path. One can see that the free distance path length is 5 and each column contains the branch values of each transition between states. For example, the second column value is 110 which means that on this branch, the output generated by the first polynomial is 1, the output of the second polynomial is 1, and the output of the third polynomial is 0. According to this, the output generated by the fourth polynomial is listed in table 5.5.

Table 5.5: Free Distance Path Branch Values

Polynomial 4	111**x**	110**x**	101**x**	110**x**	111**x**
$1+X+X^2+X^3$ (1E)	1	1	1	1	0
$1+X+X^2+X^4$ (1D)	1	1	1	0	1
$1+X^2+X^3+X^4$ (17)	1	0	1	1	1
$X+X^2+X^3+X^4$ (0F)	0	1	1	1	1

It should be noted that the four polynomials listed in table 5.5 are the ones which provide the maximum free distance of 16 as described earlier.

So for each polynomial and each punctured code rate desired (2/3, 8/18, 1/3), the puncturing matrix must be constructed to keep the minimum free distance as high as possible. This means that we must try not to delete weight 1 outputs on the free distance path.

Because the puncturing period is chosen as 8 and is different from the free distance path length (which is 5), the method to build the matrix is to try to maximize the free distance in "neighbouring 4 columns" of the matrix, in order to have a "weight balanced" matrix. For example, following are the puncturing matrices of the four polynomials, for rates 2/3, 8/18 and 1/3.

Polynomial 1E:

$$A0(R=\frac{2}{3})=\begin{bmatrix}11111110\\00001001\\00000010\\01010000\end{bmatrix}\quad A1(R=\frac{8}{18})=\begin{bmatrix}11111110\\00001001\\10101111\\01110000\end{bmatrix}\quad A2(R=\frac{1}{3})=\begin{bmatrix}11111111\\01011111\\10101111\\11110000\end{bmatrix}$$

Polynomial 1D:

$$A0(R=\frac{2}{3})=\begin{bmatrix}11111111\\00001000\\00000001\\01100000\end{bmatrix}\quad A1(R=\frac{8}{18})=\begin{bmatrix}11111111\\01010010\\10000101\\01101001\end{bmatrix}\quad A2(R=\frac{1}{3})=\begin{bmatrix}11111111\\01011010\\10101111\\11110101\end{bmatrix}$$

Polynomial 17:

$$A0(R=\frac{2}{3})=\begin{bmatrix}11111111\\01001000\\00000000\\00100010\end{bmatrix}\quad A1(R=\frac{8}{18})=\begin{bmatrix}11111111\\01001010\\10000001\\01110110\end{bmatrix}\quad A2(R=\frac{1}{3})=\begin{bmatrix}11111111\\11011110\\10100001\\01111111\end{bmatrix}$$

Polynomial 0F:

$$A0(R=\frac{2}{3})=\begin{bmatrix}11111111\\01010000\\00000101\\00000000\end{bmatrix}\quad A1(R=\frac{8}{18})=\begin{bmatrix}11111111\\01011010\\10000101\\00111000\end{bmatrix}\quad A2(R=\frac{1}{3})=\begin{bmatrix}11111111\\11011010\\10100101\\01111111\end{bmatrix}$$

5.5 Integration and Testing

At the beginning of the project the C-Codes of the STANAG 4591 Speech Codec and of the TETRA FEC Codec were made available. The required modifications to carry out this project were applied to the FEC codec.

The TETRA FEC Codec has been designed to work with its own speech codec and the AMR speech codec for use in TETRA. It contains 4 independent parts:

- speech source encoder;
- speech source decoder;
- channel encoder;
- channel decoder.

Each of these different parts will be explained in the context of how they fit in with this study.

5.5.1 Speech encoder

As we want to use the STANAG Speech Codec, this part of the TETRA speech codec was not used. Nonetheless, understanding it is needed and will assist in the study. Like all speech codecs, the TETRA version encodes different information of an audio frame prior to sending them through a channel. The structure of the output file delivered by the speech codec will be briefly discussed without much detail.

The speech source coder works with frames of 240 16-bit samples of an audio file, and for each frame gives 138 vectors of 16-bits. Each vector corresponds to an encoded bit of a computed speech parameter. In fact, in this output frame of 138 vectors, the first vector is always a null vector (it represents a BFI bit: bad frame indicator bit), followed by 137 vectors coding the audio frame parameters. This structure is presented in figure 5.5.

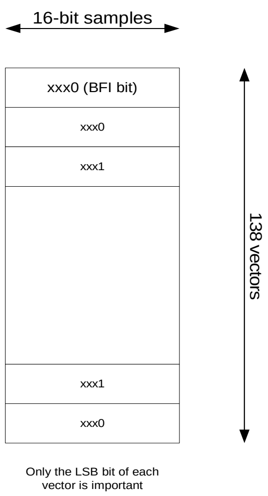

Figure 5.5: TETRA Speech Frame

The output file is a concatenation of such frames.

5.5.2 Channel encoder

The channel encoder performs 4 key functions:

- reading the input file;
- encoding the information bits;
- interleaving the data;
- writing the output file.

The most complex function is the encoding of information bits.

The channel encoder has two modes: the first one, called the "normal mode", processes two speech frames and encodes them. The second mode, called the "frame stealing mode" processes only one speech frame. For the moment, we will study only the normal mode and we will see later how to implement the frame stealing mode in order to use it with the STANAG 4591 codec.

The encoder reads two encoded speech frames, skipping the BFI before each frame, and places the bits in an array of 274 elements.

The first thing to note here is, in the case of the STANAG 4591 Codec, a TDMA frame must contain 2 or 3 speech frames and each speech contains 54 bits. It should be noted that what we call bits in this part will in reality refer to arrays of 16 bits. Therefore, each frame containing 54 bits means 54 vectors of 16 bits. This use of 16-bits vectors is preferred for compatibility with the Soft Channel Error Injection Block.

First part of the encoder performs the initialization of the parameters of the RCPC Code. The state transitions are defined and the code word values of each branch of the trellis state are computed using the three generator polynomials. At this stage, the fourth generator polynomial is defined in order to compute the fourth bit of the code word of each trellis branch. Once this is done, the RCPC encoder is ready to work.

The information bits are then classified into three classes. In our case, there are four classes; therefore some modifications are required. The number of bits allocated in each class will change, depending on bit distribution chosen for the tests. Following are the original bit classification and then the new one.

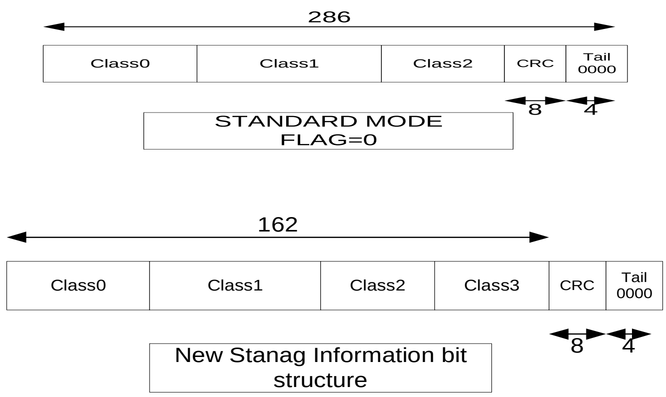

Figure 5.6: TETRA data classifications

The CRC and tail bits are also appended to the frame as indicated above. This is followed by the RCPC Coding of the information bits. Initially, the class 0 bits were not coded, just copied to the output, the class 1 bits were coded with a 2/3 rate and the class 2 bits were coded with a rate of 4/9. The new RCPC Coding encodes class 0 bits at 2/3 rate, class 1 bits at 4/9 rate, class 2 with rate 1/3 and class 3 with rate 1/4. Therefore, a new puncturing matrix, which allows generates rate 1/3 needs to be defined.

The encoded data are then interleaved. Eventually, the data are written to a file, in TETRA frame format. The 432 output values representing "0" and "1" are -127 and +127, respectively. This is required for compatibility with the following Soft Channel Error Injection Block.

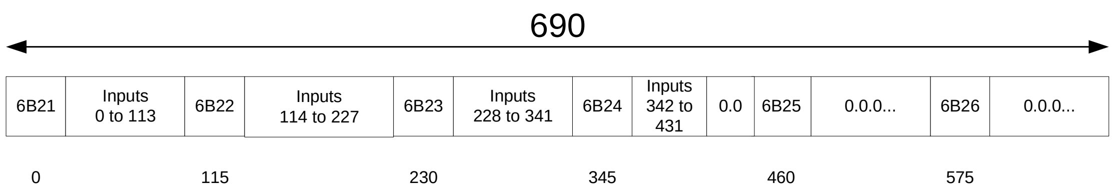

Figure 5.7: TETRA frame structure

5.5.3 Channel decoder

The modifications of the channel decoding algorithm are the same as those explained in channel encoder. Therefore, these concepts will not be covered in the present clause to avoid repetition.

5.5.4 Speech decoder

Finally, the TETRA Speech decoder has to reconstruct the audio file. For each frame, it extracts the 137 bits to construct the parameters of the speech frame prior to the synthesis of the original audio file. For each speech frame, the BFI bit is first checked: if its value is "1", it means that the frame is either meaningless (in frame stealing mode) or it is corrupted (i.e. an error has been detected by the failure of the CRC test) (see figure 5.7). In this case, the bit parameters of the previous frame are copied and used to synthesize the audio frame.

Figure 5.8: TETRA speech decoder frame

The STANAG 4591 Speech Decoder has a similar mode: if a frame erasure is detected, the bit parameters of the former frame are copied to the current one. A frame erasure occurs in the following 2 cases.

When the bits are received, 7-bit pitch information is decoded first, as they contain the mode information. If the pitch code is all-zero or has only one bit set, then the unvoiced mode is used. If two bits are set, a frame erasure is indicated.

In the unvoiced mode, some bits have been encoded with two Hamming codes (8,4) and (7,4). The (8,4) Hamming code is decoded to correct single bit errors and to detect double errors. If an uncorrectable error is detected, a frame erasure is indicated. Otherwise, the (7,4) Hamming codes are decoded, correcting single errors bit without double error detection.

The STANAG 4591 Codec has its own frame error detection function, which is not compatible with TETRA's FEC. The STANAG Speech decoder does not have a specific one in its 54 information bits that indicates that a frame should be ignored or is corrupted (i.e. a BFI bit). Therefore, at this stage, a frame stealing mode cannot be implemented in the TETRA FEC Codec, which could be compatible with the STANAG Codec, unless the source code of STANAG is changed. Moreover, the CRC bits become meaningless here, as the STANAG Codec does not use this information about a possible error in a frame. We will see how to implement the Frame stealing (FS) mode in clause 5.5.5.

5.5.5 Frame Stealing Mode - CRC Test

The TETRA Channel Codec contains a second mode, in which only a single speech frame is encoded. The frame stealing process is periodically used to replace the contents of a half slot of data with synchronization information. Figure 5.9 shows the speech decoder input data in FS mode.

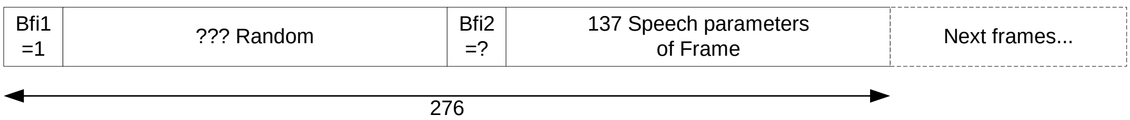

Figure 5.9: Input to the speech decoder in FS mode

When the speech decoder receives the file, it checks the first Bfi bit value (which is 1 as indicated in figure 5.9) which indicates that the FS mode is used. Hence, the parameters of the previous frame (and some default values for certain parameters) are copied to this frame and the speech is then synthesized.

In order to adapt the FS mode to the STANAG Codec, some modifications were needed. When working with the STANAG Codec, the TETRA Channel Codec normally encodes and decodes 3 speech frames. We have decided to encode only two speech frames in the FS mode and to leave the first one empty. The figure 5.10 illustrates the packet of 3 frames just after the channel decoding part:

Bfi1 =1	??? Random	Bfi2	54 speech parameters	Bfi3	54 speech parameters

165

Figure 5.10: STANAG frames decoded in FS mode

The STANAG 4591 does not process a BFI check simply because no BFI bit is sent to the speech decoder but only the speech parameters. Therefore, we have decided to simulate an FS mode, instead of checking the FS mode at the speech decoder (as done in the TETRA speech decoder). This is performed at the output of the channel decoder. If the first BFI bit is 1 (as in figure 5.10), the 54 speech parameters are the previous frame are copied to this one. The written file (without the BFI bits) is then sent to the speech decoder.

