NATIONAL CENTRE FOR RADIO ASTROPHYSICS

Release Ver 1.0 Date : 12/11/2021

Internal Technical Report

Timestamp flow in GWB and correction

S. Harshavardhan Reddy

Sanjay Kudale

Navnath Shinde

Ajith Kumar. B

B. C. Joshi

Yashwant Gupta

Objective : To provide detailed description of timestamp flow in GWB and its correction after removing bugs in the GWB code. To provide details of tests conducted and corresponding results to verify the accuracy of timestamp. Also, to provide brief history of timestamp offset with actual time since 16-antenna GWB version has been released. **For general users, the final operative instructions are provided in the conclusion section (section 14).** Issues related to DUT correction, jitter in GPS PPS/PPM pulse and timestamp offset because of Polyphase filtering will be detailed in an additional notes.

NATIONAL CENTRE FOR RADIO ASTROPHYSICS

Table of Contents

1. Introduction

The upgraded wideband correlator (GWB) was first released in August 14, 2015 as a 16-antenna correlator and was subsequently upgraded to 32-antenna version on July 04, 2017. Both the time-series and imaging data acquired with the GWB requires accurate time-stamps for referencing events in astronomical data for sources, such as radio pulsars and Fast radio bursts. Prior to May 14, 2020, timestamp recorded along with visibility data and beam data in GWB had time difference with the actual time at which the data was sampled. For visibility data, the difference was not constant and it varied with the integration (LTA1[1]) during real-time computation in GWB. For beam data, the difference was constant. This report explains the flow of timestamp in the GWB, the time difference with the actual time and corrections done to eliminate the time difference. Post corrections, GWB is released on 14/05/2020. This report also documents the timestamp offset from GWB 16-antenna version release date (14/08/2015) to 09/09/2016 and from 09/09/2016 to 04/07/2017 and provides a prescription to obtain the correct timestamp for the earlier data. We also report the time offset introduced during the FITS conversion of recorded data in GMRT format which depends on total integration i.e.; LTA1 and LTA2[2].

2. Synchronizing the start of observation with GPS PPS

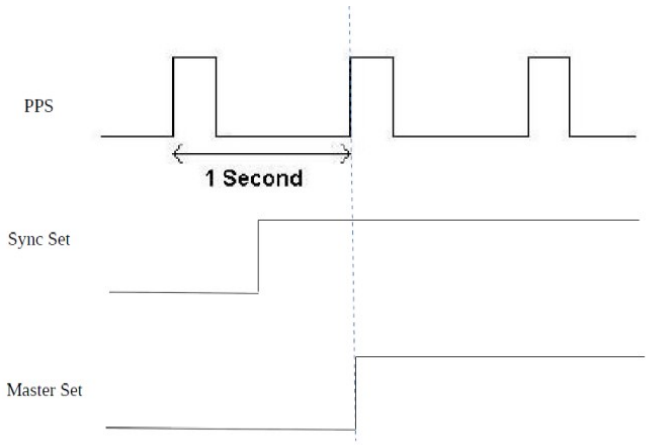

Fig. 1 **: Synchronizing in the GWB using the GPS 1-PPS signal**

In the GWB, the start of an observation is synchronized with a 1-PPS signal derived from the observatory GPS receiver. Initially, all the ROACH boards are programmed, such that they wait for a “master set” signal in the FPGA to go high for

1 LTA1 refers to the number of integrations carried during real-time computation in GWB.
2 LTA2 refers to the post integration during recording of the data in GMRT format.

starting the digitisation and packetization. Then, an initialization (init) command is given to the GWB through an online control server by the operator. Five seconds after this 'init' command, the "sync set" signal in the FPGA is made high through a telnet port. After the "sync set" signal is made high, the "master set" signal goes high at the occurrence of the next 1-PPS pulse. Once the "master set" becomes high, packets are formed with the sampled data and sent to Compute nodes through 10GbE link. Along with the sampled data, packet number information is sent for every packet. Packets are received on Compute node through 10GbE NIC. In the first Compute node, upon receiving the first packet, the local time (rounded to the second) is stored in a timeval structure (start time). This start time is shared with the host node (gwbh6) and the rest of the nodes.

3. Timestamp calculation for delay and fringe correction

The compute nodes process 256 MB of data in one iteration, which translates to 0.67108864 second (buffer time or alternatively called STA) of data for 200 MHz (8-bit per sample) and 400 MHz (4-bit per sample) bandwidth modes. For 100 MHz (8-bits per sample) bandwidth mode, buffer time is 1.34217728 seconds. The timestamp sent for calculating delay and fringe values is obtained by incrementing buffer time to the start time at every iteration. Through out this document, the timestamp offset or buffer time when mentioned in seconds is for 200 MHz bandwidth and 400 MHz bandwidth modes unless specified.

4. Data flow in GWB

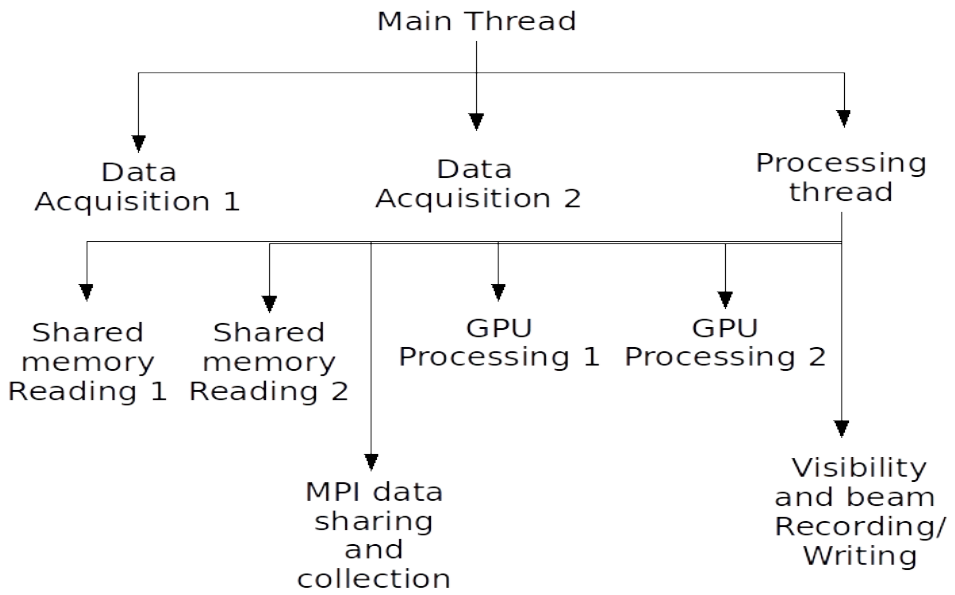

Fig. 2 : **OpenMP threads structure in GWB**

Different sections of GWB are performed in parallel using OpenMP threads. Two threads (read threads) are assigned to read raw data from shared memory, each thread for two inputs. One thread (MPI thread) for doing MPI transfers i.e.; time slicing and sharing of raw data between compute nodes and collection of results by host machines. Two threads (GPU threads) for GPU processing by both GPUs, each GPU processing half the time slice data. One thread (write thread) for writing results on to shared memories for being recorded. Double buffering ping-pong scheme is used to prevent overlap of processing and reading/writing the buffers.

