ATNP/WG2/WP407
IP/
23 June 1997

AERONAUTICAL TELECOMMUNICATIONS NETWORK PANEL

WORKING GROUP TWO

Langen 24.6.97-26.6.97

Data Link Compression Evaluation Report

Presented By Tony Whyman

Prepared by Tony Whyman

SUMMARY

This report has been expanded to for the original validation of the Mobile SNDCF Compression Algorithsm to include analysis of the Deflate compression algorithm specified in IETF RFC 1951, as an alternative to the LZW compression algorithm. This work has major cost implications for ATN users, as the original version showed, there was a 24% additional reduction of traffic volumes when the LREF and LZW compression algorithms were used together compared with either algorithm on its own. However, there are both defects in the SARPs on the use of LZW, and commercial problems in properly exploiting the LZW algorithm. Recognising the substantial cost savings that the use of such an algorithm implies, this report now explores the use of *Deflate as* an alternative to LZW; it is hoped that Deflate can achieve the same benefits without the commercial problems involved in using LZW. The view is supported by the results of the analysis.

TABLE OF CONTENTS

Preface to Issue 2.0

In Issue 2.0, this report has been expanded to include analysis of the Deflate compression algorithm specified in IETF RFC 1951, as an alternative to the LZW compression algorithm. This work has major cost implications for ATN users, as Issue 1.0 showed there was a 24% additional reduction of traffic volumes when the LREF and LZW compression algorithms were used together compared with either algorithm on its own. However, there are both defects in the SARPs on the use of LZW, and commercial problems in properly exploiting the LZW algorithm. Recognising the substantial cost savings that the use of such an algorithm implies, this report now explores the use of *Deflate as* an alternative to LZW; it is hoped that Deflate can achieve the same benefits without the commercial problems involved in using LZW.

- As was noted in Issue 1.0, the current SARPs are defective in that they simply refer to use of V.42bis for compression of data transferred by the Mobile SNDCF, and assume that that is sufficient specification. However, V.42bis is a stream compression algorithm specifically for use with the V.42 LAP-M data link protocol, and has to be adapted for use with packet mode communications and the ISO 8208 network service. This adaptation is missing from the SARPs.

- The commercial problems exist because the LZW compression algorithm used by V.42bis is covered by a patent owned by Unisys Inc., and there clearly could be problems in specifying the use of the algorithm in a SARP without negotiating the right to use the algorithm. It has to be noted that Unisys has been active in pursuing its rights to LZW and in demanding royalty payments for its use.

The Deflate algorithm offers an interesting alternative to LZW. Firstly, its specification is in the public domain and appears to be free of any encumbrances due to patents or other restrictions on the intellectual property right. Secondly, a standard ‘C’ implementation is also in the public domain and may thus be readily used to evaluate the algorithm and to incorporate in products. Thirdly, and by no means least, Deflate appears to be a very good compression algorithm that is tried and tested in general commercial use. It is the compression algorithm used by the popular compression utility *pkzip*, the *gzip* utility widely used on both Unix and MSDOS systems, and Deflate is now appearing in many other applications in data communications e.g. for compressing graphics in World Wide Web pages.

The standard ‘C’ implementation of Deflate already permits compression on a packet basis, and also has facilities for pre-loading a dictionary of known phrases. Firstly, this permits use in the ATN without modification and secondly opens up the possibility of maximising data compression and avoiding much of the learning phase for common ATC Applications. This would avoid the need to “hand craft” compression schemes for ATC Applications, as has been proposed in the past.

The analysis that was undertaken for Issue 1.0 has been repeated, using Deflate as an LZW alternative. This new analysis is incorporated in Issue 2.0, with the results contrasted with those previously obtained for LZW.

1. Initiative Reference & Title

Data Link Compression Analysis.

2. Type

Analysis of experimental data using off-line compression programs.

3. Responsible State/Organisation

EUROCONTROL

4. Contact Point

State/Organisation	Contact Details
EUROCONTROL	Mr. Henk Hof Rue De La Fusée 96 B-1030 Brussels, Belgium Tel: + 32 2 729 3329 Fax: + 32 3 729 9083 Internet: henk.hof@eurocontrol.be

5. Validation Tools Involved

These experiments made use of the following ATN systems available at the various sites:

Name	Annex B reference	Involvement
ADS Europe	tbd	ADS Europe was used as a source for the experimental data used for the analysis.

In addition, the experiments made use of the following supporting tools/functions.

Name	Annex B reference	Involvement
Compression Analysis Tools	—	The Data Compression Analysis tools are a set of MSDOS programs that analyse, compress and report on experimentally derived data. The tools implement LREF, LZW, Deflate and ACA Compression Algorithms.
Sniffer	—	A multipurpose protocol analyser has been used at EEC site to record and analyse the X.25 and Ethernet traffic as required by the experiment.

6. Validation Periods

The analysis was performed in September 1996 using data gather during the previous month, and repeated for Issue 2.0 in January 1997.

7. Validation Objectives

The following AVOs are applicable to these exercises:

AVO_454 Evaluate the compression ratio of ATN compression mechanisms for typical user application dialogues and average routing information exchanges in the following cases:

a) no compression (base reference)

b) LREF compression only

c) LREF + ACA

d) LREF + V.42bis

e) LREF + ACA + V.42bis

AVO_455 Evaluate the impact of the SNDCF Compression mechanisms on the ATN Service performance.

8. Description

8.1 Data Compression Analysis Tools

The Data Compression Analysis Tools are a set of four MSDOS programs that are used to perform the following tasks:

- LREF Data Compression Prediction
- ACA Data Compression Prediction
- LZW Data Compression Prediction
- “Deflate” Data Compression Prediction

This programs are designed to be used serially. That is, the output of one may be used as input to another, and hence the impact of using more than one compression algorithm in series, can be investigated.

8.1.1 LREF Compression Predictor

The purpose of the LREF Compression Predictor is analyse a dataset comprising CLNP PDUs and to predict the compression ratio that will be achieved using LREF compression. The dataset is typically an historical recording of a real data transfer, and the prediction will determine the accumulative compression ratio achieved after a number of test periods (e.g. 30 seconds, 1 minute, 30 minutes, etc.). The format of the input data is a plain text report as produced by the Sniffer Analysis Tool.

The LREF Compression Predictor is an MSDOS command line program. The command line syntax is:

```
complref <source file> <output file> <report file> <sample
frequency>
```

where:

<source file>	contains the input dataset in the format produced by the Sniffer line analyser used at the EEC. This file is assumed to contain timing information as well as data.
<output file>	contains the result of the compression of the dataset in Sniffer File format
<report file>	is the file used to contain the output data. This file is a text file and contains a line of text for each sample point in the form of comma delimited data. Four fields are present in each line providing, respectively: • the sample point, • the number of bits read so far from the dataset • the number of bits predicted to result from the application of the LREF compression algorithm, and • the number of LREFs assigned so far.
<sample frequency>	This is the sampling rate, in seconds. i.e. it provides the number of seconds between each sample point.

The program is written in 'C' and implements the compression of CLNP PDUs using the LREF algorithm specified in the SARPs. The implementation is non-optimised in the sense that a simple linear search of the LREF Directory is used, but is otherwise a proper implementation of the specification. The implementation includes decompression functions and was tested by compressing and decompressing sample data and then comparing the result with the original data.

8.1.2 LZW Compression Analyser

The purpose of the LZW Compression Analyser is to compress an input dataset and to determine the achieved compression ratio. The dataset is typically an historical recording of a real data transfer, and the prediction determines the accumulative compression ratio achieved after a number of test periods (e.g. 30 seconds, 1 minute, 30 minutes, etc.).

