DOI 10.5162/etc2014/7.3

Use of source coding techniques on Ariane 5 1553 data

Didier SCHOTT
AIRBUS Defense and Space
66 route de Verneuil,
78133 Les Mureaux Cedex – France
didier.schott@astrium.eads.net

Abstract:
In ETTC 2013 we have presented the use of source coding techniques on Ariane 5 measurement data [4]. This paper will now present the use of these techniques on 1553 data.

The paper will be in four parts. First we will present our analysis of the Ariane's 1553 data flow. In a second step we will describe the principal existing source coding techniques and present the trade-off we have made between these algorithms. In the third part we will compare the efficiency of the algorithms we have selected for Ariane 5 1553 flight data. Then in the fourth part we will speak about the further work we will do in this field.

Key words: Ariane 5, Telemetry, Source Coding, 1553.

1. Introduction

The data rate for Ariane 5 main telemetry system is historically limited to 1 Mbit/s. Within the framework of a CNES Launcher R&T project we have explored several tracks in order to increase the volume of data available in the telemetry link. This paper will present one of these studies: the use of source coding techniques on 1553 data.

2. Source coding for Ariane 5

2.1 Ariane 5 telemetry system

The telemetry system is not only used during validation phases (qualification flights), but also for “Commercial” flights. The system acquires the information of about 600 analogic sensors and 230 status, and spies the two avionic 1553 functional buses. It also manages an on-board memory which is used to record data for later transmission, for example when the launcher is not in the line of sight of a ground station.

The data are multiplexed by a “Central Telemetry Unit” and sent to ground in a CCSDS frame format structure.

The telemetry system is the only link with the launcher we have! Its function is to give information to the ground, in real-time and for post flight exploitation, about:

- The launcher behavior,
- The mechanical / thermal /... environment of the flight,
- Potential Anomalies and their localization,
- The trajectory, to predict payload orbits.

2.2 Source Coding Need

In the beginning of the flight the propulsion phase in the atmosphere generates a lot of vibration effects, meaning high volume of data to be transmitted to ground.

As the distance grows this data rate has to be reduced in order to guaranty the link budget. In some phases the launcher is not in visibility of the ground stations, the data are recorded to be transmitted later.

We see there the potential advantages of Source Coding:

- Increasing the quantity of data to be transmitted in the limited rate,
- Ameliorating the link budget by reducing the telemetry data rate,
- Limit the on-board memory size and speed up the restitution of recorded information.

2.3 Source coding main Requirements

The quality of the functional 1553 data is essential. All source coding techniques which

deteriorate data (“lossy” techniques) are excluded.

- Req 1: lossless algorithms

The coded data will have to be transmitted in autonomous packets (the content of a packet shall not depend of information in a previous packet). They shall contain all information mandatory to build the original data and their time-tags, with the same accuracy than non-coded parameters.

- Req 2: autonomous packets
- Req 3: same accuracy for time-tagging

3. Lossless source coding techniques

3.1 Main techniques

The main principles of lossless coding are:

- Dictionary coding: the principle is to replace a symbol or a group of symbols by a reference in a data structure (the dictionary)
  - Lempel-Ziv algorithms (LZ77 / LZ78 / LZW / LZSS ...).
- Entropic coding: each symbol is replaced by a variable length code. This code depends on the probability of the symbol in the data set (most frequent symbols gets shorter codes than less frequent ones)
  - Golomb / Rice / CCSDS [2][3],
  - Shanon Fano / Huffman / Arithmetic encoding.

These algorithms shall be non-adaptive (fixed dictionary or probability table), adaptive (the dictionary or table is built during coding) or half adaptive (two pass algorithm, first to build the dictionary, second to encode).

- Other: the redundancy in the message is eliminated by other means:
  - “Standard” RLE (the sequence ‘AAAAAAF’ is replaced by ‘6AF’)
  - 1553 specific algorithms [1]:
    - zero tracking (the “0000” value is coded by 1 bit),
    - differential encoding (unchanged data from a message to the next is coded with 1 bit),
    - adapted RLE (n 16 bits unchanged words W in a message are coded with n and W or with added bits indicating the position of the n W words).

3.2 Data Analysis

Information theory permits to evaluate the quantity of information sent by a source.

