ARL-CR-0778 ● SEP 2015

US Army Research Laboratory

High-Bandwidth Tactical-Network Data Analysis in a High-Performance-Computing (HPC) Environment: Time Tagging the Data

prepared by Brian Panneton
Technical and Project Engineering, LLC
Alexandria, VA

Jim Adametz and Jordan Franssen
QED Systems, LLC
Aberdeen, MD

under contract W91CRB-11-D-0007

ARL-CR-0778 ● SEP 2015

US Army Research Laboratory

High-Bandwidth Tactical-Network Data Analysis in a High-Performance-Computing (HPC) Environment: Time Tagging the Data

prepared by Brian Panneton
Technical and Project Engineering, LLC
Alexandria, VA

James Adametz and Jordan Franssen
QED Systems, LLC
Aberdeen, MD

under contract W91CRB-11-D-0007

REPORT DOCUMENTATION PAGE			*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)* September 2015	**2. REPORT TYPE** Final	**3. DATES COVERED (From - To)** July 2012–December 2014
4. TITLE AND SUBTITLE High-Bandwidth Tactical-Network Data Analysis in a High-Performance-Computing (HPC) Environment: Time Tagging the Data		**5a. CONTRACT NUMBER** W91CRB-11-D-0007
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Brian Panneton, Jim Adametz, and Jordan Franssen		**5d. PROJECT NUMBER**
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Technical and Project Engineering, LLC, Alexandria, VA QED Systems, LLC, Aberdeen, MD		**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) US Army Research Laboratory ATTN: RDRL-CIH-C Aberdeen Proving Ground, MD 21005		**10. SPONSOR/MONITOR'S ACRONYM(S)**
		11. SPONSOR/MONITOR'S REPORT NUMBER(S) ARL-CR-0778

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT
The analysis of data from radio-based network testing typically requires that the latency of data leaving one node and arriving at a destination be determined. To properly calculate latency, transmit and receive times of network packets must be measured, and those times must be synchronized to a common source. The Global Positioning System (GPS) is a readily available time source that can be made available at each of the distributed nodes in a network. However, applying GPS-synchronized time tags to recorded network packets has proven to be a challenge due to microsecond drift, data stream buffering, and other technical issues. The process described in this report made use of posttest processing techniques to provide packet-level time tagging with an accuracy close to 3 µs relative to Coordinated Universal Time, with a resolution of 1 µs. This enabled analysis of high-speed wireless networks where a medium-sized packet can be delivered in under 1 ms.

15. SUBJECT TERMS
tactical networks, data reduction, high-performance computing, data analysis, big data

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
					Kenneth Renard
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER (Include area code)**
Unclassified	Unclassified	Unclassified	UU	28	410-278-4678

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

1. Introduction

When analyzing network data captured at different endpoints of a radio-based network, one of the crucial metrics used to evaluate network performance is latency. To measure latency, the time tag of a packet being sent is recorded and compared to the time tag of the copy of the same packet received at the endpoint. For this to work properly, each of the packet-observing data recorders must be synchronized to a common time source. The Global Positioning System (GPS) is an ideal time source for such synchronization, as it provides a highly precise, globally accessible means to synchronize devices.[1] This type of testing routinely requires that over 1 billion packets be examined and time-tagged within an 8-h set of test records. The process described herein made use of posttest processing techniques to provide packet-level time tagging with an accuracy close to 3 µs relative to Coordinated Universal Time (UTC), with a resolution of 1 µs. This enabled analysis of high-speed wireless networks where a medium-sized packet can be delivered in under 1 ms.[2] This report describes how the US Army Research Laboratory (ARL) collaborated with the Analysis Team at the Army Test and Evaluation Command (ATEC), Aberdeen Test Center (ATC), to build a system that would leverage the capabilities of a high-performance-computing environment to reliably time-tag network packet data recorded in a High-Bandwidth Tactical Network.

2. Data Collection Overview

ATC's Advanced Distributed Modular Acquisition System (ADMAS) is a data collection device that records metadata and raw binary data in Binary Large Object (BLOb) files.[3] The recorded data are encoded into structures called Data Cuts (cuts). A cut is a slice of data wrapped in a predefined header. In general, a cut header contains the type of cut that follows the length of the recorded data and a microsecond timer value.

The timer, or ticker, is based on an ADMAS' internal clock, which increments for each microsecond that passes. This clock stores the current local time as an unsigned integer with range (0, 2^{32}). The ticker initializes to zero when the device is powered on and then resets back to zero every 2^{32} µs. Thus, a reset, known as a "rollover", occurs approximately every 1 h and 11 min. This means that in a network test lasting several hours, the ticker will roll over multiple times. Figure 1 shows a sample of the file sequence numbers, tickers, and rollover values from a long-running test.