The LZW Compression Analyser is an MSDOS command line program. The command line syntax is:

```
complzw <source file> <output file> <report file> <sample frequency>
```

where:

<source file>	contains the input dataset in the format produced by the Sniffer line analyser used at the EEC. This file is assumed to contain timing information as well as data. The timing information is not included in the scope of the compression.

<output file>	contains the result of the compression of the dataset in Sniffer File format
<report file>	is the file used to contain the output data. This file is a text file and contains a line of text for each sample point in the form of comma delimited data. Four fields are present in each line providing, respectively: • the sample point, • the number of bits read so far from the dataset, and • the number of bits predicted to result from the application of the LZW compression algorithm.
<sample frequency>	This is the sampling rate, in seconds. i.e. it provides the number of seconds between each sample point.

The ATN SARPs currently specify ITU-T V.42bis as a compression algorithm. V.42bis is derived from LZW and this test tool is intended to be an approximation to the compression that could be achieved from V.42bis. The implementation was derived from published 'C' source for LZW in "The Data Compression Book" by Mark Nelson and is very similar to that used in the Unix Compress program.

Both compression and decompression algorithms were implemented, with back-to-back testing in order to verify the implementation.

An early result from this implementation was to note that the SARPs specification needs considerable clarification and needs a change in approach to be appropriate for ATN use.

The problem is that V.42bis is specified for stream oriented communications and not packet oriented. However, in the ATN each packet must be compressed separately and then sent as a single message. The algorithm needs to be adapted for this purpose. Further, variants of the LZW algorithm were discovered during this work. The one implemented in this program uses a "fast-start" approach, starting with a small (9-bit) codeword and gradually advancing to a larger (15-bit) codeword. This appears to be particularly suitable for ATN use, where many dialogues will be short.

8.1.3 "Deflate" Compression Analyser

The purpose of the "Deflate" Compression Analyser is to compress an input dataset and to determine the achieved compression ratio. The dataset is typically an historical recording of a real data transfer, and the prediction determines the accumulative compression ratio achieved after a number of test periods (e.g. 30 seconds, 1 minute, 30 minutes, etc.).

The "Deflate" Compression Analyser is an MSDOS command line program. The command line syntax is:

```
compdfl <source file> <output file> <report file> <sample frequency>
```

where:

<source file>	contains the input dataset in the format produced by the Sniffer line analyser used at the EEC. This file is assumed to contain timing information as well as data. The timing information is not included in the scope of the compression.

<output file>	contains the result of the compression of the dataset in Sniffer File format
<report file>	is the file used to contain the output data. This file is a text file and contains a line of text for each sample point in the form of comma delimited data. Four fields are present in each line providing, respectively: • the sample point, • the number of bits read so far from the dataset, and • the number of bits predicted to result from the application of the "Deflate" compression algorithm.
<sample frequency>	This is the sampling rate, in seconds. i.e. it provides the number of seconds between each sample point.

The "Deflate" algorithm is specified in IETF RFC 1051. The standard 'C' implementation of "Deflate" was obtained from *http://quest.jpl.nasa.gov/zlib/*.

Both compression and decompression algorithms were implemented, with back-to-back testing in order to verify the implementation.

Referring to the standard 'C' implementation, the initialisation parameters were set for maximum compression and to suppress the ZLIB header and trailer; the code defaults to the ZLIB format specified in RFC 1950 and which has a short header and a 32-bit checksum as a trailer. The Z_PARTIAL_FLUSH parameter was specified on each call to the compressor, in order to ensure complete compression of each packet with the resultant compressed data being decompressible to the original packet without the need for information from the subsequent compressed block of data.

8.1.4 ACA Compression Analyser

The purpose of the ACA Compression Analyser is to compress an input dataset and to determine the achieved compression ratio. The dataset is typically an historical recording of a real data transfer, and the prediction will determine the accumulative compression ratio achieved after a number of test periods (e.g. 30 seconds, 1 minute, 30 minutes, etc.).

The ACA Compression Analyser is an MSDOS command line program. The command line syntax is:

```
compaca <source file> <output file> <report file> <sample frequency>
```

where:

<source file>	contains the input dataset in the format produced by the Sniffer line analyser used at the EEC. This file is assumed to contain timing information as well as data. The timing information is not included in the scope of the compression.
<output file>	contains the result of the compression of the dataset in Sniffer File format
<report file>	is the file used to contain the output data. This file is a text file and contains a line of text for each sample point in the form of comma delimited data. Four fields are present in each line providing,

	respectively: • the sample point, • the number of bits read so far from the dataset, and • the number of bits predicted to result from the application of the 1. ACA compression algorithm.
<sample frequency>	This is the sampling rate, in seconds. i.e. it provides the number of seconds between each sample point.

The 'C' source for this program was provided by MIT and written by Bob Grappel (MIT) and Klaus-Peter Berg (Technical University of Braunschwig) as part of their original validation work on the algorithm. It was ported into the MSDOS environment and validated using their original test suite.

8.2 Experimentally Derived Data

The data for the analysis exercises was taken from a recording made during the ADS Europe trials. The recording was made by the Sniffer Tool on the X.25 ground data link to the GES carrying data to be uplinked to an aircraft in flight. The analysis could therefore be performed on uncompressed uplink data. The recording was made over a period of approximately five hours from 10:22:21 to 15:04:40 on the day of the recording.

A particular feature of the data was a high number of ISH PDUs. Out of a total of 2010 packets monitored, 1694 were ISH PDUs. This reflected a known problem of the ADS Europe trial in that the holding time for ISH PDUs was set far too short. This fact has been taken into account in the analysis exercises with the exercises being performed both on the original data and a data set with the ISH PDUs filtered out. The impact of ISH PDUs can therefore be properly assessed.

8.3 Analysis Exercises

A total of twenty-two exercises were performed, comprising eleven exercises each run on two sets of data (the original recorded data and data filtered to remove ISH PDUs). The exercises comprised analysis of the data by each of the compression tools, and then using the tools in valid combinations (e.g. first performing LREF compression, then LZW). These exercises were:

1. Analysis by LREF
2. Analysis by LZW
3. Analysis by ACA
4. Analysis be "Deflate"
5. Analysis by LREF, then LZW
6. Analysed by LREF, then "Deflate"
7. Analysis by LREF, then ACA
8. Analysis by LREF, then ACA and finally LZW

9. Analysis by LREF, then ACA and finally “Deflate”

10. Analysis by ACA and then LZW.

11. Analysis by ACA and then “Deflate”.

In each case, the analysis provided a summary result of number of bits in, number of bits after compression and the achieved compression ratio. A report set was also produced showing the bits in, bits out figure for 30 second sample points.

8.4 Analysis of Results

The results were analysed using a set of Excel spreadsheets. This analysis resulted in two sets of results:

1. Summary results showing the overall compression ratio achieved by each exercise.

2. Convergence analysis showing the rate of convergence on the best compression ratio for each exercise.

The results could then be assessed through inspection.

9. Results

9.1 Overall Compression Achieved

Table 9-1 and Table 9-2 show, respectively, the actual compression achieved by each exercise and the overall compression ratio achieved. This data is then presented graphically in Figure 9-1.

The first observation to make is the wide variation shown in some of the exercises between the compression ratio achieved on the different datasets. This can be explained by the fact that the ISH PDU is not compressible by the LREF algorithm, while it is very compressible by the LZW and Deflate algorithms. On the other hand. the ACA algorithm can give some compression to ISH PDUs as well as CLNP PDUs. Thus LZW and Deflate give a much better compression ratio to the unfiltered data, while with LREF, it is the other way around.

