Space Network Time Distribution and Synchronization Protocol Development for Mars Proximity Link

Simon S. Woo[*], Jay L. Gao[†]
Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109

and

David Mills[‡]
Electrical and Computer Engineering, University of Delaware, Newark, DE 19716

ABSTRACT

Time distribution and synchronization in deep space network are challenging due to long propagation delays, spacecraft movements, and relativistic effects. Further, the Network Time Protocol (NTP) designed for terrestrial networks may not work properly in space. In this work, we consider the time distribution protocol based on time message exchanges similar to Network Time Protocol (NTP). We present the Proximity-1 Space Link Interleaved Time Synchronization (PITS) algorithm that can work with the CCSDS Proximity-1 Space Data Link Protocol. The PITS algorithm provides faster time synchronization via two-way time transfer over proximity links, improves scalability as the number of spacecraft increase, lowers storage space requirement for collecting time samples, and is robust against packet loss and duplication which underlying protocol mechanisms provide.

Nomenclature

NTP	=	Network Time Protocol
DSN	=	Deep Space Network
TC Packet	=	Time Correlation Packet
PITS	=	Proximity-1 Space Link Interleaved Time Synchronization
FSN	=	Frame Sequence Number
SpNTP Packet	=	Space NTP Packet

I. Introduction

KNOWING and maintaining accurate time is especially critical for correct operation of computer systems that are designed for space explorations. In particular, spacecraft at great distances from Earth require high-precision time information for navigation and communications purposes. Typically, an on-board computer carries a space clock or mission-elapsed time counter (SCLK); however, a clock can be skewed and drift due to external environmental effects such as radiation, temperature fluctuation, relativistic effects, as well as internal heat generated from software or hardware overloading. Inaccurate timing can cause other temporary or permanent cascading subsystem failures, resulting in unpredicted and undesired behaviors. In space missions, correcting and

[*]Member of the technical staff, Communication Architecture and Research Section of the Jet Propulsion Laboratory, 4800 Oak Grove Drive M/S 238-420, Pasadena, CA 91109.
[†] Senior Member of the technical staff, Communication Architecture and Research Section of the Jet Propulsion Laboratory, 4800 Oak Grove Drive M/S 238-420, Pasadena, CA 91109.
[†] Professor, Electrical and Computer Engineering at the University of Delaware, Newark, DE. 19716

synchronizing spacecraft clocks is made even more challenging due to long propagation delay from Earth. A common method for time synchronization is to perform offline processing on Earth by comparing time correlations between the spacecraft and Earth. This time correlation process requires manual resources on Earth and can introduce inaccuracy in time distribution and synchronization due to additional time lag for data processing and coordination. As the number of spacecraft increases, it is envisioned that intrinsic, ubiquitous, and distributed time synchronization and distribution in mission critical InterPlanetary communications and networks, will be necessary.

A task to develop reliable and effective space network time synchronization and distribution methods has been initiated at JPL. These methods can be applied for proximity communication scenarios in order to improve the overall network clock accuracy. This work addresses and provides the fundamental capability to distribute time when no direct communication link is available between Earth and remote spacecraft, for instance when a spacecraft is on the back side of the Mars. Therefore, we consider the in situ time distribution service for the future Mars Network where we can provide time information in GPS-denied or GPS-unavailable areas with a NTP based approach.

In addition, it may not be possible to equip every spacecraft with a high precision clock due to cost or design issues. Some missions may not require high precision time information all the time and course time synchronization is enough for some applications. In these cases, it is possible to equip some number of spacecraft acting as primary time servers to provide time distribution service to other secondary spacecraft similar to the Network Time Protocol (NTP) based approach which is widely used in terrestrial networks. As the number of spacecraft increase, we believe that this approach will provide effective time distribution architecture.

Currently, CCSDS Proximity-1 Space Link Protocol defines and provides service for time sample collection and exchange between spacecraft. The detailed algorithms for processing these time samples in order to discipline the spacecraft clocks are left for the users to develop.

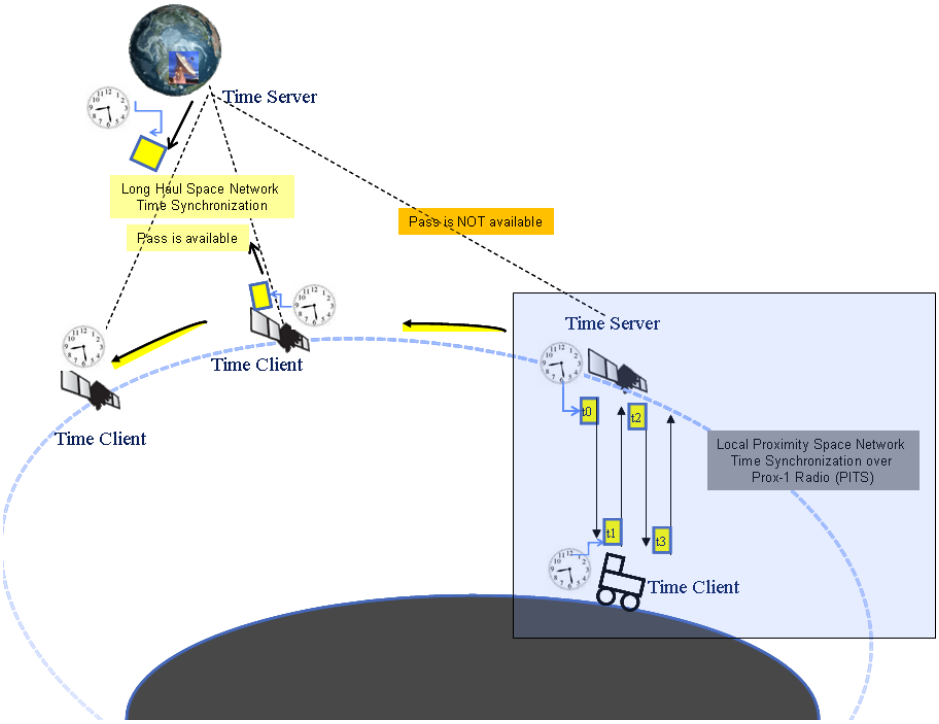

Figure1. Time Distribution for Deep Space Network

The primary objective of this work is to present an efficient, distributed, and reliable Space Network Time Distribution and Synchronization Protocol that provides accurate time synchronization and distribution service which can be directly infused into the Electra radio that implements the CCSDS Proximity-1 Space Data Link[1]. A proposed algorithm, Proximity-1 Space Link Interleaved Time Synchronization (PITS) [2] by Mills, has been designed to operate over the CCSDS Proximity-1 Space Link Protocol with minor modifications to its Medium Access Control (MAC) Sublayer. PITS can utilize both hardware-timestamps provided by the radio as well as soft-timestamps provided by a higher layer of the protocol stack for improved precision. Earth-disciplined vehicles can

operate as primary PITS servers while other spacecraft as secondary PITS servers or clients. PITS allows two-way time transfer between a primary server and a secondary server using the Supervisory Protocol Data Unit (SPDU) of Proximity-1 Protocol so that time information can be efficiently distributed. The secondary servers can distribute time information to the rest of the network in similar manner. Therefore, PITS can potentially provide the following benefits: 1) faster time synchronization via two-way time transfer over proximity link between spacecraft, 2) improved scalability as the number of spacecraft increase, 3) lower storage space requirement for collecting time samples, and 4) robust operations against packet loss and duplication which underlying protocol mechanisms provide.

