

RTCA Special Committee 186, Working Group 5

ADS-B UAT MOPS

Meeting #5

More UAT Synchronization Issues

Prepared by

Warren J. Wilson

The MITRE Corp.

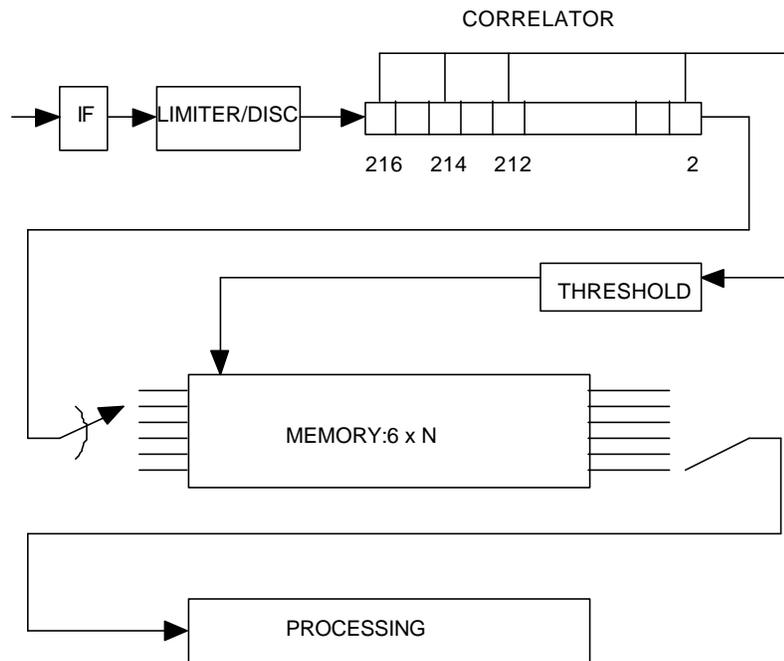
SUMMARY

In response to Action Item 4-2, some additional information regarding synchronization procedures for UAT (beyond what was discussed in UAT-WP-4-12) is provided. It is anticipated that the discussion in this paper (if deemed acceptable) will be added to the previous information to create an appendix for inclusion in the UAT MOPS.

Background

In this note we provide some additional information regarding synchronization procedures for UAT beyond what was discussed in UAT-WP-4-12. It is anticipated that the discussion in this paper (if deemed acceptable) will be added to the previous information to create an appendix for inclusion in the UAT MOPS. The areas covered here are: (1) a rationale for the particular 3 sample per bit correlation scheme described previously, (2) a discussion of the false alarm rate, and (3) a discussion of the required size of the memory device described in the previous work.

For the convenience of the reader, the high-level block diagram of the proposed receiver design is reproduced here.



Caveat: The discussions in this paper pertain to a particular implementation of a transceiver. Other techniques may also be viable.

Correlation Process

The correlation process described in UAT-WP-4-12 is designed to provide a high probability of detection, a low false alarm rate, and a reasonably accurate determination of the ideal sampling point for the information bits. To provide the necessary background, we will assume that the transmitted base band signal has been generated by passing the bit sequence through a Nyquist filter prior to the FM modulation process. The purpose of the filter is to restrict the transmitted spectrum in order to minimize interference to other nearby systems (e.g., DME). The filter is assumed to be a raised-cosine with a roll-off factor of 0.5. This is theoretically an infinite impulse response

(IIR) filter, but it can be implemented using truncation at plus and minus 3 bit periods. The resulting filter shape is shown in figure 1.