	Read threads (both threads)		**MPI Sharing**(Time slicing and data sharing)		**GPU processing (both GPU threads)**		**MPI Gathering**(Visibility and beam data)		**Recording**(Writing the results into SHM)
Iteration 0	Read buffer 0 Read buffer 1	Buffer 0 (T0) junk	MPI buffer 0 MPI buffer 1	junk junk	visi buffer 0 visi buffer 1	junk junk	output buffer 0 output buffer 1	junk junk	junk
Iteration 1	Read buffer 0 Read buffer 1	Buffer 0 (T0) Buffer 1 (T1)	MPI buffer 0 MPI buffer 1	Buffer 0 (T0) junk	visi buffer 0 visi buffer 1	junk junk	output buffer 0 output buffer 1	junk junk	junk
Iteration 2	Read buffer 0 Read buffer1	Buffer 2 (T2) Buffer 1 (T1)	MPI buffer 0 MPI buffer 1	Buffer 0 (T0) Buffer 1 (T1)	visi buffer 0 visi buffer 1	Buffer 0 (T0) junk	output buffer 0 output buffer 1	junk junk	junk
Iteration 3	Read buffer 0 Read buffer 1	Buffer 2 (T2) Buffer 3 (T3)	MPI buffer 0 MPI buffer 1	Buffer 2 (T2) Buffer 1 (T1)	visi buffer 0 visi buffer 1	Buffer 0 (T0) Buffer 1 (T1)	output buffer 0 output buffer 1	Buffer 0 (T0) junk	junk
Iteration 4	Read buffer 0 Read buffer 1	Buffer 4 (T4) Buffer 3 (T3)	MPI buffer 0 MPI buffer 1	Buffer 2 (T2) Buffer 3 (T3)	visi buffer 0 visi buffer 1	Buffer 2 (T2) Buffer 1 (T1)	output buffer 0 output buffer 1	Buffer 0 (T0) Buffer 1 (T1)	Buffer 0 (T0)
Iteration 5	Read buffer 0 Read buffer 1	Buffer 4 (T4) Buffer 5 (T5)	MPI buffer 0 MPI buffer 1	Buffer 4 (T4) Buffer 3 (T3)	visi buffer 0 visi buffer 1	Buffer 2 (T2) Buffer 3 (T3)	output buffer 0 output buffer 1	Buffer 2 (T2) Buffer 1 (T1)	Buffer 1 (T1)
Iteration 6	Read buffer 0 Read buffer 1	Buffer 6 (T6) Buffer 5 (T5)	MPI buffer 0 MPI buffer 1	Buffer 4 (T4) Buffer 5 (T5)	visi buffer 0 visi buffer 1	Buffer 4 (T4) Buffer 3 (T3)	output buffer 0 output buffer 1	Buffer 2 (T2) Buffer 3 (T3)	Buffer 2 (T2)
Iteration 7	Read buffer 0 Read buffer 1	Buffer 6 (T6) Buffer 7 (T7)	MPI buffer 0 MPI buffer 1	Buffer 6 (T6) Buffer 5 (T5)	visi buffer 0 visi buffer 1	Buffer 4 (T4) Buffer 5 (T5)	output buffer 0 output buffer 1	Buffer 4 (T4) Buffer 3 (T3)	Buffer 3 (T3)
Iteration 8	Read buffer 0 Read buffer 1	Buffer 8 (T8) Buffer 7 (T7)	MPI buffer 0 MPI buffer 1	Buffer 6 (T6) Buffer 7 (T7)	visi buffer 0 visi buffer 1	Buffer 6 (T6) Buffer 5 (T5)	output buffer 0 output buffer 1	Buffer 4 (T4) Buffer 5 (T5)	Buffer 4 (T4)

indicates block that is being processed

Fig. 3 : **Data flow between OpenMP threads in GWB (LTA1 = 1)**

In the zeroth iteration, zeroth buffer (256 MB) sampled at the start of observation (T0) is read from the shared memory by read threads. At the same time,

MPI threads, GPU threads and read threads performs processing/IO on junk data. In the first iteration, read threads read the first buffer's (T1) data, MPI threads perform time slicing and sharing on zeroth buffer's data (T0), and the rest of the threads performs processing/IO on junk data. In the second iteration, read threads read second buffer's data (T2), MPI threads perform time slicing and data sharing on T1 data and GPU threads perform correlation and beam formation processing on T0 data. In the third iteration, read threads read third buffer (T3), MPI thread performs time slicing and sharing on T2 data, GPU threads process T1 data and MPI threads perform results collection on T0 data. In the fourth iteration, read threads read fourth buffer (T4), MPI thread perform time slicing and sharing on T3 data and results collection on T1 data, GPU threads process T2 data and write thread performs writing on T0 data's results. The above mentioned scenario is for integration of one buffer i.e; LTA1 = 1. For longer integration, the GPU threads process and hold the visibility data for the number of buffers to be integrated (LTA1). MPI collection of results and writing of results is distributed among LTA1 number of iterations.

5. Flow of timestamp along with visibility data

In the zeroth iteration, start of observation time (start time) is shared by first compute node to all the other nodes (host and compute). Visibility host calculates the timestamp for delay and fringe value calculations. The timestamp and its corresponding delay and fringe values are calculated one iteration in advance and shared with the compute nodes and applied in the next iteration in phase shifting kernel in GPU processing. Hence, in the second iteration, timestamp of T1 is calculated, its delay and fringe values calculated and shared by MPI thread. GPU threads process T1 buffer in the third iteration and the delay and fringe values are applied correspondingly.

The timestamp calculated is stored in a timestamp buffer. The timestamp buffer is an array of three elements of timeval structure. The iteration count ('run count') runs from 0, and the timestamp calculated in the current iteration is stored in timestamp buffer at (run count % 3). The visibility data integrated in the GPUs and stored in global memory of GPUs, is copied into the CPU memory once in LTA1 number of iterations at the end of every (run count + 1) % LTA1 = 0 iteration. Collection of visibility data by MPI thread starts at next (run count + 1) % LTA1 = 0 iteration and is completed LTA1 buffers later. At the end of next (run count + 2) % LTA1 = 0 iteration which is LTA1 buffers later, visibility data and timestamp are written into the visibility shared memory in visibility host node.

Timestamp buffer (run count % 3)	Timestamp	Iteration number		Processing description
0		Iteration 0		
1	T0	Iteration 1		
2	T1	Iteration 2		
...	...	...		
...	...	...		
(ref count % 3)	T(ref count – 1)	Iteration 'ref count'	----->	Start of integration of visibility 'V'
...	...	...		
...	...	...		
(ref count + LTA – 1) % 3	T(ref count + LTA)	Iteration 'ref count + LTA – 1'	----->	End of integration of visibility 'V'
...	...	...		
...	...	...		
(ref count + 2 x LTA – 1) % 3	T(ref count + 2 x LTA)	Iteration 'ref count + 2 x LTA – 1'	----->	Start of MPI collection of visibility 'V'
...	...	...		
...	...	...		
(ref count + 3 x LTA – 2) % 3	T(ref count + 3 x LTA – 1)	Iteration 'ref count + 3 x LTA – 2'	----->	End of MPI collection of visibility 'V'
(ref count + 3 x LTA – 1) % 3	T(ref count + 3 x LTA)	Iteration 'ref count + 3 x LTA – 1'	----->	Start of collection of visibility 'V' into LTA buffer
...	...	...		
...	...	...		
(ref count + 4 x LTA – 4) % 3	T(ref count + 4 x LTA – 3)	Iteration 'ref count + 4 x LTA – 4'		
(ref count + 4 x LTA – 3) % 3	T(ref count + 4 x LTA – 2)	Iteration 'ref count + 4 x LTA – 3'		
(ref count + 4 x LTA – 2) % 3	T(ref count + 4 x LTA – 1)	Iteration 'ref count + 4 x LTA – 2'	----->	End of collection of visibility 'V' into LTA buffer and written into visibility shared memory. Timestamp written into the shared memory buffer is (run count – 2) % 3

