RANDOMIZED MODEL FOR OPTIMIZATION OF TELEMETRY SYSTEM DESIGN

Amir Liaghati, Nick Chang, Mahsa Liaghati
The Boeing Company
1100 Redstone Gateway Dr
Huntsville, AL 35824
amir.l.liaghati@boeing.com
chen.j.chang@boeing.com
mahsa.l.liaghati@boeing.com

ABSTRACT

The telemetry system designed for the space vehicle requires to provide constant data output, including video and instrumentation to the telemetry/transmitter box through the mission. The video data captured from the space travel is always high demand for social media or mission purpose. However, the limited telemetry bandwidth for transmitting all the high quality video data to the ground station before the end of the mission drives the system level design challenges. There are operational flight instrumentation data which takes higher priority than the video data in the telemetry bandwidth allocation. A common design approach to output all the video data is to utilize the filled data or IDLE frame with many small size IPv4 datagrams of the video. As a result, there are some video data are dropped out by the ground received equipment due to an extended period of waiting time of the receiver to collect all the defragmented IPv4 datagrams and reconstruct to a large IPv4 packet. One solution to resolve this problem is to have the telemetry processing box handling the defragmented IPv4 datagram by holding fragmented IP datagrams and reconstruct a whole IP packet before sending it to the transmitter, and yet still maintaining the vehicle telemetry system performance. This paper is going to focus on this method by developing a system level simulation tool and analyze the performance of the vehicle.

INTRODUCTION

It is difficult to downlink all the captured high quality video data in realtime due to the bandwidth constraint. One way to mitigate this problem is to rank the priority of the output telemetry. In this paper, there are two types of video formats used in the simulation: real-time, and storage. The real-time video is transmitted continuously whenever the raw video data is compressed, and the storage video is transmitted whenever there is not enough real-time time video to fill in the constant output video stream.

The video system used in a typical space systems requires to provide constant data to the telemetry/transmitter box through the mission. The current design implementation in the video

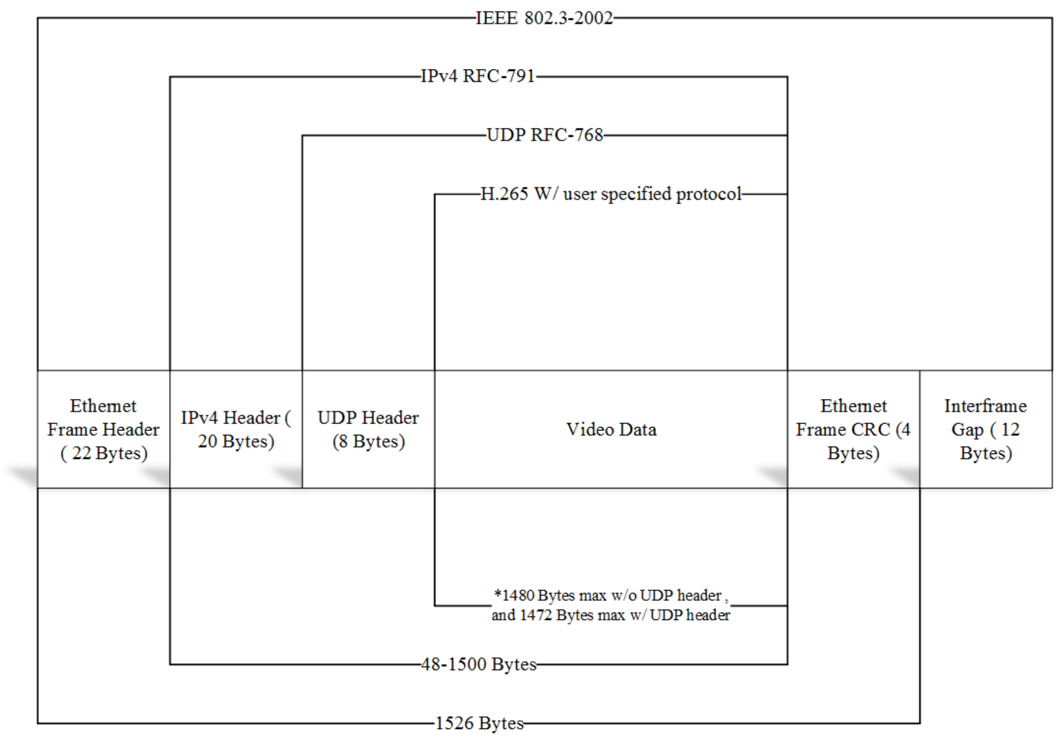
IEEE 802.3-2002
IPv4 RFC-791
UDP RFC-768
H.265 W/ user specified protocol
Ethernet Frame Header (22 Bytes)
IPv4 Header (20 Bytes)
UDP Header (8 Bytes)
Video Data
Ethernet Frame CRC (4 Bytes)
Interframe Gap (12 Bytes)
*1480 Bytes max w/o UDP header , and 1472 Bytes max w/ UDP header
48-1500 Bytes
1526 Bytes

