Ten Times more information in your Real-Time TM

David Evans[1] and Ugo Moschini[2]
ESA/ESOC, Darmstadt, Germany

One way of estimating the information content of a message is to compress it. Previous papers have shown that the information content of housekeeping telemetry packets is quite low as when they are stored in files they compress extremely well. Compressing packets and sending more of them in the same bandwidth would improve the information content of the data stream. This would bring benefits in terms of better reaction times and better spacecraft observability as well as reducing the upfront engineering effort. So far all methods proposed work on housekeeping data that has been stored on-board for transmission to the ground at some later point. This paper presents a method for compressing packets in real-time i.e. each individual packet is compressed into an equivalent smaller packet as soon as it is generated. This opens up the possibility of increasing the information content of both the playback and real-time telemetry streams with a unique process at generation time. Note that compression algorithms like ZIP cannot do this. They require a critical number of packets to be stored before the compression algorithm can be effective. This paper describes the tests run at ESA/ESOC using real spacecraft data to prove the concept. Compression ratios of ten are sometimes achieved i.e. an increase in information content of over 1000%. Only a few microseconds are needed to compress a typical packet making it suitable for real-time. The method has been tested with data from spacecraft in different environments (including safe mode) and they have shown that the compression performance is very stable. ESA has recently filed an international patent covering the method.

I. Introduction

Whereas science data is routinely compressed, housekeeping telemetry is not. This is despite the fact that many mission profiles are dominated by housekeeping telemetry (e.g. technology demonstrators) and every mission has phases dominated by it (e.g. LEOP, recovery phases). In previous work conducted by the Future Studies section of ESA/ESOC[1,2,3] we argued that there would be considerable advantages in compressing housekeeping data and no increase in risk, provided it was performed correctly. The compression algorithm must have a stable performance, be lossless and require little on-board processing power.

We showed that the CCSDS recommended compression algorithm for space (RICE[4]) could not compress housekeeping telemetry effectively and proposed an alternative algorithm. While implementing this algorithm we encountered two fundamental problems and solving those problems required a major rethink of our approach. This new approach resulted in the creation of a new algorithm that could not only compress stored housekeeping telemetry data but also data that was transmitted to the ground in real-time. This paper describes the problems, the new algorithm and the many advantages of the new approach.

In sections II-III we recap the previous algorithm's performance, general advantages and elaborate on how compression can be used to increase information content. In section IV we describe the implementation problems with that algorithm. In sections V-VII the new algorithm and its test results are presented. Sections VIII-IX deals with the robustness of the technique. Finally we present the results of tests obtained when running the new algorithm on real space hardware.

[1] Mission Concept Engineer, Future Studies Section, ESA/ESOC
[2] Young Graduate Trainee, Future Studies Section, ESA/ESOC

II. Previous Work

In our three previous papers,[1,2,3] we described an algorithm that we called bit transposition-RLE. It was extremely effective at compressing stored housekeeping telemetry for a wide variety of missions. The following experiment was set-up. A week's worth of housekeeping data was retrieved from the ESA mission archives for different ESA missions. These were then processed to extract the CCSDS source packets, thereby effectively reconstructing the original onboard packet store. These packet stores were then compressed using the bit transposition-RLE algorithm and the size of the compressed file compared to the original packet store. The results are given in Table 1. Please note that all compression ratios in this paper are given in percentage terms (size of compressed file compared to original packet store), therefore the smaller the percentage the better.

MISSION NAME	MISSION TYPE	COMPRESSION (% of original)
Columbus	Human Spaceflight	5.75%
Rosetta	Interplanetary	14.06%
Venus Express	Interplanetary	18.23%
Proba-1	Technology Demo	25.93%
Herschel	Astronomy	28.00%
Goce	Earth Observation	38.20%

Table 1. Compression performance comparison by mission for bit transposition-RLE.

The results proved that there is a considerable amount of information redundancy in housekeeping telemetry that could be removed using this simple compression algorithm.