Fig. 4 **: Visibility data and timestamp flow with iterations before correction (from 04/07/2017 to 14/05/2020)**

Tracking the actual time at which the visibility data integration started and the timestamp that is sent to the shared memory buffer gives the difference between the timestamp (sent to visibility shared memory) and actual time. As the T0 buffer GPU processing starts at second iteration (run count = 2), and the visibility data integration starts at every run count % LTA1 = 0 iteration, the actual time associated with the visibility buffer is T(run count - 2). Now, assume that the iteration at which the integration started as 'ref count', then its corresponding actual time is T(ref count - 2). At the end of next (run count + 1) % LTA1 = 0 iteration i.e.; at ref count + LTA1 – 1 iteration, visibility data (say V) is copied into CPU memory from GPU memory. At the start of next (run count + 1) % LTA1 = 0 i.e.; at ref count + (2 x LTA1) – 1 iteration, MPI collection of visibility data V starts and LTA1 buffers later collection finishes i.e; at ref count + (3 x LTA1) – 2 iteration. At the end of next (run count + 2) % LTA1 = 0 iteration i.e.; at ref count + (4 x LTA1) – 2 iteration, visibility data V is written into visibility shared memory. The timestamp from the timestamp buffer

written into the shared memory is from the ((run count - 2) % 3) element in the array. Now depending on the LTA1, the difference between the actual time (of start of integration of visibility buffer V) and timestamp written into the visibility shared memory varies.

So, the timestamp offset varies with LTA1 and the amount of time offset is delay of ((4 x (LTA1 – 1)) + 1) x buffer time (*Table 1*). For any integration outside the GPU i.e.; at the time of recording, the time offsets (with actual time) do not change as the timestamp is taken from the visibility shared memory.

Table 1 : Timestamp offsets for interferometry mode ($\Delta t^{v}_{real\text{-}time}$) from **04/07/2017 to 14/05/2020**

LTA1	**Timestamp offset ($\Delta t^{v}_{real\text{-}time}$) in seconds for 200 MHz/400 MHz Bandwidth**	**Timestamp offset ($\Delta t^{v}_{real\text{-}time}$) in seconds for 100 MHz or lower Bandwidth**
1	0.67108864	1.34217728
2	3.3554432	6.7108864
4	8.72415232	17.44830464
8	19.46157056	38.92314112
16	40.93640704	81.87281408
32	83.88608	167.77216

6. Flow of timestamp along with beam data

For beam data, LTA1 does not matter. For every buffer (256 MB) beam data is written into shared memory. At the second iteration, T0 buffer is processed by GPU threads, and in the third iteration T0 buffer MPI collection is performed by MPI thread. For IA/PA beam, T0 buffer beam data is written to beam data shared memory in fourth iteration. Along with beam data, timestamp is written into beam data shared memory. The timestamp is taken from the timestamp buffer from (run count - 1) % 3 element of the timestamp buffer which has timestamp of T2. Hence, for IA/PA beam data the timestamp is delayed by 2 buffers (*Table 2*).

Table 2 : Timestamp offsets for beam-former mode ($\Delta t^{b}_{real\text{-}time}$) from **04/07/2017 to 14/05/2020**

Beam	Timestamp offset ($\Delta t^{b}_{real\text{-}time}$) in seconds for 200 MHz/400 MHz bandwidth	Timestamp offset ($\Delta t^{b}_{real\text{-}time}$) in seconds for 100 MHz or lower bandwidth
IA/PA	1.34217728	2.68435456
Voltage (CDP)	2.01326592	4.02653184

For voltage beam (CD Pipeline), after T0 buffer MPI collection in the third iteration, because of bug in the code, T0 buffer beam data is written to beam data shared memory for CD pipeline operation in fifth iteration. Hence, for voltage beam data, the timestamp is delayed by 3 buffers (*Table 2*).

7. Correction of timestamp offset for visibility data from 14/05/2020

Timestamp buffer (run count % 4)	visi count	Timestamp	Iteration number		Processing description
0	0		Iteration 0		
1			Iteration 1		
2		T0	Iteration 2		
...	...	...	...		
...		...	...		
(visi count % 4)	visi count	T(ref count – 2)	Iteration 'ref count'	----->	Start of integration of visibility 'V'
...		...	...		
...		...	...		
		T(ref count + LTA – 3)	Iteration 'ref count + LTA – 1'	----->	End of integration of visibility 'V'
(visi count + 1) % 4	visi count + 1	T(ref count + LTA – 2)	Iteration 'ref count + LTA'	----->	Start of MPI collection of visibility 'V'
...		...	...		
...		...	...		
		T(ref count + 2 x LTA – 3)	Iteration 'ref count + 2 x LTA – 1'	----->	End of MPI collection of visibility 'V'
(visi count + 2) % 4	visi count + 2	T(ref count + 2 x LTA – 2)	Iteration 'ref count + 2 x LTA'	----->	Start of collection of visibility 'V' into LTA buffer
		...	...		
...		...	...		
...		T(ref count + 3 x LTA – 3)	Iteration 'ref count + 3 x LTA – 1'	----->	End of collection of visibility 'V' into LTA buffer and written into visibility shared memory. Timestamp written into the shared memory buffer is (visi count % 4)

***Fig. 5* : Visibility data and timestamp flow with iterations after correction (from 14/05/2020). "LTA" mentioned in the figure refers to integration during real-time computation i.e.; LTA1.**

For eliminating the timestamp offset as explained in section 5 for observation epochs prior to 14/05/2020 the following changes were implemented in the software. For correcting the timestamp offset, timestamp calculated (as mentioned in section 2) is stored in a new timestamp buffer (array of four elements, each element is a timeval structure). The timestamp is stored once in every LTA1 buffers at (run count % LTA1) = 0 iteration. At every (run count % LTA1) = 0 iteration, a new iteration counter (‘visi count’) is incremented. This new counter ‘visi count’ is used to address the new timestamp buffer at (visi count % 4) into which timestamp is stored. The timestamp stored is from the previous iteration as the timestamp is calculated one buffer in advance (to get delay and fringe values and to be shared to all the nodes for correction). At start of every run count % LTA1 = 0 iteration, visibility integration starts and ends LTA1 buffers later i.e.; at next (run count + 1) % LTA1 = 0 and the visibility data is copied into ‘visibility buffer’. At the next ‘run count’, MPI collection of visibility data starts and ends LTA1 buffers later i.e.; at next (run count + 1) % LTA1 = 0. At the next ‘run count’, visibility data collection into ‘LTA1 buffer’ starts and LTA1 buffers later i.e.; at next (run count + 1 % LTA1) = 0 collection into ‘LTA1 buffer’ ends and is written into shared memory. Along with visibility data, timestamp is picked from new timestamp buffer and written into shared memory. The timestamp picked from the new timestamp buffer is addressed at (visi count -2) % 4.

8. Experimental verification of timestamp offset in visibility data

For this experiment, input used was broadband noise modulated by GPS PPM. PPM modulated noise was fed as input to GAB RF input of a polarization of an antenna (C00 pol1). GAB LO was set to 250 MHz, Band-3 RF bandpass filter was selected, 200 MHz Baseband Low pass filter was selected. The noise power level was adjusted such that ON pulse and OFF pulse noise is clearly visible. GWB settings are 200 MHz bandwidth, 2048 spectral channels, total intensity mode. LTA1 is varied from 1 to 32 in steps of 2 i.e; 1,2,4,8,16,32. Integration is LTA1 value x 0.67108864 seconds.

