SpaceOps 2010 Conference
 Delivering on the Dream
 Hosted by NASA Mars
25 - 30 April 2010, Huntsville, Alabama

AIAA 2010-2360

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

Simon S. Woo* and 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

Time distribution and synchronization in space networking are challenging due to long propagation delays, spacecraft movements, and relativistic effects, and therefore the Network Time Protocol (NTP) designed for terrestrial networks may not work properly. In this work, we consider a time distribution protocol that is based on time message exchanges similar to Network Time Protocol (NTP) but customized for the proximity space links such as the RF links between Mars orbiters and rovers. This protocol is called the Proximity-1 Space Link Interleaved Time Synchronization (PITS) protocol and is designed with goal of integrating with the CCSDS Proximity-1 Space Data Link Protocol's existing time capture capability. 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 misordering which underlying protocol mechanisms provide. This paper describes the current progress of implementing and validating the correctness of the PITS algorithm at JPL, using the Mars Proximity link communications scenarios as our baseline.

Nomenclature

ASM	=	Attached Synchronization Marker
C&S Sublayer	=	Coding and Synchronization Sublayer
CCSDS	=	Consultative Committee for Space Data Systems
DSN	=	Deep Space Network
ET	=	Ephemeris time
FSN	=	Frame Sequence Number
MAC Sublayer	=	Medium Access Control Sublayer
NTP	=	Network Time Protocol
PITS	=	Proximity-1 Space Link Interleaved Time Synchronization
SCLK	=	Spacecraft Clock
SPDU	=	Supervisory Protocol Data Unit
SpNTP Packet	=	Space NTP Packet
TC Packet	=	Time Correlation Packet
Time-tag	=	Time value derived from the SCLK
Timestamp	=	Time represented in a coordinated time scale to be used between spacecraft for SpNTP packet exchanges
TT	=	Time Transfer
UTC	=	Coordinated Universal Time

* Member of Technical Staff, Communication Architecture and Research Section of the Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive M/S 238-420, Pasadena, CA 91109.
† Senior Member of Technical Staff, Communication Architecture and Research Section of the Jet Propulsion Laboratory, California Institute of Technology, 4800 Oak Grove Drive M/S 238-420, Pasadena, CA 91109.
‡ Professor, Electrical and Computer Engineering at the University of Delaware, Newark, DE, 19716.

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 spacecraft clock (SCLK) or mission-elapsed time counter; 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 hardware overloading. Inaccurate timing can cause other temporary or permanent cascading subsystem failures, resulting in unpredicted and undesired behaviors. In space missions, correcting and 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. The space networking time synchronization problem can be divided into two operational domains as shown in Fig. 1: deep space and proximity, creating a two-tiered architecture in time synchronization and distribution. Methods to achieve time synchronization over deep space links will be different from the proximity links due to the physical characteristics of the network as well as the capability of the end-points involved. In this paper we specifically address the method for time synchronization in the proximity domain and show how it contributes to improving the overall time accuracy of the entire network. The result of this work is aimed to provide a new fundamental capability for distributing time when no direct communication link is available between Earth and a remote spacecraft, for instance when a spacecraft is on the backside of 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.