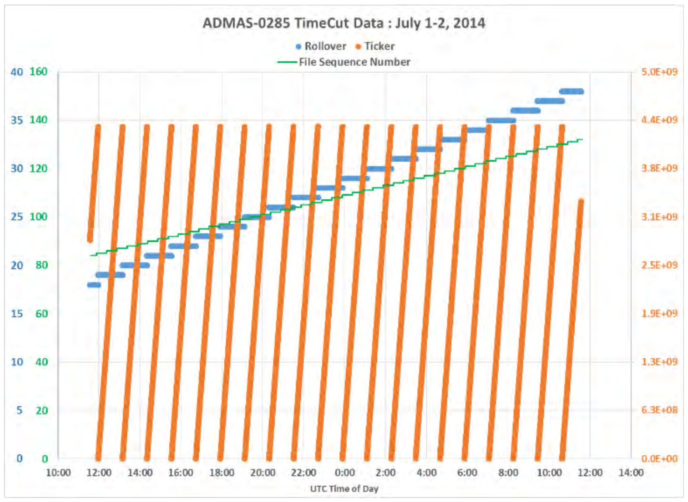


Fig. 1 Sample ADMAS time cut data

ADMAS tickers are not directly synchronized to GPS and are affected by environmental conditions. Therefore, changes in the behavior of the ticker must be taken into account. For example, with a change in temperature, the microsecond ticker can speed up or slow down, causing drift relative to GPS time. This clock drift, along with the bias between GPS time and ticker time, can be accounted for by applying corrections extracted from a Kalman filter–based[4] "clock model" of the relationship between ticker and GPS time. Aside from environmental difficulties, time correction also falls prey to data gaps and hardware malfunctions. These measurement and recording-type errors need to be addressed before the clock model can be used for correction.

3. Time Cuts

The ADMAS is capable of recording a wide variety of cut types. During tactical network testing, GPS cuts and network cuts provide the raw geospatial and network traffic data needed for analysis. These cuts, however, only contain the ADMAS' ticker value and do not include the device's ticker rollover count. To account for rollovers, time cuts are required.

Time Cuts contain a GPS time[5] matched with the ADMAS' 32-bit ticker value and 16-bit ticker rollover count. The ADMAS hardware uses field-programmable gate array (FPGA) circuitry to latch the ticker value when the GPS pulse-per-second signal is fired. The FPGA processing then captures the next GPS serial message that includes the time when the pulse fired, and the ticker and GPS times are recorded in the time cut. With the inclusion of the ticker rollover count, this data allows a clock model to be defined between the ticker values and GPS time.

4. ADMAS Stream Buffering

One stream buffer for each cut type is used in writing cuts to BLOb files. While this maintains ticker order within each cut type, the order for cuts of different types is not guaranteed. Figure 2 depicts how time, network, and GPS cuts might be ordered in a BLOb file. In this figure, cuts are represented as small rectangles, the border color represents the type of cut, and the fill color represents the ticker value of the cut. We can see that the order of tickers in the BLOb file is not sequential when viewed without regard to cut type.

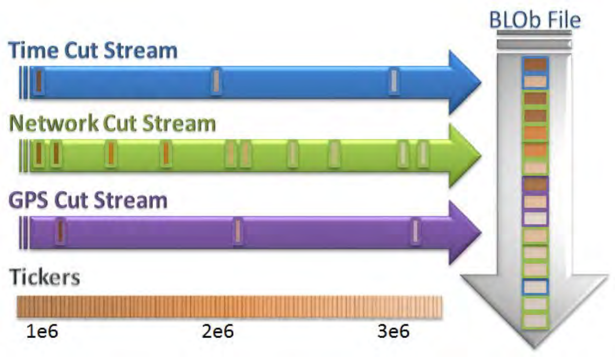


Fig. 2 How cut tickers get out of order in BLObs

In addition to the stream buffers' effect on cut ordering within a file, buffering may also result in BLOb files containing cuts recorded before the file's creation. An ADMAS will close its current BLOb file and open a new one when either 1,800 s have passed since the file was opened or the file size has exceeded 500 MB. A buffer may still contain data when the file is closed, in which case the cuts will be written to the new file. Each file is given an incremental sequence number that can be used in detecting these cases.

Because of the effects of buffering, all cuts must be separated by type before having their ticker values converted to the GPS time domain. This is required to ensure that the proper rollover value is associated with each 32-bit ticker value.

5. High-Performance-Computer (HPC) Processing

The custom data reduction software used to organize and process tens of thousands of files containing billions of data cuts runs on high-performance computers (HPCs) in a scalable, distributed fashion. This means that data are spread across hundreds of compute cores that do not necessarily share memory. The reduction software uses an approach similar to map-reduce to collect all the information needed to produce a clock model.

The process of converting all of the BLOb data cut tickers into GPS-corrected time within the HPC environment is diagramed in Fig. 3.