Concerning the CRC test, the procedure is as follows. Basically, each of the eight CRC bits is computed as being the exclusive OR of bits. After the decoding operation, the decoded CRC bit is compared to the exclusive OR of the defined bits. If the two values are the same, the decoder concludes that there is no error. If not, then an error is declared and the two BFI bits preceding the two decoded speech frames are set to one (in the case of the original channel decoder version) and the TETRA speech decoder copies the values of the previous frame's parameters (as in the FS mode).

6 Performance Evaluation

6.1 Evaluation Criteria

The performance evaluation platform used for tests is presented in the figure 6.1.

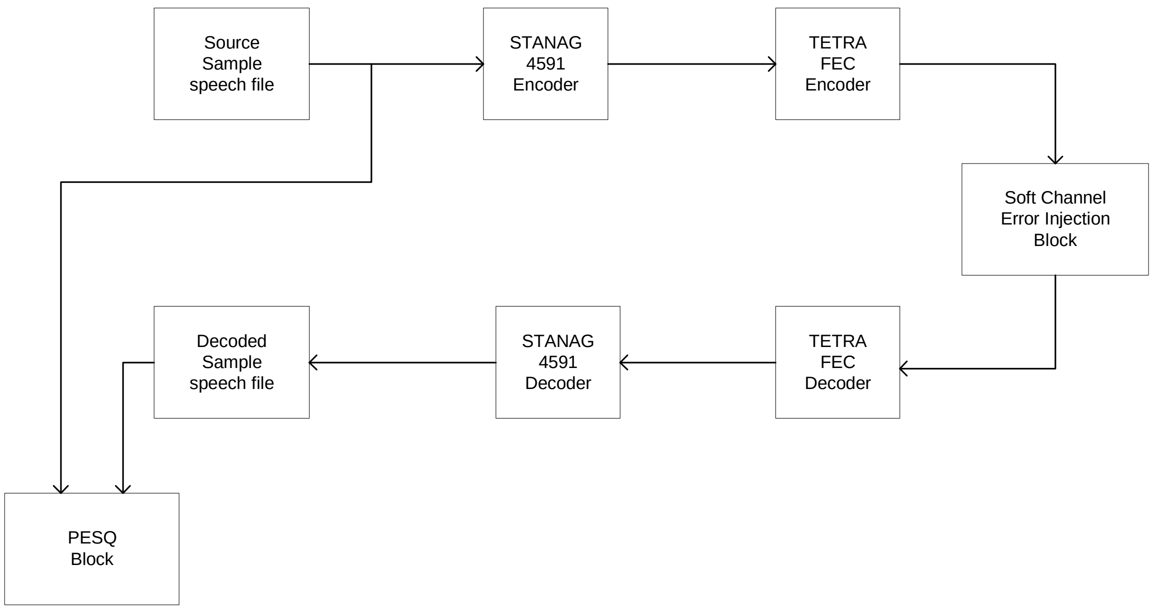

Figure 6.1: STANAG 4591 Performance Evaluation Platform

The Soft Channel Error Injection Block simulates four different noise types:

- Static Channel, with a signal on noise ratio (SNR) from 4dB to 10 dB in 1dB steps.
- Typical Urban environment, with a user moving at 5 km/h and a SNR from 10dB to 24dB in 2 dB steps.
- Typical Urban environment, with a user moving at 50 km/h and a SNR from 10 dB to 24 dB in 2dB steps.
- Hilly Terrain environment, with a user moving at 200 km/h and a SNR from 10 dB to 24 dB in 2dB steps.

Also the Perceptual Evaluation of Speech Quality (PESQ) tool is used to compare the degraded speech file with the original one. PESQ is the new International Telecommunication Union Standardization Sector (ITU-T) standard for measuring the voice quality of communications networks. Basically, the PESQ tool performs subjective tests by providing the mean mark that would mimic a group of human listeners comparing the degraded speech file with the original one. The PESQ process (prediction of perceived speech quality) is very complex and belongs to a still active research field, and its analysis is not a part of this study.

PESQ provides an output score, called Mean Opinion Score (MOS), which ranges from 1 to 5. But because people always hesitate to give a maximal mark even if the quality of a degraded file is perfect, experience shows that MOS actually ranges from 1 to 4,5. Listening tests during the study have experimentally shown that the intelligibility of a speech a file with a MOS score below 2 is very poor and below 1,6 it is almost inaudible.

We have also observed that the encoding and decoding of the speech file only by the STANAG Codec provides a MOS score of 3,119 with a slight deterioration of the original file but the degraded file still has a good quality and is intelligible.

In the following part, PESQ and listening tests under different noisy conditions are performed in order to determine which bit distribution and which polynomial provides the best performance.

The following tests have been performed as part of this study in the given time:

- Four bit distributions (based on APL metric).
- Four polynomials with the highest free distance.
- For each polynomial (associated with its puncturing pattern), PESQ scores have been obtained for TETRA channels (and all different SNR possible) provided by the Soft Error Injection Block.
- For each polynomial tested, the following 3 modes of encoding/decoding patterns have been used:
  - without frame stealing;
  - frame stealing with a 10 %;
  - frame stealing at 20 %.

Following are the four bit distributions chosen for the tests:

Table 5.6: Tested Bit Distributions

Distribution index	R_0=2/3	R_1=4/9	R_2=1/3	R_3=1/4	APL Metric (%)
1	6	132	18	6	56,687 241
2	36	84	27	15	54,269 547
3	60	36	51	15	52,623 455
4	90	12	18	42	49,485 596

These four bit distributions have been chosen in order to have a complete view of the interdependence between performance and the type of distribution. Indeed, one can see that the bit distributions are classified in a descending order of the APL metric. The first distribution has very few bits belonging to Class 0 (the lowest protection), as well as very few bits belonging to Class 3 (the highest protection). The majority of the bits belong to the intermediate protection classes.

On the contrary, the last bit distribution has many bits encoded with the lowest code rate ¼ (most protected bits) while many bits belong to the least protected class too.

The polynomials tested are listed below:

$G_{4,1}(D)= 1 + D + D^2 + D^3$(1E in hex format)

$G_{4,2}(D)= 1 + D + D^2 + D^4$(1D in hex format)

$G_{4,3}(D)= 1 + D^2 + D^3 + D^4$ (17 in hex format)

$G_{4,4}(D)= D + D^2 + D^3 + D^4$ (0F in hex format)

Hereon, the above polynomials will be referred to by their "hexadecimal" value (1E, 1D, 17, 0F).

Each polynomial is associated with a set of puncturing patterns which was listed in clause 5.

For testing purposes, a strategy needs to be adopted for CRC test. The CRC test is used to detect errors after decoding. With RCPC codes, the CRC test usually checks errors with the most protected bits. In the case of the TETRA Channel Codec, when, a CRC test fails, the BFI bit is set and the parameters of the current speech frame (LSF coefficients, pitch, gain, etc.) are replaced by those in the previous one. However, because the STANAG Codec does not include this feature, an alternative method is used. Detection of errors in bits that belong to classes 2 and 3 was the preferred approach in this study. In the event of error detection, several actions can be taken.

Recall that the TETRA Channel Codec works with 3 STANAG speech frames. The CRC test detects errors among bits of Class 2 and Class 3 but does not provide any information on which speech frames the corrupted bits belong to. Therefore, the first solution is to replace the 162 bits of the current three speech frames with the previous three in case of a CRC failure, before sending them to the STANAG speech decoder. This is illustrated in figure 6.2.

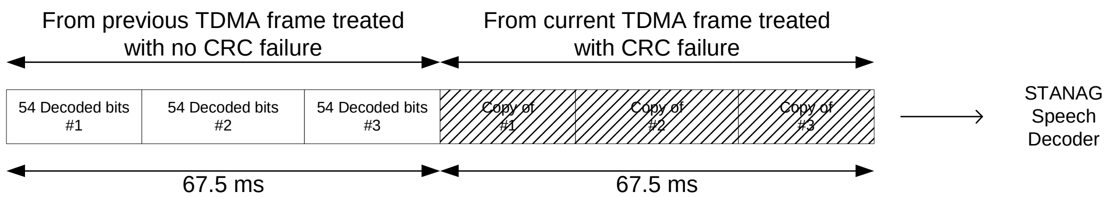

Figure 6.2: Triple Frame Replacement

The second solution involves replacing three frames in error by the last three of the previous group, in order to shorten delays in the decoded audio file as indicated below.

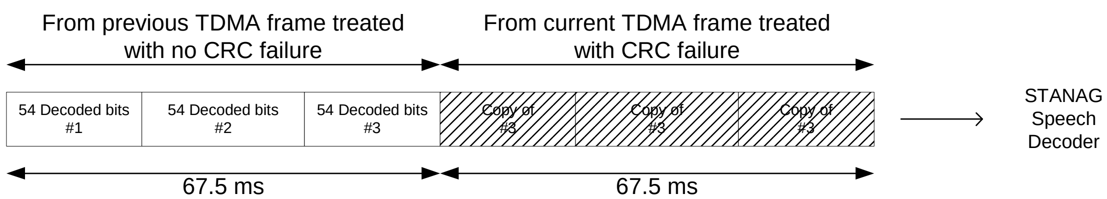

Figure 6.3: Single Frame Replacement

Because the STANAG Codec is a low rate Speech Codec, replacing one or several speech frames could cause significant degradation. The third solution proposed is to ignore the CRC test – despite failures - and to leave the frames as are (see figure 6.4).

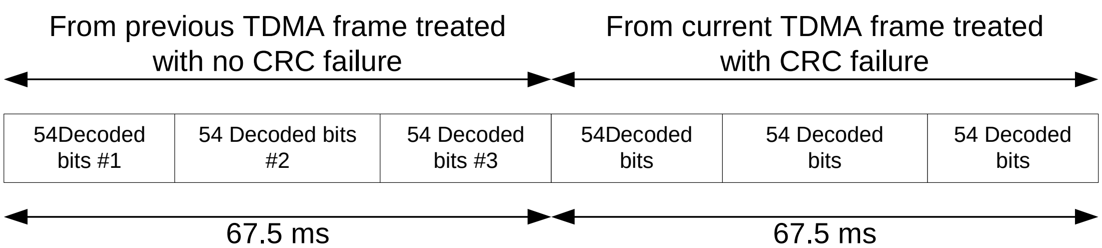

Figure 6.4: No Frame Replacement

To decide which solution is the most suitable, tests have been performed with the first bit distribution and polynomial "17". The results prove that the third solution (no replacement of frames), provide the best results. At high SNR, the performance of three options are quite similar, however with increased noise levels, the PESQ score is slightly improved with the third solution. This is combined with a slight amelioration in the listening test, which is not very audible. Therefore, this solution has been chosen for all tests. This implies that the CRC bits have no purpose in this study. A further work on the utilization of the CRC bits or adoption of an alternative strategies are left as part of future investigations.

Concerning the bit sensitivities of the STANAG 4591 Codec, the classification proposed by a previous study has been used. According to this report, 24 of the 54 bits in a speech frame need to be considered as the most significant, i.e. those need to be the most protected ones. Therefore, these bits have been placed in the highest protection classes available within the bit distributions selected. Following is a list of the number of bits allocated to the parameters and the number of these bits qualified as "the most significant" in a STANAG speech frame.

LSF Coefficients: - Stage 1: 7 of 7 bits protected

Line spectrum: - Stage 2: 4 of 6 bits protected

- Stage 3: 0 of 6 bits protected

- Stage 4: 0 of 6 bits protected

Fourier Magnitudes: 0 of 8 bits protected

Pitch: 6 of 7 bits protected

Bandpass Voicing: 1 of 4 bits protected

Aperiodic Flag (AF): 1 of 1 bit protected

Synchronization bit (Sync): 1 of 1 bit protected

Gain 2: 4 of 5 bits protected

Gain 1: 0 of 3 bits protected

TOTAL: 24 of 54 bits protected

Among these protected and unprotected bits, a "qualitative" classification has also been used in order to always protect some of them as much as possible. According to this investigation, the following bits were also identified as sensitive:

- LSF Stage 1 bits (MSBs), Pitch bits (MSBs), BPVC (MSB).
- LSF Stage 1 bits (LSBs), Pitch bits (LSBs), Gain2 bits (MSBs).
- LSF Stages 3 and 4 bits.
- Gain1 bits and Fourier magnitude bits.

The definition of the Most Significant Bits (MSBs) of a parameter (e.g. Pitch), is flexible: it can be 2, 3, 4 etc. The meaning of this is, when protecting a speech parameter, the priority is obviously to protect the MSB of this parameter before others.

6.2 Results

The results of the PESQ tests show that among the 4 bit distributions tested. It should be noted that the reference to the bit distributions in the plots is provided for one speech frame rather than three. Therefore, a bit distribution of 2-44-6-2 for a single frame is equivalent to 6-132-18-6 for three speech frames. Table 5.6 provided in the previous clause has been given for three speech frames.

Please note that the present clause provides the plots for the best polynomial 1E. The complete simulation data for all polynomials have been provided in Annex A due to its large size.