III. Housekeeping compression advantages

We also argued that implementing housekeeping telemetry compression would reap direct cost savings by enabling shorter or less ground station passes and that there were many less obvious advantages:

1) Allowing the use of smaller, cheaper ground stations (less data so improved link budgets)
2) Saving on-board power and energy (less transmission time)
3) Relaxing operational time constraints (shorter dumps so more choice on when they are done)
4) Improving interactivity with the spacecraft (shorter reaction times possible)
5) Making packet design easier (less constraints on parameter and sampling rate choice)
6) Improving mission safety (can send critical packets multiple times)

In this paper we would like to highlight a further advantage that has been cited by multiple operators: using compression to the increase the information content in housekeeping telemetry. Here the idea would be to maintain the same level of bandwidth usage for housekeeping but to use compression to allow the remote system to sample more parameters at higher frequencies. If compression ratios of ten are achieved if would correspond to an increase in information content of over 1000%. This is an information increase, not a telemetry rate increase. Parameters that compress well can be included in the housekeeping packets and sampled often with only a small impact in bandwidth usage. Therefore the actual increase in telemetry rate could be far higher than 1000%.

Without compression it is necessary to carefully select all the parameters and their respective sampling rate in the housekeeping packets. The aim is to get a balanced compromise between bandwidth use and information content. If bandwidth is really tight it could involve removing parameters from housekeeping and designing asynchronous events to flag operationally significant changes in them. Compression fundamentally changes this trade by effectively removing all "compressible" parameters, which make up the vast majority. Hence one can simply select as many of these parameters at whatever sampling rate one wants in the housekeeping, with little bandwidth usage impact. There is no need to guess if these parameter histories might be needed one day and no need to design asynchronous triggers and events for these parameters.

This results in richer, finer information on the ground to analyze. Therefore it removes the need to guess or extrapolate when trying to reconstruct on-board events. It reduces the chance of missing important information due to low sampling rates (e.g. short abnormal behavior like spikes, transients, high frequency switching). It gives the ground more chance of discovering correlations between parameter behavior and important events or trends that simply is not possible using snapshots. These are all operationally important advantages.

IV. Two problems

Although the bit transposition-RLE algorithm generated considerable interest two fundamental problems emerged. The first one was raised by operators who pointed out that housekeeping telemetry is used as input for two processes: real-time control and off-line analysis. Since the packet structures for telemetry destined for the off-line analysis process and the real-time control process are interchangeable then real-time considerations would always negate the theoretical advantages gained in the off-line analysis process. For example we argued that significant savings could be made in the operations preparation phase during packet design as the need for careful selection of parameters and sampling rates could be eliminated for those parameters that compress well. However if the same packets are used for real-time transmission then one still has to be very careful when designing the packet.

The second objection came from the on-board software area. The bit transposition-RLE algorithm requires that the different packet types are processed together, hence all packets of a particular packet type must be read from the packet store before moving onto the next packet type. They pointed out that presently packet stores are flat i.e. packets are stored sequentially as soon they are generated with no indexing or hierarchy. When the time comes to dump the packet store a pointer is set to the memory address of the last packet read and then all the packets are simply read back from this point. Any attempt to read only certain packet types from this flat store would require considerable CPU usage. While one can envisage using different packet stores for each packet type this would raise operational issues such as trying to predetermine how big each one should be in order to avoid overwrite. This would introduce unwanted complication into the mission operations concept.

We realized that the only solution to the operator's problem would need a compression algorithm that would intercept a packet when it was generated, convert it very rapidly into a smaller packet and then transmit this in the real-time stream instead of the original. If this could be done then it would also solve the on-board software problem since these smaller packets could also be stored in a flat packet store and retrieved using the present simple mechanisms.

V. The new approach

At first there seemed to be no way to do this. Most compression algorithms require a certain critical mass of data to be available before they can determine where information is redundant and then remove it. An algorithm that could compress data in the real-time telemetry stream would only have a single packet to work with and therefore not have that critical mass.