An interesting difference emerges between Deflate and LZW. LZW is clearly better at compressing the unfiltered data, whilst Deflate is better at compressing the filtered data. It appears that LZW will converge to a smaller compressed PDU size with highly repetitive

			Compressed Size After Applying Specified Algorithm (in bits)										
	Original Data (packets)	Original Data (bits)	LREF	LZW	Deflate	ACA	LREF /LZW	LREF/ Deflate	LREF /ACA	LREF /ACA /LZW	LREF/ ACA/ Deflate	ACA /LZW	ACA/ Deflate
Recorded Data	2,010	661,480	532,984	108,896	134,336	641,256	85,192	120,120	532,920	85,264	120,192	106,960	133,200
Filtered Data	316	200,712	72,216	78,040	65,064	180,488	55,112	52,520	72,152	55,112	52,592	76,264	64,816

Table 9-1 Compression Achieved by each Exercise

		Compression Ratios										
		LREF	LZW	Deflate	ACA	LREF /LZW	LREF/ Deflate	LREF /ACA	LREF /ACA /LZW	LREF/ ACA/ Deflate	ACA /LZW	ACA/ Deflate
Recorded Data		1.241088	6.07442	4.924071	1.031538	7.764579	5.506827	1.241237	7.758022	5.503528	6.184368	4.966066
Filtered Data		2.779329	2.571912	3.08484	1.112052	3.641893	3.82163	2.781794	3.641893	3.816398	2.631805	3.096643

Table 9-2 Compression Ratios Achieved by each Exercise

data, whilst Deflate is general better with a more normal distribution. As the large number of ISH PDUs in the unfiltered data was the result of a wrong assumption, and in operational use, the number of ISH PDUs is likely to be much lower, the implication is that the two algorithms should give broadly similar results in a more normal traffic mix. Furthermore, ISH PDUs are only exchanged after the Route Initiation phase in the optional non-use of IDRP. When IDRP is used, ISH PDUs are not exchanged. Therefore Deflate should be expected to give better results than LZW with operational ATN data.

Looking at the detail of what happened, inspection of the detailed output showed that once the compression algorithm had converged, LZW could compress each ISH PDU down to two octets - the theoretical minimum. In the data used, most of the early PDUs were ISH PDUs so it is possible to show very clearly how the LZW algorithm converges. Figure 9-2 shows the first fifty lines of the output of the LZW compressor when acting on the experimental data, and illustrates a very smooth convergence to a minimal packet size.

This should be compared with Figure 9-3, which shows the convergence of Deflate on the same data. Instead of converging down to 2 octets, it goes into a repetitive loop outputting four different code sequences of 4 to 5 octets in length. Hence the difference in performance for the unfiltered data. On the other hand, the convergence rate is much faster, and Deflate achieves its maximum compression after only three packets. In comparison, LZW takes twenty three packets to achieve the same.

This observation probably explains why Deflate gives better compression on more random data streams than does LZW. The very rapid convergence of Deflate permits high compression levels to be achieved with only a small amount of redundancy in the data, whilst the slower converging LZW gets a higher compression ratio out of data streams with a higher degree of redundancy.

The ACA appears to give significantly less benefits that the other algorithms. The best compression ratio achieved is 1.11, when run on its own on the filtered data. This is a useful but not spectacular contribution. However, when run after LREF, there seems almost nothing left for it to compress, and its contribution is only to the third decimal place. This is in sharp contrast to LZW and Deflate, which both find much to work on even after LREF has done its work. This observation is in line with original criticism of the ACA algorithm in that while there appeared to be redundancy in NSAP Addresses, this was both small in comparison to the user data and would anyway be minimised by the LREF or Deflate algorithms.

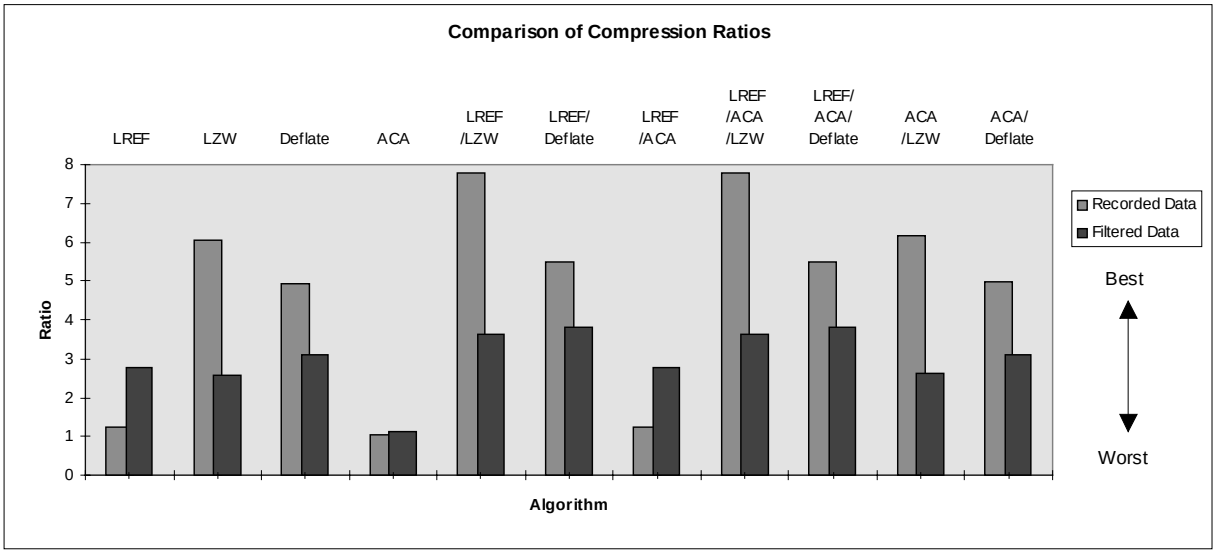

Figure 9-1

The combination of LREF and LZW is clearly the best for unfiltered data, whilst LREF and Deflate is best for the filtered data stream. On its own LREF marginally out-performs LZW on filtered CLNP only data stream, whilst Deflate out performs both. The situation is reversed on the unfiltered data stream, as LREF cannot compress ISH PDUs, and LZW finds these very compressible, LZW is about twice as good on the original data stream. Deflate also finds these compressible, but not by so much. On a more normal data stream, LZW should out-perform LREF wherever there is a significant number of ISH PDUs. Deflate should always out perform LREF, and will out perform LZW unless there is an unusually high number of ISH PDUs.

Both LREF and LZW, and LREF and Deflate appear to be complementary. When LREF and LZW are used in combination, there appears to be a gain in the compression ratio of between 1 and 1.5, which is a very significant result. When LREF and Deflate are used together, the gain is not so impressive, but this is due to Deflate achieving a better performance on its own. Together LREF and Deflate still outperform LREF and LZW on the filtered data stream, with a compression ratio of 3.82 compared with 3.64.

Looking at the filtered data stream alone, the data transferred is reduced by 23.7% using LREF and LZW combined, compared with LREF on its own, and 29.4% compared with LZW on its own. The improvement is even better on the unfiltered data stream.

Similarly, on the filtered data, the data transferred is reduced by 27.3% using LREF and Deflate combined, compared with LREF on its own, and 19.3% compared with Deflate on its own. The original filtered data stream of 200,712 bits is reduced to 55,112 bits when using LREF and LZW, and to 52,520 when using LREF and Deflate, and there is thus good reason to use both LREF and either LZW or Deflate, in series.

It should also be noted that LREF/ACA/LZW seem to interfere with each other. The result is slightly worse on the original data stream compared with LREF/LZW, while no difference is made to the filtered dataset. We see a similar problem with LREF/ACA/Deflate.