Using ‘xtract’ tool a single channel time series was extracted and plotted after extrapolating the timeseries from the start time of recording. Plots below show the time series of a single channel (channel number 800) timeseries plotted with respect to extrapolated time from start time. As mentioned in section 4, it can be seen (on left side plots) that there is a timestamp offset which is varying with LTA1 and it has been corrected (on right side plots). Exact amount of offset and the timestamp’s accuracy is difficult to obtain from the plots sample time resolution.

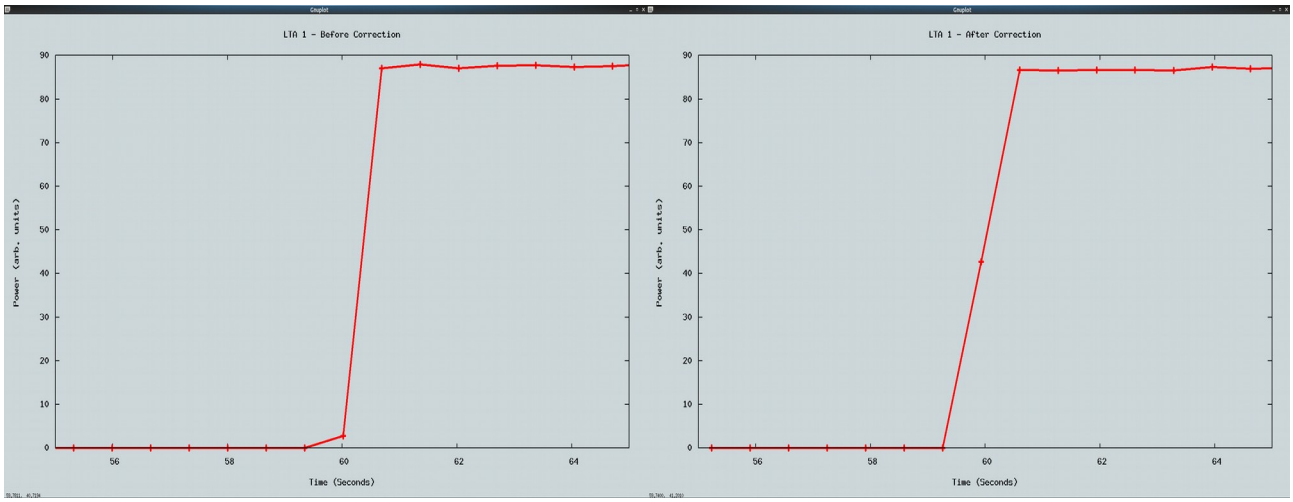

Fig. 6 **: Plot showing timeseries of a channel from visibility data for LTA1 = 1. Left side – Before timestamp offset correction, Right side – Post timestamp offset correction. Timestamp offset before correction is 1 LTA1 buffer i.e; 0.67108864 seconds. "LTA" mentioned in the figure refers to integration during real-time computation i.e; LTA1.**

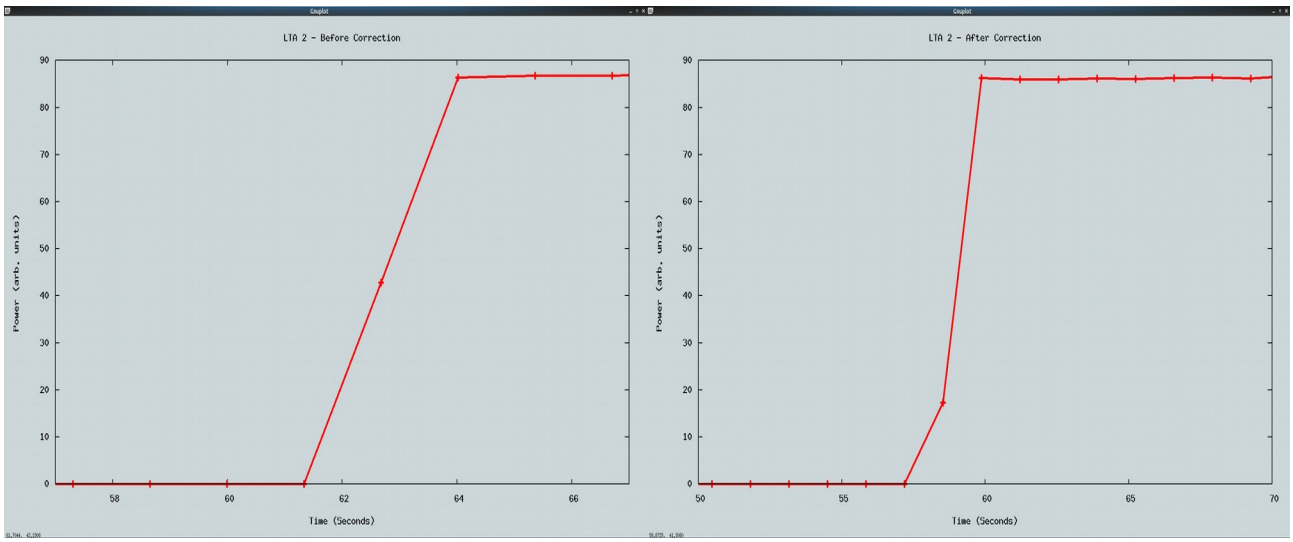

Fig. 7 **: Plot showing timeseries of a channel from visibility data for LTA1 = 2. Left side – Before timestamp offset correction, Right side – Post timestamp offset correction. Timestamp offset before correction is 5 LTA1 buffers i.e; 3.35 seconds. "LTA" mentioned in the figure refers to integration during real-time computation i.e; LTA1.**

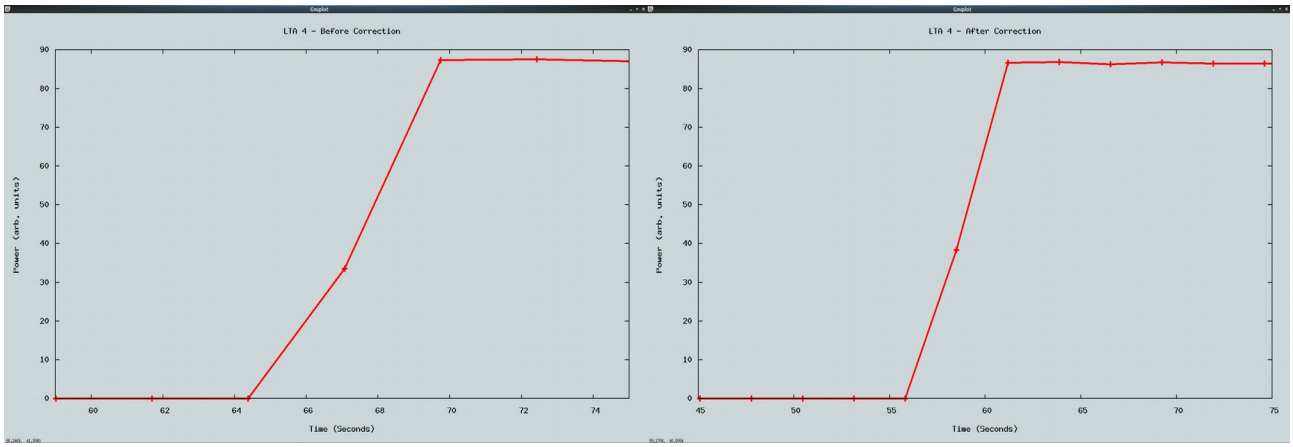

Fig. 8 **: Plot showing timeseries of a channel from visibility data for LTA1 = 4. Left side – Before timestamp offset correction, Right side – Post timestamp offset correction. Timestamp offset before correction is 13 LTA1 buffers i.e; 8.72 seconds. "LTA" mentioned in the figure refers to integration during real-time computation i.e; LTA1.**