This paper describes the current progress of implementing and validating the correctness and design of the PITS algorithm and protocol at JPL considering the Mars Proximity link communications scenarios as a baseline for analytical evaluation and simulation. We reveal interesting and important design issues. Also, preliminary simulation and testing results will be described.

II. Time Stamping Principles

One way to achieve clock synchronization between two nodes is based on message based time information exchanges, where this approach used in NTP is widely adopted for terrestrial network time synchronization. A node that has accurate time information is defined as a *time server*. A node who wants to be synchronized to a time server is a *time client*. It is the time distribution service requester. Generally, a time server and a time client exchange time information several times and find relative time differences to each other. To better illustrate the time synchronization process, we denote node A as a time server and node B as a time client for convenience in Fig 2. We assume that the paths from A to B and B to A are symmetric during the time synchronization process. This means that the distance stays the same and link characteristics are stochastically equivalent. We assume that one way light time (OWLT) is half of the round trip time (RTT) delay. We precisely define T_1 as a transmit-timestamp of node A when A sends time packet to node B where typically time information is encapsulated in the network time protocol (NTP) packet. We define T_2 as a receive-timestamp of node B when node B receives the NTP packet from node A. Similarly, T_3 and T_4 are the transmit-time stamp at node B and receive-timestamp at node A respectively. When each node obtains the ***four consecutive timestamps***: (i.e.: T_1, T_2, T_3, and T_4 for node A and T_3, T_4, T_5, and T_6 for node B in Fig 2), RTT and offset can be calculated using the following equations:

For A,

- $\text{offset} = \frac{1}{2}\{[(T_2 - T_1)] + [T_3 - T_4]\}$ (1)
- $\text{delay} = [T_4 - T_1] - [T_3 - T_2]$ (2)

For B,

- $\text{offset} = \frac{1}{2}\{[(T_4 - T_3)] + [T_5 - T_6]\}$ (3)
- $\text{delay} = [T_6 - T_3] - [T_5 - T_4]$ (4)

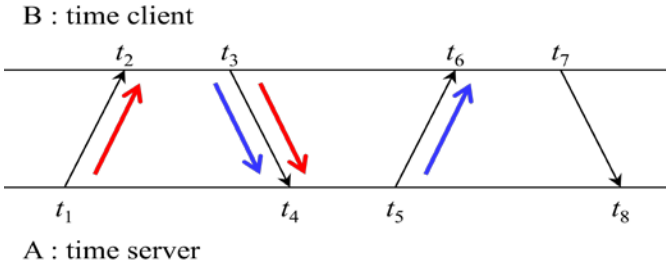

Figure 2. Illustration of offset and round trip calculation between A and B

As shown in Fig. 2, it requires collecting four consecutive timestamps in order to calculate the round trip time (RTT) delay and relative time difference at each side. Therefore, we need to complete all A→B, B→A, and A→B

transmissions successfully in order to compute the correct RTT and offset. We define the ***NTP process*** as the time that node A transmits NTP packets to node B till both A and B successfully collect four consecutive timestamps. The number of samples to collect or the duration of collection time highly depends on the Clock Discipline Algorithm. For the scope of this work, the details of the Clock Discipline Algorithm and grooming algorithm are not discussed. In reality, it is not trivial to obtain the accurate transmit and receive-time stamps due to various hardware constraints. However, there are some general principles that are typically employed to minimize timestamping errors [2]:

-Inbound and outbound Time Packet sizes should be known to each other in advance and the NIC should provide reliable/accurate data rate

-The propagation delay measured from the first bit sent in a packet to the first bit received on each direction of transmission must be the same

-$T1$ and $T3$ must be captured from the preamble timestamp, where a preamble timestamp is captured as near to the start of the packet as possible (the preferred point follows the start-of-frame (SOF) octet and before the first octet of the data)

-$T2$ and $T4$ must be the captured from the trailer timestamp, where a trailer timestamp is captured as near to the end of the packet as possible (on transmit the preferred point follows the last octet of the data and before the frame check sequence (FCS) on receive the preferred point follows the last octet of the FCS) : Also, different combinations of preamble and trailer timestamp can be available at the sender and receiver. In order to provide the high accuracy of time stamping, it is strongly recommended that each side agrees on what type of time stamping strategy it will employ in advance. If hardware is not available to timestamp the packet, less accurate software-timestamp (or driver-stamp) by software or driver is also available.

-A software timestamp is captured as close to the system I/O call as possible: More details on techniques to obtain and approximate time stamps are further explained in [2]. In [2], Mills suggested that if the delays are reciprocal (delays on inbound and outbound are the same or statistically equivalent) and the packet lengths the same, software timestamps are equivalent to hardware timestamps.

For CCSDS Proximity-1 Space Link Protocol, the data rate between the orbiter and the landed asset can be different. Therefore, time stamping strategy should be agreed upon in advance.

III. CCSDS Proximity-1 Space Data Link and its Time Correlation and Distribution Services

CCSDS Proximity-1 Space Link Protocol is used for communicating between a Mars orbiter and rovers and has demonstrated the successful communications capability over a Mars proximity link. In this section, we briefly summarize and describe the existing Time Services in CCSDS Proximity-1 Space Link Protocol and compare the existing on board time service with the proposed PITS algorithm.

Time Service in CCSDS Proximity-1 Space Link Protocol

Currently, CCSDS Proximity-1 Space Link Protocol defines the Time Collection, Time Correlation, and Time Distribution services [1], where Time Services can be used for time correction, synchronization, and time-derived ranging measurements purposes. First, we assume that two spacecraft A and B collect time samples as follows:

Time Collection Service

1. Spacecraft A initiates the Time Distribution service and radiates Supervisory Protocol Data Unit (SPDU) frames to spacecraft B where the vehicle controller (VC) initiates the 'Time Distribution service' and the MAC layer sets the details of the duration of time collection interval, T, or the number of samples, n, parameters. Time samples frames SPDUs are sent with Expedited QoS service and not as Sequence controlled QoS.