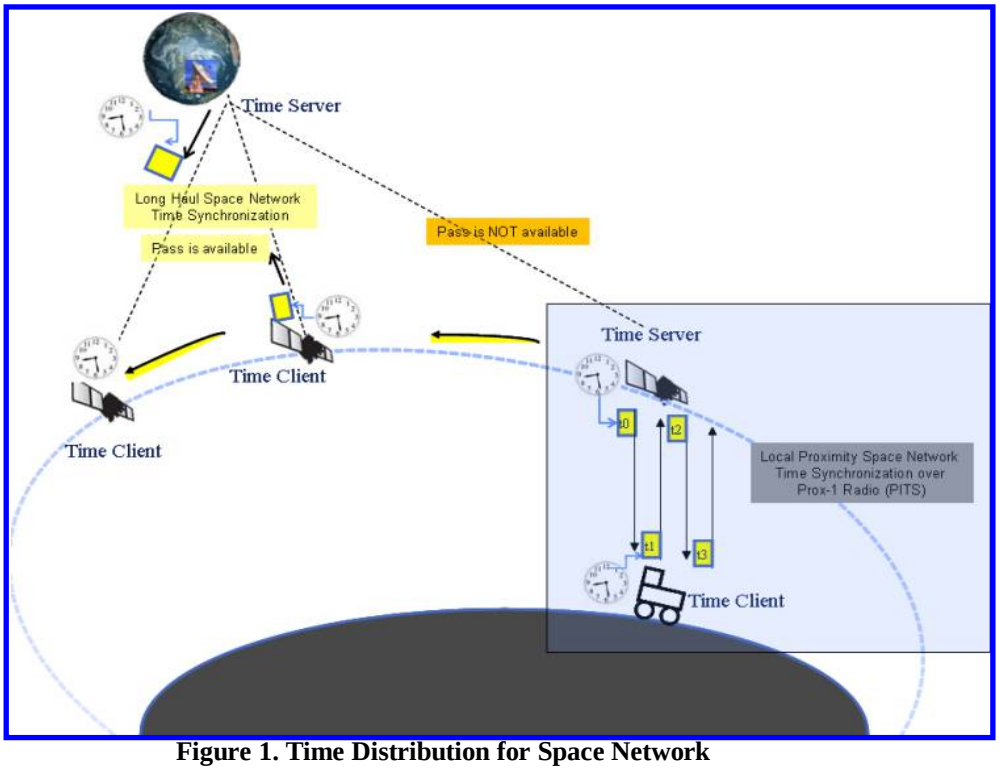

Figure 1. Time Distribution for Space Network

Also, from a practical point of view, it may not be possible to equip every spacecraft with a high precision clock due to mission cost or design issues. Some missions may not require high precision time information all the time and coarse timing is enough for some applications. In these cases, it is possible to designate a number of spacecraft as primary time servers to provide time distribution service to other second-tier spacecraft, similar to the NTP-based

approach which is widely used in terrestrial networks. As the number of spacecraft increases, we believe that this approach will provide effective time distribution architecture.

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

The primary objective of this work is therefore to present an efficient, distributed, and reliable time exchange and processing algorithm that can be directly infused into the CCSDS Proximity-1 Space Data Link. The 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 and Proximity-1 Supervisory Protocol Data Unit (SPDU) format. PITS can utilize both soft-timestamps provided by a higher layer of the protocol stack as well as hardware-timestamps provided by the radio for improved precision. Earth-disciplined vehicles can operate as primary PITS time servers while other spacecraft as secondary PITS servers or clients. PITS allows two-way time transfer between a primary server and a secondary server or a client 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 time packet loss and misordering over space links.

This paper describes the current progress of implementing and validating the correctness of the PITS algorithm, using the Mars Proximity link communications scenarios as our baseline for analytical evaluation and simulation, and the important design issues are revealed through our test and simulation activities.

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[3] 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*. 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 as shown 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 time packet. We define T_2 as a receive-timestamp of node B when node B receives the time 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 delay and offset can be calculated using the following equations:

For A,

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

$$\text{delay} = [T_4 - T_1] - [T_3 - T_2] \quad (2)$$

For B,

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

$$\text{delay} = [T_6 - T_3] - [T_5 - T_4] \quad (4)$$

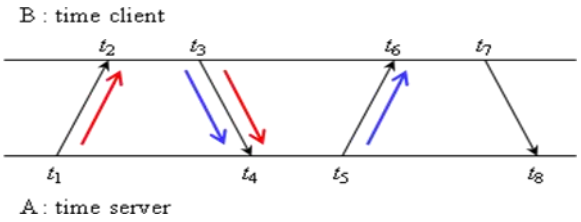

Figure 2. Illustration of offset and round trip calculation between A and B

As shown in Fig. 2, 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 time 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 depends on the mission operation requirements and specifications.

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 approaches that can be applicable in CCSDS Proximity-1 Space Link Protocol for accurate timestamping:

-Time information must be captured from the preamble timestamp, where a preamble timestamp is captured as near to the start of the packet as possible. In CCSDS Proximity-1 Space Link Protocol, time stamping ASM can provide the accurate transmit-time information. In order to increase the accuracy, we need to know the delay from the ASM via Reed Solomon and convolutional encoder and delays can be accurately measured in the lab testing. For example, if a frame is transmitted with the maximum packet size 2048 Bytes at the minimum rate of 1Kbps, it will take about 32s for queuing and data transmission. In CCSDS Proximity-1 Space Link Protocol, C&S Sublayer can time tag this information.