Figure 1: IEEE 802.3 Standard

control unit compresses the captured raw data into high quality video, and transmits the compressed video to the telemetry and the transmitter unit in IP/UDP data format. Figure 1 provides a high level video format diagram. The video message structure follows the IEEE 802.3 standard where a large IP packet is defragmented into multiple smaller size datagrams if the IP packet size is greater than 1500 bytes[1, 2, 3].

Problem Statement

One problem with the current system design implementation is that the storage video data are transmitted as multiple 1500 bytes datagrams, and the datagrams are transmitted only when there is not enough realtime video to keep the constant output rate. The scattered IP datagrams are not transmitted at the same time, and sometimes the receiver is not able to decode the received storage datagram due to extended waiting time in order to de-fragment the datagrams into the IP frame.

Proposed Method

The proposed method for improving the system performance without impacting the video system or the overall vehicle architecture is by adding two additional memory buffer in the telemetry

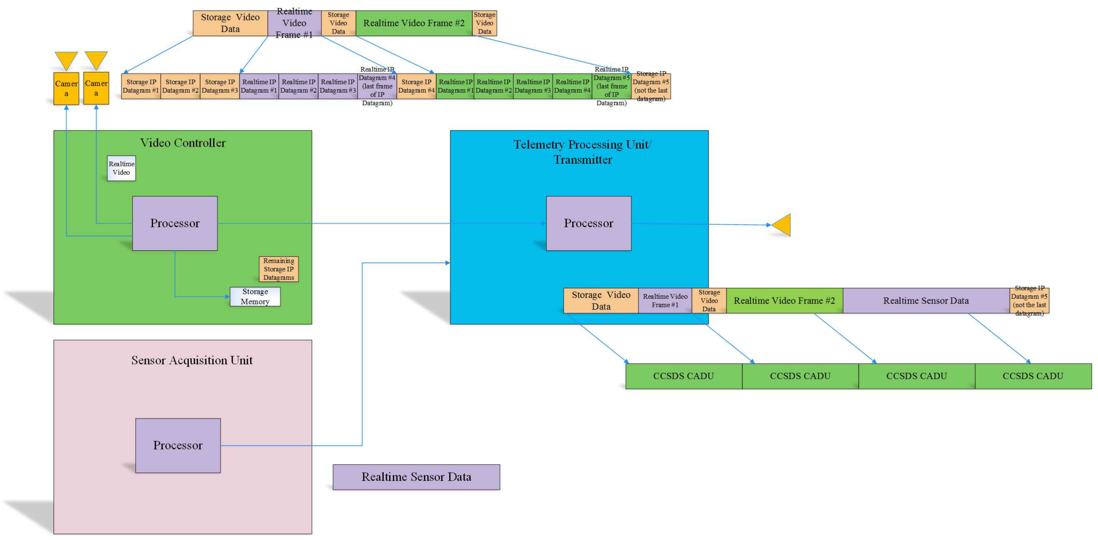

Figure 2: Conventional Method Block Diagram

processing unit. One memory is used as buffer of defragmented video, and the other memory is used as storage buffer of the reconstructed video data. The processor inside the telemetry processing unit forwards the real-time telemetry with CCSDS [4, 5, 6] format to the RF transmitter, and throttles the received fragmented IP datagrams to the buffer Memory. Upon receiving the last fragmented IP datagram, the telemetry processing unit will defragment the datagrams into the single IP packet, and store this large IP packet to the storage memory. This throttled IP packet will be transmitted whenever there is not enough data for the received real-time data. In this paper, a Matlab simulation model is developed to simulate the telemetry system performance by generating the random frames as the inputs to the model, aggregating the received frames and transmit with a constant output rate. There are two methods discussed in this paper about different ways of processing the aggregated received data and compare the system efficiency. In the model, random frame generator is used to generate real time and non-real time IP frames that can construct larger frames. Figure 3 shows the proposed method.