We assumed that the C&S physical layer in CCSDS Prox-1 provides the capability of time-tagging the trailing
bytes of the ASM synch which is also used in the PITS algorithm. Both the sequence numbers as well as the time tag of each transmitted and received frame are stored in the Sent-time and Received-time buffer at the MAC layer respectively.

2. Both spacecraft A and B collect and store 1) the transmitted and 2) received time and the 3) sequence number of the time sample SPDUs for duration T or the number of samples n which are specified in the Time Sample field in SPDU.

Time Correlation Service

3. Then, initiator A creates Proximity Time Correlation (TC) Packets SPDU which contains the time tag, frame sequence number (FSN), and time tag direction and sends it to B. Then, B can calculate the time correlation. After that, B creates Proximity-1 TC Packet and sends TC Packets to A. Finally, A can compute the time correlation as well.

Time Distribution/Transfer Service

4. After completing time sample collection, the Universal Time Coordinated (UTC) time transfer process begins using the TIME DISTRIBUTION (UTC Time Transfer) directive over the Proximity link by creating Time Transfer (TT) Packet SPDU. This packet contains the correlation between UTC and the recipient's clock. (However, we believe that this time directive requires additional processing time and this may not reflect the accurate time offset.) Each creates Time Transfer (TT) SPDU containing the time correlation information and exchanges with each other.

5. After receiving TT SPDU, a receiving spacecraft updates the clock based on the time correlation information. Therefore, A and B simultaneously exchange the time-tags during this process.

In order to achieve higher accuracy in timestamping, several additional important observations are made in [1] regarding the existing time services such as a priori knowledge about time collection buffer size and known internal signal path delays. Also, [1] recommends the simultaneous collection of TC packets in both directions. In this work, we like to address solutions to some of these described issues within the PITS protocol.

IV. PITS algorithm

PITS is a general time exchange protocol which defines how to exchange time information between two spacecraft. PITS provides one possible way of implementing efficient and reliable time packet exchange that can be used for CCSDS Proximity-1 Space Link Protocol Time Collection, Correlation, and Distribution services. The core functionalities of PITS are similar to the ones used in terrestrial NTP [3]. However, the PITS algorithm is designed to be used for time synchronizing and distribution between a Mars orbiter and landed assests over the CCSDS Proximity-1 Space Link with minimal modifications. PITS is benficial when the direct time synchronization from Earth to rover is difficult.

Further, PITS can automatically cope with packet drops and reception of duplicated packets. Therefore, there is no worry for harmful or defective time samples, since the collected time samples are checked by the underlying protocol logic. This can be achieved by cross-checking the stored state variables and received time information in the Space NTP (SpNTP) packet. More details are described in [2]. Therefore, this check routine can be easily implementable with existing time tagging methods and can be placed into the MAC Sublayer.

In particular, we assume that a Mars orbiter carries an accurate on board clock and has *in flight time distribution* capability to distribute time to landed assets. We assume that the relative movements of orbiter and rovers are stationary during the time synchronization process as before. PITS can operate in the three modes: 1) Basic Mode and 2) Interleaved Mode, and 3) Broadcast Mode and the more details are described in [2]. In this work, we focus on the Basic and Interleaved Modes.

Proximity-1 Space Link Interleaved Time Synchronization (PITS)

The goal of PITS is to provide the method of capturing and exchanging timestamps more accurately and reliably in order to minimize time difference between the timestamp tagged at the spacecraft and true clock information. In the PITS algorithm, time information is exchanged via Space NTP (SpNTP) and this time information can be easily encapsulated as a SPDU. Time information contains the origin timestamp, receive-timestamp and transmit-timestamp as shown in the following Fig. 3.

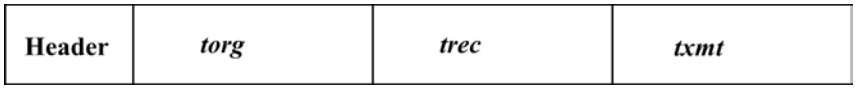

Figure 3. Space NTP (SpNTP) Packet

A time server and a time client exchange the SpNTP packet that contains the following three fields in the payload,

- t_{org} : origin-timestamp
- t_{rec} : receive-timestamp
- t_{xmt} : transmit-timestamp

These timestamps are updated with new time information when a new SpNTP packet is generated. In addition, each server and client has the following three state variables in the spacecraft memory to store the timestamps received from SpNTP packet:

- *rec* : receive-timestamp
- *dst* : destination-timestamp
- *aorg* : origin-timestamp
- *borg* : origin-timestamp used for Interleaved Mode

Updating state variables with time information received from SpNTP is slightly different for Basic and Interleaved mode and is explained in the next section.

Basic Symmetric Mode

The example of packet exchanges between two spacecraft using Basic Symmetric Mode is given below. The payload in SpNTP packet as well as state variables at the time server and client are shown at each time instant. The state variables and packet payload updates occur according to the *transmit* and *receive process*. In particular, the time in blue boxes indicate the system clock, where the time that is stamped either at transmission or reception of SpNTP packet.

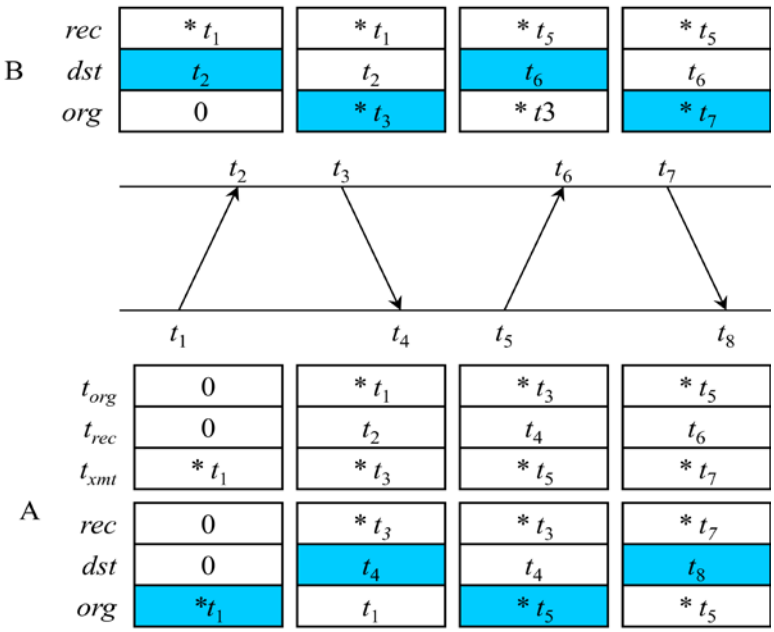

Figure 4. Illustration of Basic Process showing SpNTP packet and state variables[2]