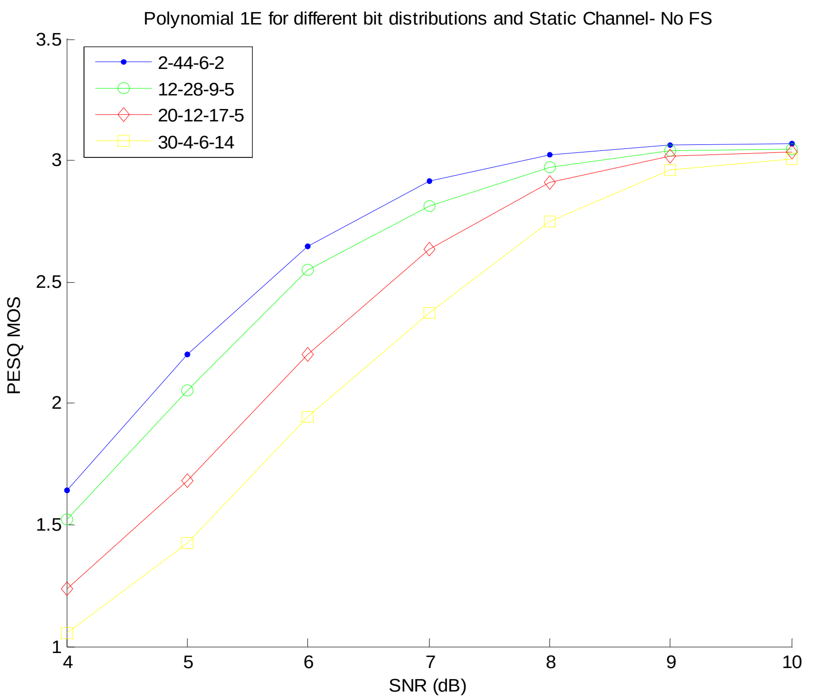

Figure 6.5: Polynomial 1E, Static Channel, No FS

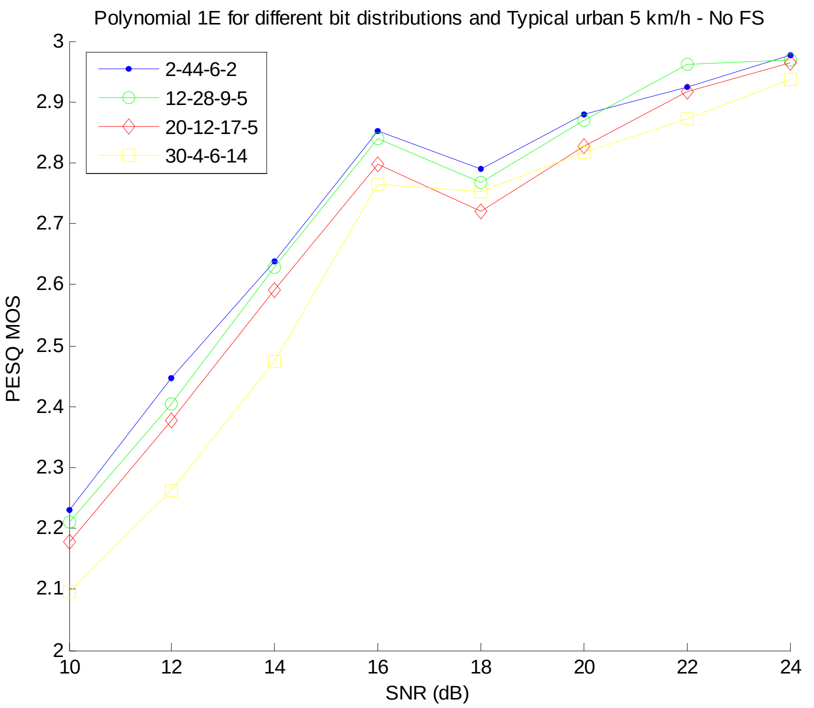

Figure 6.6: Polynomial 1E, TU5 Channel, No FS

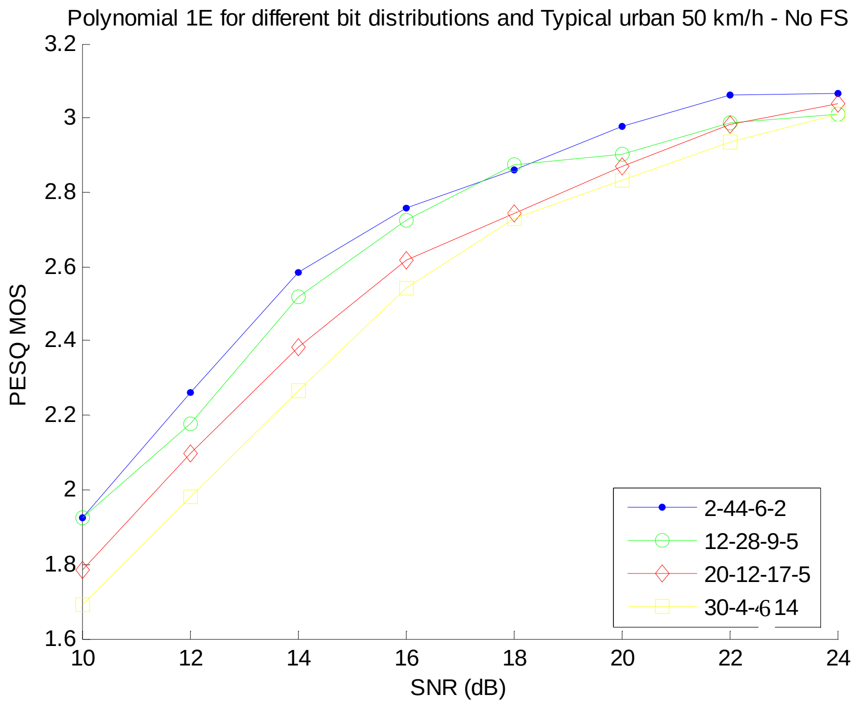

Figure 6.7: Polynomial 1E, TU50 Channel, No FS

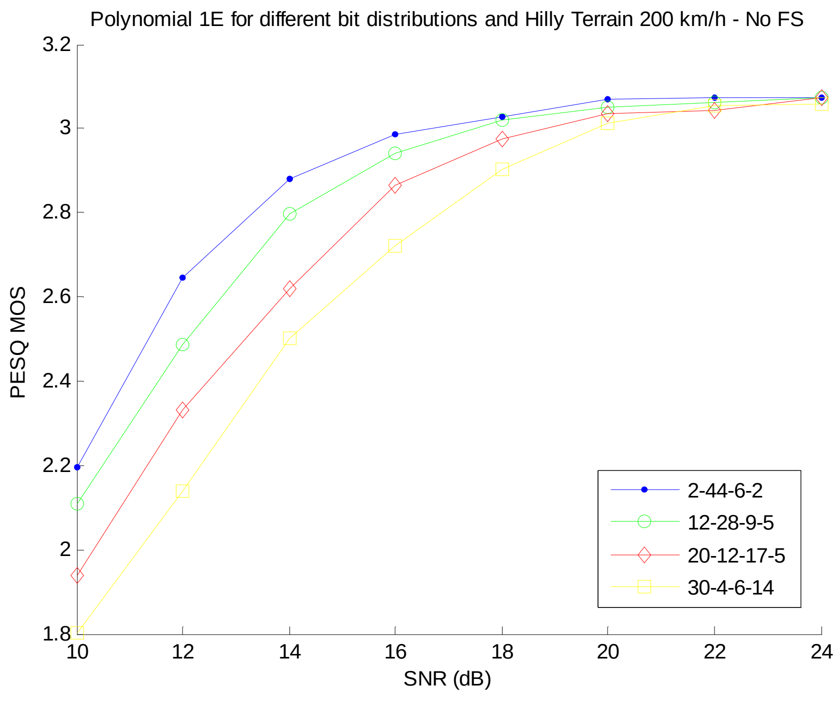

Figure 6.8: Polynomial 1E, HT200 Channel, No FS

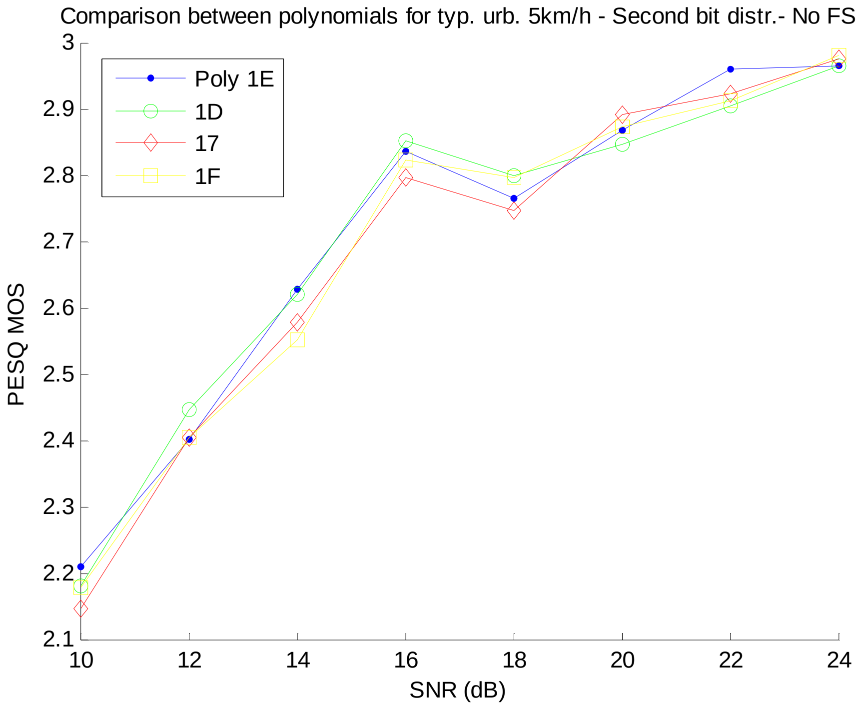

Figure 6.9: Distribution 12-28-9-5 Performance, TU5 Channel, No FS

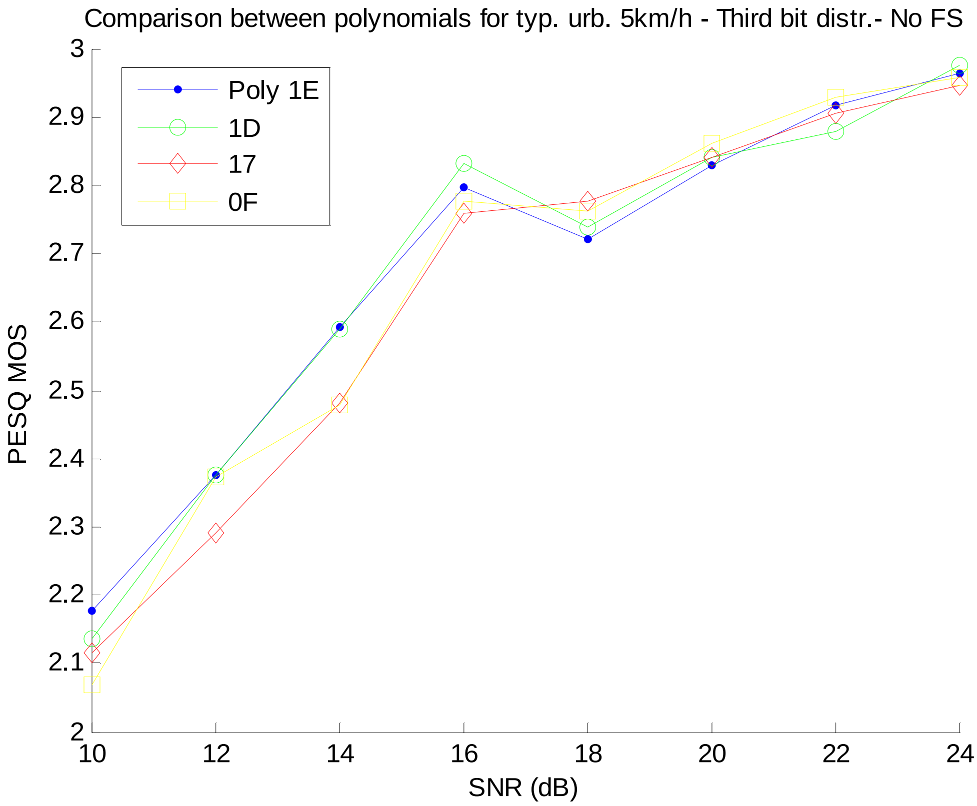

Figure 6.10: Distribution 20-12-17-5 Performance, TU5 Channel, No FS

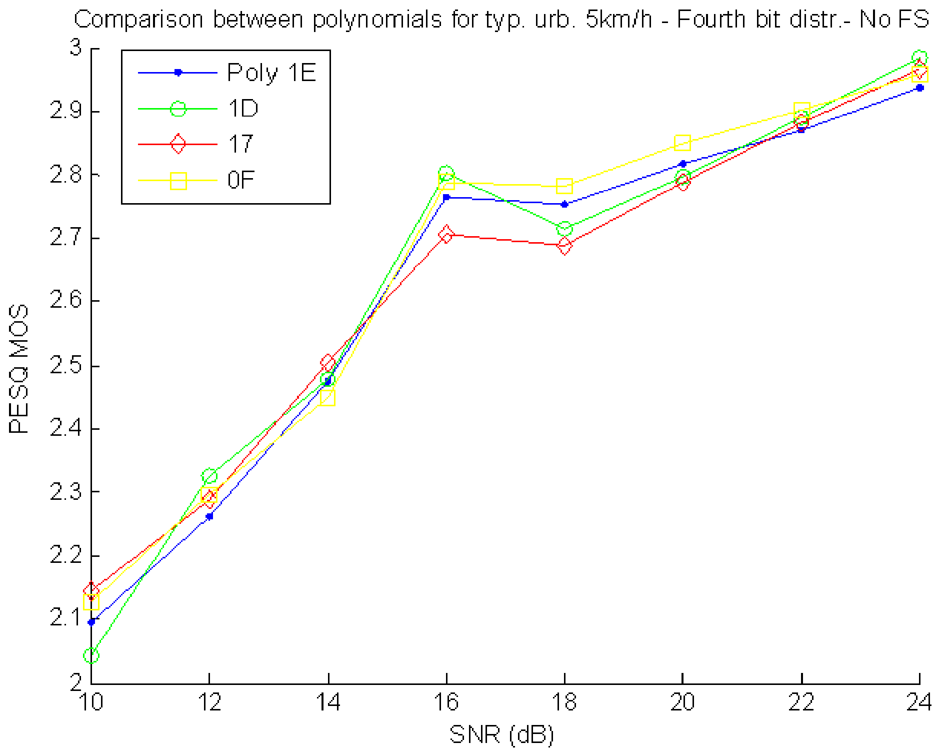

Figure 6.11: Distribution 30-4-6-14 Performance, TU5 Channel, No FS

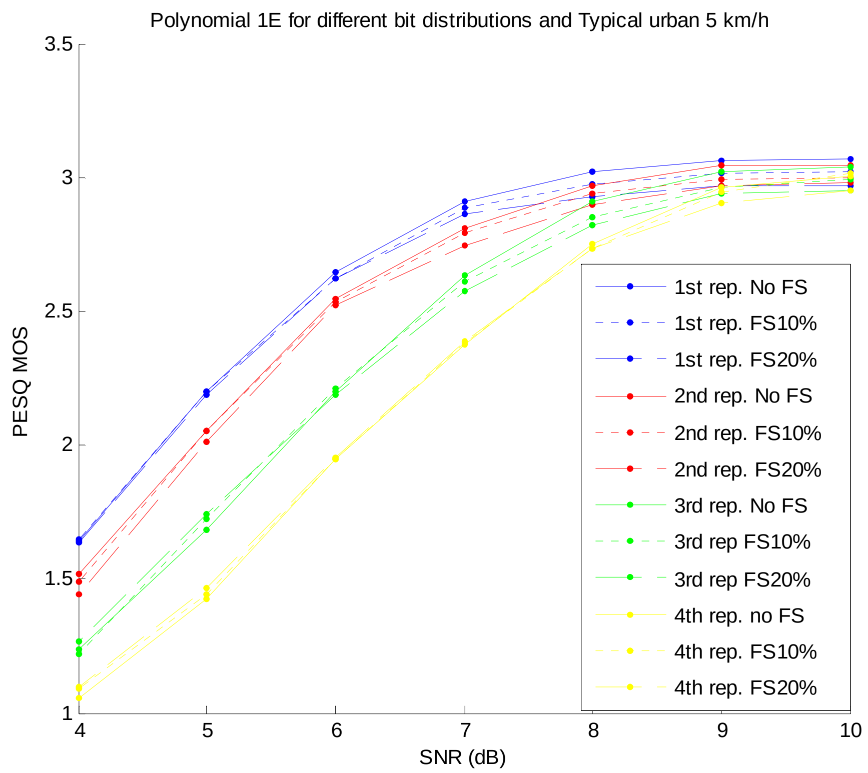

Figure 6.12: Polynomial 1E, TU5 Channel