If p_i is the probability of appearance of the message i, the mean quantity of information (or Entropy) of the source is given by the relation:

$$H = -\sum_i p_i \log(p_i)$$

A low entropy figure means that the source sends a lot of redundant information. We calculate the entropy with the data of Ariane L549 flight (1/10/2009):

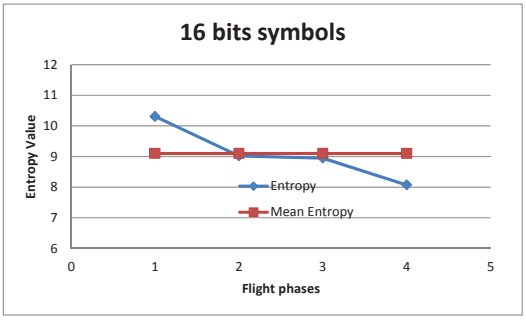

Fig. 1: entropy with 16 bits symbols

Globally the figures are rather high at the beginning of the flight (phase 1) but diminish in the next phases. It shall be easier to code the data in the last phase.

In our analysis we realize that some 16 bits figures appear more frequently than others:

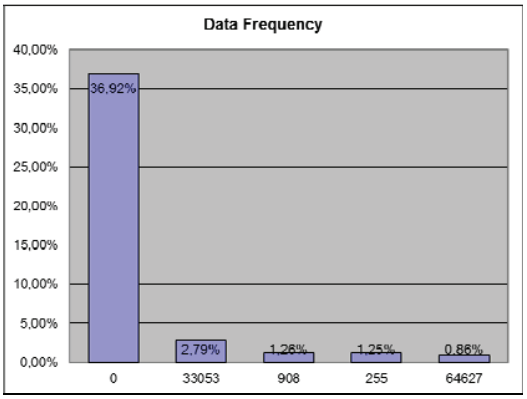

Fig. 2: frequency of some 16 bits values in the messages

The « 0000 » value appears very frequently in the messages (~37 % of the data words !).

Then we estimate how data evoluate in the messages:

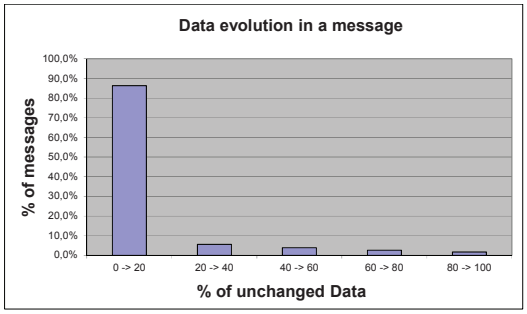

Fig. 3: proportion of unchanged data in each message (comparison of word n+1 in a message with word n)

For about 86 % of the messages, the data do evolve in each message.

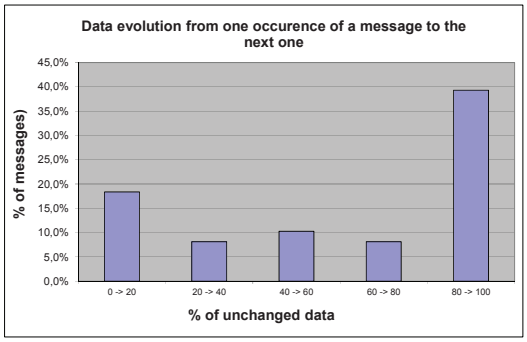

Fig. 4: proportion of unchanged data from one message to the next (comparison of each word of a message at t with corresponding words of the same message at t+dt)

For 40 percent of the 1553 messages, 80 to 100 % of the data don't evolve from an occurrence of a message to the next one.

4. Algorithm selection and description – first step

The results shown in figures 2 and 4 are very interesting. We decide in a first step to test algorithms based on 0 value detection, and evolutions in messages.

Two of the three algorithms described in [1] are clearly well adapted to this need: Zero Tracking and Differential Encoding.

The RLE encoding is not adapted to our need, due to the low proportion of unchanged data shown in figure 3.

4.1 Zero Tracking Encoding [1]

Depending on the message length, 1 to 2 words are created at the beginning of the coded message. The content of these words indicate the position of the "0000" in the message.

In the example hereafter, the added word is CBD7. The position of the "0000" values is given by each bit of this word (ZT column):

Word Count (Hex)	Input Data (Hex)	ZT	Encoded Data (Hex)
0	0000	1	CBD7
1	0000	1	FFFF
2	FFFF	0	0059
3	0059	0	AC9F
4	0000	1	0486
5	AC9F	0	F5A9
6	0000	1	
7	0000	1	
8	0000	1	
9	0000	1	
A	0486	0	
B	0000	1	
C	F5A9	0	
D	0000	1	
E	0000	1	
F	0000	1	

Table 1: Zero tracking encoding example

1 indicates that the corresponding word equals "0000".