A. *Telemetry Design Option 1*

Figure 4 shows the block diagram for the proposed formatting scheme 1. In this process, a random frame generator is used to generate IP datagrams that can be reconstructed into an IP packet. If a random frame generator generates number 8, it means it takes 8 smaller frames to reconstruct one IP packet. $Capacity$ is used which represents the maximum amount of data in frames, that can be filled in allowable data stream and it is fixed. For instance, the capacity can be 10, therefore the $Capacity$ can be filled up with frames up to 10, and if it exceeds that number it will wait until the next packet. After generating the real-time and non-real time frames, the algorithm will reformat the frames. In option 1, the real-time frames always have the priorities and fill up the packet first. If there is enough room for more frames, the algorithm checks to see if the non-real time frame

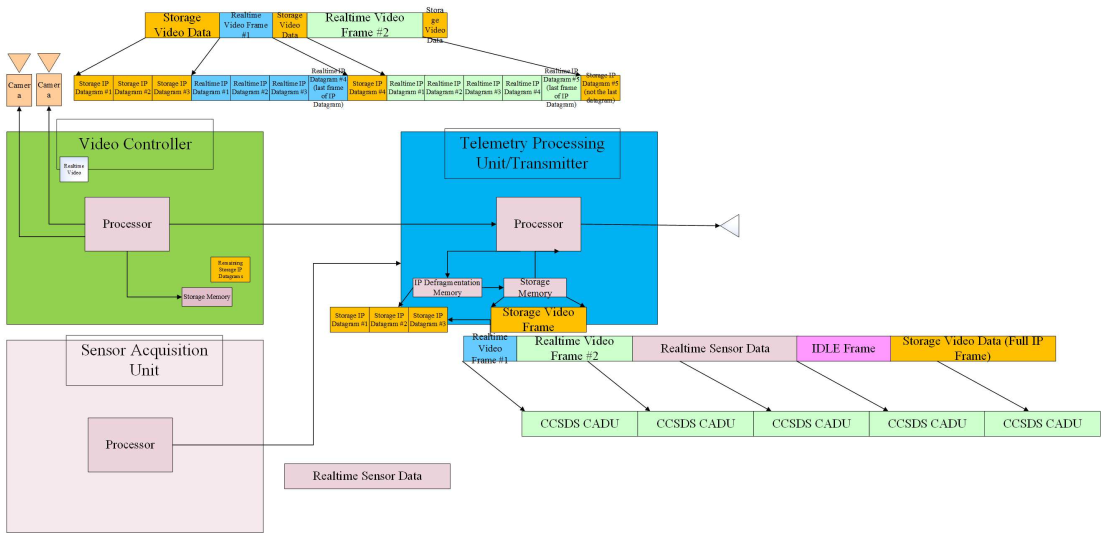

Figure 3: Proposed Method Block Diagram

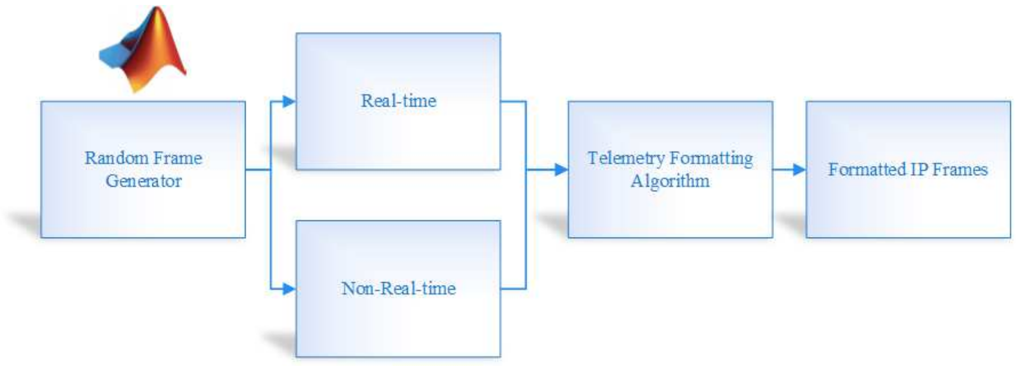

Figure 4: Matlab Model Block Diagram

is smaller or equal to the remaining space. If that is true, then the rest of the empty spot will be filled with non-real time frame. However, if the quantity of non-real time frame is greater than the empty slot, then the algorithm fills up the empty slot with the filled data. Filled data could be Idle Pattern or other non-real time data which does not require larger frames or sample based data. As a result, the non-real time frame will wait until next available slot to fill in. For example,

Frames1 = [2 8 3 6 9 . . .]
Frames2 = [7 7 7 1 . . .].

The output will be as the following:

Output = [2 7 1 8 2 3 7 . . .].