6.3 Additional TU 50 results

Following discussions within WG5, the additional results were presented for the TU50 channel with no frame stealing (No FS). The polynomial used are 0F, 1D, 1E and 17 in hexadecimal format and the bit distributions are 2-44-6-2, 12-28-9-5, 2-12-17-5 and 30-4-6-14.

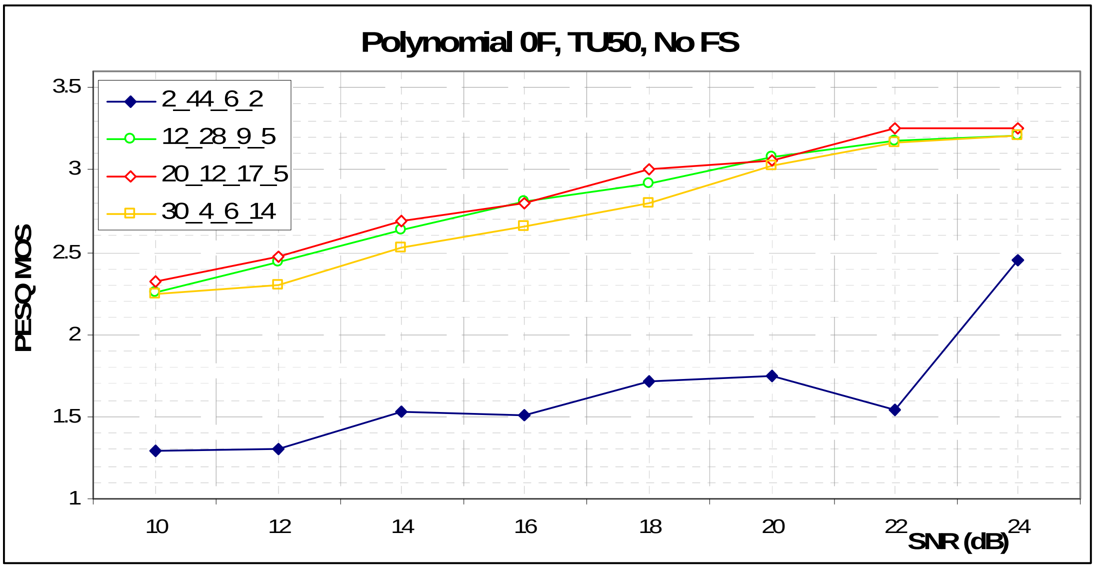

Figure 6.13: Polynomial 0F Results

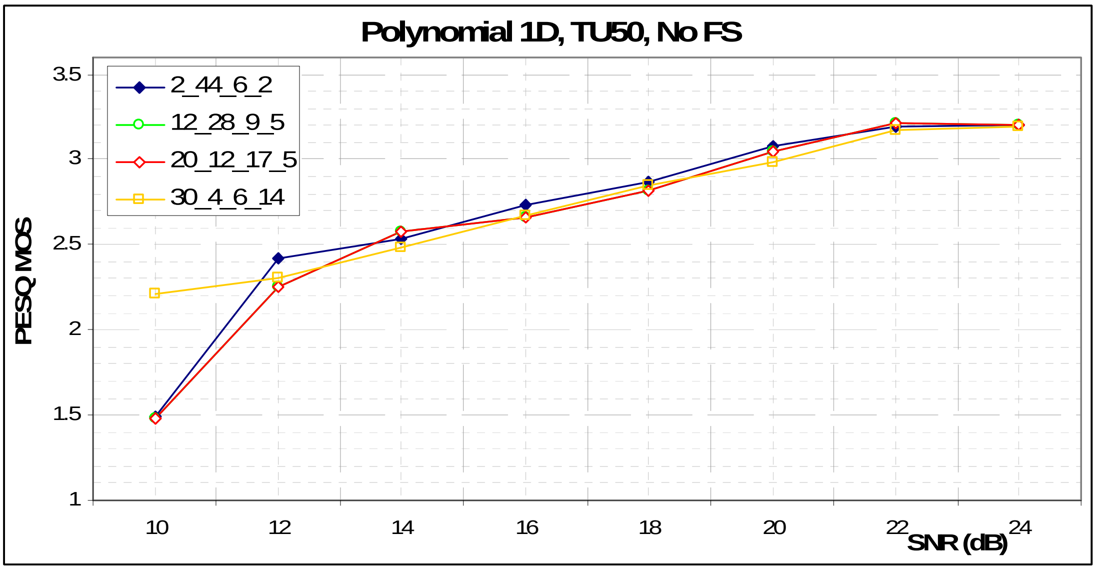

Figure 6.14: Polynomial 1D Results

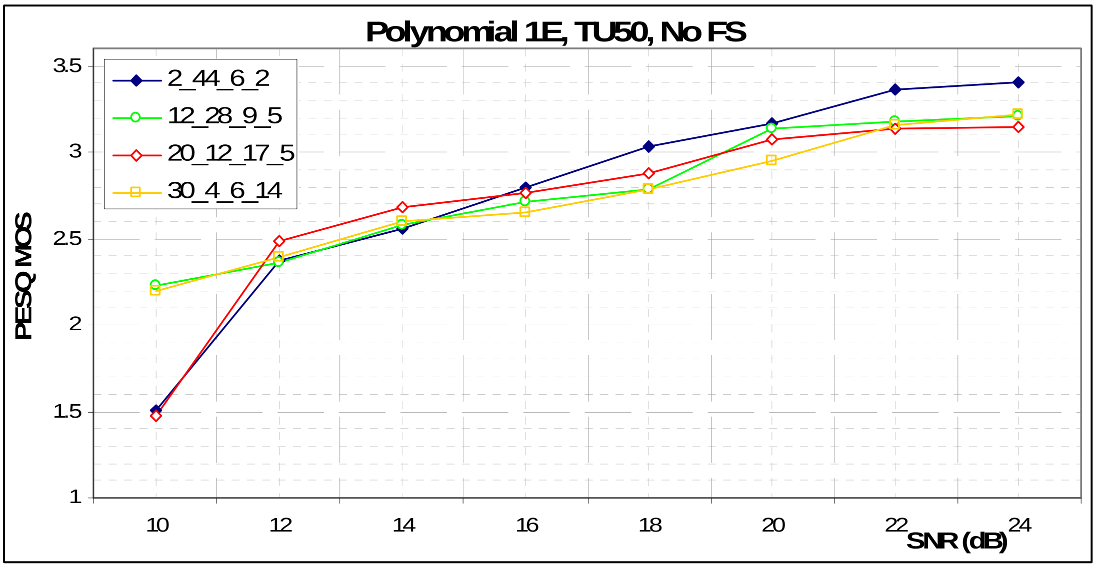

Figure 6.15: Polynomial 1E Results

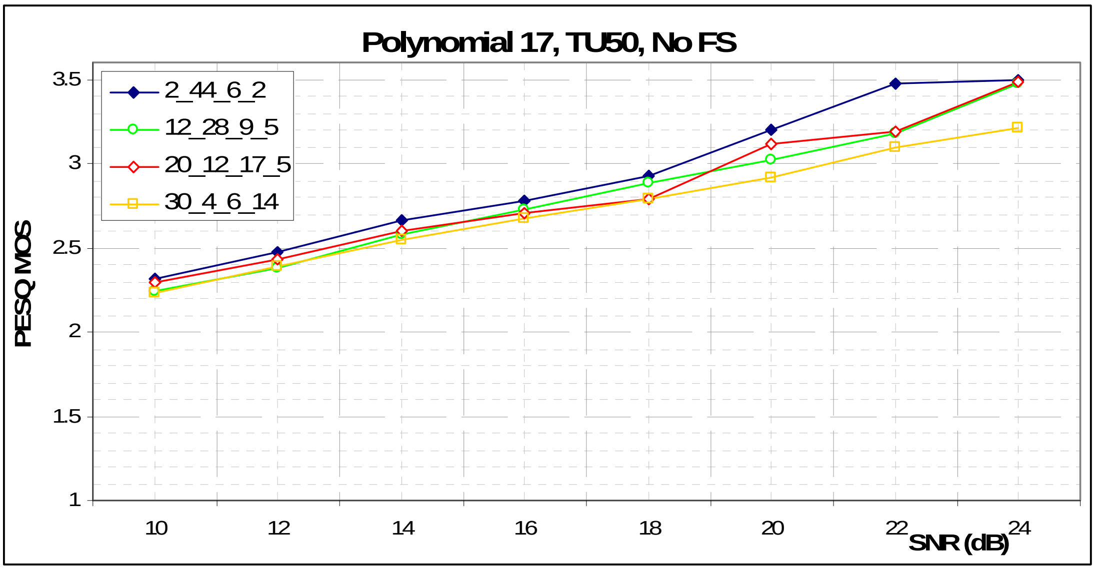

Figure 6.16: Polynomial 17 Results

7 Summary

Result presented in figure 6.15 can directly be compared with the results in figures 6.8, 6.13, 6.14 and 6.16 can be compared to the data provided in Annex A of the present document.