A server A initiates the synchronization process by sending a SpNTP packet to B as shown in Fig. 4, where this is defined as *Transmit Process* and pseudo codes [2] are given in Fig. 5. The flow diagram shown in Fig. 7. is another description of Transmit Process in Fig. 5. Initially, each t_{org} and t_{rec} stores receive state variable, *rec*, and destination timestamp state variables, *dst*, respectively as shown in Fig. 6. Just before transmitting an origin timestamp, state variable *aorg* stores the current time, *clock* in the After SpNTP packet transmission, t_{xmt}, contains the clock information.

```
t_org = rec
t_rec = dst
If (x == 0) {                    /* basic mode */
   aorg = clock
   t_xmt = aorg
} else {                         /* interleaved mode */
   if (x > 0) {
      aorg = clock
      t_xmt = borg
   } else {
      borg = clock
      t_xmt = aorg
   }
   x = -x
}
```

Figure 5. Pseudo codes of Transmit Process for 1) Basic and 2) Interleaved Mode

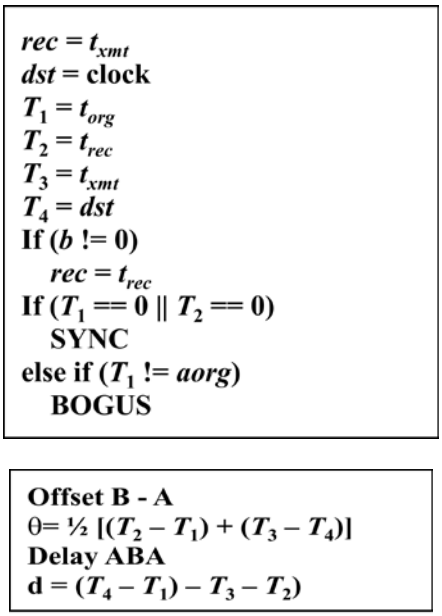

Figure 6. Pseudo codes of Basic Receive Process

Upon reception of a SpNTP packet at B, B updates its state variable *rec, dst*, and *org* and transmits SpNTP packet with new time information according to the *Receive Process*. The pseudo codes and flow chart of Receive Process are provided in Fig. 6 and Fig. 8, respectively. After receiving a SpNTP packet from A, *rec* state variable in B stores t_{xmt} timestamp from the received SpNTP packet. Also, *dst*, state variable stores the current received *clock* at B.

In addition, each temporal variable T_1, T_2, T_3, and T_4 stores the t_{org}, t_{rec}, t_{xmt} and , *dst* respectively to compute the offset and round trip delay described in Eq. (1) through (4) as shown in Fig. 6. This received process is also described in the flow chart in Fig 8. Therefore, the *Transmit Process* occurs just before transmitting a Space NTP packet and the *Receive Process* occurs right after receiving a SpNTP packets [2].