Frame	Time	Source	M Bit	Bytes	Data
1	10:22:21.6709	DTE	0	35	
	410880200020003c00000508e0013a048a55290048a432080229162214e0ac601008b1				
2	10:22:31.6725	DTE	0	20	81c160f0985c361f1189c562f1920c6e3b1f2c40
3	10:22:41.6749	DTE	0	17	90c1a110a8643a211296462552e0047a41
4	10:22:51.6764	DTE	0	14	98c8e69269bca62d3a8d4727b308
5	10:23:01.6796	DTE	0	12	9fcce4b36944e6593c9f1620
6	10:23:11.6812	DTE	0	12	a54926b279c4aa8d2da454a0
7	10:23:21.6828	DTE	0	10	a9d069b57a2d42b73080
8	10:23:31.6844	DTE	0	10	ae532ad4495ceec52080
9	10:23:41.6876	DTE	0	8	b1d56874fb44be41
10	10:23:51.6892	DTE	0	8	b550a9d58b0dbcb1
11	10:24:01.6907	DTE	0	7	b857acd59a8cc2
12	10:24:11.6923	DTE	0	6	bb596db47bac
13	10:24:21.6963	DTE	0	7	bddb2e76ea4c82
14	10:24:31.6971	DTE	0	6	bfdcac182a94
15	10:24:41.6995	DTE	0	7	c257ecf7ec1962
16	10:24:51.7011	DTE	0	5	c45e2e98c8
17	10:25:01.7043	DTE	0	5	c6df6b5758
18	10:25:11.7059	DTE	0	5	c860717528
19	10:25:21.7083	DTE	0	5	c9e1b2b308
20	10:25:31.7090	DTE	0	5	cb62b258c8
21	10:25:41.7122	DTE	0	5	ccde6d2588
22	10:25:51.7138	DTE	0	4	ce63ea50
23	10:26:01.7162	DTE	0	4	cfe1e610
24	10:26:11.7178	DTE	0	4	d0e62410
25	10:26:21.7210	DTE	0	4	d1e6ea50
26	10:26:31.7217	DTE	0	4	d2dea410
27	10:26:41.7241	DTE	0	3	d3e7a0
28	10:26:51.7257	DTE	0	3	d4dd60
29	10:27:01.7294	DTE	0	3	d56320
30	10:27:11.7303	DTE	0	3	d5d4a0
31	10:27:21.7318	DTE	0	3	d64c20
32	10:27:31.7342	DTE	0	3	d6c820
33	10:27:41.7374	DTE	0	3	d71620
34	10:27:51.7384	DTE	0	2	d7c0
35	10:28:01.7408	DTE	0	2	d7c0
36	10:28:11.7424	DTE	0	2	d7c0
37	10:28:21.7455	DTE	0	2	d7c0
38	10:28:31.7471	DTE	0	2	d7c0
39	10:28:41.7487	DTE	0	2	d7c0
40	10:28:51.7505	DTE	0	2	d7c0
41	10:29:01.7537	DTE	0	2	d7c0
42	10:29:11.7553	DTE	0	2	d7c0
43	10:29:21.7567	DTE	0	2	d7c0
44	10:29:31.7591	DTE	0	2	d7c0
45	10:29:41.7623	DTE	0	2	d7c0
46	10:29:51.7634	DTE	0	2	d7c0
47	10:30:01.7658	DTE	0	2	d7c0
48	10:30:11.7674	DTE	0	2	d7c0
49	10:30:21.6996	DTE	0	2	d7c0
50	10:30:31.7020	DTE	0	2	d7c0

Figure 9-2 LZW Convergence for ISH PDU Compression

Frame	Time	Source	M Bit	Bytes	Data
1	10:22:21.6709	DTE	0	34	
	6a52626460619063601071675	06f740d0d627475766400112ec18c0cc79898220002			
2	10:22:31.6725	DTE	0	6	a889a00a8000
3	10:22:41.6749	DTE	0	4	22ac0220
4	10:22:51.6764	DTE	0	5	8008ab0008
5	10:23:01.6796	DTE	0	5	20c22a0002
6	10:23:11.6812	DTE	0	5	88b00a8000
7	10:23:21.6828	DTE	0	4	22ac0220
8	10:23:31.6844	DTE	0	5	8008ab0008
9	10:23:41.6876	DTE	0	5	20c22a0002
10	10:23:51.6892	DTE	0	5	88b00a8000
11	10:24:01.6907	DTE	0	4	22ac0220
12	10:24:11.6923	DTE	0	5	8008ab0008
13	10:24:21.6963	DTE	0	5	20c22a0002
14	10:24:31.6971	DTE	0	5	88b00a8000
15	10:24:41.6995	DTE	0	4	22ac0220
16	10:24:51.7011	DTE	0	5	8008ab0008
17	10:25:01.7043	DTE	0	5	20c22a0002
18	10:25:11.7059	DTE	0	5	88b00a8000
19	10:25:21.7083	DTE	0	4	22ac0220
20	10:25:31.7090	DTE	0	5	8008ab0008
21	10:25:41.7122	DTE	0	5	20c22a0002
22	10:25:51.7138	DTE	0	5	88b00a8000
23	10:26:01.7162	DTE	0	4	22ac0220
24	10:26:11.7178	DTE	0	5	8008ab0008
25	10:26:21.7210	DTE	0	5	20c22a0002
26	10:26:31.7217	DTE	0	5	88b00a8000
27	10:26:41.7241	DTE	0	4	22ac0220
28	10:26:51.7257	DTE	0	5	8008ab0008
29	10:27:01.7294	DTE	0	5	20c22a0002
30	10:27:11.7303	DTE	0	5	88b00a8000
31	10:27:21.7318	DTE	0	4	22ac0220
32	10:27:31.7342	DTE	0	5	8008ab0008
33	10:27:41.7374	DTE	0	5	20c22a0002
34	10:27:51.7384	DTE	0	5	88b00a8000
35	10:28:01.7408	DTE	0	4	22ac0220
36	10:28:11.7424	DTE	0	5	8008ab0008
37	10:28:21.7455	DTE	0	5	20c22a0002
38	10:28:31.7471	DTE	0	5	88b00a8000
39	10:28:41.7487	DTE	0	4	22ac0220
40	10:28:51.7505	DTE	0	5	8008ab0008
41	10:29:01.7537	DTE	0	5	20c22a0002
42	10:29:11.7553	DTE	0	5	88b00a8000
43	10:29:21.7567	DTE	0	4	22ac0220
44	10:29:31.7591	DTE	0	5	8008ab0008
45	10:29:41.7623	DTE	0	5	20c22a0002
46	10:29:51.7634	DTE	0	5	88b00a8000
47	10:30:01.7658	DTE	0	4	22ac0220
48	10:30:11.7674	DTE	0	5	8008ab0008
49	10:30:21.6996	DTE	0	5	20c22a0002
50	10:30:31.7020	DTE	0	5	88b00a8000

Figure 9-3 ISH PDU Convergence for Deflate

9.2 Convergence Analysis

A considerable amount of data was collected during each exercise and this is summarised in Figure 9-4 through to Figure 9-14. Each shows the results of the same exercise run on each of the two datasets.

In the case of LREF, convergence is very rapid. In the experimental data, only a single LREF was used. This was established 840 seconds into the trial, and hence an early convergence should be expected. The LREF overhead is incurred every time a new LREF is created and such an LREF is needed for each data path between systems that is required. In the ADS Trial only a single Airborne and Ground System were in communication and hence only a single LREF was established. In operational use the number of LREFs will not however increase with the number of applications but systems. Hence, in practice, the observed behaviour is not likely to be that different from this trial.