We began by reexamining why the bit transposition-RLE algorithm worked so well with housekeeping packets. These packets use fixed bit position mapping to determine where parameters start and finish which means that they must allocate a fixed number of bits for each parameter. This is efficient in the sense that no parameter labeling is required but it also means that each parameter must be allocated a sufficiently large enough bit field to cover all its entire possible dynamic range. However it is very rare that a parameter uses its entire dynamic range and so while most parameters change value from one housekeeping packet to another the majority of their bits do not.

The bit transposition-RLE algorithm exploits this property by reading the same bit position from multiple packets in sequence. We realized this property could be exploited in another way; by performing a bit XOR of the newly generated packet with the last packet generated of that type. This would produce a new packet that contained mainly zeros at the bit level (for those bits that had not changed state). This could be then be compressed by run-length encoding those bits. By adding a new header to this compressed data and a trailer to make it an integer number of bytes, we can create a new smaller packet from the original longer one.

Inspection of the XORed packets revealed that some bits were changing state much more often than remaining in their present state. Hence adding a simple inversion of the XOR result for that bit would produce a zero more often and improve the compression. This could be reversed in the decompression algorithm. A similar approach was taken for those bits that tended to change state at the same time. Here we found that XORing the two results produced a zero more often and this could also be reversed in the decompression algorithm. This was called horizontal correlation. Both inversion and horizontal correlation are applied to all the subsequent algorithms in this paper.

We called the technique the "basic" algorithm and used it to compress a week's worth of stored housekeeping data for several ESA missions. The results are shown in Figure 2. This shows that this basic algorithm was indeed capable of producing significant compression but was less performant than bit transposition-RLE results presented in Table 1.

VI. Improvements on the basic algorithm

The bit transposition-RLE results proved that it was possible to remove more information redundancy from the data than the basic algorithm and so we began to search for methods to improve it. We considered how a-priori information could be exploited. Our first idea was to calculate the probability that each bit would change state from one packet to another based on historical data. The ground could then pre-load an order in which to read the bits in the XORed packet based on those values. This would have the effect of grouping bits with a high probability of being zero after the XOR and this would make the RLE compression more effective. A similar idea had already been proposed for compressing and storing housekeeping telemetry on the ground by Staudinger et al[5], therefore we called this algorithm "Staudinger". The results are shown in Figure 2. One can see that it had a very good performance, beating the basic algorithm (and even bit transposition-RLE) in all considered cases.

However our overall aim was to produce a system that could compress the real-time telemetry stream therefore it was important to test how much time it took to compress an average packet. We ran speed trials using a notebook equipped with an Intel Core i7 processor to gain relative speed information between the techniques. The results are shown in Figure 3 and highlight that the Staudinger algorithm was taking over twice as long to compress a packet on average compared to the basic algorithm. This was not good news for a real-time compression system.

VII. POCKET

As well as being slower than the basic algorithm, Staudinger had other disadvantages when applied to our problem. It has a complicated ground-space interface because it requires the loading of a specific order table for each bit in every packet type. Also Staudinger points out in his paper that is not stable for "real data". Staudinger proposed a recalculation of the order tables every hour to compensate and this would not be practical for us.

We therefore started looking for a solution that would have a better performance than the basic algorithm but would be faster, stable and have a simpler ground-space interface. The algorithm we created was christened POCKET (Probability Of Change masKEd Transformation). The idea behind POCKET is to exploit probability information available in historical data to produce a simple bit mask packet rather than a bit order table. The ground then sends this mask packet to the spacecraft and the on-board algorithm uses this in a series of bit-wise operations to compress the newly generated packet with very little calculation involved. This gives the algorithm a very simple ground-space interface and makes the algorithm very fast.

The POCKET algorithm has a ground element and an on-board element which are described in the steps below.