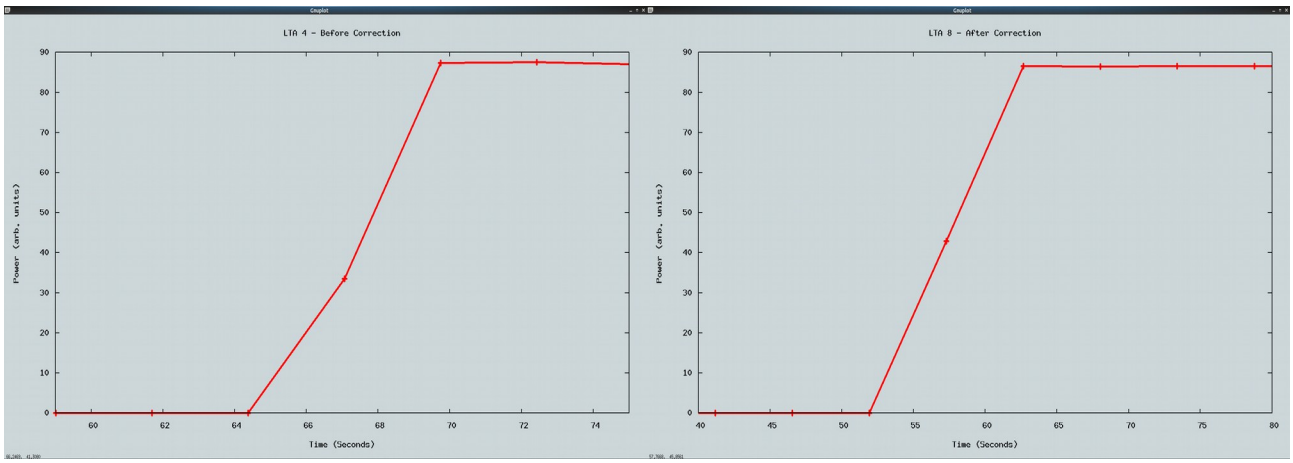

Fig. 9 **: Plot showing timeseries of a channel from visibility data for LTA1 = 8. Left side - Before timestamp offset correction, Right side - Post timestamp offset correction. Timestamp offset before correction is 29 LTA1 buffers i.e; 19.46 seconds. "LTA" mentioned in the figure refers to integration during real-time computation i.e.; LTA1.**

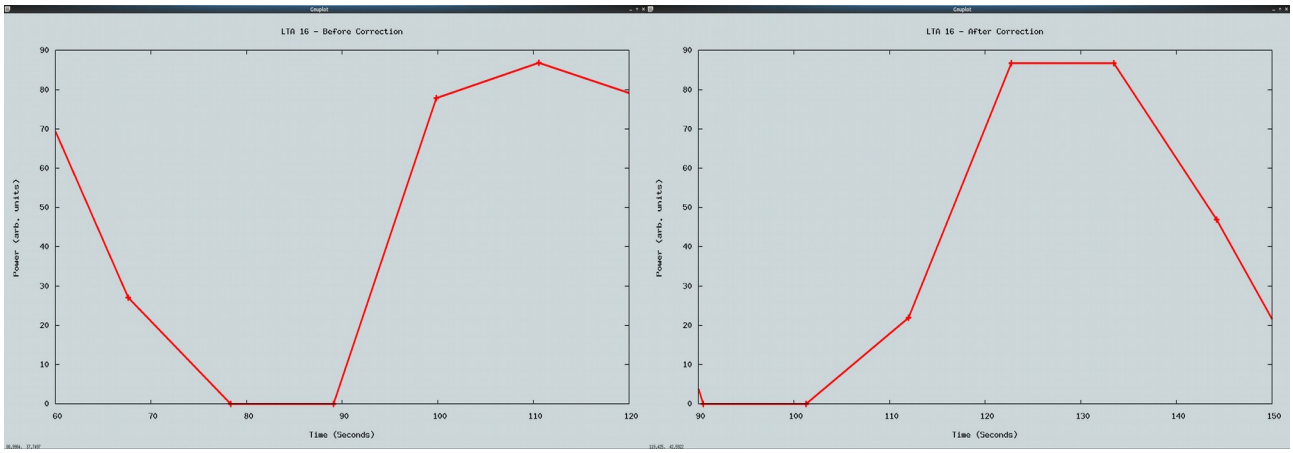

Fig. 10 **: Plot showing timeseries of a channel from visibility data for LTA1 = 16. Left side – Before timestamp offset correction, Right side – Post timestamp offset correction. Timestamp offset before correction is 61 LTA1 buffers i.e; 40.93 seconds. "LTA" mentioned in the figure refers to integration during real-time computation i.e; LTA1.**

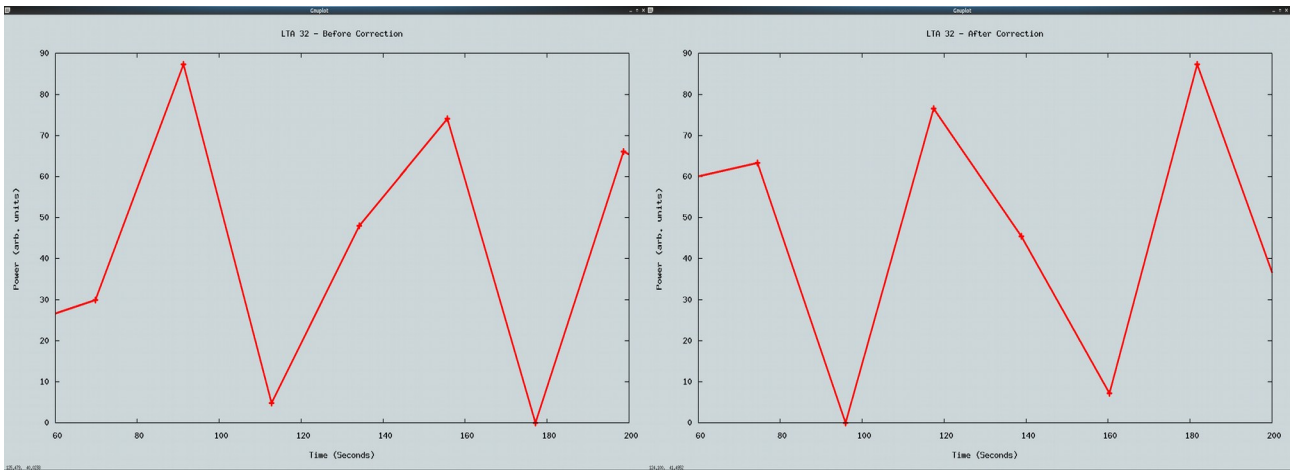

Fig. 11 **: Plot showing timeseries of a channel from visibility data for LTA1 = 32. Left side – Before timestamp offset correction, Right side – Post timestamp offset correction. Timestamp offset before correction is 125 LTA1 buffers i.e; 83.88 seconds. "LTA" mentioned in the figure refers to integration during real-time computation i.e; LTA1.**

9. Correction of timestamp offset for beam data from 14/05/2020

For eliminating the timestamp offset as explained in section 6 for observation epochs prior to 14/05/2020 the following changes were implemented in the software. For beam data, there is no integration and every buffer's beam data is sent to shared memory after GPU processing and MPI collection of beam data from all compute nodes. Hence, the pipeline delay with respect to timestamp calculated is three buffers i.e; one for timestamp calculation (calculated in advance), one for GPU processing and one for MPI collection of results. Hence, when the beam data is written to shared memory the timestamp is picked from the old timestamp buffer addressed at (run count – 3) % 4.