The other values follow the added word in their appearance order.

4.2 Differential Encoding [1]

As in previous algorithm 1 to 2 words are added at the beginning of the coded message. The content of these words indicate the position of the unchanged value in consecutive messages.

In the example hereafter, the added word is 2022. The position of the modified values is given by each bit of this word (DT column):

Word Count (Hex)	Previous Data (Hex)	Current Data (Hex)	DT	Encoded Data (Hex)
0	0054	0054	0	2022
1	0815	0815	0	AF00
2	AF58	AF00	1	4567
3	0000	0000	0	AAAA
4	0000	0000	0	
5	6542	6542	0	
6	FFFF	FFFF	0	
7	9542	9542	0	
8	BC65	BC65	0	
9	0000	0000	0	
A	0000	4567	1	
B	0000	0000	0	
C	8966	8966	0	
D	8966	8966	0	
E	5634	AAAA	1	
F	0054	0054	0	

Table 2: differential encoding example

1 indicates that the value has changed; the modified values follow the first coded word.

0 indicates an unchanged value => this algorithm needs a first reference message.

5. Tests

5.1 Algorithms efficiency – maximum performance

First we test the algorithms with the data of the whole flight.

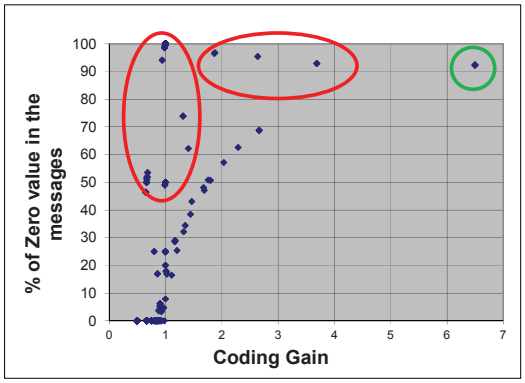

Fig. 5: zero-tracking algorithm – gain compared with % of zero values in messages

The coding gain increases with the proportion of zero values in message, to a maximum value of 6.5 (in green) … but there are curious results in red: even with a great proportion of zero values, the coding gain is lower than expected. This is due to the messages length: all the corresponding messages have only one to four data words.

For the whole flight the Zero-tracking algorithm gives a coding gain of 1.31.

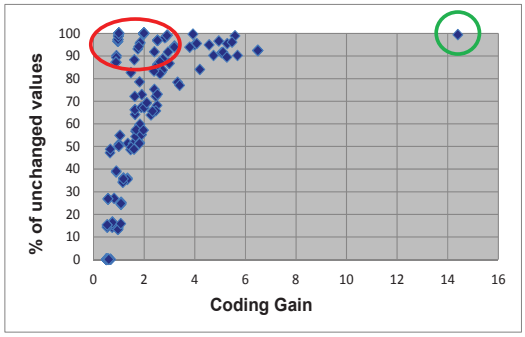

Fig. 6: differential encoding algorithm – gain compared with % of unchanged values in messages

The coding gain also increases with the % of unchanged values, to a maximum of 14.4 in green. The curious results in red are also due to the short message length.

For the whole flight the differential encoding algorithm gives a coding gain of 2.35.

5.2 Algorithms efficiency with data assembled in packets

In a second step we assemble the data in packets before coding, as they will be processed on-board. Each packet will contain the data of n occurrences of the same message. This grouping has no impact with Zero-tracking algorithm (coding of each message independently).

With the differential encoding algorithm there is an impact: each new packet must contain a non-coded reference message to be compliant with Req. 2. We test the algorithm with different packet size (number of messages in a packet) and different duration (accumulation of data during a limited time):

Packet Size	Gain	Duration (s)	Gain
8	2,018	5	2,266
16	2,173	10	2,309
32	2,260	20	2,331
64	2,306	40	2,343
128	2,330	80	2,348
256	2,342	160	2,351
512	2,348	320	2,353
1024	2,351	640	2,354
2048	2,353	Max	2,354
4096	2,354		
Max	2,354		

Table 3: gain for different packet size and different duration - Max: 1 packet contains all occurrences of 1 message

The maximum coding gain is almost reached for packets with 512 or more messages. But as some messages are much slower than others, this approach is not the good one (too much time needed to make big packets).

The duration approach is a better one. For 10 second on, the gain (2.31) is near the maximum (2.35).