It is clearly shown that using this method, there will be additional delay of transmission of the non-real time frames and the filled data will increase. This method could be more efficient in the cases where the quantity of both real and non-real frames is equal to the allowable packets. This model uses a random number generator to consider all the possible combinations. Random means the probabilities of each number from 1 to 9 (assuming $maxframesize < capacity$) is same, meaning the probability is uniformly distributed as shown in the following equation:

$$f(n) = \frac{1}{n} \qquad \text{for } n = 1, 2, .., n \tag{1}$$

Figure 4 shows the block diagram for option 1. equation 2 shows the output when there is enough room for both real and non-real data.

if $capacity - Real > Non\_real$

$$\begin{aligned} output = [&output, Real, Non\_real, \\ &capacity - (Real + Non\_real))] \end{aligned} \tag{2}$$

Equation 3 shows the equation for the array $indicators$, where 0 represents $Real$, 1 represents Non_real, and 2 represents $Filled\_Data$.

$$indicators = [indicators, 0, 1, 2] \tag{3}$$

Equation 4 is used for the output when there is no room for Non-realtime, as a result, the filled data is used after the real time frame. Equation 5 shows the $indicator$ array. Since there is no room for non-real, indicator 2 is used for filled data as shown in equation 5.

else if $capacity - Real(i1) < Non\_real$

$$output = [output, Real(i1), capacity - Real]; \tag{4}$$

$$indicators = [indicators, 0, 2], \tag{5}$$

Equations 6, and 7 are used for a special case where Non-real can be fitted without perfectly after the real-time to fill the $Capacity$ entirely without any room for filled data. For this special case, only $indicator$ 0, and 1 are used.

else if $capacity - Real = Non\_real$

$$output = [output, Real, Non\_real] \tag{6}$$

$$indicators = [indicators, 0, 1] \tag{7}$$

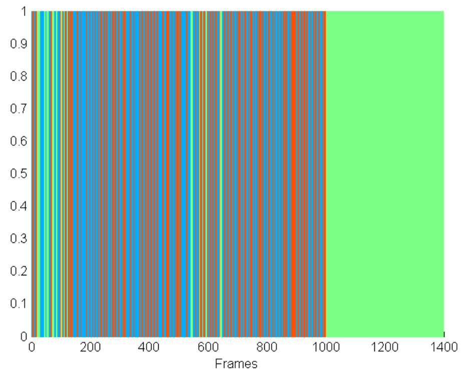

Figure 5: Result for Option1

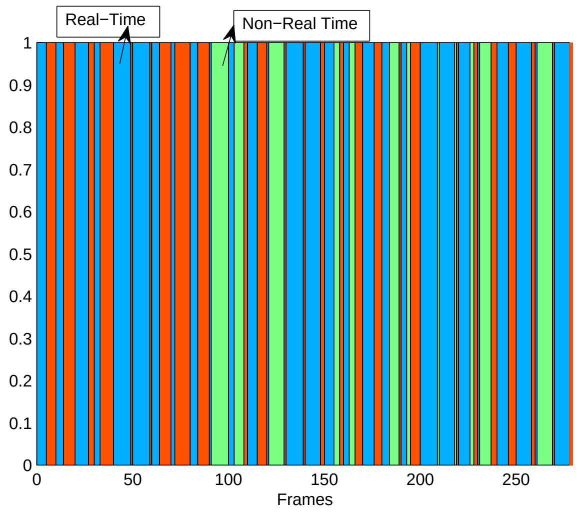

Figure 6: Result for Option1(zoomed in)