10. Experimental verification of timestamp offset in beam data

For this experiment, input used was broadband noise modulated by GPS PPM. PPM modulated noise was fed as input to GAB RF input of a polarization of an antenna (C03 pol1). GAB LO was set to 250 MHz, Band-3 RF bandpass filter was selected, 200 MHz Baseband Low pass filter was selected. The noise power level was adjusted such that ON pulse and OFF pulse noise is clearly visible. GWB settings are 200 MHz bandwidth, 1024 spectral channels, total intensity mode, beam integration – 4 FFTs (20.48 microseconds). In the GAC configuration, only the input where PPM modulated noise was connected was selected (C03 pol1 here). Beam data was recorded for few minutes.

A program was written in C-language, to extract a single channel as a time series from the beam data and written to a file. Also, along with the time series, timestamp was extrapolated from the start time of recording obtained from the header file and written to a file. For the extrapolation of timestamp, only the seconds and nanoseconds were considered and extrapolated. So, in the case when the timestamps are corrected (no offset), the ON pulse should start at multiple of 60 seconds (e.g. : 60, 120, 180 ----) and OFF pulse should start at multiple of 30 seconds (e.g. : 30,90,120,150 ----). Plots attached above shows the timeseries of a channel extracted from CDP data, IA beam data and PA beam data.

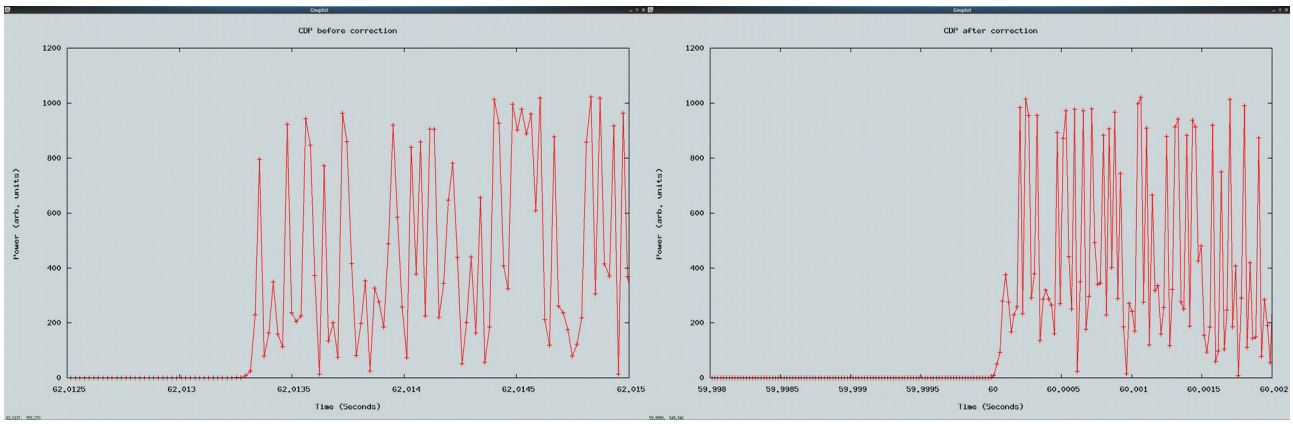

Fig. 12 **: Plot showing timeseries of a channel from CD pipeline data. Left side – Before timestamp offset correction, Right side – Post timestamp offset correction**.

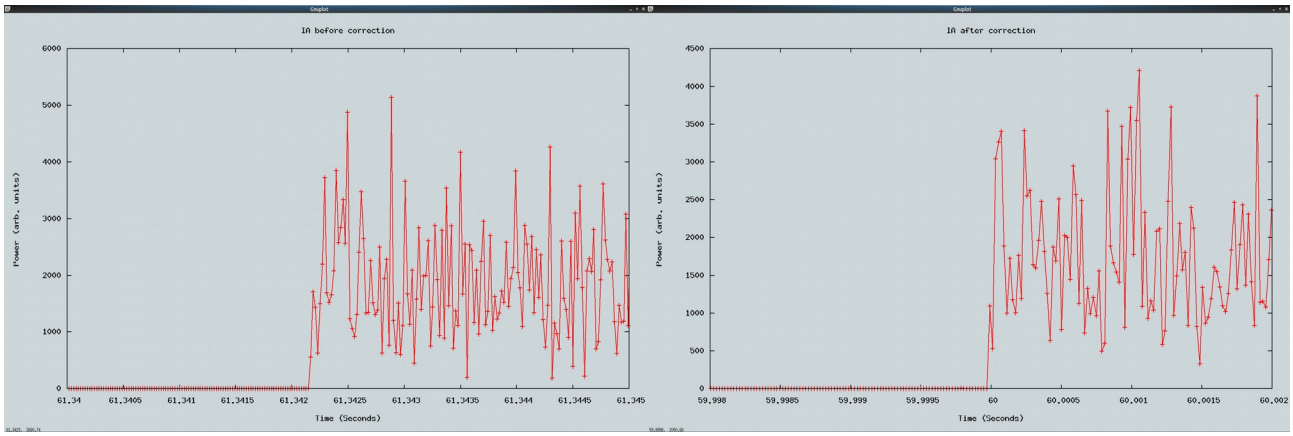

Fig. 13 **: Plot showing timeseries of a channel from IA beam data. Left side – Before timestamp offset correction, Right side – Post timestamp offset correction.**

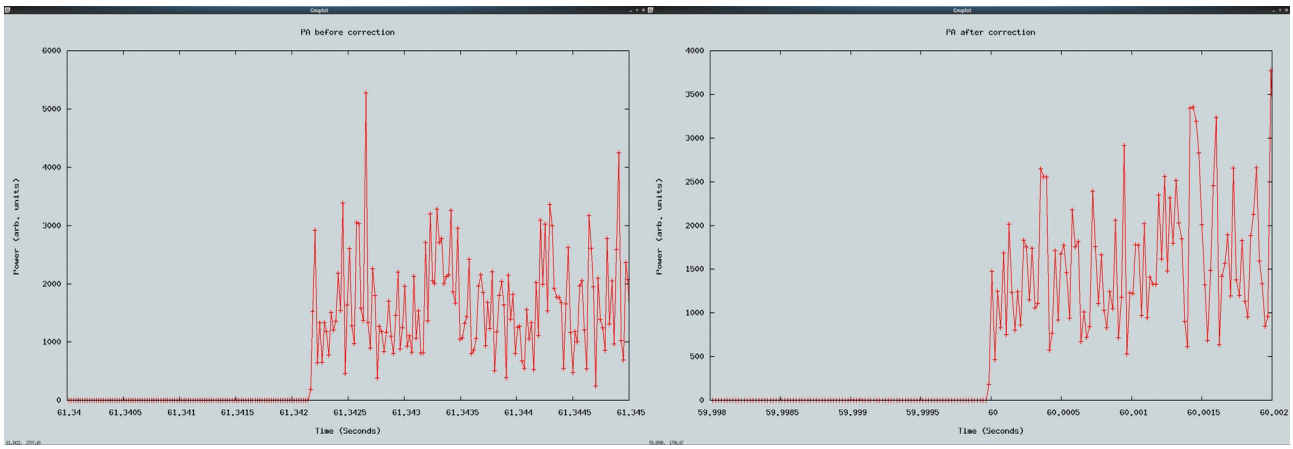

Fig. 14 **: Plot showing timeseries of a channel from PA beam data. Left side – Before timestamp offset correction, Right side – Post timestamp offset correction.**

11. Timestamp offset from 14/08/2015 to 09/09/2016

On 14/08/2015, a 16 - antenna version of GWB was released for internal users. That version had timestamp offsets which were partly corrected on 09/09/2016. The amount of offsets varied for interferometry depending on the integration (LTA1) and the amount of offset was constant for beam-former. That version had provision for only two beams each of which could be configured as either IA beam or PA beam.