Let's see the performance of the algorithms with this duration approach for the whole flight:

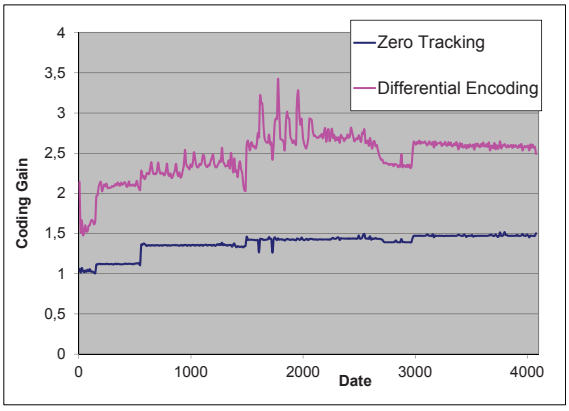

Fig. 7: Algorithms comparison with 10 seconds packets

Differential encoding is systematically better than Zero-Tracking, with a coding gain varying from 1.5 to 3.5.

6. Algorithm selection and description – second step

The algorithms we have tested were finally not so efficient, comparing to those tested for telemetry data [4]. So we decide to make additional tests with some of these algorithms: RLE, adaptive LZW and CCSDS.

6.1 RLE algorithm

The RLE encoding replaces symbol repetitions in a message by the number of repetitions followed by the symbol:

AAAAAAAF -> 7AF

For 1553 data we choose the following data structure to code the packets:

1 bit	7 bits	8 bits	1 bit	7 bits	8 bits	...	8 bits
1 : indicates repetition	Repetition count	Symbol to be repeted	0 : no repetition	Number n of symbols	Symbol 1	S ...	Symbol n

Fig. 8: RLE data structure - 8 bit counter / 8 bits 0 to FF symbols (RLE 8/8). Symbols are coded when the number of repetition is > 2.

Alternate versions with 16 bits counters and/or symbols were also tested, but were less efficient.

6.2 Adaptive LZW algorithm

The Lemper-Zil-Welch algorithm is a substitution type coding. The encoder and decoder have the same dictionary containing the individual symbols. The encoder searches the symbol to be coded in the dictionary and transmit only its position in the dictionary. In the same step it creates a new entry in the dictionary by concatenating the symbol with the previous ones.

Coding algorithm:

```
D = Null;
while (read a char C) do
   if (DC exists in the dictionary) then
      D=DC;
   else
      add DC to dictionary;
      write the D code;
      D = C;
   end if
end while
write the D code;
```

The main interest of this algorithm is that there is no need to transmit the dictionary: the encoder and decoder generate directly this dictionary during the coding/decoding. We choose 8 bits symbols (for 16 bits symbols the initial 65536 values initial dictionary would be too important).

6.3 CCSDS algorithm

The algorithm is built with two functional blocks ([2], [3]):

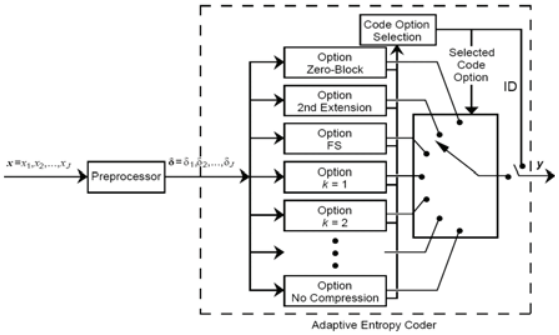

Fig. 9: CCSDS algorithm functional diagram

Pre-processor:

Each n bits sample is compared with an estimation (generally the previous value). The difference between estimation and real value (n+ 1 bits) is transformed in an n bits integer by a specific function. These d[i] coded values are transmitted to the second block.

Adaptive Entropy Coder:

The d[i] output symbols from the pre-processor are coded in parallel with different methods:

- Option "zero block": when the bloc is constituted only with 0 values, only the number of blocs with 0 is transmitted,
- Option "2nd extension" (SE): the J symbols are paired and transformed in J/2 coded information y[i], with y[i]=(d[i] + d[i+1])(d[i] + d[i+1] + 1)/2 + d[i+1],
- Option "Fundamental Sequence" (FS format): the 'm' value is coded with m zeroes followed by one 1 (0: 1 | 1: 01 | 2: 001 | 3: 0001...),
- Option "sample splitting" (Rice) – k order (k=1 à x): the n-k most significant bits are coded with FS format, the k less significant bits are grouped and placed after the coded MSB,
- Option "No compression": direct transmission of the input bloc.