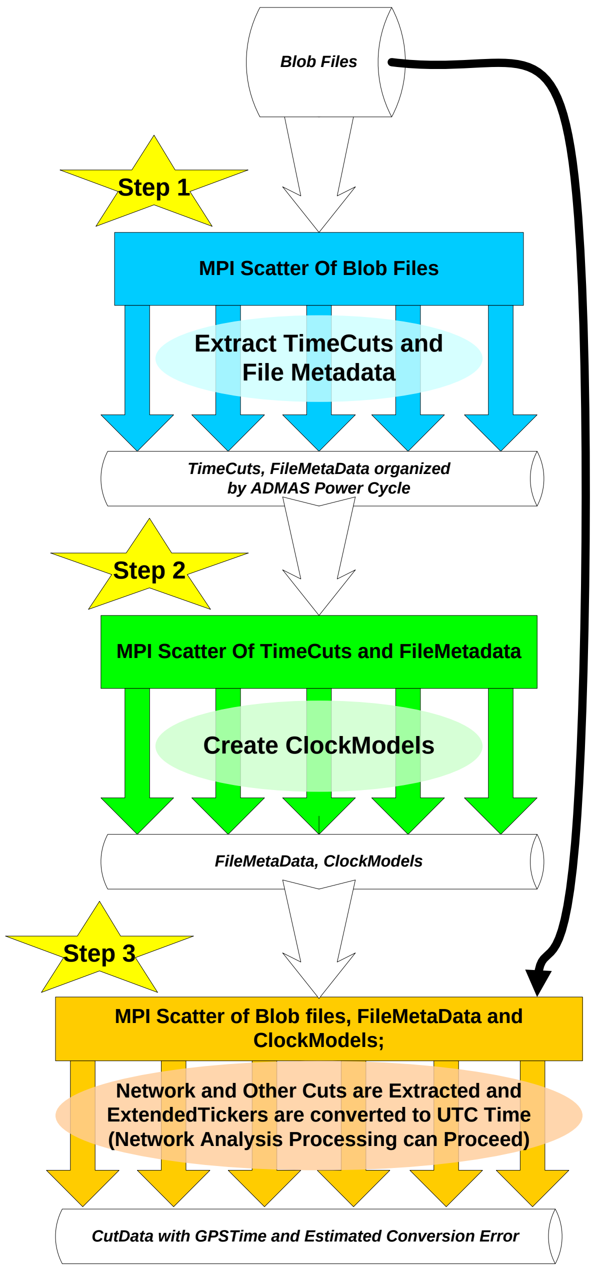

Blob Files
Step 1
MPI Scatter Of Blob Files
Extract TimeCuts and File Metadata
TimeCuts, FileMetaData organized by ADMAS Power Cycle
Step 2
MPI Scatter Of TimeCuts and FileMetadata
Create ClockModels
FileMetaData, ClockModels
Step 3
MPI Scatter of Blob files, FileMetaData and ClockModels;
Network and Other Cuts are Extracted and ExtendedTickers are converted to UTC Time (Network Analysis Processing can Proceed)
CutData with GPSTime and Estimated Conversion Error

Fig. 3 Time conversion process

Step 1 distributes the individual BLOb files across message-passing interface (MPI) worker processes. The purpose of this step is to extract all time cuts and metadata from each file, which includes the following:

- Number of ticker rollovers at the start of the file
- Indicator if the file contains one or more ticker rollovers for each cut type

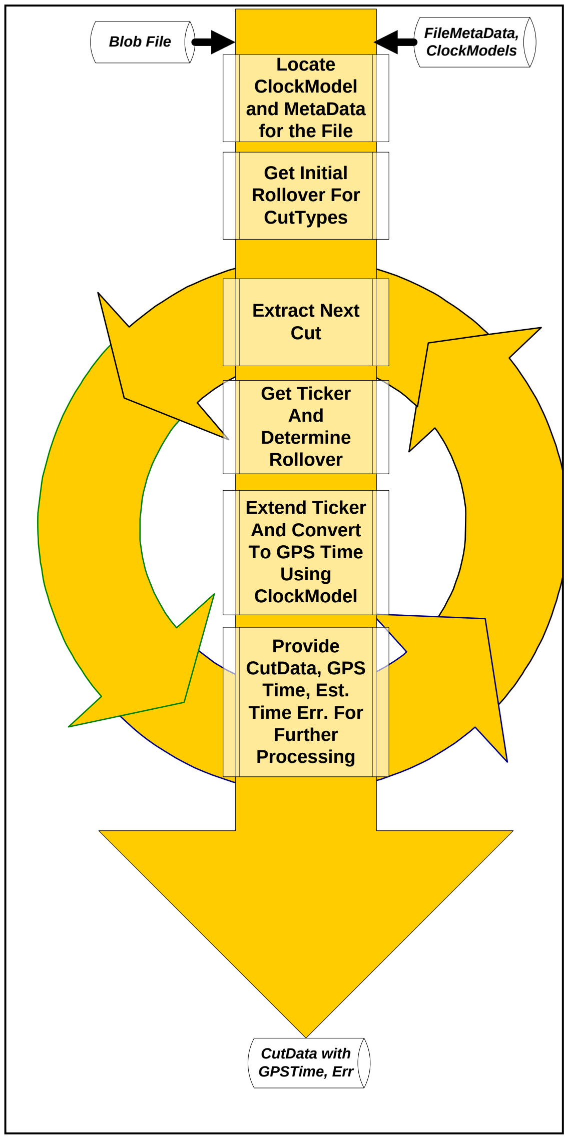
- File name
- File sequence number since the last startup
- Types of data cuts in the file

The time cuts and metadata are serialized back to a disk for further processing in step 2.

Step 2 distributes the time cuts and metadata across the MPI worker processes; the distribution is segmented by data collector device and a “boot sequence” number. The boot sequence is derived from observations of the file sequence number present in each BLOb file. When the sequence goes backwards as time moves forwards, a reboot of the ADMAS has occurred. The reboot causes the relationship between GPS time and ticker time to be reset. This forces the creation of a new clock model for the time period following each reboot.