The results presented in the present document indicate that the PESQ comparisons made between the source and decoded audio files is not scalable to represent those made between the STANAG encoded files and the channel decoded files.

Following are the main conclusions derived from this study:

- APL metric was found to be a good indicator for the suitability of a chosen bit distribution.
- Polynomial 1E was found to perform the best in terms of PESQ and listening tests.
- The CRC test was found to have insignificant effect on the STANAG 4591 speech codec.
- This study has established the key concepts in evaluating various parameters that affect the performance of the STANAG 4591 speech codec. Further results can easily be obtained using the software supplied.

8 Conclusions

WG 5 concluded that the performance differential provided by the MELPe voice coder coupled with additional FEC was not sufficient to merit its inclusion as an additional voice coder within the TETRA standards currently.

9 Further Work

Even though this study has provided significant results and a large simulation data set, the following working items could be worthy of further investigation at a future point:

- Puncturing patterns could further be optimized to improve the performance.
- The remaining bit distributions except the 4 best ones can be simulated for further verification of the suitability of the APL metric.
- Bit sensitivities can be further investigated for improving the protection schemes investigated in this study.
- Encoding 2 speech frames with a suitable frame signalling scheme can also be investigated.

More generally, the inclusion of further additional voice coders within the TETRA standards as options to the mandatory ACELP voice coder is still possible. The selection criteria for additional voice coders and the engineering necessary to optimize their performance within the TETRA frame structure will depend on the requirements of users and network operators.

Annex A: Complete simulation data

The present clause provides the complete simulation data for the best 4 polynomials identified in this study.

A.1 Distribution 2-44-6-2

A.1.1 Polynomial 17 ($1+ X^2 + X^3 +X^4$)

Without CRC

sc_10_soft	3,076
sc_9_soft	3,074
sc_8_soft	3,061
sc_7_soft	2,950
sc_6_soft	2,713
sc_5_soft	2,197
sc_4_soft	1,643

tu5_24_soft	2,980	tu50_24_soft	3,025	ht200_24_soft	3,075
tu5_22_soft	2,905	tu50_22_soft	3,043	ht200_22_soft	3,075
tu5_20_soft	2,855	tu50_20_soft	2,996	ht200_20_soft	3,070
tu5_18_soft	2,775	tu50_18_soft	2,858	ht200_18_soft	3,034
tu5_16_soft	2,865	tu50_16_soft	2,713	ht200_16_soft	2,968
tu5_14_soft	2,574	tu50_14_soft	2,524	ht200_14_soft	2,873
tu5_12_soft	2,426	tu50_12_soft	2,282	ht200_12_soft	2,622
tu5_10_soft	2,252	tu50_10_soft	1,932	ht200_10_soft	2,242

Frame stealing= 10 %

sc_10_soft	3,024
sc_9_soft	3,023
sc_8_soft	3,013
sc_7_soft	2,894
sc_6_soft	2,693
sc_5_soft	2,177
sc_4_soft	1,617

tu5_24_soft	2,914	tu50_24_soft	2,970	ht200_24_soft	3,023
tu5_22_soft	2,856	tu50_22_soft	3,000	ht200_22_soft	3,023
tu5_20_soft	2,789	tu50_20_soft	2,939	ht200_20_soft	3,008
tu5_18_soft	2,743	tu50_18_soft	2,819	ht200_18_soft	2,997
tu5_16_soft	2,819	tu50_16_soft	2,682	ht200_16_soft	2,941
tu5_14_soft	2,526	tu50_14_soft	2,489	ht200_14_soft	2,840
tu5_12_soft	2,410	tu50_12_soft	2,254	ht200_12_soft	2,605
tu5_10_soft	2,225	tu50_10_soft	1,908	ht200_10_soft	2,202

Frame stealing= 20 %

sc_10_soft	2,973
sc_9_soft	2,970
sc_8_soft	2,975
sc_7_soft	2,879
sc_6_soft	2,662
sc_5_soft	2,177
sc_4_soft	1,608

tu5_24_soft	2,923	tu50_24_soft	2,936	ht200_24_soft	2,972
tu5_22_soft	2,847	tu50_22_soft	2,957	ht200_22_soft	2,964
tu5_20_soft	2,773	tu50_20_soft	2,918	ht200_20_soft	2,969
tu5_18_soft	2,719	tu50_18_soft	2,748	ht200_18_soft	2,952
tu5_16_soft	2,814	tu50_16_soft	2,648	ht200_16_soft	2,902
tu5_14_soft	2,504	tu50_14_soft	2,497	ht200_14_soft	2,817
tu5_12_soft	2,368	tu50_12_soft	2,224	ht200_12_soft	2,601
tu5_10_soft	2,238	tu50_10_soft	1,904	ht200_10_soft	2,218

A.1.2 Polynomial 1E ($1+ X + X^2 +X^3$)

Without CRC

sc_10_soft	3,068
sc_9_soft	3,062
sc_8_soft	3,022
sc_7_soft	2,913
sc_6_soft	2,644
sc_5_soft	2,203
sc_4_soft	1,625

tu5_24_soft	2,976	tu50_24_soft	3,066	ht200_24_soft	3,074
tu5_22_soft	2,923	tu50_22_soft	3,062	ht200_22_soft	3,075
tu5_20_soft	2,880	tu50_20_soft	2,978	ht200_20_soft	3,071
tu5_18_soft	2,789	tu50_18_soft	2,859	ht200_18_soft	3,029
tu5_16_soft	2,852	tu50_16_soft	2,760	ht200_16_soft	2,986
tu5_14_soft	2,637	tu50_14_soft	2,583	ht200_14_soft	2,881
tu5_12_soft	2,447	tu50_12_soft	2,264	ht200_12_soft	2,645
tu5_10_soft	2,231	tu50_10_soft	1,925	ht200_10_soft	2,194

Frame stealing= 10 %

sc_10_soft	3,020
sc_9_soft	3,014
sc_8_soft	2,972
sc_7_soft	2,889
sc_6_soft	2,620
sc_5_soft	2,202
sc_4_soft	1,646

tu5_24_soft	2,942	tu50_24_soft	3,001	ht200_24_soft	3,032
tu5_22_soft	2,874	tu50_22_soft	2,993	ht200_22_soft	3,007
tu5_20_soft	2,836	tu50_20_soft	2,906	ht200_20_soft	3,016
tu5_18_soft	2,724	tu50_18_soft	2,843	ht200_18_soft	2,990
tu5_16_soft	2,806	tu50_16_soft	2,708	ht200_16_soft	2,938
tu5_14_soft	2,547	tu50_14_soft	2,476	ht200_14_soft	2,820
tu5_12_soft	2,393	tu50_12_soft	2,232	ht200_12_soft	2,566
tu5_10_soft	2,236	tu50_10_soft	1,949	ht200_10_soft	2,233

Frame stealing= 20 %

sc_10_soft	2,969
sc_9_soft	2,966
sc_8_soft	2,930
sc_7_soft	2,862
sc_6_soft	2,620
sc_5_soft	2,186
sc_4_soft	1,635

tu5_24_soft	2,900	tu50_24_soft	2,964	ht200_24_soft	2,948
tu5_22_soft	2,847	tu50_22_soft	2,940	ht200_22_soft	2,948
tu5_20_soft	2,805	tu50_20_soft	2,882	ht200_20_soft	2,971
tu5_18_soft	2,712	tu50_18_soft	2,819	ht200_18_soft	2,962
tu5_16_soft	2,784	tu50_16_soft	2,698	ht200_16_soft	2,890
tu5_14_soft	2,541	tu50_14_soft	2,422	ht200_14_soft	2,774
tu5_12_soft	2,405	tu50_12_soft	2,223	ht200_12_soft	2,529
tu5_10_soft	2,248	tu50_10_soft	1,937	ht200_10_soft	2,246

A.1.3 Polynomial 1D ($1+ X + X^2 +X^4$)

Without CRC

sc_10_soft	3,076
sc_9_soft	3,076
sc_8_soft	3,054
sc_7_soft	2,949
sc_6_soft	2,728
sc_5_soft	2,226
sc_4_soft	1,627

tu5_24_soft	2,975	tu50_24_soft	3,061	ht200_24_soft	3,075
tu5_22_soft	2,920	tu50_22_soft	3,043	ht200_22_soft	3,071
tu5_20_soft	2,986	tu50_20_soft	2,971	ht200_20_soft	3,066
tu5_18_soft	2,772	tu50_18_soft	2,867	ht200_18_soft	3,052
tu5_16_soft	2,865	tu50_16_soft	2,734	ht200_16_soft	2,990
tu5_14_soft	2,624	tu50_14_soft	2,519	ht200_14_soft	2,829
tu5_12_soft	2,421	tu50_12_soft	2,206	ht200_12_soft	2,534
tu5_10_soft	2,243	tu50_10_soft	1,911	ht200_10_soft	2,177

Frame stealing= 10 %

sc_10_soft	3,023
sc_9_soft	3,022
sc_8_soft	3,009
sc_7_soft	2,896
sc_6_soft	2,700
sc_5_soft	2,189
sc_4_soft	1,628

tu5_24_soft	2,937	tu50_24_soft	3,015	ht200_24_soft	3,022
tu5_22_soft	2,883	tu50_22_soft	2,995	ht200_22_soft	3,019
tu5_20_soft	2,820	tu50_20_soft	2,926	ht200_20_soft	3,014
tu5_18_soft	2,735	tu50_18_soft	2,825	ht200_18_soft	2,992
tu5_16_soft	2,822	tu50_16_soft	2,703	ht200_16_soft	2,953
tu5_14_soft	2,584	tu50_14_soft	2,505	ht200_14_soft	2,813
tu5_12_soft	2,378	tu50_12_soft	2,134	ht200_12_soft	2,533
tu5_10_soft	2,200	tu50_10_soft	1,885	ht200_10_soft	2,143

Frame stealing= 20 %

sc_10_soft	2,972
sc_9_soft	2,970
sc_8_soft	2,969
sc_7_soft	2,845
sc_6_soft	2,671
sc_5_soft	2,164
sc_4_soft	1,619

tu5_24_soft	2,911	tu50_24_soft	2,962	ht200_24_soft	2,970
tu5_22_soft	2,874	tu50_22_soft	2,942	ht200_22_soft	2,968
tu5_20_soft	2,814	tu50_20_soft	2,902	ht200_20_soft	2,968
tu5_18_soft	2,710	tu50_18_soft	2,790	ht200_18_soft	2,955
tu5_16_soft	2,804	tu50_16_soft	2,649	ht200_16_soft	2,918
tu5_14_soft	2,560	tu50_14_soft	2,484	ht200_14_soft	2,780
tu5_12_soft	2,371	tu50_12_soft	2,163	ht200_12_soft	2,501
tu5_10_soft	2,223	tu50_10_soft	1,890	ht200_10_soft	2,158

A.1.4 Polynomial 0F ($X + X^2 + X^3 + X^4$)

Without CRC

sc_10_soft	3,076
sc_9_soft	3,072
sc_8_soft	3,017
sc_7_soft	2,955
sc_6_soft	2,662
sc_5_soft	2,173
sc_4_soft	1,650

tu5_24_soft	2,978	tu50_24_soft	3,066	ht200_24_soft	3,074
tu5_22_soft	2,912	tu50_22_soft	3,003	ht200_22_soft	3,075
tu5_20_soft	2,858	tu50_20_soft	2,937	ht200_20_soft	3,055
tu5_18_soft	2,775	tu50_18_soft	2,888	ht200_18_soft	3,026
tu5_16_soft	2,844	tu50_16_soft	2,740	ht200_16_soft	2,972
tu5_14_soft	2,593	tu50_14_soft	2,488	ht200_14_soft	2,841
tu5_12_soft	2,443	tu50_12_soft	2,218	ht200_12_soft	2,571
tu5_10_soft	2,263	tu50_10_soft	1,925	ht200_10_soft	2,242

Frame stealing= 10 %

sc_10_soft	3,024
sc_9_soft	3,021
sc_8_soft	2,986
sc_7_soft	2,896
sc_6_soft	2,649
sc_5_soft	2,174
sc_4_soft	1,576

tu5_24_soft	2,934	tu50_24_soft	3,013	ht200_24_soft	3,024
tu5_22_soft	2,901	tu50_22_soft	2,995	ht200_22_soft	3,026
tu5_20_soft	2,797	tu50_20_soft	2,882	ht200_20_soft	3,007
tu5_18_soft	2,739	tu50_18_soft	2,834	ht200_18_soft	3,015
tu5_16_soft	2,781	tu50_16_soft	2,801	ht200_16_soft	2,955
tu5_14_soft	2,568	tu50_14_soft	2,506	ht200_14_soft	2,858
tu5_12_soft	2,446	tu50_12_soft	2,254	ht200_12_soft	2,617
tu5_10_soft	2,209	tu50_10_soft	2,023	ht200_10_soft	2,182