The ground element:

1) Use historical data to determine the predictability of each bit in a data packet based on the state of the bit in the last packet generated with inversion and/or horizontal correlation applied.
2) For each bit in the packet decide whether its state is predictable i.e. its state can be predicted with a better probability than a certain threshold.
3) Use this information to create a mask packet with bits in a one state for all predictable bit positions and in a zero state for the rest.
4) Upload the mask packet to the spacecraft along with inversion and horizontal correlation instructions.

The on-board element is described below and in Figure 1:

5) When a new packet is generated perform an XOR with the previous packet of that type.
6) Perform inversion/horizontal correlation instructions as necessary. The resulting packet is called the processed packet.
7) Perform a bitwise AND between the processed packet and the ground loaded mask packet. This will change all the unpredictable bits in the packet to a zero state.
8) Perform bit run length encoding on the resulting packet and write the results to an output buffer.
9) Extract the unpredictable bits from the processed packet (information in the mask packet) and append them to the output buffer in read order.
10) Add a header and a trailer to the output buffer to create the compressed packet.
11) Transmit or store the compressed packet as required.

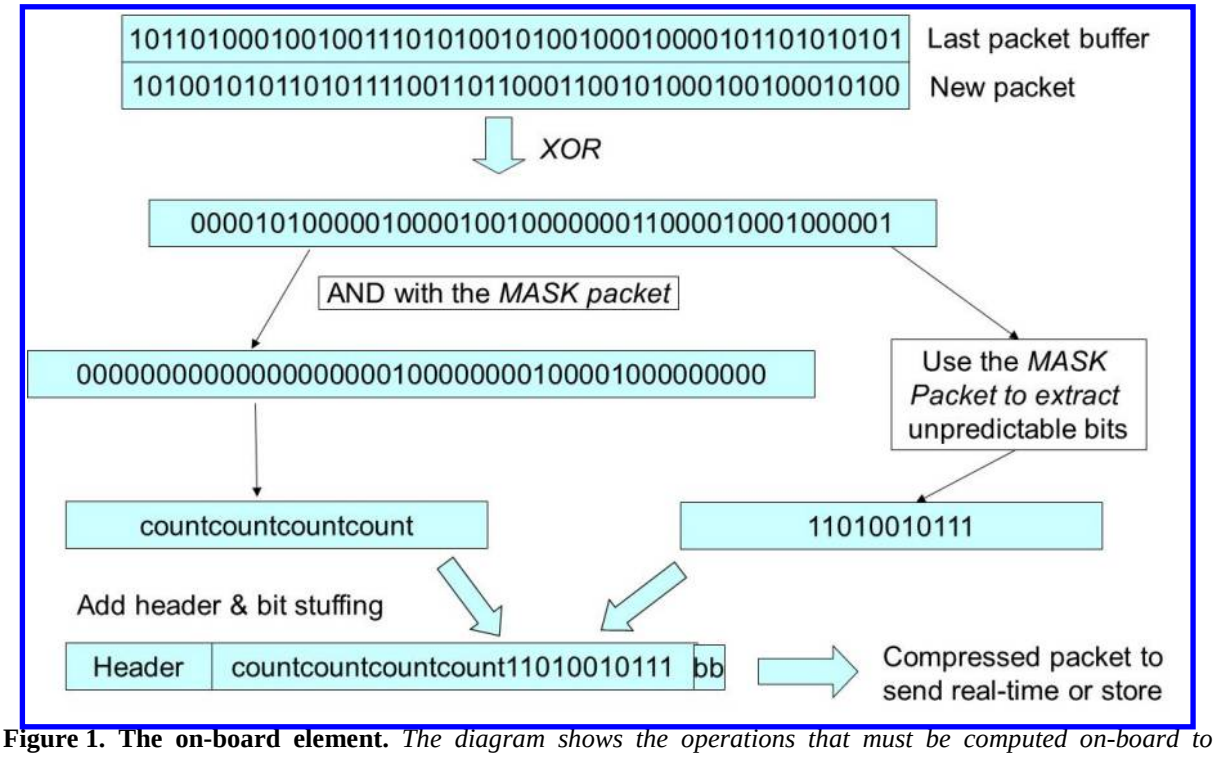

Figure 1. The on-board element. *The diagram shows the operations that must be computed on-board to compress a single packet before it is streamed or stored. The "XOR" arrow implies points 5 and 6 of the previous numbered list, involving the initial XORing, inversion and horizontal correlation to produce the processed packet.*