In that version, triggering happened at negative edge of the GPS PPS pulse but start time was taken as the start of second which resulted in 0.5 seconds offset i.e; timestamp taken is advanced by 0.5 seconds than the actual time. The difference between the positive and the negative edge of the GPS was smaller than 0.5 seconds by an amount of $\Delta t^{i}_{GPS\text{-}PPS}$. Additionally, this edge had a larger jitter and both these are given in *Table 5*. For calculation of timestamp offset $\Delta t^{v}_{real\text{-}time}$ in *Table 3* and $\Delta t^{b}_{real\text{-}time}$ in *Table 4* mean offset of GPS jitter from *Table 5* has been considered. The mean offset of GPS jitter will have an error of 0.0566008 μs (*Table 5*)**.** For the integer delay correction, the sampled data stream (for all antennas) is block shifted by 57344 bytes (143.36 microseconds) in order to allow for positive and negative delay corrections with respect to a reference location in the array. This block shift was not accounted which resulted in 143.36 microseconds of offset.

Because of bug in the code, in the interferometry mode timestamp was delayed by LTA1 number of buffers. So, the accumulated timestamp offset was delay of ((LTA1 x 0.67108844) – (0.5 + $\Delta t^{i}_{GPS\text{-}PPS}$) – 0.00014336) seconds (*Table 3*). For the beam-former mode, irrespective of the either IA beam or PA beam timestamp offset was delay of 0.17096239 seconds (*Table 4*).

Table 3 : Timestamp offsets for interferometry mode ($\Delta t^{v}_{real\text{-}time}$) from **14/08/2015 to 09/09/2016**

LTA1	**Timestamp offset ($\Delta t^{v}_{real\text{-}time}$) in seconds for 200 MHz/400 MHz bandwidth**	**Timestamp offset ($\Delta t^{v}_{real\text{-}time}$) in seconds for 100 MHz or lower bandwidth**
1	0.17096239	0.84190767
2	0.84205103	2.18465839
4	2.18422831	4.86901295
8	4.86858287	10.23772207
16	10.23729199	20.97514031
32	20.97471023	42.44997679

Table 4 : Timestamp offsets for beam-former mode ($\Delta t^{b}_{real\text{-}time}$) from **14/08/2015 to 09/09/2016**

Beam	**Timestamp offset ($\Delta t^{b}_{real\text{-}time}$) in seconds for 200 MHz/400 MHz bandwidth**	**Timestamp offset ($\Delta t^{b}_{real\text{-}time}$) in seconds for 100 MHz or lower bandwidth**
IA/PA	0.17096239	0.84190767

Table 5 : Timestamp offsets ($\Delta t^{j}_{GPS\text{-}PPS}$) and jitter for negative GPS edge which was in use from **14/08/2015 to 09/09/2016**

Unit	**Mean offset ($\Delta t^{j}_{GPS\text{-}PPS}$) in µs**	**Median offset ($\Delta t^{j}_{GPS\text{-}PPS}$) in µs**	**Jitter (µs)**
GPS1 (+ve) vs GPS2 (-ve)	-17.115173	-17.118	0.0566008

12. Timestamp offset from 09/09/2016 to 04/07/2017

On 09/09/2016, offsets because of negative GPS PPS pulse triggering and integer delay were removed. But because of bug in the code time offsets changed to delay of ((LTA1 + 1) x 0.67108864) seconds for the interferometry mode (*Table 6*) and delay of 1.34217728 seconds for the beam-former mode.

Table 6 : Timestamp offsets for interferometry mode ($\Delta t^{v}_{real\text{-}time}$) from **09/09/2016 to 04/07/2017**

LTA1	**Timestamp offset ($\Delta t^{v}_{real\text{-}time}$) in seconds for 200 MHz/400 MHz bandwidth**	**Timestamp offset ($\Delta t^{v}_{real\text{-}time}$) in seconds for 100 MHz or lower bandwidth**
1	1.34217728	2.68435456
2	2.01326592	4.02653184
4	3.3554432	6.7108864
8	6.03979776	12.07959552
16	11.40850688	22.81701376
32	22.14592512	44.29185024

Table 7 : Timestamp offsets for beam-former mode ($\Delta t^{b}_{real\text{-}time}$) from **09/09/2016 to 04/07/2017**

Beam	**Timestamp offset ($\Delta t^{b}_{real\text{-}time}$) in seconds for 200 MHz/400 MHz bandwidth**	**Timestamp offset ($\Delta t^{b}_{real\text{-}time}$) in seconds for 100 MHz or lower bandwidth**
IA/PA	1.34217728	2.68435456

13. Timestamp offset in FITS conversion

Once GWB visibility data is tagged with time, there is no further timestamp offset introduced in record integration. GWB for visibility data always tags timestamp which is beginning of the buffers integrated in one. Further in *record* there is a provision to integrate visibility in which it will always attach timestamp of first visibility record of those are integrated in one. During conversion of lta data to fits, software *gvfits* gives various options for user either to move the visibility record timestamp to middle of integration or to leave untouched or to add an arbitrary offset to timestamp. **By default, *listscan/gvfits* choose option to move timestamp to the middle of integration.** However, we noted a bug in *gvfits* older versions i.e. version 2.03 and earlier, that it adds some offset to visibility timestamp in addition to moving it to centre of integration. In moving timestamp to the centre of integration block, timestamp recorded with visibility data is added with 'timestamp_offset' where 'timestamp_offset' is needed (0.5 x LTA1 x LTA2 x STA), where STA=0.67108864 second for 200 MHz & 400 MHz bandwidth mode and STA=1.34217728 second for 100 MHz and decimated mode. However, in *gvfits* version 2.03 and earlier the 'timestamp_offset' is calculated as (0.5 x ((LTA1 x LTA2) – 16) x STA), thereby introducing an extra offset to actual timestamp. **This problem is corrected in future versions of *listscan/gvfits, i*.e. version 2.04 and later, released for GTAC cycle40 (version 2.04 released on 11th May 2021).** FITS data carry information in history table which version of *gvfits* is used for the conversion from GMRT LTA data to FITS data. *Table 9* and *Table 10* gives overview of few combinations of LTA commonly used, the time offset correction needed and the correction applied for 200 MHz/400 MHz and 100 MHz bandwidth respectively by *gvfits* version 2.03 and earlier. By looking at *Table 8*, we note that there is only one correction ($\Delta t_{offline}$) needed to offset for given bandwidth. Following correction can be applied to correct recorded time to get accurate time

$$\text{timestamp\_corrected} = \text{timestamp\_recorded} + \Delta t_{offline}$$

where $\Delta t_{offline}$ = 5.36870912 sec for 200 MHz/400 MHz bandwidth and $\Delta t_{offline}$ = 10.73741824 sec for 100 MHz and lower bandwidth modes.