During step 2, the clock models are fed the time cuts in time order. Once the clock models are done processing all of the time cuts, the clock model runtime parameters are serialized to a disk for the next step in processing.

Step 3 redistributes the original BLOb files, file metadata, and the clock models across the MPI worker processes; the distribution is segmented by BLOb file. During this iterative step (Fig. 4), each data cut is extracted from a given BLOb file, and the ticker is converted to a GPS time using the clock model associated with the file. The data cut and the clock model–derived GPS time with estimated time conversion error are provided for further processing.



Blob File
FileMetaData, ClockModels
Locate ClockModel and MetaData for the File
Get Initial Rollover For CutTypes
Extract Next Cut
Get Ticker And Determine Rollover
Extend Ticker And Convert To GPS Time Using ClockModel
Provide CutData, GPS Time, Est. Time Err. For Further Processing
CutData with GPSTime, Err

Fig. 4 Conversion process: step-3 details

6. Employing the Clock Model

Employing a clock model in step 3 starts with standardizing ticker times for cuts using the ticker rollover counts found in time cuts. The ticker rollover counts from time cuts are applied across the other cut types to convert all tickers to 48-bit unsigned integers. These 48-bit extended tickers define the cut order and can be correlated to epoch time using the clock model. The clock model details are discussed later in this report.

Before beginning step 3, one must have the rollover properly aligned with the data, otherwise the data for that particular file will be off by a factor of 2^{32} µs (~1 h, 11 m). It is also important to keep track of the rollover count per file because it will be needed when the files are redistributed for further processing. The HPC reduction software performs the actual time conversion when the file is reprocessed.

During step 2, processing of the cut data within each BLOb file must be done in time-order to generate the correct rollover information for each file before it is redistributed, otherwise the application of the clock model cannot be done on the fly. Part of this information includes having the correct rollover count for the start of the file. To determine the rollover count, the file metadata must be collected in one place. It then gets sorted by the file creation time. Once sorted, missing rollovers can be detected during time cut data gaps. The processing of the data is broken into 2 significant parts: rollover collection and rollover application.

7. Rollover Collection: Assumptions

The initial design goal for rollover collection was to identify ticker rollovers and calculate the extended tickers of cuts on a per-file basis. This method, however, made some reasonable assumptions.

The first assumption was that there is, at a maximum, one rollover per file. This assumption was based on the fact that the ticker value resets once every 1 h and 11 min, and a BLOb file spans roughly 30 min or less. With the ADMAS configuration intact, this was a safe assumption.

The next assumption was that a decrease in the ticker value for cuts of the same type must be caused by a ticker rollover. This assumption was based on the fact that tickers could only increment and cuts would be buffered in order.

While the second assumption is theoretically sound, anomalous data dubbed *rogue cuts* can cause false positives in detecting ticker rollovers. A rogue cut is produced when valid, but erroneous data are written into a well-structured cut.[6] In these cases, the ticker value is a random number, so it may appear that a rollover occurred when it did not. The algorithm to detect and reject rogue cuts is discussed in the Appendix.

8. Rollover Collection: Detecting Rollovers

In order to apply the clock model to each file independently, each file's starting rollover count needed to be determined. To accomplish this, the following metadata from each file was collected:

- The file creation time

- The file sequence number
- Whether or not a rollover was detected in the file
- The minimum and maximum time cuts

The file creation time and sequence number are included in the file metadata and are readily available. With files ordered by the creation times, sequence numbers can be compared to delineate power cycles in the ADMAS collector.

A power cycle is the period of time between device resets. Since sequence numbers are incremental, a file is considered to be the start of a power cycle when the sequence number is less than or equal to the sequence number of the previous file. Each power cycle resets the rollover count to zero and requires a separate clock model.

The rollover count increments at most once per file based on the previous assumptions, thus allowing per file rollover detection. By keeping track of the high-order bit of the ticker, we could detect a rollover once the value flips from one to zero, indicating the ticker value decreased. The change of the high-order bit represents 2^{31} µs, which is large enough (roughly 35 min) to avoid the time variability of the unordered ticker values. Because there is a separate buffer for each cut type, the ticker could appear to roll over multiple times—once for each cut type—but would only be recorded as one occurrence.

If a rollover had been detected in a file, the rollover values in the minimum and maximum time cuts could be used to determine the relative position of the rollover and, in turn, calculate the file's starting rollover count. The relative position of the ticker rollover falls into one of the following categories:

1. Before the minimum time cut
2. Between the minimum and maximum time cuts
3. After the maximum time cut
4. Unknown position

The rollover is determined to be in category 1 if the rollover count in the minimum time cut is greater than the maximum rollover in the previous file. In this case, the current file's starting rollover must be one less than the minimum time cut rollover count.

The rollover falls into category 2 if the minimum time cut rollover is one less than the maximum time cut rollover. In this case, the file's starting rollover count would be equal to the minimum time cut rollover count.

If the rollover does not fall into either category 1 or 2, then, by default, it is attributed to category 3 and must have occurred after the maximum time cut. As with category 2, the file's starting rollover count is equal to the minimum time cut rollover count.

Category 4 sums up all the cases that do not fit into the other categories. Primarily, rollovers detected in a file are considered in this category when there are no time cuts recorded. However, complications with cut buffering can also cause the position of the rollover to become uncertain.