The compression results of the POCKET algorithm are presented in Figure 2. One can see that it performs almost as good as the Staudinger algorithm even though it is much simpler and easier to configure. Looking at Figure 3 one can see that it is significantly faster than the Staudinger algorithm and even faster than the basic algorithm.

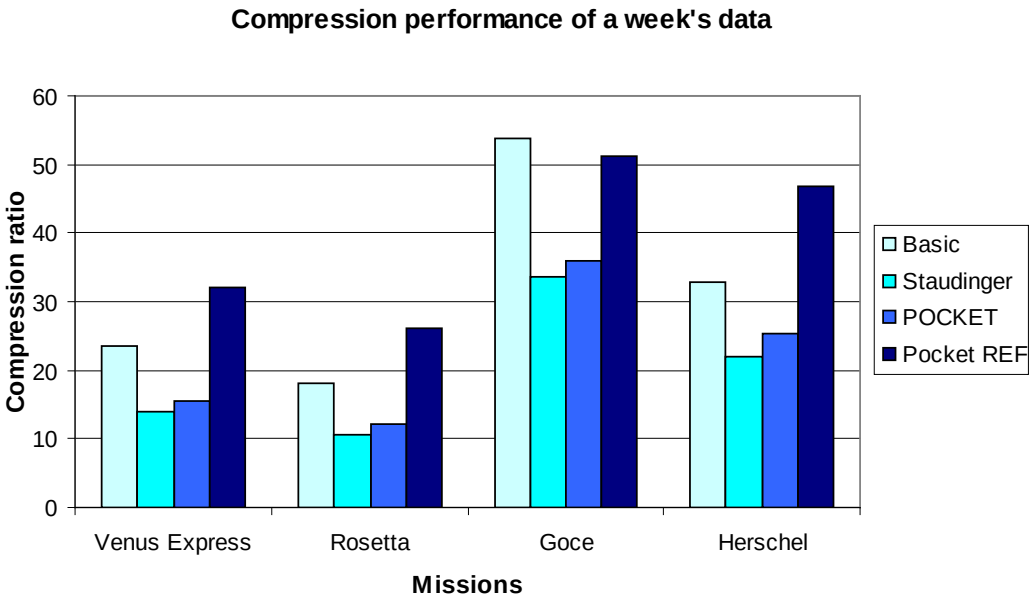

Figure 2. Compression performance comparison by mission for the four methods illustrated in sections V, VI, VII and VIII. *The figure shows the compression ratio percentages by mission, as size of the compressed packets compared to the original packet stores. For every mission, a week's worth of stored housekeeping data has been considered.*