Table 8 : Timestamp offsets for FITS conversion **version 2.03 and earlier**

Timestamp offset ($\Delta t_{offline}$) in seconds for 200 MHz/400 MHz bandwidth	**Timestamp offset ($\Delta t_{offline}$) in seconds for 100 MHz or lower bandwidth**
5.36870912	10.73741824

Table 9 : Time offsets introduced and the correct timestamp with respect to timestamp recorded with visibility data for various combinations during FITS conversion (for version 2.03 and earlier) for 200 MHz/400 MHz bandwidth

Integration (sec)	**LTA1 x LTA2**	**Correction needed**	**Correction applied**
2.684352	4	1.342176	-4.026528
5.368704	8	2.684352	-2.684352
10.737408	16	5.368704	0.000000
21.474815	32	10.737408	5.368704
42.949631	64	21.474816	16.106112

Table 10 : Time offsets introduced and the correct timestamp with respect to timestamp recorded with visibility data for various combinations during FITS conversion (for version 2.03 and earlier) for 100 MHz or lower bandwidth

Integration (sec)	**LTA1 x LTA2**	**Correction needed**	**Correction applied**
5.368704	4	2.684354	-8.053062
10.737408	8	5.368708	-5.368708
21.474815	16	10.737416	0.000000
42.949631	32	21.474832	10.737416
85.899262	64	42.949664	32.212248

14. Conclusion

For observations such as monitoring of pulsar time of arrival it is necessary to timestamp data consistently across epochs of observations. This constraint demands that timestamps are generated identically at all epochs. Since the difference between the recorded timestamp and the actual timestamp is zero since 14/05/2020 we are

considering this epoch to be the reference epoch. And to be consistent with this reference epoch all the timestamps for all the data prior to 14/05/2020 needs to be corrected using the following prescription. And the prescription is

timestamp actual = timestamp recorded – $\Delta t^{v}_{real\text{-}time}$ + $\Delta t_{offline}$ **--- for visibility data**

timestamp actual = timestamp recorded – $\Delta t^{b}_{real\text{-}time}$ **--- for beam data**

where *timestamp actual* is the actual timestamp at the time the data was acquired and *timestamp recorded* is the timestamp recorded with the data and $\Delta t^{v}_{real\text{-}time}$ is the difference or offset between these two for visibility data for that particular epoch as given in *Table 1*, *Table 3* and *Table 6* and $\Delta t^{b}_{real\text{-}time}$ is the difference or offset between these two for beam data for that particular epoch as given in *Table 2*, *Table 4* and *Table 7*. **For visibility and beam data recorded after 14/05/2020, $\Delta t^{v}_{real\text{-}time}$ and $\Delta t^{b}_{real\text{-}time}$ are zero.** $\Delta t_{offline}$ is an offset introduced in FITS conversion using version 2.03 and earlier as given in *Table 8*. **In the later versions of *gvfits*, i.e. version 2.04 and later, $\Delta t_{offline}$ is zero.** So user needs to apply this expression given above to make the time referencing of his/her data consistent for all epochs.

Note : Polyphase Filtering (PFB) mode was released on 21/04/2021 at the start of GTAC cycle 40. In this mode, a small offset in time was found whose exact value is given in section 7 of the document given in this link[3]. Users who used PFB mode for their regular observations can contact the operations group for further details.

For ease of finding the offsets to be applied, the tables 1 to 10 are again given below in the appendix.

Appendix 1 : Offsets for visibility data ($\Delta t^{v}_{real\text{-}time}$)

For visibility data acquired **after 14/05/2020**, timestamp recorded reflects the actual timestamp i.e; **timestamp ($\Delta t^{v}_{real\text{-}time}$) offset is zero**.

For visibility data acquired from **04/07/2017 to 14/05/2020**

LTA1	Timestamp offset ($\Delta t^{v}_{real\text{-}time}$) in seconds for 200 MHz/400 MHz Bandwidth	Timestamp offset ($\Delta t^{v}_{real\text{-}time}$) in seconds for 100 MHz or lower Bandwidth
1	0.67108864	1.34217728
2	3.3554432	6.7108864

3 https://www.gmrt.ncra.tifr.res.in/subsys/digital/DigitalBackend/target_files/GWB/DOC/PFB_in_GWB_version2.pdf

4	8.72415232	17.44830464
8	19.46157056	38.92314112
16	40.93640704	81.87281408
32	83.88608	167.77216

For visibility data acquired from **09/09/2016 to 04/07/2017**

LTA1	**Timestamp offset ($\Delta t^{v}_{real\text{-}time}$) in seconds for 200 MHz/400 MHz bandwidth**	**Timestamp offset ($\Delta t^{v}_{real\text{-}time}$) in seconds for 100 MHz or lower bandwidth**
1	1.34217728	2.68435456
2	2.01326592	4.02653184
4	3.3554432	6.7108864
8	6.03979776	12.07959552
16	11.40850688	22.81701376
32	22.14592512	44.29185024

For visibility data acquired from **14/08/2015 to 09/09/2016**

LTA1	**Timestamp offset ($\Delta t^{v}_{real\text{-}time}$) in seconds for 200 MHz/400 MHz bandwidth**	**Timestamp offset ($\Delta t^{v}_{real\text{-}time}$) in seconds for 100 MHz or lower bandwidth**
1	0.17096239	0.84190767
2	0.84205103	2.18465839
4	2.18422831	4.86901295
8	4.86858287	10.23772207
16	10.23729199	20.97514031
32	20.97471023	42.44997679

Appendix 2 : Offsets for beam data ($\Delta t^{b}_{real\text{-}time}$)

For beam data acquired **after 14/05/2020**, timestamp recorded reflects the actual timestamp i.e; **timestamp ($\Delta t^{b}_{real\text{-}time}$) offset is zero**.

For beam data acquired from **04/07/2017 to 14/05/2020**

Beam	**Timestamp offset ($\Delta t^{b}_{real\text{-}time}$) in seconds for 200 MHz/400 MHz bandwidth**	**Timestamp offset ($\Delta t^{b}_{real\text{-}time}$) in seconds for 100 MHz or lower bandwidth**
IA/PA	1.34217728	2.68435456
Voltage (CDP)	2.01326592	4.02653184

For beam data acquired from **09/09/2016 to 04/07/2017**

Beam	**Timestamp offset ($\Delta t^{b}_{real\text{-}time}$) in seconds for 200 MHz/400 MHz bandwidth**	**Timestamp offset ($\Delta t^{b}_{real\text{-}time}$) in seconds for 100 MHz or lower bandwidth**
IA/PA	1.34217728	2.68435456
Voltage (CDP)	Mode not available	Mode not available

For beam data acquired from **14/08/2015 to 09/09/2016**

Beam	**Timestamp offset ($\Delta t^{b}_{real\text{-}time}$) in seconds for 200 MHz/400 MHz bandwidth**	**Timestamp offset ($\Delta t^{b}_{real\text{-}time}$) in seconds for 100 MHz or lower bandwidth**
IA/PA	0.17096239	0.84190767
Voltage (CDP)	Mode not available	Mode not available

Appendix 3 : Timestamp offset during FITS conversion ($\Delta t_{offline}$)

For FITS conversion ***gvfits*** **version 2.04 and later**, **timestamp offset ($\Delta t_{offline}$) is zero.**

Timestamp offsets for FITS conversion ***gvfits*** **version 2.03 and earlier**

Timestamp offset ($\Delta t_{offline}$) in seconds for 200 MHz/400 MHz bandwidth	**Timestamp offset ($\Delta t_{offline}$) in seconds for 100 MHz or lower bandwidth**
5.36870912	10.73741824

Appendix 4 : Timestamp offset ($\Delta t^{j}_{GPS\text{-}PPS}$) and jitter for negative GPS edge which was in use from 14/08/2015 to 09/09/2016

Unit	**Mean offset ($\Delta t^{j}_{GPS\text{-}PPS}$) in µs**	**Median offset ($\Delta t^{j}_{GPS\text{-}PPS}$) in µs**	**Jitter (µs)**
GPS1 (+ve) vs GPS2 (-ve)	-17.115173	-17.118	0.0566008