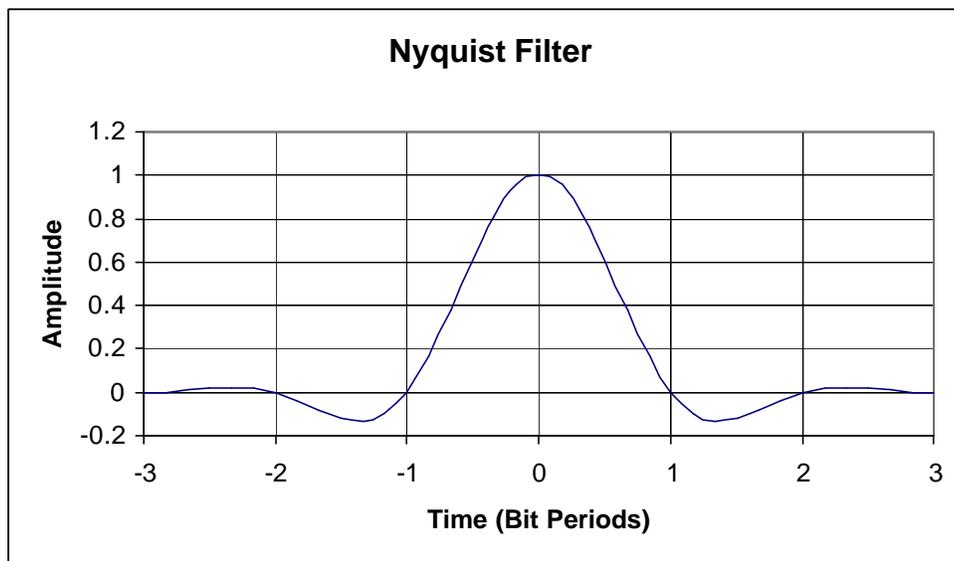


Figure 1. Base band Transmit Filter. Truncated Raised-Cosine Nyquist Filter With Roll-Off Factor = 0.5

When the synchronization sequence is passed through this filter the result is a fluctuating signal that can be used to modulate the transmitted frequency. The profile for the sequence is shown in figure 2. In the figure, the synchronization sequence is represented so that a 1 is a shift up in frequency by F_0 and a 0 is a shift down by F_0 . For UAT, the value of F_0 is given by 312.5 kHz. The correct frequency values for the synchronization sequence are achieved at the 36 integer values spanning 0 through 35. Due to the action of the filter, the instantaneous frequency smoothly varies from +1 to -1 at noninteger values. Sometimes the instantaneous frequency is actually larger than 1. Note that prior to the synchronization sequence (i.e., during ramp up), the waveform is assumed to be modulated with zeroes as specified in the MOPS.

A small portion of the sequence is shown in figure 3. This picture emphasizes that the frequency deviation is often reduced significantly between the ideal sampling points. This fact might seem to indicate that the most reliable performance would be available to a scheme that relies only on these robust bits (i.e., only use one sample per bit). This may be the case; but, as we will show below, such a method may not accurately determine the location of the waveform's "sweet spot." To address this issue, the method proposed in UAT-WP-4-12 uses every other sample of 6 samples per bit period to generate a correlation score (number of correct bits) that can vary between 0 and 108 for any given measurement. The general idea behind this scheme is that when the central sample is indeed at the center of the bit period, the 2 "outrigger" samples will typically have the same polarity. On the other hand, if the central sample is off by as little as one sixth of a bit period, some of the outrigger samples will have a high probability of being incorrect (refer to figure 3).