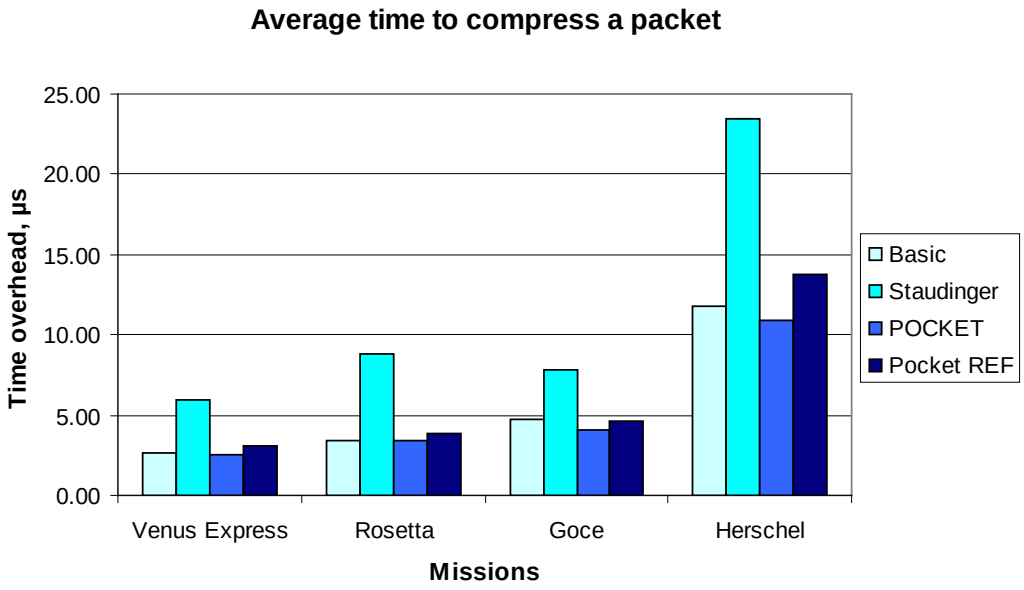

Figure 3. Speed performance comparison by mission for the four methods illustrated in sections V, VI, VII and VIII. *The figure shows the average time, expressed in microseconds, needed to compress a packet of each mission on an Intel i7 notebook.*

VIII. POCKET Robustness

In earlier papers,[1,2,3] we argued that compression does not increase the risk of information loss if the information in a telemetry frame can be uncompressed without needing any information contained in other telemetry frames. The use of an XOR with the previous packet means that the compressed packet is effectively a delta change packet. One must work forwards from a reference packet in which the absolute state was known to calculate the present absolute state. It is possible to work backwards from a reference packet in the future as well. If a packet is lost then all information will be lost (or delayed) until the next reference packet is transmitted. Hence reference packets must be sent at intervals. The frequency with which one sends these reference packets depends on the probability that a packet will be lost and how much risk one is prepared to take.

One could simply use uncompressed packets as reference packets but there is also a simple way of configuring POCKET to produce reference packets. POCKET REFERENCE packets are not based on an XOR with a previous packet but an XOR with a pre-loaded ground packet which predicts the state of each bit. A corresponding mask packet is used which is based on the probability of the prediction being correct. The results are given in Figure 2 and show that POCKET can produce reference packets with a good level of compression. Also Figure 3 shows that the POCKET REFERENCE is much faster than Staudinger.

IX. Stability results

In order to test the robustness of the POCKET algorithm we set up the following experiment to determine its performance over time. A week's worth of housekeeping data of Venus Express was split in chunks corresponding to the seven days of the week. The first chunk only was used as historical data to produce the mask packet and the inversion/correlation instructions, which were then applied to compress the packets of the remaining six days. The results are shown in the Figure 4. It shows that POCKET performance remains remarkably stable over time only decreasing from 15% to 16% compression over one week without updating the mask packet. On the other hand when a similar experiment was performed with POCKET in reference mode then significant degradation was seen (21% to 36%). As reference packets are likely to only be a small part of the data stream the impact would be limited.