9. Buffering Effect on Rollover Determination

The ADMAS hardware has a 128-KB buffer size limit that must be reached before the cuts get written to a file. If the cuts are small enough, then a large number of cuts can be stored in the buffer and not written until a new file has been opened. So far, this has only been witnessed with network cuts.

This creates a problem when the buffered cuts written to a new file contain a rollover. Since some different cut types from the same period of time will have been written to 2 different files, a rollover can be detected in both. This makes it appear that an extra rollover occurred and, in some cases, can even cause 2 rollovers to appear in a single file.

This case breaks a few of the assumptions required to accurately produce extended ticker values for each cut. While detecting 2 files in sequence both with rollovers quickly identifies this case, the interwoven nature of the cut types makes calculating extended tickers a difficult task.

10. Data Gaps

Another difficult issue to overcome is gaps in the data. Gaps can be attributed to missing files or the monitored radio losing GPS satellite connectivity. Incorrect configurations and faulty hardware may also cause missing data. When combined with the buffering issue, a situation exists where it is impossible to determine the ticker rollover count needed to calculate extended tickers.

11. Rollover Application

Dealing with buffering and data gaps with the initial approach resulted in a lot of uncertainty for the file's starting rollover count. Another approach can overcome the multiple rollovers and simplify the majority of the categorizing code from the previous approach. Instead of detecting rollovers for the entire file by examining high order bits of *any* 2 consecutive cuts, detection occurs per cut type stream (refer back to Fig. 2). By recording each ticker value that starts a new rollover on a per-cut-type basis, it is possible keep count of how many rollovers occurred in the file. The location of rollover occurrence within the file is no longer needed.

When calculating extended ticker values, we use the same algorithms to order the file metadata and handle power cycles. Starting with the first file, a simple rollover count is maintained per cut type. In addition, the first post-rollover ticker value is kept per rollover observation.

With valid time cuts within the file, the data gap problem is not very difficult. Even when time cuts are not in every file, there is a chance that the rollover can be guessed. This depends on the number of contiguous files missing time cuts and the sparseness of the other cut types. Without any time cuts, there are no reference points for predicting rollover counts.

12. Missing Rollovers

Missing rollovers are corrected by taking the difference between the rollover count for time cuts and the rollover count for the current cut type. This algorithm allows for overcorrection by marking the rollover count with a maximum of one higher than it actually is. This is a case caused by buffering, but it can easily be detected. The rollover guess is applied to the tickers of the last time cut and the first data cut of the current cut type. If the last time cut is less than the first current cut type, the rollover for the current cut type has been overcorrected. The detection is valid when the time cut stream has a rollover in the file and the current cut type stream does not. Subtracting one from the data cut's starting file rollover corrects this instance.

Missing rollovers also occur when the data are sparse. In this case, the rollover is truly missed. When the value of the ticker is on the upper half of the 32-bit value, it ends up being overcorrected by one rollover. Taking the distance between the minimum and maximum ticker of both the current cut type and the time cuts allows this case to be detected. If the difference of the current is less than 25% of the time cuts and it is on the high end of the 32 bits, it was overcorrected. Subtracting one rollover will correct this case.

13. Clock Model Application

Once the starting rollover count per cut type is determined correctly, the clock model can be employed to convert a ticker to GPS time. The input value for conversion by the clock model is a rollover-extended ticker value. A simple addition of the ticker and the left bit shift of rollover count by 32 bits creates the rollover extended ticker.

The clock model used is a second-order Kalman filter of the clock offset between ticker-time and GPS time. This model implements a tracking filter that uses as measurements the difference between local ticker time (extended ticker roughly converted to seconds) and GPS time, which provides a phase difference in seconds between the 2 clock sources (see Fig. 5 for an example). The filter maintains 3 state values (Fig. 6) for the model: Phase-Difference, Skew, and Aging-Noise. The relationship between these states is analogous to a single dimension Position, Velocity, Acceleration (PVA) model. Skew is the first derivative of Phase-Difference, and Aging-Noise is the second derivative. The design for the filter is documented in the case study of WIN-T IOTE ClockModel Issues.[7]