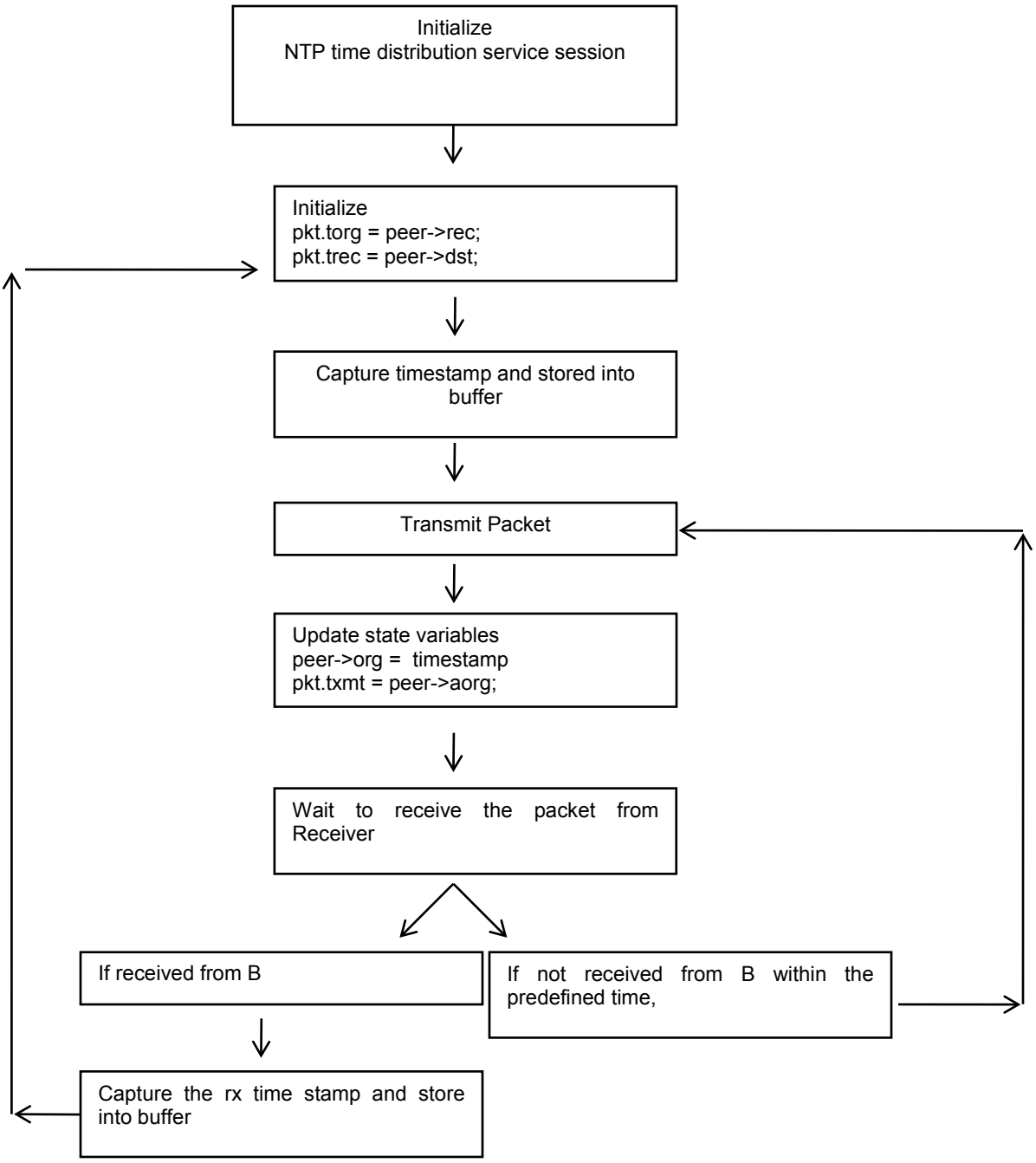

Figure 7. Flow Chart of Basic Transmit Process

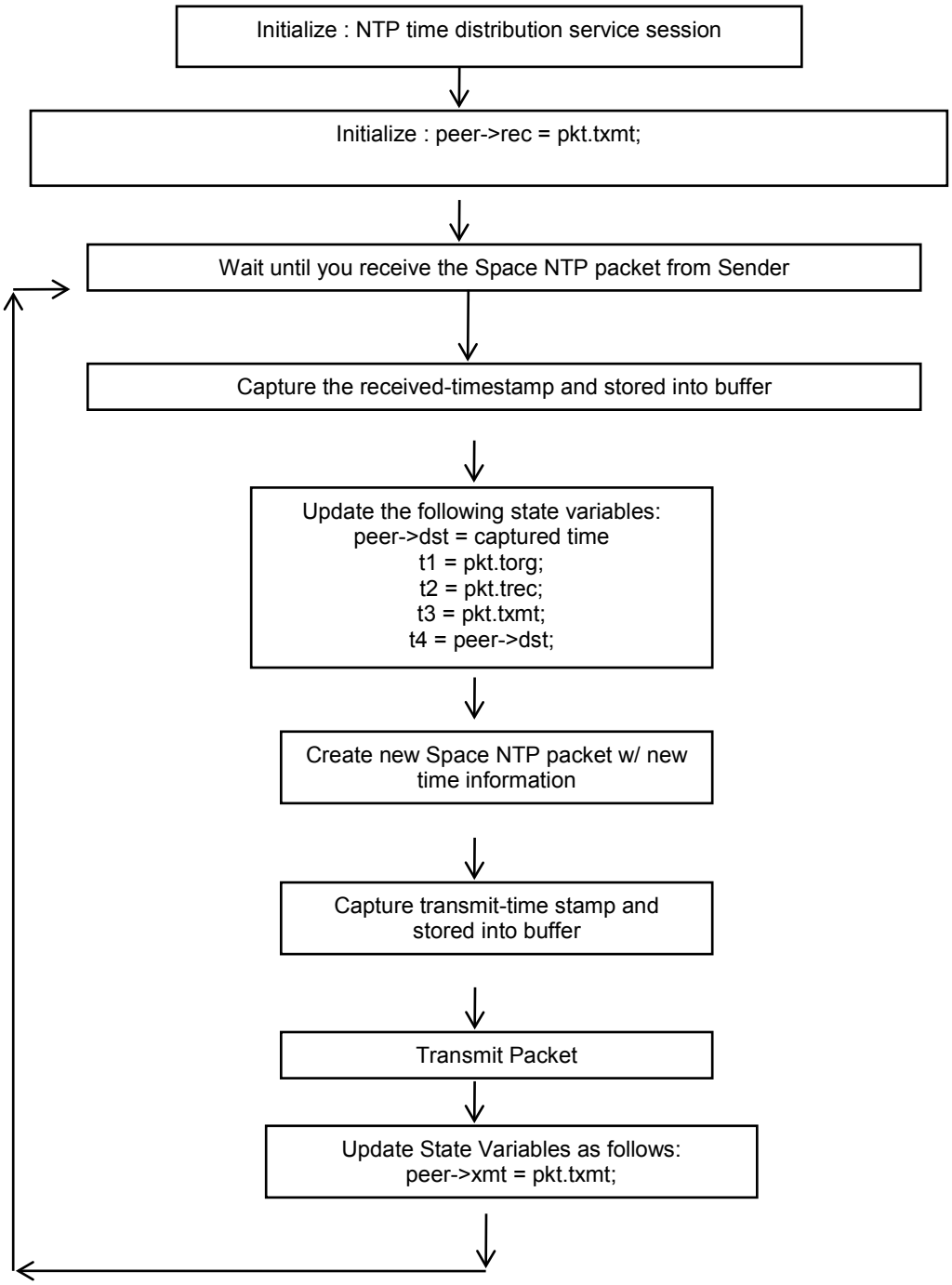

Figure 8. Flow Chart of Basic Receive Process

Interleaved Symmetric Mode

First, the SpNTP packet used in the interleaved mode has the same format as the one used in the Basic Mode. This provides the backward compatibility when each Basic Mode or Interleaved Mode has to switch to another mode. On the other hand, interleaved mode has one more state variable, *borg*. This, *borg*, holds the original timestamp every odd (or even) time and *aorg* also holds the same information for every even (or odd) time as shown in Fig.5.

Another important aspect in the Interleaved Mode is that the transmit-timestamp is captured right after the SpNTP packet has been sent unlike Basic Mode. Hence, the current transmit-timestamp is sent in the next transmit-timestamp, t_{xmt} , using the next SpNTP Packet transmission, whereas the receive-transmit timestamp is immediately available as soon as a packet has been received. The following figure provides the example of exchanging space packets with the Interleaved Mode. In Fig 9, the first transmit-timestamp, t_1, at A is actually being sent at the second round, at time t_5. The first transmit-timestamp of B, t_3, when B transmits packet to A for the first time, is sent in the following transmission time at t_7. On the other hand, the received timestamp, t_2 at B is immediately encapsulated and sent in the following SpNTP packet transmission unlike transmit-timestamps. Therefore, the new receive-timestamp and the transmit-timestamp information from the other side are interleaved and available at every other round.