LZW appears to converge at an acceptable rate and achieves its optimal performance about 1000 seconds into the trial (after about 4KB of data is exchanged), of which about 1.2KB is CLNP. The effect of the initial surplus of ISH PDUs is very obvious from these figures, showing an initial high compression ratio, dropping down to a more realistic level once application data is exchanged.

Deflate has similar convergence to LZW. The effect of the ISH PDUs is less pronounced and, on the unfiltered data, the general improvement in compression can be readily seen.

As expected, a very flat result is achieved from ACA.

In the case of the combined algorithms, the results are generally additive from the individual performance of the algorithms, showing their complementary nature. The exception to this was the combinations of LREF/ACA/LZW and LREF/ACA/Deflate. In these cases, the algorithms appear to interfere with each other, resulting in lower compression any combination of two of those algorithms tested.

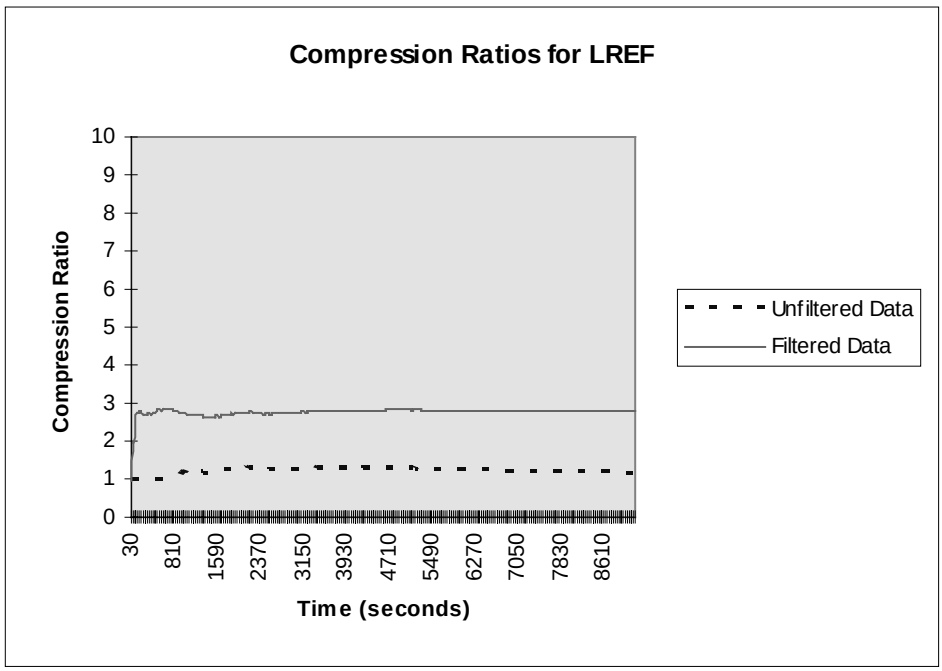

Figure 9-4

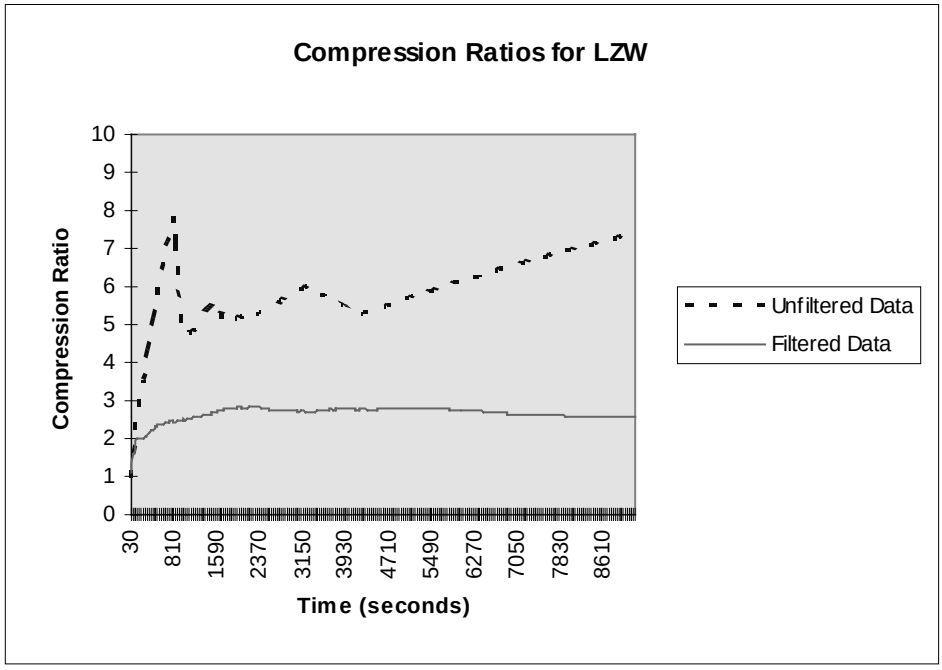

Figure 9-5

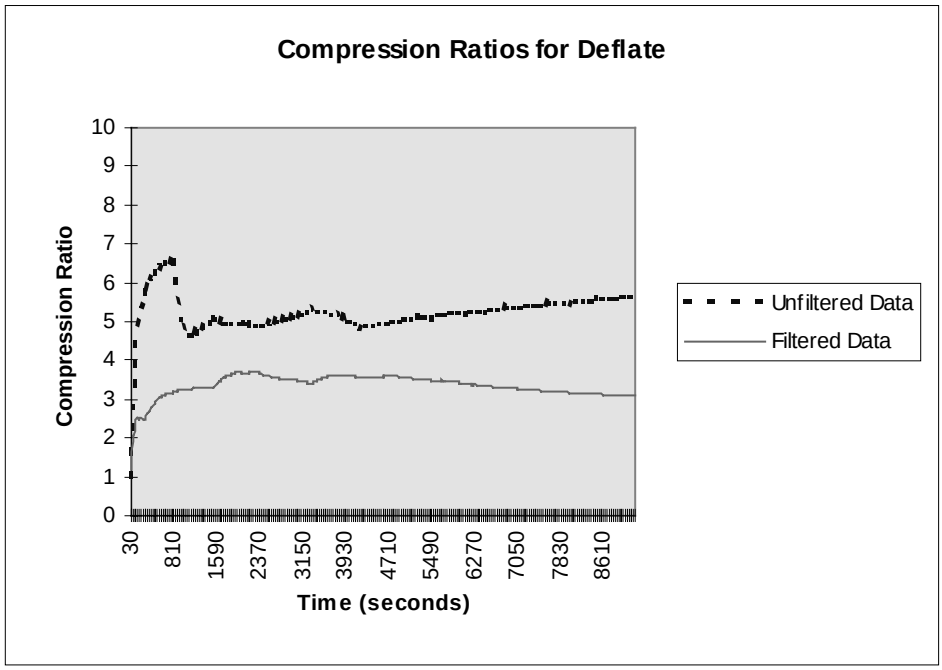

Figure 9-6

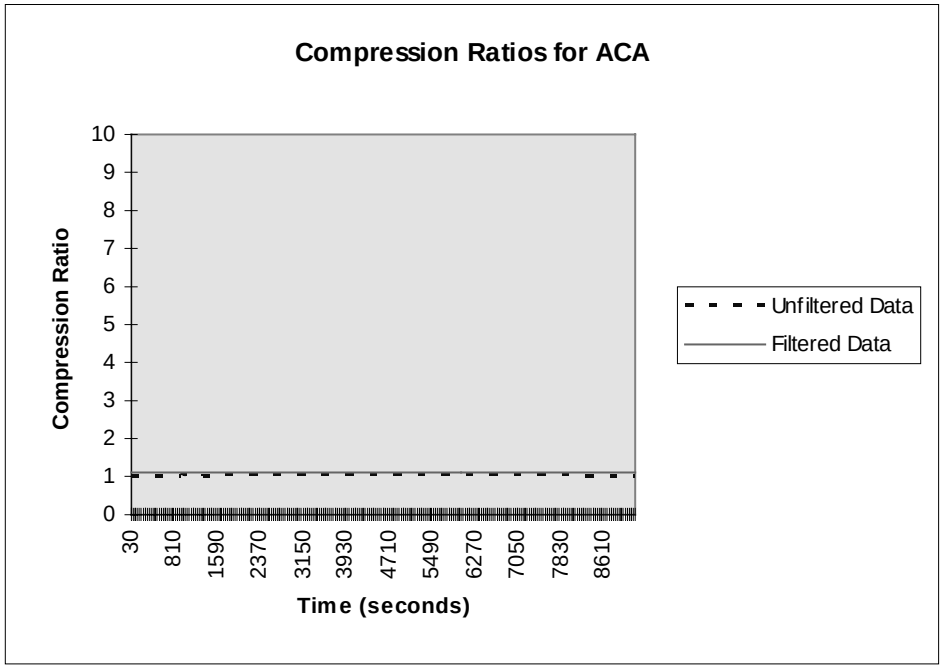

Figure 9-7

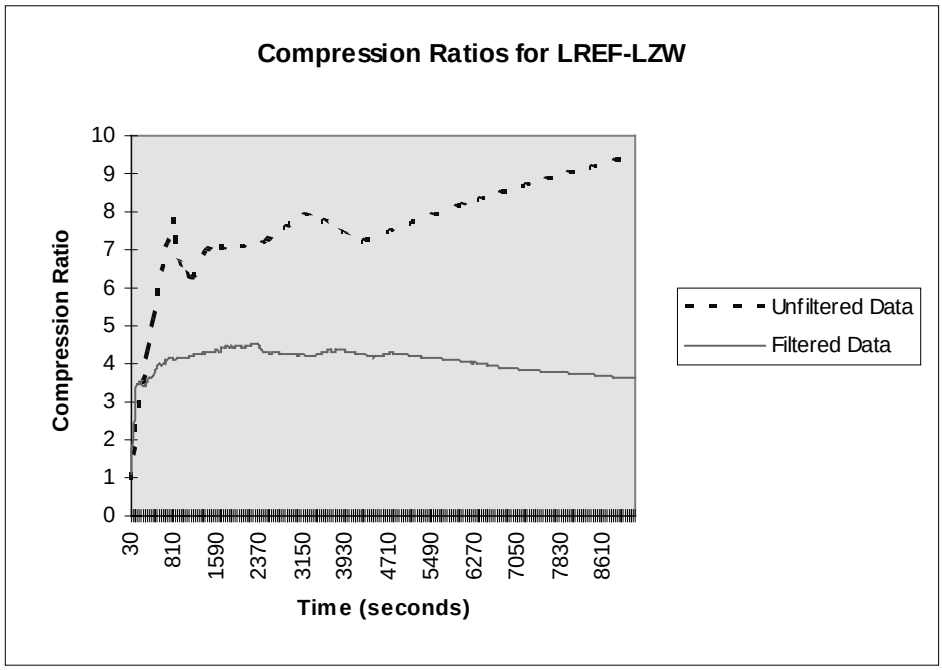

Figure 9-8