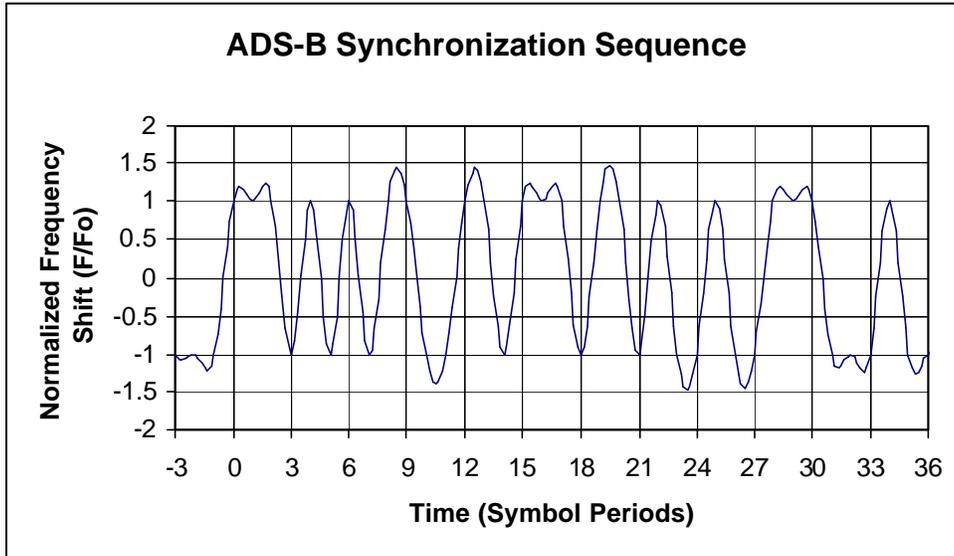


Figure 2. ADS-B Synchronization Sequence

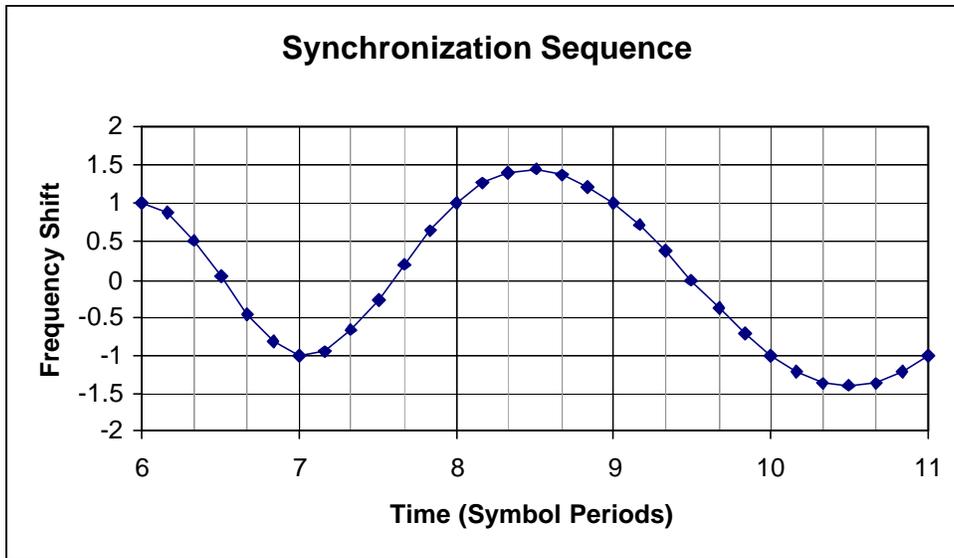


Figure 3. A Portion Of the ADS-B Synchronization Sequence

To analyze how this works we need to establish some mathematical notation. First, assume that the synchronization sequence is given by the vector

$$\vec{B} = (b_1, b_2, b_3, \dots, b_{36}).$$

Each component of this vector is +1 or -1 depending on whether the bit is a one or a zero. From this we can define a new vector, \vec{F} , whose components are all zero except that

$$F_{6i} = b_i.$$

We can now define a third vector that specifies the correlator tap weights:

$$T_i = F_i + F_{i+2} + F_{i+4}.$$

All the odd tap weights are 0. There are 108 nonzero tap weights (the even ones from 2 to 216).

If the incoming sample sequence at any given time (again, expressed as +1 or -1) is given the notation

$$S_i \quad \text{with} \quad i = 1 \text{ to } 216,$$

then the correlation value is given by

$$Corr = \left(108 + \sum_{n=1}^{108} T_{2n} S_{2n} \right) / 2.$$

This counts the number of correct samples out of a possible 108.

To estimate performance, we will assume that the incoming sample sequence is generated by a valid synchronization sequence (as portrayed in figures 2 and 3) plus additive white Gaussian noise. (These are only estimates, certain approximations are inherent in this analysis.) In other words,

$$\begin{aligned} S_i = +1 & \quad \text{if} \quad x_i = \sum N_{ij} F_j + g_i \geq 0 \\ S_i = -1 & \quad \text{if} \quad x_i = \sum N_{ij} F_j + g_i < 0 \end{aligned}$$

where the N_{ij} are the Nyquist filter coefficients and the g_i are Gaussian noise samples. It can be shown that when noise is absent from the input sequence and there is perfect time alignment between the input and the expected sequence the correlation score will be 108. With finite noise and a possible offset, the expected correlation score can be approximated by

$$\langle Corr(\mathbf{g}, m) \rangle = 108 - \frac{1}{2} \sum_{i=1}^{108} \operatorname{erfc} \left(T_{2i} x_{2i+m} \sqrt{\frac{\mathbf{g}}{2}} \right)$$

where the value m is an integer that indicates the relative timing offset in units of sample periods. g is the signal-to-noise ratio (SNR). The results of this equation for various values of SNR and offset are shown in figure 4.