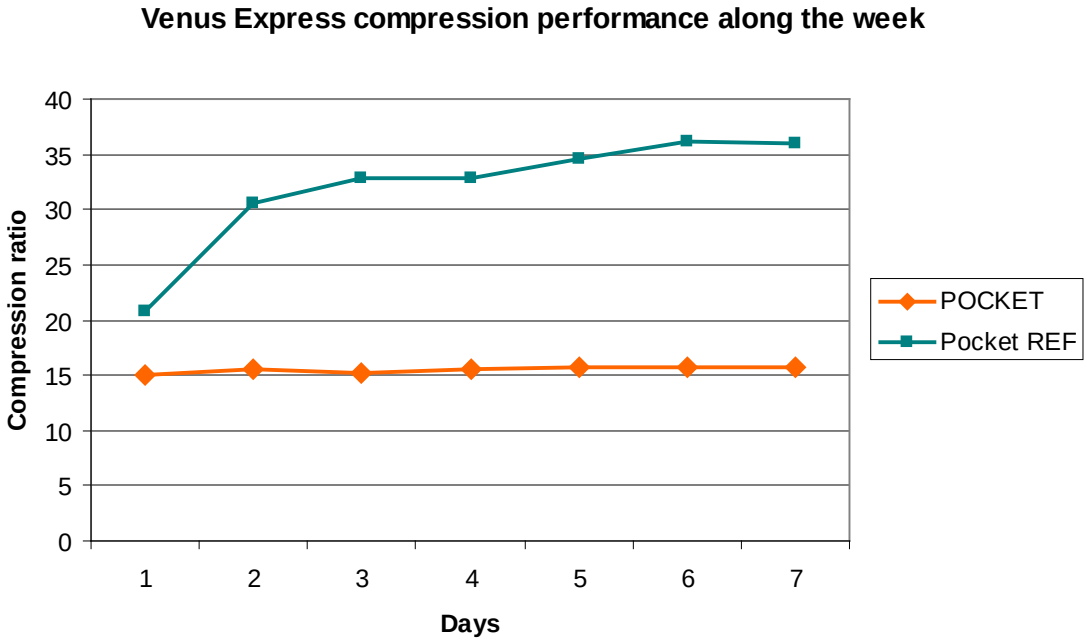

Figure 4. Stability comparison between POCKET and POCKET REFERENCE. *The figure shows that the compression ratios of POCKET remain stable day by day even if the mask packet was created seven days before. POCKET REFERENCE shows a considerable degradation: it is more sensitive to the evolution of the telemetry.*

We also wanted to test if the algorithm performance was stable when a major change occurred. The following experiment was performed. Mask packets and inversion/correlation instructions were created on historical telemetry, stored during a day when Venus Express behaved nominally. That configuration was then used to compress telemetry packets produced on a day when the satellite entered safe-mode. The results are given in Figure 5 and they show that POCKET performance remains remarkably stable even after safe mode entry, only decreasing from 16% to 19%. On the other hand when POCKET was run in reference mode then a significant degradation was seen (21% to 52%). This shows that when a spacecraft enters safe mode many status values will change but the dynamic range of most parameters remains similar.