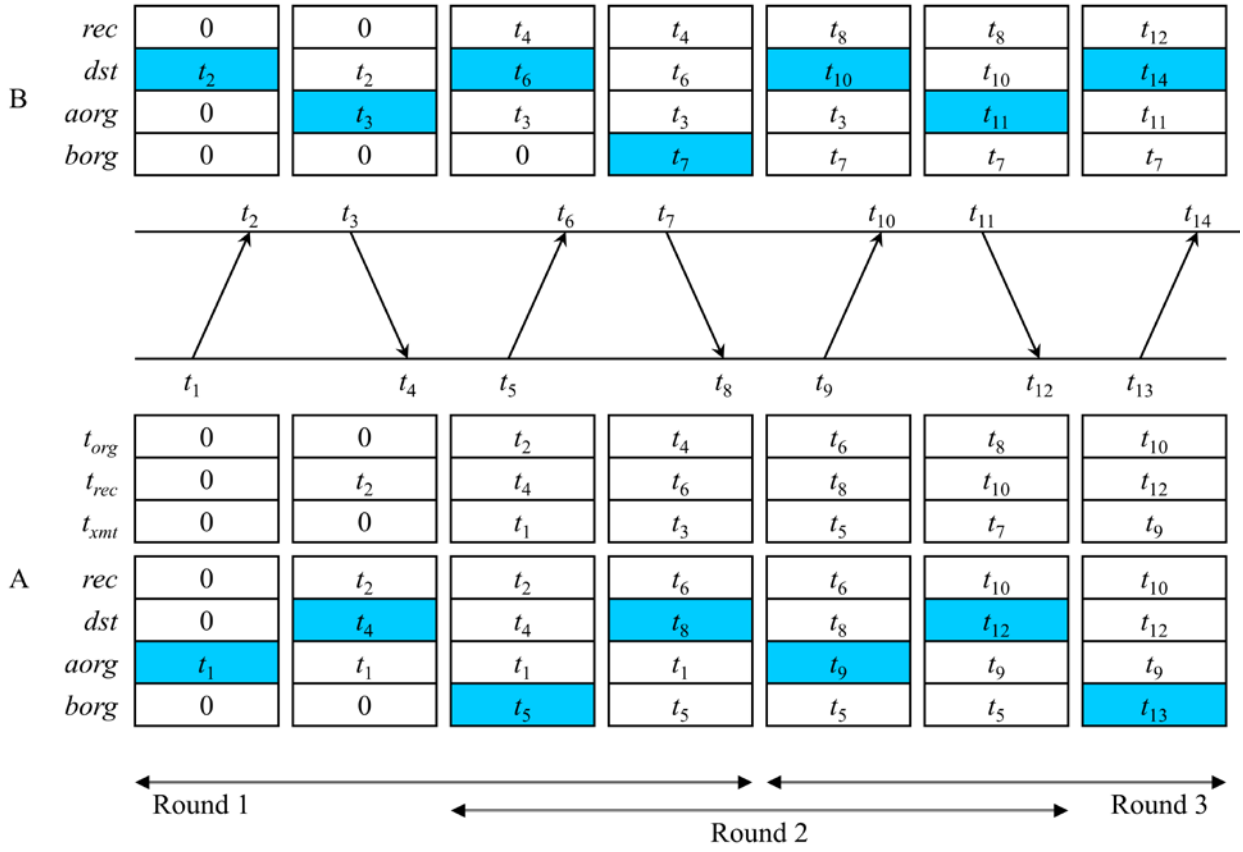

Figure 9. Illustration of Interleaved Process showing SpNTP packet and state variables[2]

```
If (x > 0)
   T1 = aorg
else
   T1 = borg
T2 = rec
T3 = txmt
T4 = dst
rec = trec
dst = clock
if (torg == 0 || T2 == 0 || T3 == 0) {
   SYNC
} else if (torg != 0 && torg != T4)
   BOGUS
}
```

Figure 10. Pseudo codes of Interleaved Receive Process

On the other hand, the synchronization duration of Interleaved Mode takes two more rounds to achieve the time synchronization because of sending the transmit timestamp in the next packet transmission round. However, interleaved approach provides more accurate time stamping since it captures the timestamp right after packet has actually been sent. Therefore, with twice as many time packet exchanges, the Interleaved Mode can achieve more accurate time information since it can capture the time that is closest to the real packet transmission time.

In addition, PITS can detect and filter possible errors caused from receiving duplicate or lost packets by comparing the stored state variables and current received timestamp information in the SpNTP packet. Further, PITS is able to detect and autonomously recover from these unexpected events that can possibly occur during the time synchronization process, providing additional level of protocol protection. Therefore, these features allow detecting logical protocol errors that cannot be checked by mere CRC checks. Further, a protocol automatically restarts new synchronization process after detecting errors or loss. The details are provided in [2].

The existing time collection method in CCSDS Prox-1 allows collecting a number of data samples that can be averaged over for Time Correlation calculation. However, this approach cannot resolve if there are bad time samples or lost time packets. Once Time Collection Service begins, the existing approach would not able to control or stop the process or cannot validate the collected time samples on the fly. However, PITS is capable of checking the correctness of received samples on the fly and ignore if time samples are detected faulty.

The key differences between the proposed PITS algorithm and the conventional CCSDS Proximty-1 Time Collection services are described in Fig. 11, in terms of protocol interactions and time packet exchanges. As shown in the following figure 8, in the existing time collection service, a time server keeps sending SPDU frames until the requested time interval or to collect the number of samples. A time client passively timetags the received SPDU frames. On the other hand, in PITS, when a time client sends back the new SPDU frames as soon as it receives the SPDU sent from a time server. The SPDU time information is updated according to the piggybacked *Transmit Process* and *Receive Process*. In this way, PITS provides much more intact time information between a time server and a client. The time stamps collected at the server and the client are tightly coupled together. Therefore, each spacecraft needs to switch back and forth between *Transmit* and *Receive Process* accordingly when it transmits and receives Space NTP packet. PITS will require additional latency to send SPDUs back and forth to collect time samples. PITS time collection occurs concurrently whereas, in the existing approach, one waits until the completion of the other.