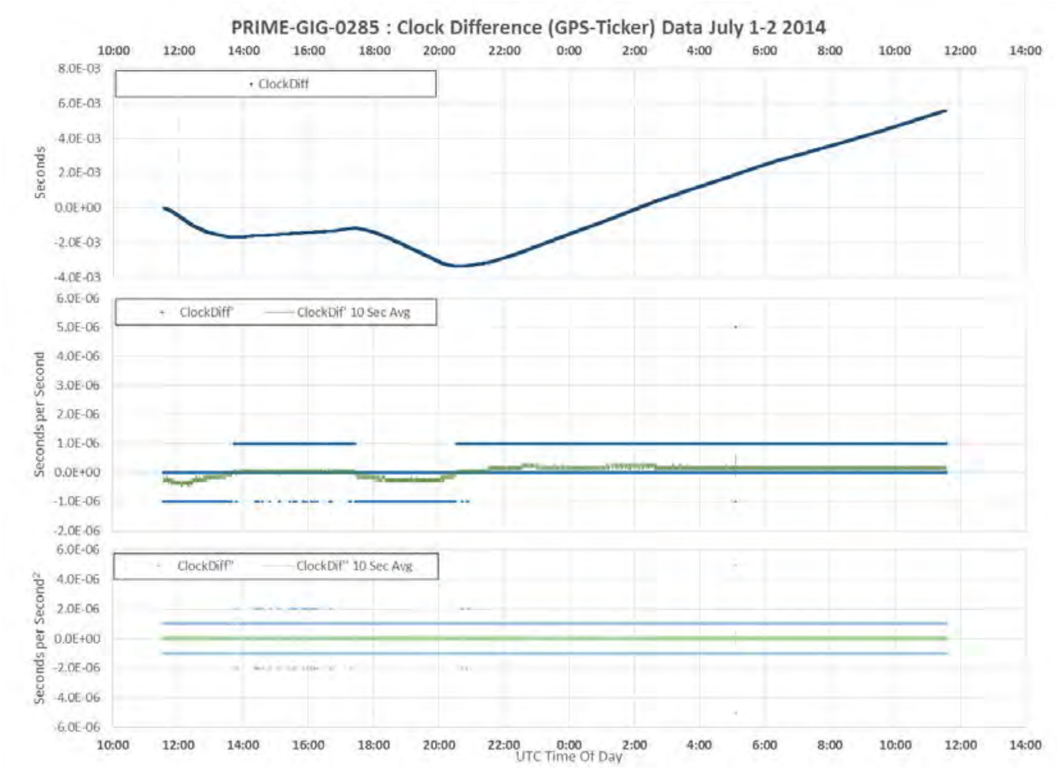


Fig. 5 Sample long-running ADMAS clock differences (3 clock model states)

$$\boldsymbol{ExtendedTicker = 2^{32} * Rollover + Ticker}$$

$$\begin{bmatrix} PhaseDiff_t \\ Skew_t \\ Aging_t \end{bmatrix} = \begin{bmatrix} PhaseDiff_{t-} \\ Skew_{t-} \\ Aging_{t-} \end{bmatrix} * \begin{bmatrix} 1 & \Delta t & \frac{\Delta t^2}{2} \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix}$$

Fig. 6 Clock model state update equation

The clock model maintains a covariance matrix (Fig. 7), which is used to generate an estimated error of the conversion from ticker time to GPS time. When the estimated error of a conversion exceeds a threshold, the data associated with that ticker will be marked for limited analytical use.

$$P_t = \begin{bmatrix} \partial PhaseDiff_t^{\,2} & \partial PhaseDiff_t * \partial Skew_t & \partial PhaseDiff_t * \partial Aging_t \\ \partial PhaseDiff_t * \partial Skew_t & \partial Skew_t^{\,2} & \partial Skew_t * \partial Aging_t \\ \partial PhaseDiff_t * \partial Aging_t & \partial Skew_t * \partial Aging_t & \partial Aging_t^{\,2} \end{bmatrix}$$

Fig. 7 Clock model covariance matrix

Prior to the ATC/ARL HPC efforts, the model employed for ATC network analysis was originally implemented in Java. The model design was ported over to Python and was enhanced to enable the extraction/distribution of the clock model states and covariance within the MPI environment. To accomplish this, a pass is made through the data files, and all time cuts are extracted from each file and fed to a separate clock model class instantiated for each ADMAS boot cycle (refer to step 2 in Fig. 3). The ordered time cuts are fed into each model, and the state/covariance data is extracted and stored at 30-s intervals[8] based on the GPS time.

Finally, the clock models are distributed with BLOb files, and data cuts are extracted from the BLObs. Each data cut ticker value is rollover extended, and the GPS conversion is applied. The data cut with its GPS time tag and estimated time conversion error is now ready for the complex task of communications processing, where accurate time tags are critical for generating latency and network load calculations.

14. Conclusion

ARL's Computational and Information Sciences Directorate, in collaboration with the Analysis Team at ATEC, ATC, has implemented a system that leverages the capabilities of an HPC environment to reliably time-tag network packet data recorded in a High-Bandwidth Tactical Network. Recent test events where over 1 TB of test data was collected in each 24-h period were successfully processed using this system.

15. References and Notes

1. Dana P, Penrod B. The role of GPS in precise time and frequency dissemination. Greenbelt (MD): National Aeronautics and Space Administration; 1990 [accessed 2015 Jan 27]. http://ilrs.gsfc.nasa.gov /docs/timing/gpsrole.pdf.

2. 700 bytes = 5,600 bits over a 50 Mbps wireless network => 112 µs.

3. Army Aberdeen Test Center (US). VISION BLOb description. Aberdeen Proving Ground (MD): Army Aberdeen Test Center (US), Instrumentation Development Division; 2014.

4. Kalman RE. A new approach to linear filtering and prediction problems. Transactions of the ASME Journal of Basic Engineering. 1960;82(D):35–45 [accessed 2015 Jan 13]. https://www.cs.unc.edu/~welch/kalman /media/pdf/Kalman1960.pdf.

5. ADMAS records GPS time (epoch time) as the number of whole seconds since January 1, 2000.

6. The reasons for the existence of these rogue cuts is still under investigation by the ADMAS development team. These represent a very small fraction of the total cuts recorded (on the order of 0.00001% of cuts recorded).