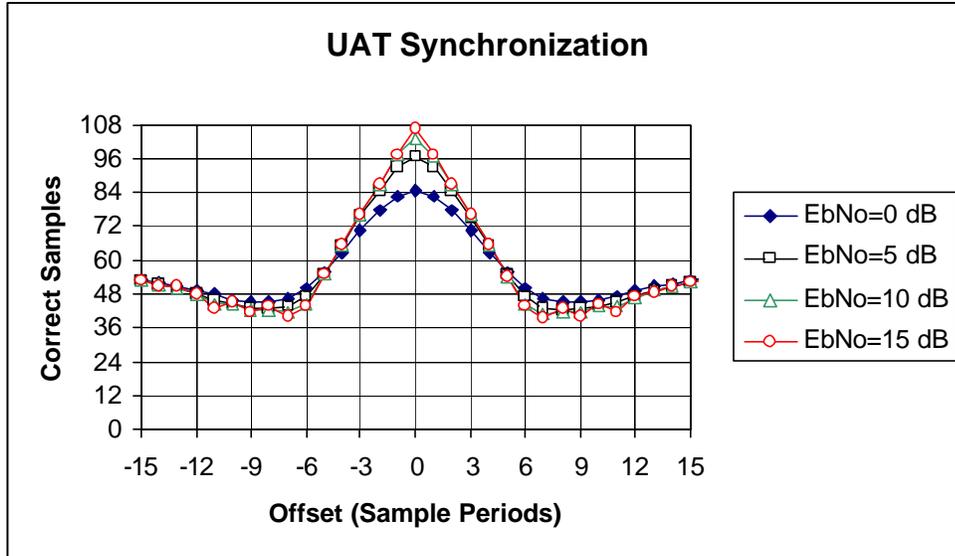


Figure 4. Performance of the Three Sample per Bit Scheme

Note that the curves in figure 4 have some desirable features. In particular the curves are quite peaked whenever the SNR (or EbNo) is greater than 5 dB. This feature should make it comparatively easy to identify the sample time with the smallest offset. This performance can be compared with a scheme based on one sample per bit. In that case the tap weights would be given by

$$T_i = F_i$$

and the correlation value would be given by

$$Corr = \left(36 + \sum_{n=1}^{36} T_{6n} S_{6n} \right) / 2.$$

The expected correlation value is given by an equation similar to the one for the three-sample case. It is evaluated for various values of SNR and time offset in figure 5. In the one sample per bit scheme the correlation “peak” is actually a plateau, and at high SNR there is a very good likelihood that several samples will have the maximum possible value. The “sweet spot” will be very difficult to identify. This provides the rationale for the first scheme.

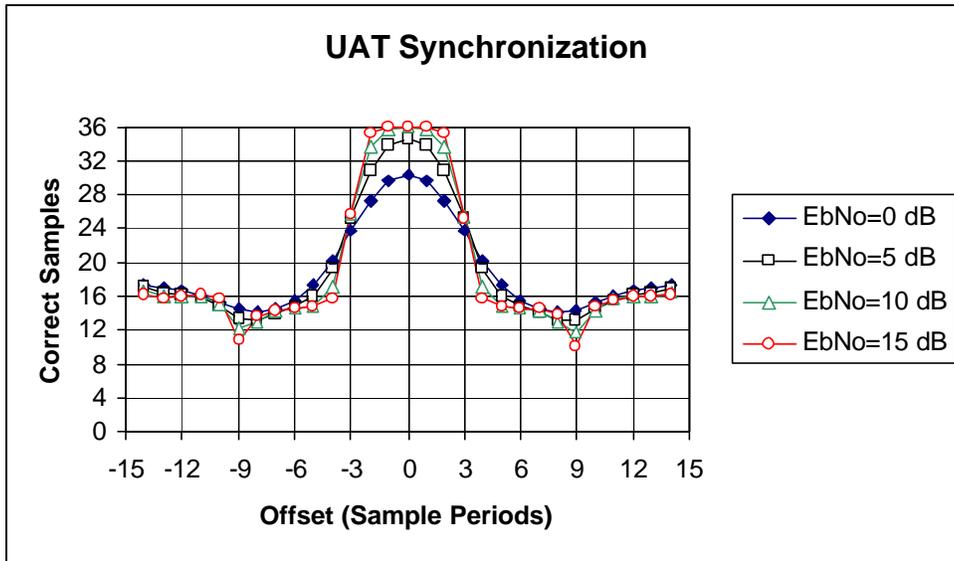


Figure 5. Performance of the One Sample per Bit Scheme

False Alarm Rate

The false alarm rate is related to the probability that a random bit sequence (or simply noise) creates a correlation value that exceeds the synchronization threshold. A reasonable rate of false alarms can be tolerated provided that the receiver has a re-trigger capability. If all of the 108 samples that are used to create the correlation value were statistically independent, then we could write the false alarm probability as