Figure 5 shows an example of the output frames using option 1 for 100 different iterations. Blue represents the real-time frames, green represents the non-real-time frames, and red represent the filled data. Figure 6 shows the output telemetry format option 1 (zoomed in). It is clearly shown that the non-real-time frames are pushed back to the right, and when all the real-time frames are transmitted, the rest will be used to transmit the remaining non-real-time frame. It can be seen clearly that it takes about 1400 frames to transmit all the real and non-real video data. The reason is because when there is no room for non-real time frames, filled data is used. This leads us to a more efficient approach as described in option2 described in the next section.

B. Telemetry Design Option 2

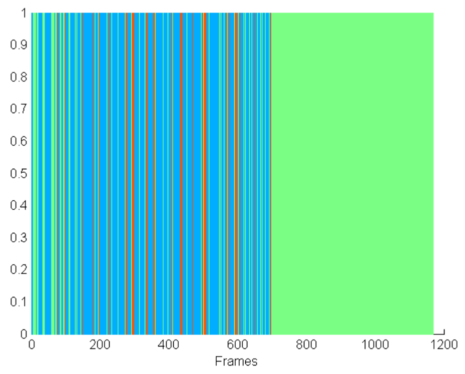

Figure 7: Result for Option2

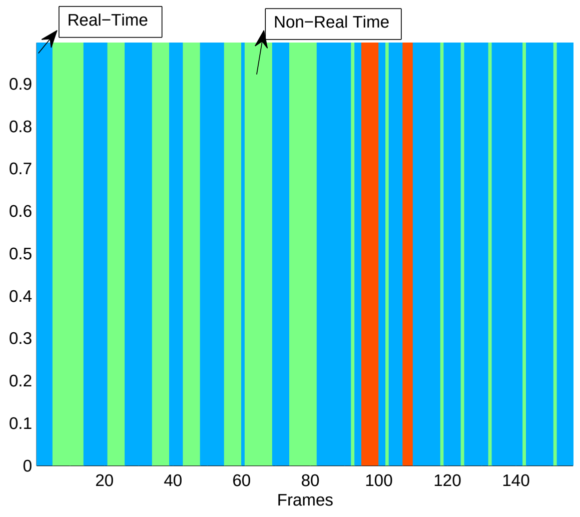

Figure 8: Result for Option2 (zoomed in)

Due to the large amount of filled data in option 1, a new option is suggested for more optimum formatting of the frames. One of the novel features of this option is that when there is no room for non-real-time frames, the empty slot starts by filling the non-real time frames and the reaming will be in the following packet. This will be optimum and the data will be transmitted sooner than option 1. Filled data is used only when the frame numbers for both real and non-real are small for a few sequence, the remaining slot will be filled with filled data. Figure 7 shows an example of the output frames using option 2 for 100 different iterations, and figure 8 shows the output telemetry format option 2 (zoomed in). It is clearly shown that the filled data is reduced significantly because the non-real time is formatted after real-time regardless of capacity.

For option 2, a variable $rest$ is introduced, and its value represents the rest of the frame when there is not enough room to fill up the current packet. In this option, the main objective is to packetize real and non-real time frame back to back to minimize the use of fill data. The following equations are used for a case where there is room for current real-time, non-real time, and the next real-time. Sometimes only portion of the next real-time might be fitted in the current packet, as a result, the rest will carry for the next packet. As a result the next real-time value needs to be updated.

$$output = [output, Real(i1), Non\_real(i2), \tag{8}$$

$$capacity - (Real(i1) + Non\_real(i2))] \tag{9}$$

$$Real(i1+1) = Real(i1+1) - (capacity - (Real(i1) + Non\_real(i2)))$$

$$indicators = [indicators, 0, 1, 0] \tag{10}$$

Note, there will be several cases that can occur in option 2. In this paper, only equations for one of the cases are shown.