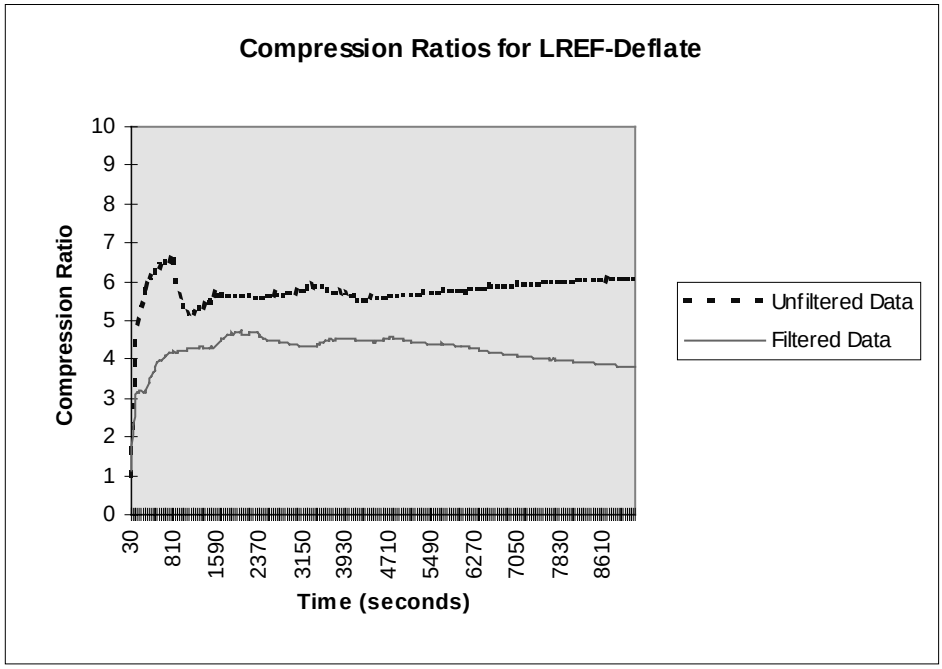

Figure 9-9

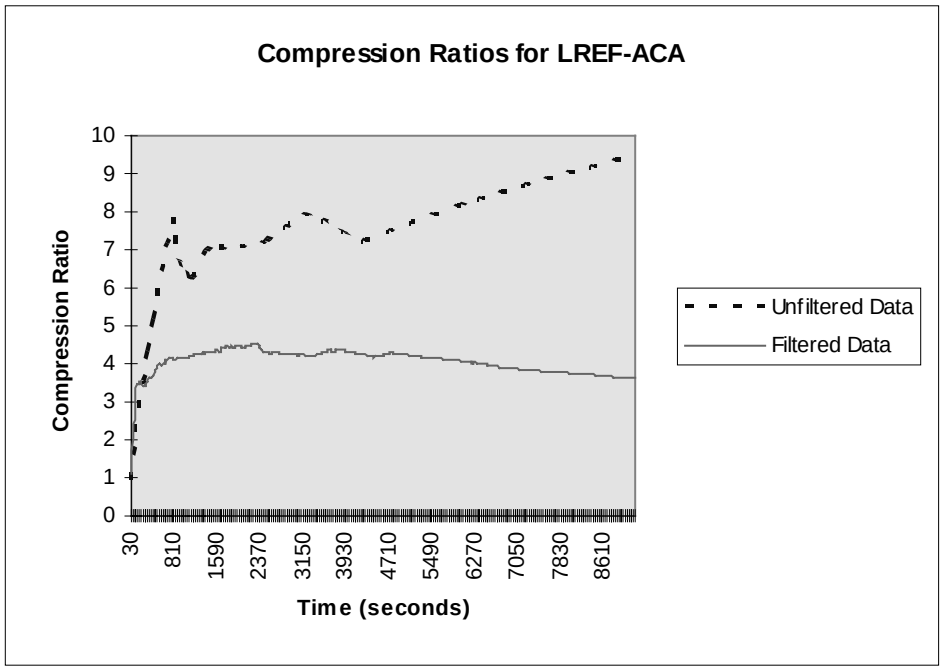

Figure 9-10

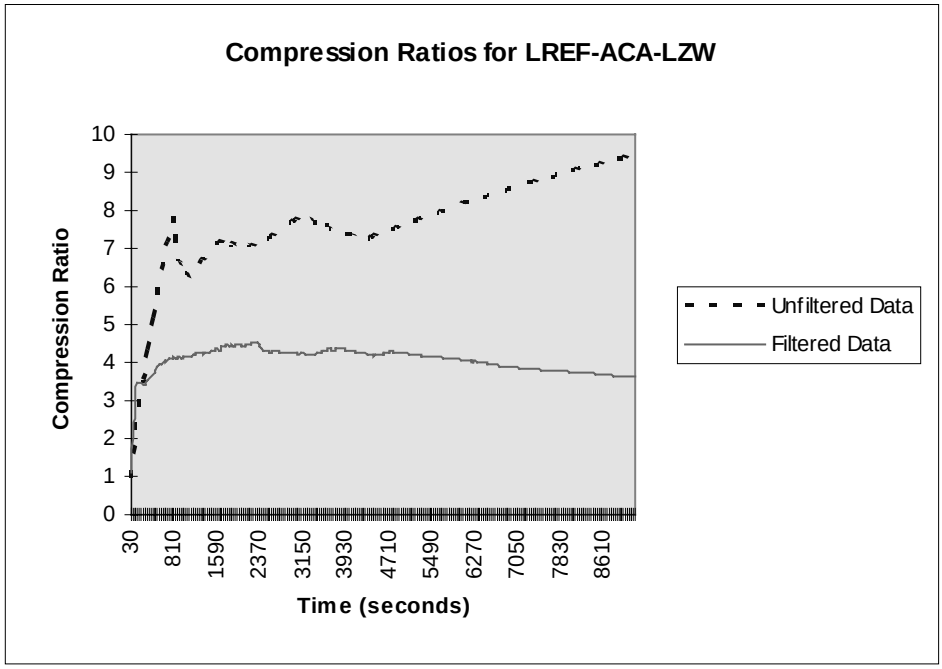

Figure 9-11

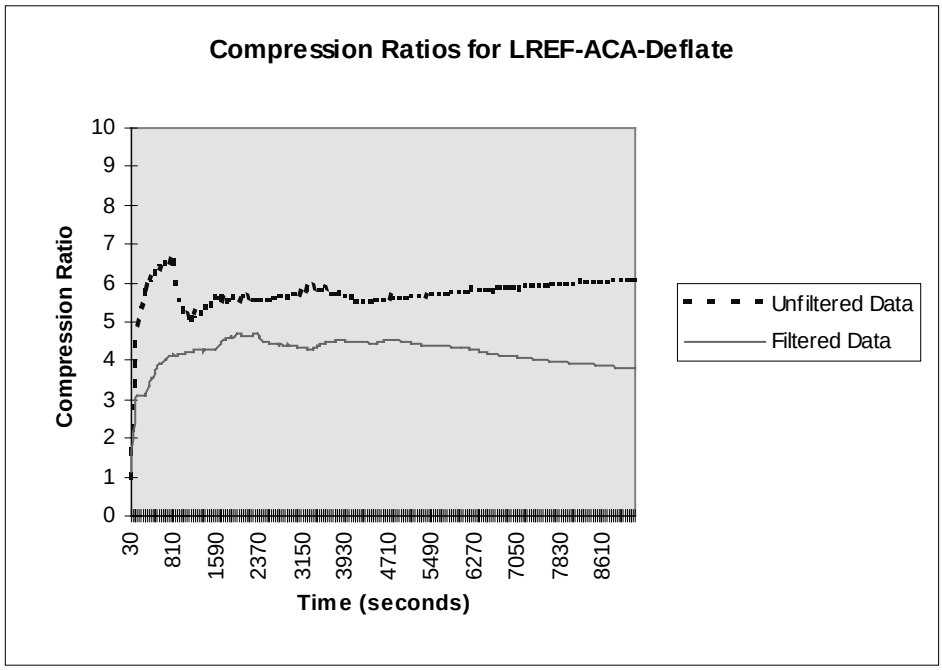

Figure 9-12

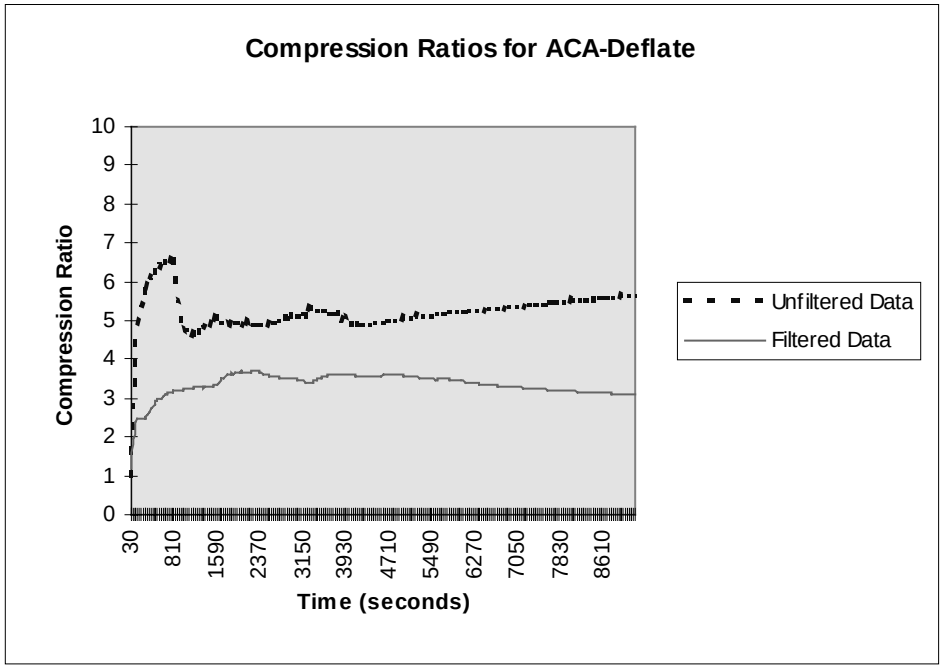

Figure 9-13

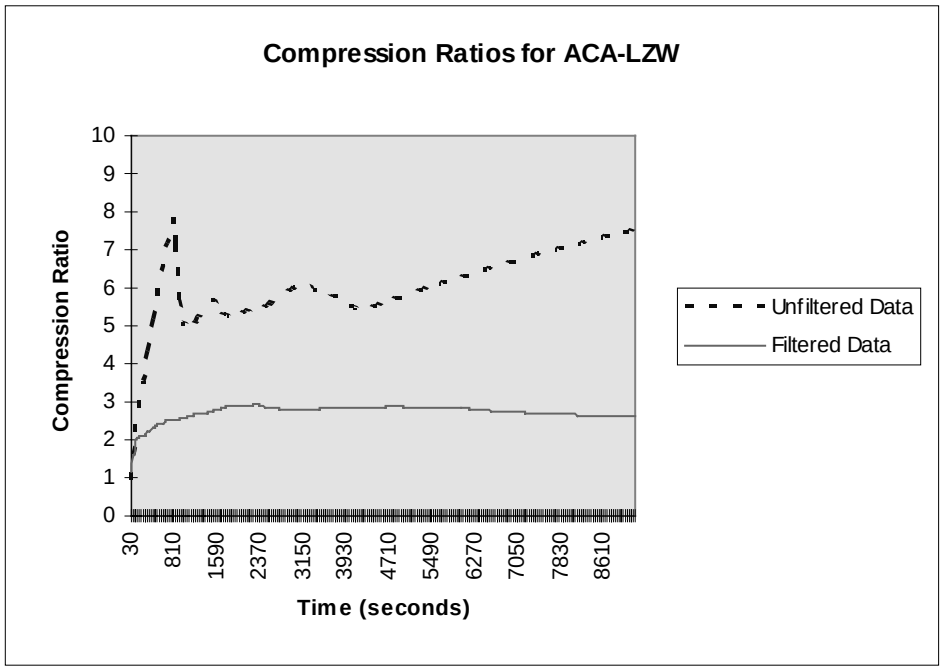

Figure 9-14

9.3 Conclusion

The conclusions from the analysis are very clear. The LREF, LZW and Deflate algorithms are all useful and LREF is complementary to either LZW or Deflate. There is a clear benefit in using LREF in combination with one of the others, which will translate into reduced costs for the users and better use of limited bandwidth.