The coder chooses the option which gives the best compression ratio. The coded bloc is transmitted with a header which identifies the option.

We kept the data structure used for telemetry data:

- 8 bits symbols
- One reference value(s) in each bloc
- New "no compression / no reference" option, for incomplete blocs which will be transmitted directly before the pre-processor stage.
- Option coded with four bits

6.4 Data preparation before coding

The data must be organized differently for the CCSDS algorithm. So we decide to pre-process the data packet before coding, by adapting for 1553 data what we do in [4]. A first simple transposition was foreseen, with 8 bits or 16 bits symbols:

Mes 1 :	ABCD	1230	4567	9874		
Mes 2 :	ABCD	1230	4568	9874		
Mes 3 :	ABCD	1235	4567	9874		
Initial data bloc:						
ABCD	1230	4567	9874	ABCD	1230	…
4568	9874	ABCD	1235	4567	9874	
8 bits transposed data bloc:						
ABAB	ABCD	CDCD	1212	1230	3035	,,,
4545	4567	6867	9898	9874	7474	
16 bits transposed data bloc for 16 bits algorithms:						
ABCD	ABCD	ABCD	1230	1230	1235	,,,
4567	4568	4567	9874	9874	9874	

Fig. 10: pre-processing 1 – simple transposition

We realize that this transposition is not sufficient for CCSDS, because of its functioning based on the evolution of a physical parameter. A new transposition has been tested:

Mes 1 : 8B39 01E6				
Mes 2 : 8B39 01E6				
...				
Mes 124 : 8B39 01E6				
8 bits transposition :				
8B 8B …..(124)….. 8B 39 39 ....39 01 01 ....01 E6 E6 .....E6				
CCSDS Coding				
28 BF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FE 00 …				
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 …				
3F FF FF FF FF FF FF FF FF FF FF FF FF FF FC 00 00 …				
00 00 00 00 00 00 00 00 00 00 07 FF FF FF FF FF FF …				
FF FF FF FF FF FF FF FF FF 80 00 00 00 00 00 00 00 …				
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 …				
01 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF E				
=> Coding Gain 3,92				
New transposition			**CCSDS coding**	
8B 8B .....(124).. 8B		->	08 B8	
39 39 39		->	03 98	
01 01 01		->	00 18	
E6 E6 .....E6		->	0E 68	
=> Coding Gain 62 !!!				

Fig. 11: pre-processing 2 – new transposition with 124 identical messages – each 8 bits resulting column is coded independently.

6.5 Results

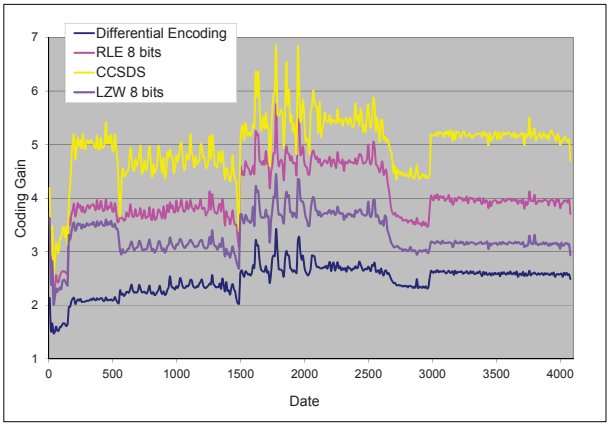

Fig. 12: Algorithms comparison with 10 seconds packets – CCSDS, RLE and LZW used pre-processed data with the new algorithm.

CCSDS algorithm gives the best coding gain in all flight phases, varying from 2.6 to 6.8.

8. Conclusion and future work

This study shows that the same CCSDS algorithm can be used for 1553 data and telemetry data, with high coding gain.

We decide to continue its evaluation:

- Noise / disturbance sensibility
- System aspects (behavior in case of data loss)
- Implantation of the algorithm in on-board type components (evaluation of the calculation time)
- Transmission aspects in CCSDS frame ("real" compression ratio)

References

[1] IEEE Aerospace Conference 2008: Application of Data Compression to the MIL-STD-1553 Data Bus (Russell W. Duren and Michael W. Thompson)

[2] CCSDS 121.0-B-1: Recommendation for space data system standards Lossless data compression, Blue book

[3] CCSDS 120.0-G-2: Recommendation for space data system standards Lossless data compression, Green book

[4] ETTC 2013 Conference: Use of source coding techniques on Ariane 5 telemetry data (Didier Schott).