Result

In this paper, there are 10 different iterations of generated random frames used in this study. Figure 9 shows the result for number of filled frame between option 1 and 2 for 10 different iterations. In summary, the option 1 requires more filled data than option 2 as we expected for different iterations. Figure 10 shows the result for the same iterations for the total number of frames transmitted. As predicted, the total number of frames for option 1 is greater than option 2. This implies that the telemetry system can transmit the data more efficiently in option 2 than option 1. Option 1 will be optimum when most of the real and non-real time frames are smaller than the capacity. Option 2 is more optimum by minimizing the filled data, however, the algorithm is more complex. Also, note that in option 2 the following real-time might shift to the right if non-real carries to the next frame slot, however, the data will be transmitted to the ground much earlier.

Summary

This work proposes a novel telemetry scheme for more efficient data rate optimization. Due to an extended period of waiting time to reconstruct the non-real time video frames, some frames

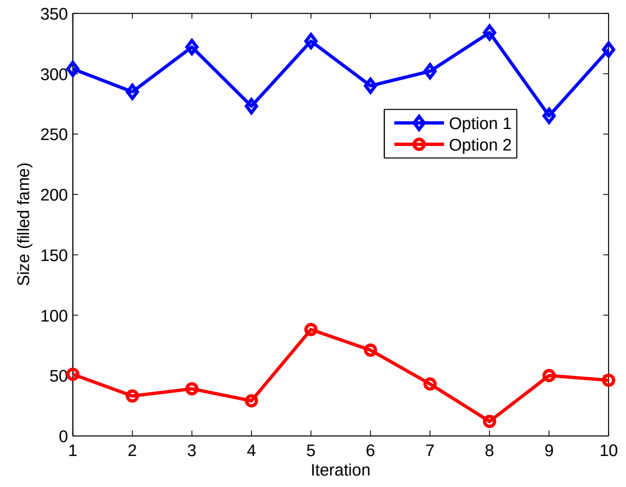

Figure 9: Result

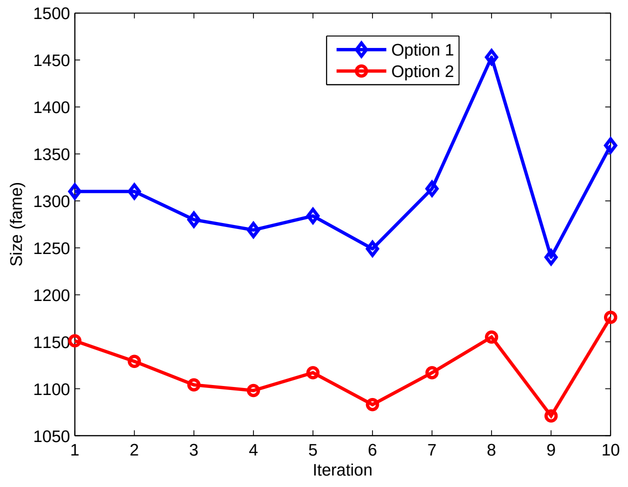

Figure 10: Result

can be dropped. As a result, the proposed two schemes can resolve this issue by fragmenting the defregmented storage video frames before transmitting to RF-transmitter. In option 1, more filled data is used, but the real-time is not shifted (delayed). Option 2 is very optimum by reducing the filled data, and transmitting the data to the ground faster.

REFERENCES

[1] J. Postel, "User datagram protocal (RFC 768)," August 1980.

[2] DARPA INTERNET PROGRAM, "Internet protocol (RFC 791)," September 1981.

[3] IEEE Computer Society, "IEEE Standard 802.3," June 2005.

[4] Consultive Committee for Space Data Systems (CCSDS), "Encapsulation service (131.1-B-2 Blue Book)," Oct. 2009.

[5] Consultive Committee for Space Data Systems (CCSDS), "Aos space data link protocol (732.0-B-2 Blue Book)," July 2006.

[6] Consultive Committee for Space Data Systems (CCSDS), "Tm synchronization and channel coding (131.0-B-2 Blue Book)," August 2011.