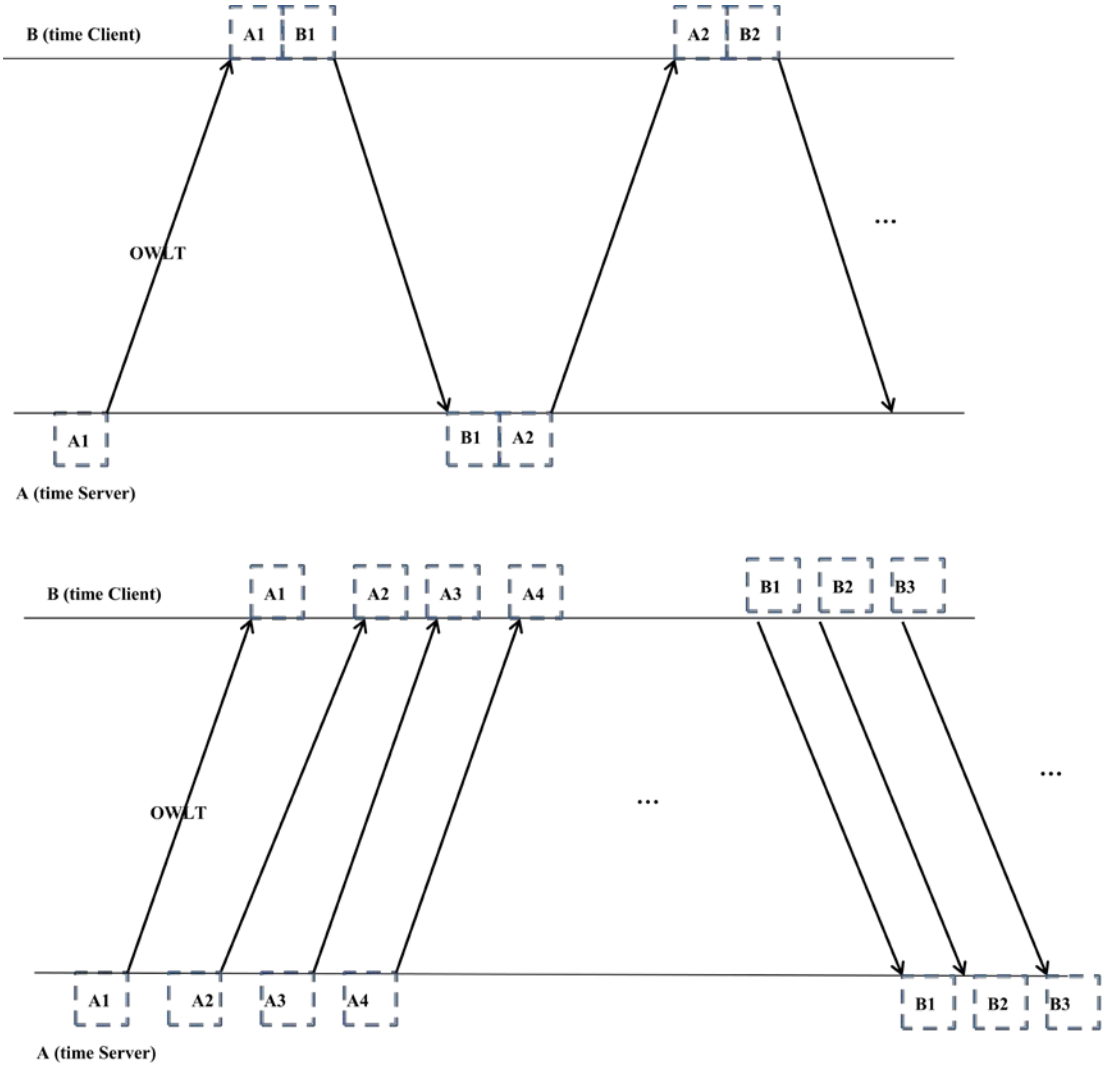

Figure 11. A side-by-side comparison between the proposed PITS algorithm and the conventional CCSDS Proximty-1 Time Collection services

V. Integrating PITS with CCSDS Proximity-1 Space Link Protocol

The MAC Sublayer is the place that most of time services interactions occur in CCSDS Proximity-1 Space Link Protocol, where the MAC Sublayer directly interacts and interfaces with Frame Sublayer, Coding and Synchronization (C&S) Subayer, and Physical Sublayer. The MAC Sublayer has a capability to instruct the C&S Sublayer to intercept a 24-bit ASM Sync marker used to delimit C&S frames, check CRC-32, captures a timetag from the Spacecraft clock (SCLK), and store the frame type and sequence number from the 5-octet frame header. Therefore, PITS can simply reuse the existing time collection capabilities for time tagging.

Therefore, MAC Sublayer can create SpNTP packets and encapsulate those into SPDU frames and transmit and receive to and from the other remote MAC. In particular, each SpNTP Packet can be transmitted as an expedited SPDU. Then, the SPDU that contains SpNTP packet is transmitted first with the current transmit and received state variables followed by the SPDU header. After that, the transmit time tag buffer is read, converted to UTC and saved

in transmit state variables for the next message in the Interleaved Mode. Therefore, the state variables can be located in the MAC Sublayer to store the received time information. Therefore, a SpNTP packet sent as a SPDU frame is shown below:

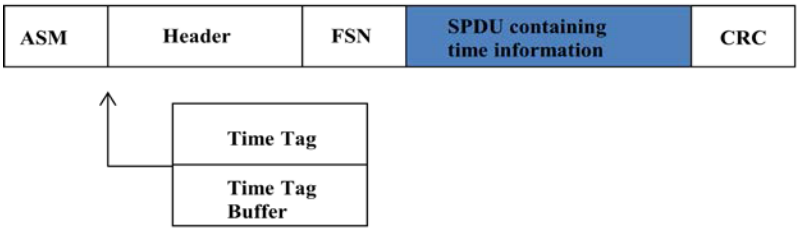

Figure 12. The proposed CCSDS Proximiy-1 Frame including SPDU SpNTP Packet

Note[2]*:* In this narrative the term ***time tag*** refers to a value derived from the SCLK, while the term ***timestamp*** refers to a coordinated time scale such as Coordinated Universal Time (UTC). This is important since the buffering and coding functions can delay a *time tag* in the frame data payload with many and varying seconds.

Other Considerations

Generally, Time Collection and Distribution service can be considered as a passive task, requiring periodic time synchronization and distribution. Therefore, PITS will require small bandwidth to collect time samples. Also, PITS requires relatively small storage space to store the state variables and time samples, whereas Time Collection Services in CCSDS Prox-1 will potentially require a large on board buffer space to store the time samples which need to be predetermined in advance and this space will grow as the number of sample sizes. Hence, this proposed PITS algorithm can be used for *In Situ Time Distribution Service*, and provide sufficient bandwidth and memory margin for other flight software and hardware operations.

VI. Implementations and Results

In this work, we designed the abstraction of the PITS model and implemented the software version of PITS mainly using C and socket program. A socket program provides an abstraction where applications can get access to the network and can read or receive remote data. In particular, for simplification, we used the software level clock, soft timestamping, (or network layer timestamping) using “OS Kernel system call” method to timetag the receive and transmit times. Through simulation, we have verified the functionality of the PITS algorithm by running PITS in two Linux machines, one functioning as a time server who provides the time distribution service and the other being a time client who requests the time distribution service as follows.