7. Adametz J, McGowan J. Case study of WIN-T IOTE ClockModel issues. Aberdeen Proving Ground (MD): Army Aberdeen Test Center (US), Instrumentation Development Division; 2012.

8. The 30-s interval was empirically determined to be a useful clock model storage interval. Because of the stability of the ticker to GPS relationship, very little estimated error growth was observed to occur with this size interval.

INTENTIONALLY LEFT BLANK.

Appendix. Rogue Cut Detection

A custom-designed algorithm based on ticker values was used to detect and reject rogue cuts recorded in Binary Large Object (BLOb) files. This Appendix will endeavor to explain the algorithm used and prove its efficacy.

Random tickers in rogue cuts were detected by comparing the differences between 3 sequential ticker values. To determine if a ticker is that of a rogue cut, the algorithm first calculates the change between the ticker in question and the one before it, the change between the ticker in question and the one after it, and the difference between those 2 changes. The following equations describe this step, where t_0 is the ticker in question, t_- is the previous ticker, and t_+ is the next ticker sequentially.

$$\Delta t_{-0} = t_0 - t_-$$
$$\Delta t_{0+} = t_+ - t_0$$
$$\Delta\Delta t_{-0+} = \Delta t_{0+} - \Delta t_{-0}$$

Next, the algorithm generates a score (S) for the ticker being examined by comparing the difference between the 2 changes ($\Delta\Delta t_{-0+}$) and the change from the ticker in question to the one after (Δt_{0+}).

$$S(t_-, t_0, t_+) = \frac{\Delta\Delta t_{-0+}}{\Delta t_{0+}} = \frac{\Delta t_{0+} - \Delta t_{-0}}{\Delta t_{0+}} = \frac{t_+ - 2t_0 + t_-}{t_+ - t_0}$$

In the current implementation of the algorithm, a ticker (t_0) is determined to be from a rogue cut if the calculated score is between 1.5 and 2.5, with 2.0 representing the score with the highest confidence level. This relationship is described in the following expression:

$$1.5 < S(t_-, t_0, t_+) < 2.5$$

The equation is based on the high likelihood that a rogue ticker will be far from the expected ticker value. In these cases, either Δt_{-0} will be positive and Δt_{0+} will be negative or Δt_{-0} will be negative and Δt_{0+} will be positive. They will also have similar magnitudes. In these cases, the expression above will yield a value close to 2.

The limits 1.5 and 2.5 provide a necessary buffer for maximizing rogue cut detection and minimizing false positives. The limits are designed to avoid the accidental removal of legitimate cuts, particularly those where either Δt_{-0} or Δt_{0+} is very large because of time gaps or rollovers.

In the simple case where there is no rogue ticker or rollover, all 3 tickers will be ordered least to greatest, and Δt_{-0} and Δt_{0+} will both be positive. In this case, the expression will always return a score less than one and fall outside the limits for detecting a rogue cut.

Special consideration also needs to be taken for rollovers. Tickers that appear immediately before or after a rollover could be misinterpreted as denoting rogue cuts if the limits used by the algorithm are not sufficiently constrictive. The range of scores produced by these tickers can be calculated by knowing the maximum elapsed time between t_- and t_+, the tickers before and after the ticker being examined. In the case of this algorithm, a BLOb file containing cuts has a maximum time span of 30 min (1.8×10^9 µs). This implies that t_+ occurs at most 1.8×10^9 µs after t_- with a value that is negatively offset by the rollover value (2^{32}). t_- occurs within 1.8×10^9 µs before the rollover, and t_+ occurs within 1.8×10^9 µs after the rollover.

For a ticker (t_0) occurring just before the rollover (Fig. A-1), the maximum score determined by the algorithm is 1.419 as shown in the following calculations:

Premise:

(1) $30 \text{ min} = 1.8 \times 10^9\ \mu s$

(2) $2^{32} - 1.8 \times 10^9 \leq t_- < t_0 \leq 2^{32} - 1$

(3) $0 \leq t_+ < 1.8 \times 10^9 - 1$

(4) $(2^{32} + t_+) - t_- \geq 1.8 \times 10^9$

$\equiv t_+ - t_- \geq 2^{32} - 1.8 \times 10^9$

$\equiv (t_+ - t_0) - (t_- - t_0) \geq 2^{32} - 1.8 \times 10^9$

$\equiv (t_+ - t_0) \geq (t_- - t_0) + 2^{32} - 1.8 \times 10^9$

Calculation:

(1) $S(t_-, t_0, t_+) = \frac{t_+ - 2t_0 + t_-}{t_+ - t_0} = 1 + \frac{t_- - t_0}{t_+ - t_0}$

(2) $\min\limits_{t_-, t_0, t_+} S(t_-, t_0, t_+) = S(2^{32} - 2, 2^{32} - 1, 0) = 1 + \frac{1}{2^{32} - 1} \cong 1$

(3) $\boxed{\max\limits_{t_-, t_0, t_+} S(t_-, t_0, t_+) = S(2^{32} - 1.8 \times 10^9, 2^{32} - 1, 0) = 1 + \frac{1.8 \times 10^9 - 1}{2^{32} - 1} \cong 1.419}$