Frame stealing= 20 %

sc_10_soft	2,973
sc_9_soft	2,967
sc_8_soft	2,935
sc_7_soft	2,844
sc_6_soft	2,609
sc_5_soft	2,161
sc_4_soft	1,565

tu5_24_soft	2,903	tu50_24_soft	2,978	ht200_24_soft	2,972
tu5_22_soft	2,877	tu50_22_soft	2,956	ht200_22_soft	2,958
tu5_20_soft	2,779	tu50_20_soft	2,876	ht200_20_soft	2,979
tu5_18_soft	2,716	tu50_18_soft	2,786	ht200_18_soft	2,972
tu5_16_soft	2,780	tu50_16_soft	2,741	ht200_16_soft	2,888
tu5_14_soft	2,541	tu50_14_soft	2,500	ht200_14_soft	2,815
tu5_12_soft	2,425	tu50_12_soft	2,238	ht200_12_soft	2,602
tu5_10_soft	2,158	tu50_10_soft	2,002	ht200_10_soft	2,181

A.2 Distribution 12-28-9-5

A.2.1 Polynomial 1E ($1+ X + X^2 +X^3$)

Without CRC

sc_10_soft	3,047
sc_9_soft	3,043
sc_8_soft	2,970
sc_7_soft	2,813
sc_6_soft	2,549
sc_5_soft	2,055
sc_4_soft	1,520

tu5_24_soft	2,968	tu50_24_soft	3,011	ht200_24_soft	3,075
tu5_22_soft	2,961	tu50_22_soft	2,986	ht200_22_soft	3,062
tu5_20_soft	2,870	tu50_20_soft	2,903	ht200_20_soft	3,052
tu5_18_soft	2,767	tu50_18_soft	2,876	ht200_18_soft	3,021

tu5_16_soft	2,839	tu50_16_soft	2,724	ht200_16_soft	2,942
tu5_14_soft	2,629	tu50_14_soft	2,521	ht200_14_soft	2,798
tu5_12_soft	2,404	tu50_12_soft	2,177	ht200_12_soft	2,488
tu5_10_soft	2,211	tu50_10_soft	1,923	ht200_10_soft	2,107

Frame Stealing 10 %

sc_10_soft	2,997
sc_9_soft	2,995
sc_8_soft	2,937
sc_7_soft	2,794
sc_6_soft	2,537
sc_5_soft	2,054
sc_4_soft	1,491

tu5_24_soft	2,919	tu50_24_soft	2,955	ht200_24_soft	3,020
tu5_22_soft	2,884	tu50_22_soft	2,938	ht200_22_soft	3,019
tu5_20_soft	2,828	tu50_20_soft	2,855	ht200_20_soft	3,003
tu5_18_soft	2,711	tu50_18_soft	2,847	ht200_18_soft	2,995
tu5_16_soft	2,822	tu50_16_soft	2,690	ht200_16_soft	2,934
tu5_14_soft	2,560	tu50_14_soft	2,479	ht200_14_soft	2,760
tu5_12_soft	2,370	tu50_12_soft	2,155	ht200_12_soft	2,473
tu5_10_soft	2,176	tu50_10_soft	1,931	ht200_10_soft	2,102

Frame stealing 20 %

sc_10_soft	2,980
sc_9_soft	2,969
sc_8_soft	2,901
sc_7_soft	2,744
sc_6_soft	2,522
sc_5_soft	2,011
sc_4_soft	1,445

tu5_24_soft	2,891	tu50_24_soft	2,955	ht200_24_soft	3,020
tu5_22_soft	2,862	tu50_22_soft	2,938	ht200_22_soft	3,019
tu5_20_soft	2,792	tu50_20_soft	2,855	ht200_20_soft	3,003

tu5_18_soft	2,724	tu50_18_soft	2,847	ht200_18_soft	2,995
tu5_16_soft	2,787	tu50_16_soft	2,690	ht200_16_soft	2,934
tu5_14_soft	2,545	tu50_14_soft	2,479	ht200_14_soft	2,760
tu5_12_soft	2,340	tu50_12_soft	2,155	ht200_12_soft	2,473
tu5_10_soft	2,163	tu50_10_soft	1,931	ht200_10_soft	2,102

A.2.2 Polynomial 1D ($1+ X + X^2 +X^4$)

Without CRC

sc_10_soft	3,076
sc_9_soft	3,071
sc_8_soft	3,021
sc_7_soft	2,826
sc_6_soft	2,440
sc_5_soft	1,815
sc_4_soft	1,326

tu5_24_soft	2,966	tu50_24_soft	3,050	ht200_24_soft	3,059
tu5_22_soft	2,907	tu50_22_soft	2,999	ht200_22_soft	3,060
tu5_20_soft	2,848	tu50_20_soft	2,926	ht200_20_soft	3,054
tu5_18_soft	2,802	tu50_18_soft	2,861	ht200_18_soft	3,003
tu5_16_soft	2,855	tu50_16_soft	2,670	ht200_16_soft	2,926
tu5_14_soft	2,623	tu50_14_soft	2,372	ht200_14_soft	2,801
tu5_12_soft	2,449	tu50_12_soft	2,112	ht200_12_soft	2,485
tu5_10_soft	2,183	tu50_10_soft	1,892	ht200_10_soft	2,105

Frame Stealing 10 %

sc_10_soft	3,023
sc_9_soft	3,018
sc_8_soft	2,978
sc_7_soft	2,804
sc_6_soft	2,396
sc_5_soft	1,804
sc_4_soft	1,320

tu5_24_soft	2,908	tu50_24_soft	2,975	ht200_24_soft	3,010

tu5_22_soft	2,874	tu50_22_soft	2,981	ht200_22_soft	3,009
tu5_20_soft	2,790	tu50_20_soft	2,790	ht200_20_soft	3,010
tu5_18_soft	2,755	tu50_18_soft	2,823	ht200_18_soft	2,981
tu5_16_soft	2,813	tu50_16_soft	2,644	ht200_16_soft	2,885
tu5_14_soft	2,586	tu50_14_soft	2,332	ht200_14_soft	2,762
tu5_12_soft	2,377	tu50_12_soft	2,109	ht200_12_soft	2,457
tu5_10_soft	2,140	tu50_10_soft	1,861	ht200_10_soft	2,098

Frame Stealing 20 %

sc_10_soft	2,971
sc_9_soft	2,981
sc_8_soft	2,939
sc_7_soft	2,791
sc_6_soft	2,385
sc_5_soft	1,780
sc_4_soft	1,338

tu5_24_soft	2,879	tu50_24_soft	2,938	ht200_24_soft	2,955
tu5_22_soft	2,838	tu50_22_soft	2,926	ht200_22_soft	2,951
tu5_20_soft	2,770	tu50_20_soft	2,844	ht200_20_soft	2,963
tu5_18_soft	2,714	tu50_18_soft	2,790	ht200_18_soft	2,929
tu5_16_soft	2,781	tu50_16_soft	2,618	ht200_16_soft	2,848
tu5_14_soft	2,537	tu50_14_soft	2,343	ht200_14_soft	2,734
tu5_12_soft	2,338	tu50_12_soft	2,084	ht200_12_soft	2,443
tu5_10_soft	2,156	tu50_10_soft	1,899	ht200_10_soft	2,099

A.2.3 Polynomial 17 ($1+ X^2 + X^3 + X^4$)

Without CRC

sc_10_soft	3,062
sc_9_soft	3,043
sc_8_soft	3,019
sc_7_soft	2,827
sc_6_soft	2,540
sc_5_soft	1,973
sc_4_soft	1,378

tu5_24_soft	2,978	tu50_24_soft	3,017	ht200_24_soft	3,062
tu5_22_soft	2,925	tu50_22_soft	3,036	ht200_22_soft	3,070
tu5_20_soft	2,893	tu50_20_soft	2,971	ht200_20_soft	3,071
tu5_18_soft	2,749	tu50_18_soft	2,850	ht200_18_soft	3,032
tu5_16_soft	2,799	tu50_16_soft	2,715	ht200_16_soft	2,925
tu5_14_soft	2,581	tu50_14_soft	2,504	ht200_14_soft	2,797
tu5_12_soft	2,406	tu50_12_soft	2,185	ht200_12_soft	2,491
tu5_10_soft	2,148	tu50_10_soft	1,823	ht200_10_soft	2,034

Frame Stealing 10 %

sc_10_soft	3,017
sc_9_soft	2,999
sc_8_soft	2,963
sc_7_soft	2,820
sc_6_soft	2,539
sc_5_soft	1,994
sc_4_soft	1,375

tu5_24_soft	2,930	tu50_24_soft	2,964	ht200_24_soft	3,009
tu5_22_soft	2,890	tu50_22_soft	2,980	ht200_22_soft	3,007
tu5_20_soft	2,841	tu50_20_soft	2,901	ht200_20_soft	2,998
tu5_18_soft	2,719	tu50_18_soft	2,843	ht200_18_soft	2,967
tu5_16_soft	2,786	tu50_16_soft	2,690	ht200_16_soft	2,923
tu5_14_soft	2,536	tu50_14_soft	2,497	ht200_14_soft	2,768
tu5_12_soft	2,354	tu50_12_soft	2,152	ht200_12_soft	2,453
tu5_10_soft	2,143	tu50_10_soft	1,822	ht200_10_soft	2,061

Frame Stealing 20 %

sc_10_soft	2,986
sc_9_soft	2,972
sc_8_soft	2,939
sc_7_soft	2,775
sc_6_soft	2,489
sc_5_soft	1,978
sc_4_soft	1,371

tu5_24_soft	2,893	tu50_24_soft	2,913	ht200_24_soft	2,966
tu5_22_soft	2,865	tu50_22_soft	2,933	ht200_22_soft	2,977
tu5_20_soft	2,821	tu50_20_soft	2,863	ht200_20_soft	2,957
tu5_18_soft	2,696	tu50_18_soft	2,804	ht200_18_soft	2,948
tu5_16_soft	2,782	tu50_16_soft	2,656	ht200_16_soft	2,876
tu5_14_soft	2,509	tu50_14_soft	2,503	ht200_14_soft	2,722
tu5_12_soft	2,351	tu50_12_soft	2,183	ht200_12_soft	2,419
tu5_10_soft	2,132	tu50_10_soft	1,803	ht200_10_soft	2,054

A.2.4 Polynomial 0F ($X + X^2 + X^3 + X^4$)

Without CRC

sc_10_soft	3,070
sc_9_soft	3,054
sc_8_soft	3,018
sc_7_soft	2,847
sc_6_soft	2,483
sc_5_soft	1,944
sc_4_soft	1,387

tu5_24_soft	2,983	tu50_24_soft	3,048	ht200_24_soft	3,076
tu5_22_soft	2,915	tu50_22_soft	3,020	ht200_22_soft	3,071
tu5_20_soft	2,876	tu50_20_soft	2,961	ht200_20_soft	3,060
tu5_18_soft	2,800	tu50_18_soft	2,912	ht200_18_soft	3,045
tu5_16_soft	2,826	tu50_16_soft	2,692	ht200_16_soft	2,944
tu5_14_soft	2,553	tu50_14_soft	2,498	ht200_14_soft	2,778
tu5_12_soft	2,406	tu50_12_soft	2,142	ht200_12_soft	2,431
tu5_10_soft	2,179	tu50_10_soft	1,848	ht200_10_soft	2,082

Frame Stealing 10 %

sc_10_soft	3,015
sc_9_soft	3,012
sc_8_soft	2,974
sc_7_soft	2,808
sc_6_soft	2,470
sc_5_soft	1,921
sc_4_soft	1,397

tu5_24_soft	2,922	tu50_24_soft	3,000	ht200_24_soft	3,024
tu5_22_soft	2,976	tu50_22_soft	2,980	ht200_22_soft	3,025
tu5_20_soft	2,827	tu50_20_soft	2,924	ht200_20_soft	2,996
tu5_18_soft	2,746	tu50_18_soft	2,858	ht200_18_soft	2,977
tu5_16_soft	2,806	tu50_16_soft	2,638	ht200_16_soft	2,906
tu5_14_soft	2,498	tu50_14_soft	2,453	ht200_14_soft	2,744
tu5_12_soft	2,381	tu50_12_soft	2,150	ht200_12_soft	2,398
tu5_10_soft	2,156	tu50_10_soft	1,866	ht200_10_soft	2,085