On the other hand, the ACA appears to offer only a marginal benefit on its own and no benefit when used in combination with the other algorithms.

In the expected operational ATN with IDRP used air/ground, the combination of LREF and Deflate should give the best results with a combined compression ratio of 3.82 achievable. The total data transferred can thus be reduced to almost 25% of the original data size. On its own, Deflate gives very good results with a compression ratio of 3.08 achieved, compared with LREF's 2.78, and LZW's 2.57. If we were only able to use one compression algorithm then clearly, Deflate should be preferred.

Deflate is therefore a technically better solution that LZW and is free of patent/IPR problems. It thus appears to be a better all round data compression algorithm.

LREF is already specified in the SARPs, and no problems were found with the specification during this work.

If LZW were to be specified for ATN use, then it should be noted that the SARPs, for reasons outline in 8.1.2, do not contain an adequate specification for an LZW like algorithm and include only a reference to ITU-T V.42bis. It is believed that V.42bis cannot be used without adaptation for the ATN and that such adaptation must be standardised. Furthermore, V.42bis is probably not the most appropriate LZW derivation for use in the ATN. Work would therefore be needed on this subject (see section 10.1).

If Deflate were to be specified for ATN use, then the IETF RFC 1951 specification appears adequate, together with the standard 'C' implementation, to either be referenced from the SARPs, or incorporated within them. The ATN specific requirements to use Deflate appear straightforward and are discussed in Appendix A. During this work, some areas of future work with Deflate were noticed, and these are discussed in 10.2.

AVO_454 and AVO_455 can be considered as having been fully achieved by these exercises. The results give both detailed information on the operation of the specified algorithms and the advantages of their use for the service user.

10. Future Work

10.1 On LZW

The adaptation of the LZW algorithm used was developed rapidly for this analysis work and cannot therefore be regarded as necessarily the most suitable for the ATN, or as having been fully validated. However, this analysis work has shown the benefits that can be achieved using an LZW derivative.

If an LZW derived algorithm is to be used in the ATN, work would thus need to be done on the detailed analysis of the algorithm and its full validation prior to its incorporation into the SARPs. This would need to be part of the future work program of WG2. Furthermore, it should again be noted that part of the LZW algorithm is subject to a patent held be Unisys. While the validity of patents on algorithms is an area of controversy, the validity of the patent will need to be investigated, together with any royalties that might have to be paid for its use.

10.2 On Deflate

Deflate can give improved results if dictionaries of likely to occur phrases are pre-loaded into both the compressor and decompressor. However, no work was done on this subject in this validation exercise. This a valid area for future work as such dictionaries have the potential for additional reductions in the cost of air/ground data link usage.

Certainly, this possibility should be taken into account in any use of Deflate in the ATN and, in particular, for the use of a given dictionary to be negotiated during the connection establishment phase. This is important to enable new dictionaries to be phased in when new ATC Applications are introduced that change the set of likely to occur phrases.

It should also be considered as to whether an additional error checksum should be included with each compressed packet. Such a checksum would be taken on the original, uncompressed data, and would be used to verify the proper operation of the decompressor. This is potentially important, as an otherwise undetected error in the compressed data not only affects the packet in which it occurs, but it also affects subsequent packets due to the fact that the compression algorithm retains state information between each packet. Such a checksum would be used to detect this class of error and force a reset of the compression algorithm in order to re-synchronise compressor and de-compressor. It should be noted that a similar issue exists with the use of LZW.

IETF RFC 1950 specifies a 32 bit “adler 32” checksum for use with the ZLIB compression format. This is an improved version of the 16-bit Fletcher Algorithm used in the ISO 8073 COTP. However, this would impose an extra four octets for every packet if used in the ATN and, given the small packet size would not seem to be justifiable. On the other hand, the Fletcher algorithm would impose only a 2 octet overhead and would appear to provide more than adequate protection.

A two octet overhead on each compressed packet would, for example, have reduced the achieved compression ratio for LREF in combination with Deflate, from 3.82 to 3.48 on the filtered data stream.

11. Recommendations

As a result of this work, the following recommendations are made:

1. That the ICAO Address Compression Algorithm (ACA) is removed from the ATN SARPs on the grounds that it offers no user benefit when the other compression algorithms are used.

2. That the use of V.42bis is removed from the ATN SARPs, to be replaced by the use of the Deflate algorithm.

3. The use of the Deflate algorithm includes a per packet checksum on the uncompressed data.

4. That the LREF algorithm is considered as mandatory for use and that the implementation and use of the Deflate algorithm be strongly recommended.

5. Future work on the development of dictionaries of “likely to occur phrases” for use with Deflate is considered for the WG2 future work programme, and that provision is made for the negotiation of such dictionaries in the Mobile SNDCF.

Appendix A - Inclusion of Deflate in the ATN Internet SARPs

If Deflate is to be specified by the ATN Internet SARPs, then two main changes are required:

1. To the Mobile SNDCF Call Setup procedures specified in 5.7.6 of the ATN Internet SARPs, in order to include negotiation of the use of the Deflate algorithm, and an optional phrase dictionary.

2. To include the specified use of the Deflate Algorithm and a per packet checksum.

Changes Required to 5.7.6 of the ATN Internet SARPs

1. In 5.7.6.1, note 2, Deflate needs to be added to the list of Data Compression Procedures.

2. In Figure 5-7.2, Table 5-7.2 and 5.7.6.2.1.5, a “bit” needs to be assigned for proposing the use of Deflate. It is suggested that bit 8 is assigned for this purpose.

3. Further, in Figure 5-7.2 and 5.7.6.2.1.5, a procedure needs to be reserve for the negotiation of compression dictionaries. It is proposed that a 2 octet identifier be reserved for identifying dictionaries in future amendments to the SARP, and that the capability is added to include a variable length list of such identifiers in the call user data, with an empty list implying no dictionaries proposed. When Fast Select is not available, at most one dictionary could be proposed.

4. In 5.7.6.2.2.2, call acceptance procedures will need to be added to accept one or none of the proposed dictionaries.

5. In 5.7.6.2.2.3, call rejection procedures will need to be added to reject the call when the proposed dictionary is not known and Fast Select is not available.

6. In 5.7.6.2.2.4 and Figure 5.7-3, the call accept format will need to be updated to respond with the identify of the accepted dictionary, if any.

7. In Table 5.7-3 an additional diagnostic needs to be defined for call rejection due to the proposed dictionary not being supported.

8. At the same level in the outline as 5.7.6.3 “Local Reference Compression Procedures”, the procedures for compressing each packet using Deflate need to be specified. The key requirements are:

    - Specification of the IETF RFC 1951 Deflate format together with a suitable checksum on the original uncompressed data;

    - A requirement that the compressor is flushed after the compression of each packet i.e. so that each compressed packet may be decompressed without requiring the receipt of the next packet;

    - A requirement that the virtual circuit is reset when a checksum failure is detected, and that the compression algorithm is reset whenever the virtual circuit is reset.