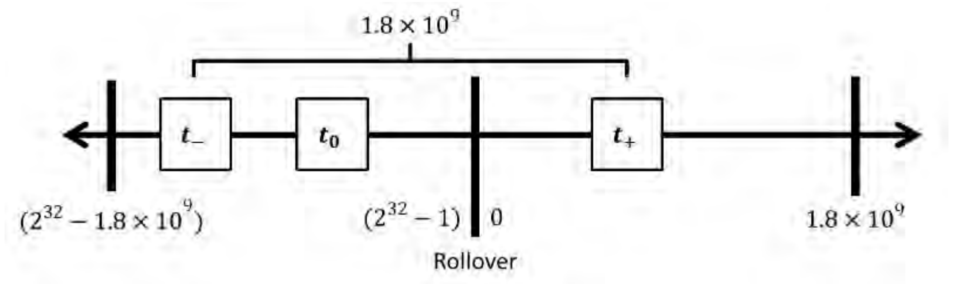


Fig. A-1 A ticker (t_0) occurring just before a rollover

For a ticker (t_0) occurring just after the rollover (Fig. A-2), the minimum score determined by the algorithm is 3.386 as shown in the following calculations:

Premise:

(1) $30 \text{ min} = 1.8 \times 10^9 \ \mu s$

(2) $2^{32} - 1.8 \times 10^9 < t_- \leq 2^{32} - 1$

(3) $0 \leq t_0 < t_+ \leq 1.8 \times 10^9 - 1$

(4) $(2^{32} + t_+) - t_- \geq 1.8 \times 10^9$

$\equiv t_+ - t_- \geq 2^{32} - 1.8 \times 10^9$

$\equiv (t_+ - t_0) - (t_- - t_0) \geq 2^{32} - 1.8 \times 10^9$

$\equiv (t_+ - t_0) \geq (t_- - t_0) + 2^{32} - 1.8 \times 10^9$

Calculation:

(1) $S(t_-, t_0, t_+) = \frac{t_+ - 2t_0 + t_-}{t_+ - t_0} = 1 + \frac{t_- - t_0}{t_+ - t_0}$

(2) $\boxed{\min_{t_-, t_0, t_+} S(t_-, t_0, t_+) = S(2^{32} - 1, 0, 1.8 \times 10^9 - 1) = 1 + \frac{2^{32} - 1}{1.8 \times 10^9 - 1} \cong 3.386}$

(3) $\max_{t_-, t_0, t_+} S(t_-, t_0, t_+) = S(2^{32} - 1, 0, 1) = 1 + \frac{2^{32} - 1}{1} = 2^{32}$

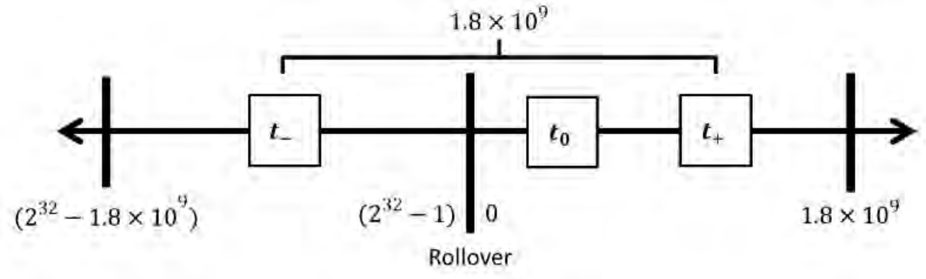


Fig. A-2 A ticker (t_0) occurring just after a rollover

The full range of scores for ticker rollovers and other natural ticker values fall outside the range identifiable as rogues. Though expanding the limits of rogue cut detection to scores between 1.419 and 3.386 might seem reasonable given the above calculations, special consideration is given to the effects of data stream buffering within the data recorder. Buffering can artificially increase the apparent time span covered by a file, especially in cases where there are very few cuts. The bounds of 1.5 and 2.5 were experimentally found to be the most accurate for the use case, detecting more than 99.8% of rogue cuts for medium- to high-packet* networks.

The algorithm used to detect and reject rogue cuts looks for random ticker values by comparing each ticker to its neighbors. A score is calculated for each ticker, and

*Defined here as having an average packet rate greater than one packet per second.

those with scores that fall inside the predefined limits are identified as rogue. The limits chosen for the use case have been shown to effectively detect rogue cuts and to not produce any false positives. Thus, the efficacy of the algorithm has been proven.

List of Symbols, Abbreviations, and Acronyms

ADMAS	Advanced Distributed Modular Acquisition System
ARL	US Army Research Laboratory
ATC	US Army Aberdeen Test Center
ATEC	US Army Test and Evaluation Command
BLOb	binary large object
FPGA	field-programmable gate array
GPS	Global Positioning System
HPC	High-Performance Computing
IP	Internet Protocol
PVA	Position, Velocity, Acceleration
UTC	Coordinated Universal Time

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
2 (PDF)	DIRECTOR US ARMY RESEARCH LAB RDRL CIO LL IMAL HRA MAIL & RECORDS MGMT
1 (PDF)	GOVT PRINTG OFC A MALHOTRA
1 (PDF)	TECH AND PROJ ENGR LLC B PANNETON
1 (PDF)	QED SYSTEMS LLC J ADAMETZ J FRANSSEN
1 (PDF)	DIR USARL RDRL CIN S K RENARD

INTENTIONALLY LEFT BLANK.