Frame Stealing 20 %

sc_10_soft	2,972
sc_9_soft	2,976
sc_8_soft	2,955
sc_7_soft	2,798
sc_6_soft	2,462
sc_5_soft	1,887
sc_4_soft	1,397

tu5_24_soft	2,909	tu50_24_soft	2,968	ht200_24_soft	2,972
tu5_22_soft	2,868	tu50_22_soft	2,930	ht200_22_soft	2,963
tu5_20_soft	2,793	tu50_20_soft	2,871	ht200_20_soft	2,963
tu5_18_soft	2,733	tu50_18_soft	2,820	ht200_18_soft	2,947
tu5_16_soft	2,779	tu50_16_soft	2,639	ht200_16_soft	2,857
tu5_14_soft	2,499	tu50_14_soft	2,476	ht200_14_soft	2,704
tu5_12_soft	2,384	tu50_12_soft	2,126	ht200_12_soft	2,344
tu5_10_soft	2,171	tu50_10_soft	1,871	ht200_10_soft	2,083

A.3 Distribution 20-12-17-5

A.3.1 Polynomial 1E ($1+ X + X^2 +X^3$)

Without CRC

sc_10_soft	3,037
sc_9_soft	3,019
sc_8_soft	2,909
sc_7_soft	2,635
sc_6_soft	2,199
sc_5_soft	1,684
sc_4_soft	1,239

tu5_24_soft	2,963	tu50_24_soft	3,039	ht200_24_soft	3,074
tu5_22_soft	2,917	tu50_22_soft	2,981	ht200_22_soft	3,044
tu5_20_soft	2,828	tu50_20_soft	2,869	ht200_20_soft	3,034
tu5_18_soft	2,720	tu50_18_soft	2,744	ht200_18_soft	2,975
tu5_16_soft	2,797	tu50_16_soft	2,616	ht200_16_soft	2,864
tu5_14_soft	2,591	tu50_14_soft	2,383	ht200_14_soft	2,619
tu5_12_soft	2,376	tu50_12_soft	2,100	ht200_12_soft	2,332
tu5_10_soft	2,177	tu50_10_soft	1,783	ht200_10_soft	1,939

Frame Stealing 10 %

sc_10_soft	2,990
sc_9_soft	2,962
sc_8_soft	2,849
sc_7_soft	2,611
sc_6_soft	2,211
sc_5_soft	1,722
sc_4_soft	1,219

tu5_24_soft	2,890	tu50_24_soft	2,993	ht200_24_soft	3,005
tu5_22_soft	2,856	tu50_22_soft	2,930	ht200_22_soft	2,995
tu5_20_soft	2,761	tu50_20_soft	2,831	ht200_20_soft	2,975
tu5_18_soft	2,686	tu50_18_soft	2,716	ht200_18_soft	2,931

tu5_16_soft	2,766	tu50_16_soft	2,580	ht200_16_soft	2,811
tu5_14_soft	2,547	tu50_14_soft	2,352	ht200_14_soft	2,607
tu5_12_soft	2,359	tu50_12_soft	2,076	ht200_12_soft	2,295
tu5_10_soft	2,146	tu50_10_soft	1,746	ht200_10_soft	1,962

Frame Stealing 20 %

sc_10_soft	2,954
sc_9_soft	2,937
sc_8_soft	2,825
sc_7_soft	2,575
sc_6_soft	2,190
sc_5_soft	1,742
sc_4_soft	1,268

tu5_24_soft	2,877	tu50_24_soft	2,950	ht200_24_soft	2,953
tu5_22_soft	2,846	tu50_22_soft	2,898	ht200_22_soft	2,954
tu5_20_soft	2,752	tu50_20_soft	2,799	ht200_20_soft	2,928
tu5_18_soft	2,672	tu50_18_soft	2,679	ht200_18_soft	2,894
tu5_16_soft	2,763	tu50_16_soft	2,556	ht200_16_soft	2,789
tu5_14_soft	2,547	tu50_14_soft	2,366	ht200_14_soft	2,609
tu5_12_soft	2,351	tu50_12_soft	2,071	ht200_12_soft	2,267
tu5_10_soft	2,129	tu50_10_soft	1,756	ht200_10_soft	1,923

A.3.2 Polynomial 1D ($1+ X + X^2 +X^4$)

Without CRC

sc_10_soft	3,066
sc_9_soft	3,031
sc_8_soft	2,916
sc_7_soft	2,541
sc_6_soft	2,182
sc_5_soft	1,676
sc_4_soft	1,256

tu5_24_soft	2,974	tu50_24_soft	3,029	ht200_24_soft	3,082
tu5_22_soft	2,878	tu50_22_soft	2,996	ht200_22_soft	3,064
tu5_20_soft	2,840	tu50_20_soft	2,907	ht200_20_soft	3,069
tu5_18_soft	2,737	tu50_18_soft	2,771	ht200_18_soft	2,971
tu5_16_soft	2,833	tu50_16_soft	2,620	ht200_16_soft	2,885
tu5_14_soft	2,590	tu50_14_soft	2,364	ht200_14_soft	2,641
tu5_12_soft	2,377	tu50_12_soft	2,143	ht200_12_soft	2,324
tu5_10_soft	2,135	tu50_10_soft	1,768	ht200_10_soft	1,903

Frame Stealing 10 %

sc_10_soft	3,018
sc_9_soft	2,993
sc_8_soft	2,890
sc_7_soft	2,508
sc_6_soft	2,185
sc_5_soft	1,669
sc_4_soft	1,248

tu5_24_soft	2,936	tu50_24_soft	2,989	ht200_24_soft	3,024
tu5_22_soft	2,834	tu50_22_soft	2,936	ht200_22_soft	3,025
tu5_20_soft	2,793	tu50_20_soft	2,874	ht200_20_soft	3,017
tu5_18_soft	2,675	tu50_18_soft	2,735	ht200_18_soft	2,933
tu5_16_soft	2,780	tu50_16_soft	2,575	ht200_16_soft	2,834
tu5_14_soft	2,553	tu50_14_soft	2,344	ht200_14_soft	2,658
tu5_12_soft	2,338	tu50_12_soft	2,099	ht200_12_soft	2,316
tu5_10_soft	2,131	tu50_10_soft	1,752	ht200_10_soft	1,919

Frame Stealing 20 %

sc_10_soft	2,948
sc_9_soft	2,934
sc_8_soft	2,851
sc_7_soft	2,497
sc_6_soft	2,171
sc_5_soft	1,631
sc_4_soft	1,254

tu5_24_soft	2,893	tu50_24_soft	2,940	ht200_24_soft	2,988
tu5_22_soft	2,803	tu50_22_soft	2,921	ht200_22_soft	2,961
tu5_20_soft	2,804	tu50_20_soft	2,812	ht200_20_soft	2,959
tu5_18_soft	2,669	tu50_18_soft	2,673	ht200_18_soft	2,887
tu5_16_soft	2,782	tu50_16_soft	2,574	ht200_16_soft	2,759
tu5_14_soft	2,532	tu50_14_soft	2,329	ht200_14_soft	2,602
tu5_12_soft	2,345	tu50_12_soft	2,082	ht200_12_soft	2,286
tu5_10_soft	2,109	tu50_10_soft	1,736	ht200_10_soft	1,933

A.3.3 Polynomial 17 ($1+ X^2 + X^3 +X^4$)

Without CRC

sc_10_soft	3,022
sc_9_soft	2,942
sc_8_soft	2,837
sc_7_soft	2,487
sc_6_soft	2,055
sc_5_soft	1,577
sc_4_soft	1,128

tu5_24_soft	2,945	tu50_24_soft	3,013	ht200_24_soft	3,073
tu5_22_soft	2,906	tu50_22_soft	2,982	ht200_22_soft	3,062
tu5_20_soft	2,840	tu50_20_soft	2,904	ht200_20_soft	3,049
tu5_18_soft	2,776	tu50_18_soft	2,727	ht200_18_soft	2,955
tu5_16_soft	2,760	tu50_16_soft	2,568	ht200_16_soft	2,828
tu5_14_soft	2,480	tu50_14_soft	2,316	ht200_14_soft	2,585
tu5_12_soft	2,291	tu50_12_soft	2,122	ht200_12_soft	2,309
tu5_10_soft	2,116	tu50_10_soft	1,778	ht200_10_soft	1,905

Frame stealing 10 %

sc_10_soft	2,947
sc_9_soft	2,917
sc_8_soft	2,785
sc_7_soft	2,479

sc_6_soft	2,060
sc_5_soft	1,590
sc_4_soft	1,182

tu5_24_soft	2,877	tu50_24_soft	2,966	ht200_24_soft	3,020
tu5_22_soft	2,842	tu50_22_soft	2,934	ht200_22_soft	2,999
tu5_20_soft	2,785	tu50_20_soft	2,890	ht200_20_soft	3,003
tu5_18_soft	2,731	tu50_18_soft	2,685	ht200_18_soft	2,937
tu5_16_soft	2,710	tu50_16_soft	2,525	ht200_16_soft	2,762
tu5_14_soft	2,486	tu50_14_soft	2,309	ht200_14_soft	2,580
tu5_12_soft	2,263	tu50_12_soft	2,115	ht200_12_soft	2,308
tu5_10_soft	2,077	tu50_10_soft	1,770	ht200_10_soft	1,941

Frame stealing 20 %

sc_10_soft	2,940
sc_9_soft	2,898
sc_8_soft	2,788
sc_7_soft	2,480
sc_6_soft	2,044
sc_5_soft	1,578
sc_4_soft	1,187

tu5_24_soft	2,882	tu50_24_soft	2,915	ht200_24_soft	2,978
tu5_22_soft	2,830	tu50_22_soft	2,881	ht200_22_soft	2,952
tu5_20_soft	2,779	tu50_20_soft	2,835	ht200_20_soft	2,959
tu5_18_soft	2,729	tu50_18_soft	2,668	ht200_18_soft	2,886
tu5_16_soft	2,708	tu50_16_soft	2,534	ht200_16_soft	2,754
tu5_14_soft	2,442	tu50_14_soft	2,306	ht200_14_soft	2,555
tu5_12_soft	2,255	tu50_12_soft	2,106	ht200_12_soft	2,307
tu5_10_soft	2,038	tu50_10_soft	1,772	ht200_10_soft	1,920

A.3.4 Polynomial 0F ($X + X^2 + X^3 + X^4$)

Without CRC

sc_10_soft	3,062
sc_9_soft	3,045
sc_8_soft	2,911
sc_7_soft	2,605
sc_6_soft	2,139
sc_5_soft	1,586
sc_4_soft	1,197

tu5_24_soft	2,959	tu50_24_soft	3,053	ht200_24_soft	3,075
tu5_22_soft	2,928	tu50_22_soft	2,961	ht200_22_soft	3,064
tu5_20_soft	2,860	tu50_20_soft	2,882	ht200_20_soft	3,034
tu5_18_soft	2,761	tu50_18_soft	2,816	ht200_18_soft	3,024
tu5_16_soft	2,777	tu50_16_soft	2,605	ht200_16_soft	2,906
tu5_14_soft	2,478	tu50_14_soft	2,318	ht200_14_soft	2,624
tu5_12_soft	2,373	tu50_12_soft	2,116	ht200_12_soft	2,315
tu5_10_soft	2,070	tu50_10_soft	1,749	ht200_10_soft	1,937

Frame stealing 10 %

sc_10_soft	3,014
sc_9_soft	2,991
sc_8_soft	2,887
sc_7_soft	2,549
sc_6_soft	2,139
sc_5_soft	1,582
sc_4_soft	1,241

tu5_24_soft	2,907	tu50_24_soft	2,986	ht200_24_soft	3,027
tu5_22_soft	2,869	tu50_22_soft	2,913	ht200_22_soft	3,009
tu5_20_soft	2,777	tu50_20_soft	2,831	ht200_20_soft	2,999
tu5_18_soft	2,719	tu50_18_soft	2,750	ht200_18_soft	2,956
tu5_16_soft	2,745	tu50_16_soft	2,543	ht200_16_soft	2,866
tu5_14_soft	2,408	tu50_14_soft	2,291	ht200_14_soft	2,611
tu5_12_soft	2,328	tu50_12_soft	2,084	ht200_12_soft	2,313
tu5_10_soft	2,078	tu50_10_soft	1,797	ht200_10_soft	1,887

Frame stealing 20 %

sc_10_soft	2,983
sc_9_soft	2,960
sc_8_soft	2,841
sc_7_soft	2,535
sc_6_soft	2,183
sc_5_soft	1,600
sc_4_soft	1,166

tu5_24_soft	2,869	tu50_24_soft	2,951	ht200_24_soft	2,971
tu5_22_soft	2,851	tu50_22_soft	2,890	ht200_22_soft	2,965
tu5_20_soft	2,781	tu50_20_soft	2,798	ht200_20_soft	2,954
tu5_18_soft	2,711	tu50_18_soft	2,737	ht200_18_soft	2,905
tu5_16_soft	2,718	tu50_16_soft	2,526	ht200_16_soft	2,809
tu5_14_soft	2,454	tu50_14_soft	2,282	ht200_14_soft	2,566
tu5_12_soft	2,313	tu50_12_soft	2,087	ht200_12_soft	2,237
tu5_10_soft	2,071	tu50_10_soft	1,818	ht200_10_soft	1,880