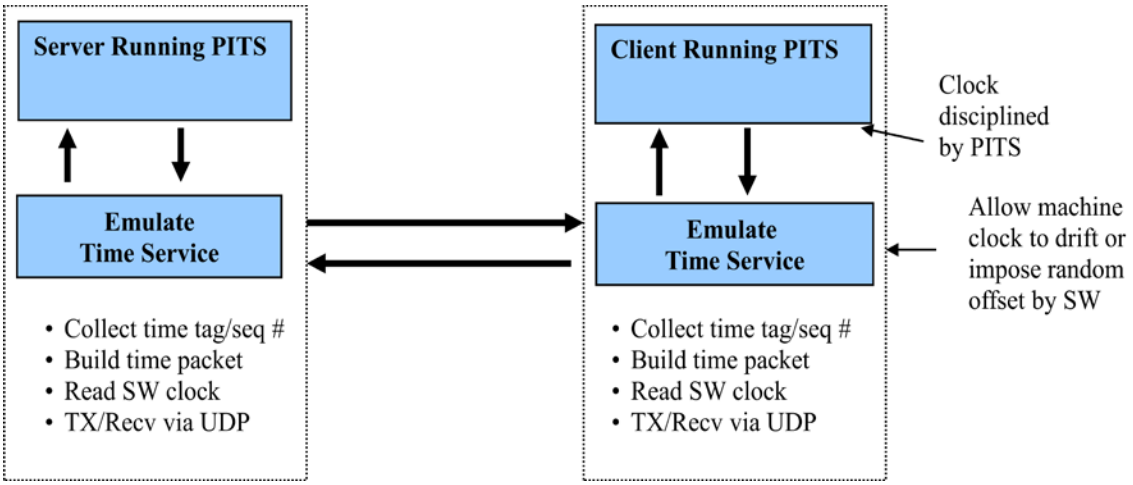

Figure 13. Simulation Setup

Since we do not have actual CCSDS Proximity-1 Space Link Protocol hardware or emulator available for testing, we could not test full integration interfaces and hardware timestamping strategy. However, we believe that software implementation is adequate enough to capture the packet exchange behaviors disciplined by the PITS algorithm and offset and round trip time estimates.

We used, *gettimeofday*, for evaluating our PITS algorithm. The *gettimeofday* resolution, while not fixed by a standard, may or may not be the same as the system clock's resolution. These parameters vary between hardware and operating systems, and also between machines with identical hardware and operating systems because these parameters can be modified during operating system compilation and/or system boot in many operating systems. Based on these assumptions, specifically, we have implemented the various subroutine calls used in *Transmit Process* and *Receive Process* in Fig 4 and 5. The main part of developed subroutine works as follows: subroutine call order at a receiver is when a packet arrives at the host the kernel will trigger the driver interrupt and call inbound packet driver interrupt. This will occur at or near the network layer. Later, when the kernel passes the packet to the network application layer, an operating system notification will trigger the call receive packet from *peer*. Then at the protocol's discretion, transmit packet to *peer* will be called, which will pass the return packet to the kernel at the network application layer. Later, at or near the network layer the kernel will trigger the driver interrupt and call outbound packet driver interrupt.

We chose two Linux machines that are physically close to each other, where not many computers are connected in this network in order to minimize traffic caused by other machines. We initially drift the clock in the client machine. Following is the simulation results that we collected for several experiments. As we can see, the computed offset using Eq (1) and (3) between A and B at each side has almost the same magnitude but opposite sign. Both calculated the RTT are about the same for 1000 different simulation runs.

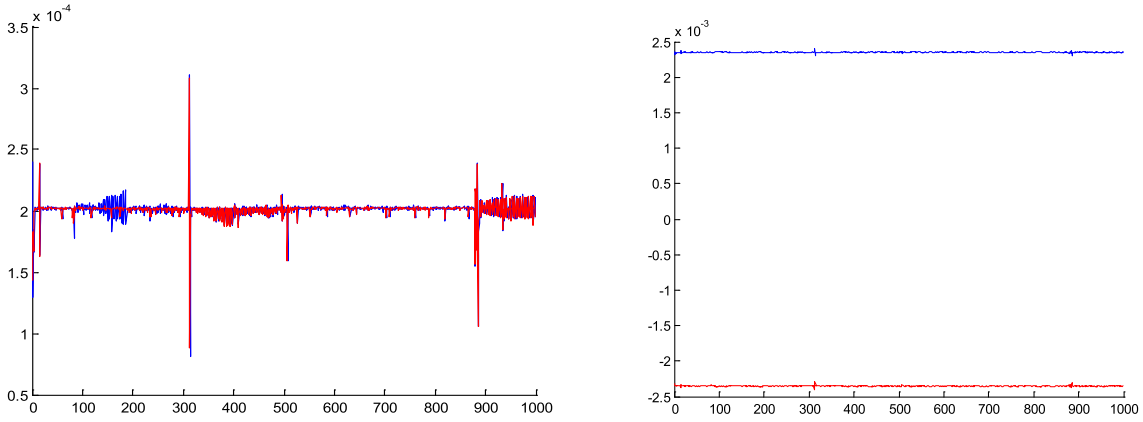

Figure 14. Offset and Delay results obtained between two Linux machines

The results accurately estimates the offset and RTT. We have validated that the PITS algorithm and protocol is working properly between two Linux machines. However, there were other services, OS kernel processes, and network traffic that were generated from each Linux machines that can potentially impact the performance and we could not completely isolate those. Therefore, we have seen some spikes or outliers. Therefore, soft timestamps were fluctuating a bit. However, we have observed and demonstrated that on average, the PITS algorithm correctly computed the RTT and offset.

VII. Future Work and Conclusions

In this work, we presented the framework to provide the on board time distribution capability at the spacecraft which would be a used component for *In Situ Time Distribution Service*. We verified the correctness of the PITS algorithm, validated the design, and provided the simulation results. As the number of space assets increases and the mission become more complex, we believe that the proposed time distribution approach will be an extremely effective method to distribute time to other spacecraft in proximity. Further prototype development and hardware integration and testing will be interesting extensions of the proposed work to show the timing accuracy improvement by exploiting the benefit of Interleaved Mode. This will further solidify the implementation and design issues arising from integrating PITS with Proximity Radio. Also, future work may be necessary for time attack or incorporate with security.

In particular, the proposed PITS can be extended to multi-spacecraft missions such as the Terrestrial Planet Finder (TPF) mission at JPL to provide the periodic time updates. In addition, PITS can be considered an option for the Space-Based Positioning, Navigation, and Timing (PNT) component. Therefore, PITS is expected to significantly reduce the amount time and resources required for time synchronization and distribution services in Mars proximity link while providing time information accuracy.

Acknowledgments

The authors would like to thank Dr. David Mills at University of Delaware for providing the PITS algorithm and invaluable explanations. Also, the authors would like to acknowledge John Veregge at JPL for originally implementing the PITS algorithm. The research was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

References

[1] CCSDS Proximity-1 Space Link Protocol, CCSDS 211.0-B-4: https://ccsds.org
[2] Mills, D.L., “Computer Network Time Synchronization: the Network Time Protocol on Earth and in Space”, to be published in 2010.
[3] The Network Time Protocol (NTP), http://www.ntp.org/