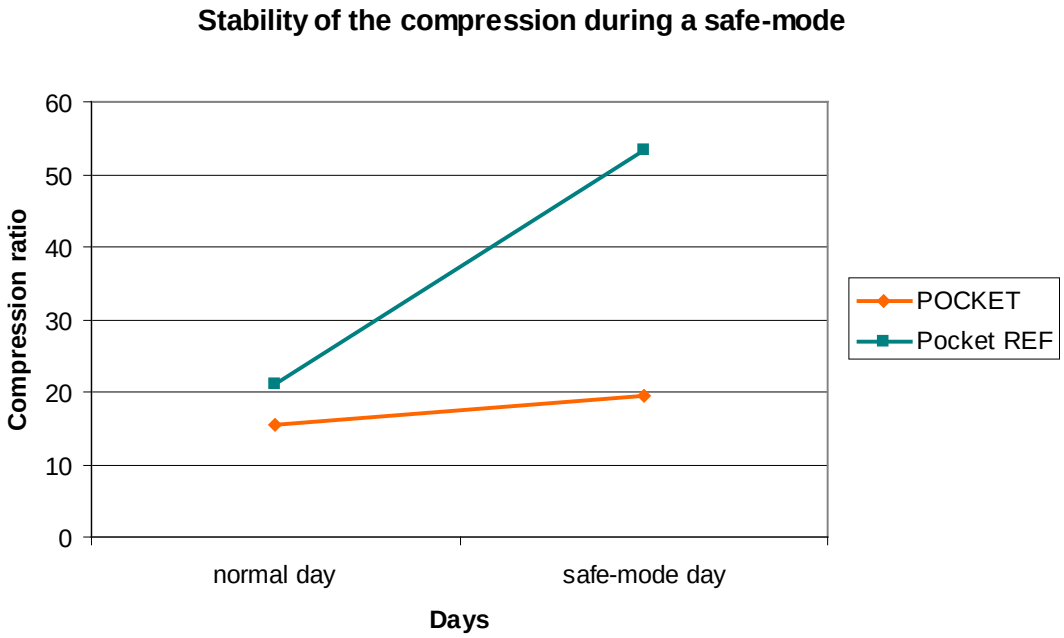

Figure 5. Compression performance comparison of POCKET and POCKET REFERENCE after a safe-mode entry. *The figure shows that the compression ratio of POCKET remains noticeably stable during the safe-mode period, even when the mask packets and inversion/correlation instructions have not been updated. POCKET REFERENCE shows a substantial performance degradation, being more sensitive to status changes in the telemetry.*

X. On-board software testing results

POCKET compression has been selected as the baseline compression technique for compressing the formation flying data on PROBA-3. The project is interested in increasing the information content in the housekeeping data related to the sensors and actuators used for formation flying. This is essential as this is a technology demonstration mission and so the more information one can gather during the formation flying experiments the higher value those experiments will have. The project is presently in Phase B. As part of the preparations POCKET was tested on the PROBA-3 target hardware a LEON-2 50 MHz processor. It could easily compress the 6680 bytes/second of formation flying measurement data with the CPU usage varying between 3.5% and 5.5%. The compression performance achieved with this sort of formation flying data was between 20% and 25%.

ESA has also started to investigate the use of POCKET in other areas. An industrial study will be starting in 2012 investigating its use on telecommunications satellites (which are typically real-time control only) as well as another one looking at its use in telepresence systems.

XI. Conclusion

We have presented a new algorithm for compressing spacecraft housekeeping data called POCKET. This is capable of compressing packets as they are generated. It shows a very good performance when tested on real housekeeping data from various ESA missions. It is very fast since it relies mainly on bit operations that are carried out at the data word level. In all cases considered so far, this will be fast enough to compress the real-time data stream. This ability means that the numerous advantages of housekeeping compression can be applied to both the real-time and the playback telemetry streams, including significantly increasing the information content of housekeeping telemetry. This can be achieved with a single algorithm and with little change to the present operation concepts or on-board telemetry storage systems. It also opens up new markets such as the control of telecommunications satellites, real-time industrial processes, robotics and telepresence.

Acknowledgments

The authors would like to thank Tomas Laroche and Spacebel of Belgium for conducting the testing on the LEON 2 processor.

References

[1]Martinez-Heras, J.A., Evans, D., Timm, R., "Housekeeping Telemetry Compression: When, how and why bother?", *International Conference on Advances in Satellite and Space Communications (SPACOMM 2009)*, July 20 – 25, 2009.

[2]Evans, D., Martinez-Heras, J.A., Timm, R., "Housekeeping Data: Can You Afford Not to Compress It?", *SpaceOps 2010*, April 26 – 30, 2010.

[3]Evans, D., Martinez-Heras, J.A., Timm, R., "An Idea from the World of Housekeeping Data Compression", *2nd Int. WS on On-Board Payload Data Compression (OBPDC)*, October 28 – 29, 2010.

[4]CCSDS 120.0-G-2, *"Lossless Data Compression, Green Book"*, The Consultative Committee for Space Data Systems, December 2006.

[5]Staudinger, P., Hershey, J., Grabb, M., Joshi, N., Ross, F., Nowak, T., "Lossless compression for archiving satellite telemetry data", *Aerospace Conference Proceedings, 2000 IEEE*, Vol. 2, pp. 299-304.