A.4 Distribution 30-4-6-14

A.4.1 Polynomial 1E (1+ X + X^2 +X^3)

Without CRC

sc_10_soft	3,007
sc_9_soft	2,963
sc_8_soft	2,750
sc_7_soft	2,375
sc_6_soft	1,947
sc_5_soft	1,426
sc_4_soft	1,057

tu5_24_soft	2,936	tu50_24_soft	3,011	ht200_24_soft	3,058
tu5_22_soft	2,871	tu50_22_soft	2,935	ht200_22_soft	3,056
tu5_20_soft	2,816	tu50_20_soft	2,834	ht200_20_soft	3,012
tu5_18_soft	2,752	tu50_18_soft	2,732	ht200_18_soft	2,903

tu5_16_soft	2,765	tu50_16_soft	2,541	ht200_16_soft	2,720
tu5_14_soft	2,475	tu50_14_soft	2,268	ht200_14_soft	2,501
tu5_12_soft	2,263	tu50_12_soft	1,980	ht200_12_soft	2,138
tu5_10_soft	2,095	tu50_10_soft	1,690	ht200_10_soft	1,801

Frame stealing 10 %

sc_10_soft	3,014
sc_9_soft	2,945
sc_8_soft	2,736
sc_7_soft	2,381
sc_6_soft	1,948
sc_5_soft	1,446
sc_4_soft	1,089

tu5_24_soft	2,891	tu50_24_soft	2,992	ht200_24_soft	3,006
tu5_22_soft	2,840	tu50_22_soft	2,877	ht200_22_soft	2,996
tu5_20_soft	2,803	tu50_20_soft	2,787	ht200_20_soft	2,974
tu5_18_soft	2,728	tu50_18_soft	2,672	ht200_18_soft	2,903
tu5_16_soft	2,724	tu50_16_soft	2,461	ht200_16_soft	2,717
tu5_14_soft	2,415	tu50_14_soft	2,236	ht200_14_soft	2,477
tu5_12_soft	2,216	tu50_12_soft	1,951	ht200_12_soft	2,156
tu5_10_soft	2,063	tu50_10_soft	1,718	ht200_10_soft	1,813

Frame stealing 20 %

sc_10_soft	2,954
sc_9_soft	2,902
sc_8_soft	2,734
sc_7_soft	2,388
sc_6_soft	1,951
sc_5_soft	1,466
sc_4_soft	1,097

tu5_24_soft	2,862	tu50_24_soft	2,922	ht200_24_soft	2,969
tu5_22_soft	2,801	tu50_22_soft	2,880	ht200_22_soft	2,973
tu5_20_soft	2,779	tu50_20_soft	2,762	ht200_20_soft	2,930
tu5_18_soft	2,707	tu50_18_soft	2,659	ht200_18_soft	2,854
tu5_16_soft	2,724	tu50_16_soft	2,454	ht200_16_soft	2,679
tu5_14_soft	2,429	tu50_14_soft	2,216	ht200_14_soft	2,469
tu5_12_soft	2,219	tu50_12_soft	1,956	ht200_12_soft	2,127
tu5_10_soft	2,055	tu50_10_soft	1,707	ht200_10_soft	1,835

A.4.2 Polynomial 1D ($1+ X + X^2 +X^4$)

Without CRC

sc_10_soft	3,065
sc_9_soft	3,007
sc_8_soft	2,836
sc_7_soft	2,450
sc_6_soft	1,904
sc_5_soft	1,393
sc_4_soft	0,998

tu5_24_soft	2,983	tu50_24_soft	3,014	ht200_24_soft	3,043
tu5_22_soft	2,890	tu50_22_soft	2,898	ht200_22_soft	3,054
tu5_20_soft	2,797	tu50_20_soft	2,833	ht200_20_soft	3,015
tu5_18_soft	2,716	tu50_18_soft	2,694	ht200_18_soft	2,965
tu5_16_soft	2,803	tu50_16_soft	2,531	ht200_16_soft	2,759
tu5_14_soft	2,478	tu50_14_soft	2,263	ht200_14_soft	2,496
tu5_12_soft	2,325	tu50_12_soft	2,015	ht200_12_soft	2,130
tu5_10_soft	2,043	tu50_10_soft	1,708	ht200_10_soft	1,798

Frame stealing 10 %

sc_10_soft	3,008
sc_9_soft	2,963
sc_8_soft	2,790
sc_7_soft	2,390
sc_6_soft	1,922
sc_5_soft	1,417
sc_4_soft	1,043

tu5_24_soft	2,924	tu50_24_soft	2,950	ht200_24_soft	3,003
tu5_22_soft	2,832	tu50_22_soft	2,884	ht200_22_soft	3,009
tu5_20_soft	2,741	tu50_20_soft	2,810	ht200_20_soft	2,950
tu5_18_soft	2,651	tu50_18_soft	2,665	ht200_18_soft	2,914
tu5_16_soft	2,751	tu50_16_soft	2,490	ht200_16_soft	2,719
tu5_14_soft	2,464	tu50_14_soft	2,248	ht200_14_soft	2,505
tu5_12_soft	2,312	tu50_12_soft	2,007	ht200_12_soft	2,147
tu5_10_soft	2,041	tu50_10_soft	1,755	ht200_10_soft	1,770

Frame stealing 20 %

sc_10_soft	2,946
sc_9_soft	2,927
sc_8_soft	2,794
sc_7_soft	2,435
sc_6_soft	1,921
sc_5_soft	1,438
sc_4_soft	1,070

tu5_24_soft	2,894	tu50_24_soft	2,929	ht200_24_soft	2,959
tu5_22_soft	2,806	tu50_22_soft	2,837	ht200_22_soft	2,956
tu5_20_soft	2,721	tu50_20_soft	2,779	ht200_20_soft	2,930
tu5_18_soft	2,622	tu50_18_soft	2,643	ht200_18_soft	2,846
tu5_16_soft	2,727	tu50_16_soft	2,473	ht200_16_soft	2,683
tu5_14_soft	2,421	tu50_14_soft	2,250	ht200_14_soft	2,479
tu5_12_soft	2,287	tu50_12_soft	2,013	ht200_12_soft	2,158
tu5_10_soft	2,043	tu50_10_soft	1,715	ht200_10_soft	1,770

A.4.3 Polynomial 17 ($1+ X^2 + X^3 +X^4$)

Without CRC

sc_10_soft	3,024
sc_9_soft	2,957
sc_8_soft	2,753
sc_7_soft	2,394
sc_6_soft	1,956
sc_5_soft	1,428
sc_4_soft	1,103

tu5_24_soft	2,965	tu50_24_soft	3,014	ht200_24_soft	3,071
tu5_22_soft	2,883	tu50_22_soft	2,910	ht200_22_soft	3,049
tu5_20_soft	2,788	tu50_20_soft	2,832	ht200_20_soft	3,009
tu5_18_soft	2,689	tu50_18_soft	2,683	ht200_18_soft	2,919
tu5_16_soft	2,705	tu50_16_soft	2,511	ht200_16_soft	2,752
tu5_14_soft	2,505	tu50_14_soft	2,267	ht200_14_soft	2,436
tu5_12_soft	2,289	tu50_12_soft	1,955	ht200_12_soft	2,093
tu5_10_soft	2,144	tu50_10_soft	1,725	ht200_10_soft	1,812

Frame stealing 10 %

sc_10_soft	2,979
sc_9_soft	2,921
sc_8_soft	2,723
sc_7_soft	2,358
sc_6_soft	1,970
sc_5_soft	1,437
sc_4_soft	1,129

tu5_24_soft	2,909	tu50_24_soft	2,988	ht200_24_soft	3,027
tu5_22_soft	2,825	tu50_22_soft	2,884	ht200_22_soft	3,009
tu5_20_soft	2,746	tu50_20_soft	2,779	ht200_20_soft	2,972
tu5_18_soft	2,650	tu50_18_soft	2,642	ht200_18_soft	2,876
tu5_16_soft	2,672	tu50_16_soft	2,479	ht200_16_soft	2,742
tu5_14_soft	2,422	tu50_14_soft	2,254	ht200_14_soft	2,477
tu5_12_soft	2,222	tu50_12_soft	1,940	ht200_12_soft	2,089
tu5_10_soft	2,091	tu50_10_soft	1,713	ht200_10_soft	1,807

Frame stealing 20 %

sc_10_soft	2,979
sc_9_soft	2,921
sc_8_soft	2,723
sc_7_soft	2,358
sc_6_soft	1,970
sc_5_soft	1,437
sc_4_soft	1,144

tu5_24_soft	2,876	tu50_24_soft	2,940	ht200_24_soft	2,972
tu5_22_soft	2,807	tu50_22_soft	2,852	ht200_22_soft	2,964
tu5_20_soft	2,725	tu50_20_soft	2,755	ht200_20_soft	2,913
tu5_18_soft	2,634	tu50_18_soft	2,628	ht200_18_soft	2,851
tu5_16_soft	2,664	tu50_16_soft	2,440	ht200_16_soft	2,695
tu5_14_soft	2,407	tu50_14_soft	2,262	ht200_14_soft	2,396
tu5_12_soft	2,255	tu50_12_soft	1,965	ht200_12_soft	2,075
tu5_10_soft	2,051	tu50_10_soft	1,718	ht200_10_soft	1,796

A.4.4 Polynomial 0F ($X + X^2 + X^3 + X^4$)

Without CRC

sc_10_soft	3,061
sc_9_soft	2,989
sc_8_soft	2,839
sc_7_soft	2,443
sc_6_soft	1,994
sc_5_soft	1,400
sc_4_soft	1,111

tu5_24_soft	2,958	tu50_24_soft	2,997	ht200_24_soft	3,054
tu5_22_soft	2,903	tu50_22_soft	2,913	ht200_22_soft	3,055
tu5_20_soft	2,849	tu50_20_soft	2,867	ht200_20_soft	3,012
tu5_18_soft	2,782	tu50_18_soft	2,689	ht200_18_soft	2,932
tu5_16_soft	2,789	tu50_16_soft	2,566	ht200_16_soft	2,779
tu5_14_soft	2,450	tu50_14_soft	2,303	ht200_14_soft	2,550
tu5_12_soft	2,298	tu50_12_soft	2,018	ht200_12_soft	2,235
tu5_10_soft	2,128	tu50_10_soft	1,747	ht200_10_soft	1,913

Frame stealing 10 %

sc_10_soft	3,001
sc_9_soft	2,935
sc_8_soft	2,756
sc_7_soft	2,446
sc_6_soft	1,970
sc_5_soft	1,413
sc_4_soft	1,079

tu5_24_soft	2,903	tu50_24_soft	2,965	ht200_24_soft	3,020
tu5_22_soft	2,854	tu50_22_soft	2,889	ht200_22_soft	2,998
tu5_20_soft	2,770	tu50_20_soft	2,850	ht200_20_soft	2,988
tu5_18_soft	2,741	tu50_18_soft	2,672	ht200_18_soft	2,922
tu5_16_soft	2,737	tu50_16_soft	2,521	ht200_16_soft	2,764
tu5_14_soft	2,448	tu50_14_soft	2,273	ht200_14_soft	2,566
tu5_12_soft	2,272	tu50_12_soft	1,999	ht200_12_soft	2,222
tu5_10_soft	2,123	tu50_10_soft	1,748	ht200_10_soft	1,850

Frame stealing 20 %

sc_10_soft	2,970
sc_9_soft	2,921
sc_8_soft	2,727
sc_7_soft	2,443
sc_6_soft	1,970
sc_5_soft	1,434
sc_4_soft	1,090

tu5_24_soft	2,898	tu50_24_soft	2,921	ht200_24_soft	2,969
tu5_22_soft	2,831	tu50_22_soft	2,854	ht200_22_soft	2,966
tu5_20_soft	2,741	tu50_20_soft	2,822	ht200_20_soft	2,931
tu5_18_soft	2,689	tu50_18_soft	2,665	ht200_18_soft	2,856
tu5_16_soft	2,672	tu50_16_soft	2,490	ht200_16_soft	2,710
tu5_14_soft	2,420	tu50_14_soft	2,236	ht200_14_soft	2,521
tu5_12_soft	2,266	tu50_12_soft	1,998	ht200_12_soft	2,210
tu5_10_soft	2,142	tu50_10_soft	1,733	ht200_10_soft	1,823

History

Document history		
V1.1.1	July 2007	Publication