$$P_{fa} = (0.5)^{108} \sum_{n=0}^{108-T} \frac{108!}{n!(108-n)!}$$

where T is the synchronization threshold. In that case the false alarm rate (number per second) would be

$$FAR = 5 \times 10^6 P_{fa}$$

since the ADS-B portion of each second is 0.8 s and the samples are taken at a rate of 6.25 MHz. This method is clearly not correct because it fails to take into account the finite bandwidth involved in the overall synchronization process. The narrow IF filter limits the bandwidth, and the addition of three samples per bit acts as a crude version of a digital filter matched to the transmitter Nyquist filter. Neighboring estimates of the correlation score are *not* independent. An extreme manifestation of this effect would interpret the synchronization as being based on only 36 independent bit samples occurring once each bit period. This would mean that the false alarm probability would be given by

$$P_{fa} = (0.5)^{36} \sum_{n=0}^{36-T/3} \frac{36!}{n!(36-n)!}$$

and the FAR by

$$FAR = 833333 P_{fa} .$$

In actuality the truth probably lies somewhere between these two extremes. As an *ad hoc* guess, we can try the following:

$$P_{fa} = (0.5)^{NE} \sum_{n=0}^{NE-T(NE/108)} \frac{NE!}{n!(NE-n)!}$$

and

$$FAR = 2.5 \times 10^6 (NE/108) P_{fa}$$

where NE is the effective number of bits. In figure 6 we plot the FAR versus T for three hypothetical values of NE . Together with these curves we have plotted the measured data as reported in UAT-WP-4-18. (Note that there was a typographical error in the table in that paper, and the threshold “93” should have been reported as “90.”) The curve fits well if we assume the effective number of bits in the correlation is 72.

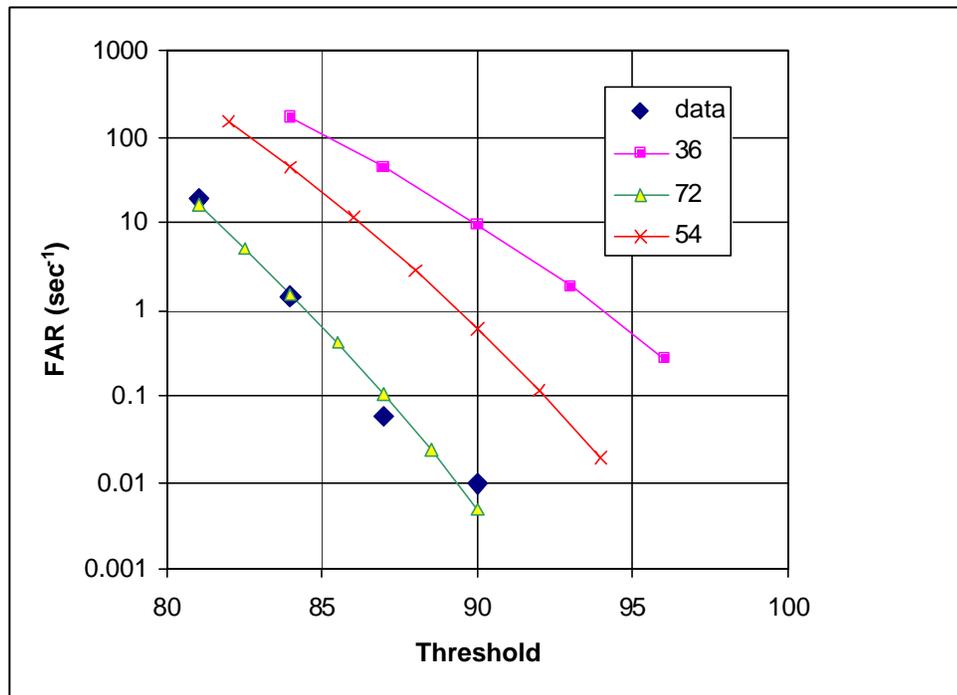


Figure 6. False Alarm Rate versus Threshold
(Parameter is Effective Number of Bits, NE)

The measurements indicate that the threshold, which is currently set at 94, could be lowered without generating a large number of false alarms per second. This threshold lowering could increase the robustness of the synchronization process, if desirable. However, it should be borne in mind that the measurements were made with an IF filter whose bandwidth was nominally 1.75 MHz. This filter will probably be replaced by a narrower one. A narrower filter may result in lowering the value of NE . If that is the case, and if the analysis presented here is correct, the FAR might increase somewhat. Therefore, the FAR should be re-measured with the final filter in place to determine if the threshold can be safely lowered.

Required Memory Size

The block labeled “Memory: 6xN” is provided to allow the receiver to sort out numerous (possibly overlapping) received signals and feed potentially valid bit streams into the “processing” block in the form of a sequence of long ADS-B RS code words. This block is essentially an elaborate de-interleaver. It is assumed that the bits are fed into the memory at a rate of 6.25 Mbps and can be read out at a rate of 6.25 Mbps at the same time. In other words, it must be capable of reading in one bit and reading out one bit during the same clock cycle (160 nanosecond). The question addressed here is the required size of the memory, or “What should N be?”

To answer that question, a simulation was created which generated 2200 ADS-B message starts each second. This corresponds to the worst-case loading scenario. These messages were then sorted according to time-of-arrival. It was then assumed that every one of these messages would result in a crossing of the synchronization threshold, i.e., every message would need to pass through to the processing block. That is actually an unrealistic assumption since many of the synchronization attempts may fail due to self-interference. Thus, this analysis will provide an overestimate of the memory requirement.

The simulation operated by marching along in time and noting each message start. When a message start was encountered, a backlog of bits to be processed was added to a running total. This total was incremented when encountering new messages and decremented by feeding bits into the processing block. The maximum size of the backlog is the length of the memory, N. The simulation was run repeatedly to determine the probability that the backlog would overflow versus the memory size. This is equivalent to the probability that an ADS-B message will be dropped because it gets overwritten before it can be transferred to the processing block. The results of the simulation are shown in figure 7.

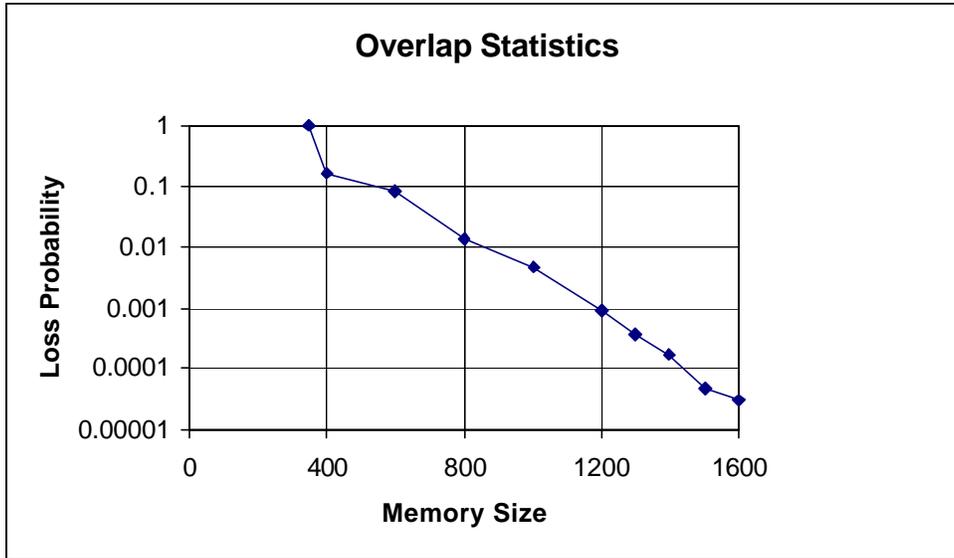


Figure 7. Memory Length Determination

Figure 7 indicates that if the memory length is 1200 bits, the probability that an ADS-B message will be dropped is 0.001. Thus, the total memory size required to implement the sorting function would be about $6 \times 1200/8 = 900$ bytes.