-Alternatively, a software timestamp needs to be captured as close to the system I/O call as possible. If 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. However, this is normally not the case for space links, where low speeds for telemetry and high speeds for data. Although space link itself is symmetrical if we neglect spacecraft motion, there are asymmetrical components with using different packet sizes, data rates, and coding. These asymmetrical components can be also measured in the hardware testing and the measured delays can be appropriately accounted for software timestamping.

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:

A. Time Collection Service

1. Spacecraft A initiates the Time Distribution service and radiates U-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 U-frames are sent with Expedited QoS service.

2. We assumed that the C&S physical layer in CCSDS Proximity-1 Space Link Protocol provides the capability of time-tagging the trailing bytes of the ASM. 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.

3. Both spacecraft A and B collect and store 1) the transmitted and 2) received time and the 3) sequence number of the time sample U-frames for duration *T* or the number of samples *n*.

B. Time Correlation Service

4. Then, initiator A creates Proximity-1 Time Correlation (TC) packets which contains the time tag, frame sequence number (FSN), and time tag direction and sends TC packets 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 in the same way.

C. Time Distribution/Transfer Service

5. After completing time sample collection, the Coordinated Universal Time (UTC) transfer process begins using the TIME DISTRIBUTION (UTC Time Transfer) directive over the Proximity link by creating Time Transfer (TT) packet. This packet contains the correlation between UTC and the recipient's clock. Each creates Time Transfer (TT) containing the time correlation information and exchanges with each other. (However, we believe that this time directive requires additional processing time.)

6. After receiving TT packet, 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 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 to improve the accuracy. In this work, we like to address solutions to some of these described issues within the PITS protocol.

IV. PITS algorithm

The Proximity-1 Space Link Interleaved Time Synchronization (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 exchanges that can be integrated with CCSDS Proximity-1 Space Link Protocol. The core functionalities of PITS are similar to the ones used in terrestrial NTP[3]. In particular, the interleaved mode of PITS adopts the interleaved mode of NTP Interleaved On-Wire Protocol[4] which was first implemented and successfully tested during the summer of 2008. 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 Protocol with minimal modifications. PITS is benficial when the direct time synchronization from Earth to rover is difficult.

In addition, PITS can automatically cope with dropped or out-of-order 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. Therefore, this check routine can be easily implementable with existing time tagging methods and can be placed into the MAC Sublayer. For duplicate packet receptions, we may have to use other methods or external logic to detect.

In order to actually send a SpNTP packet, time tags captured by the Physical Sublayer must be converted to coordinate time. Each spacecraft maintains state variables necessary to convert its uncorrelated SCLK to coordinate time. These variables are initialized at the primary servers from messages received from Earth. Secondary servers initialize these variables using PITS.

The PITS protocol operates in a symmetric manner rather than client-server or master-slave relationship. The symmetric mode allows that any involved spacecraft can potentially distribute time to other spacecraft. A time server and a time client can be switched dynamically. For example, if Mars orbiter's clock is failed due to some reasons, then the rover that is synchronized to Earth can provide time to Mars orbiter. Therefore, the selection of time server depends on which end has been synchronized by Earth. Throughout this work, however, 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. This result can be easily extended to the case where rover provides time to a Mars orbiter. Further, we assume that the relative movements of orbiter and rovers are stationary during the time synchronization process as before. PITS can be operated in the following two modes: 1) Basic Symmetric Mode and 2) Interleaved Symmetric Mode. The details of each mode are explained in the next section.

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 SpNTP packets and this time information can be easily encapsulated in a SPDU. A time server and a time client exchange the SpNTP packets 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 or received. In addition, each server and client has the following 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 and handling state variables with time information received from SpNTP packets are slightly different between Basic and Interleaved Mode and these differences are explained in the next section.

A. PITS Basic Symmetric Mode

The example of packet exchanges between two spacecraft using Basic Symmetric Mode is given in Fig. 3. The payload in SpNTP packet as well as state variables at the time server and client are shown at each time instant. In particular, in Fig. 3, the time in dotted boxes indicate the spacecraft system clock, where the time that is stamped either at transmission or reception of SpNTP packet. The state variables and packet payload updates occur according to the *transmit* and *receive process* defined in Fig. 4 and 5.

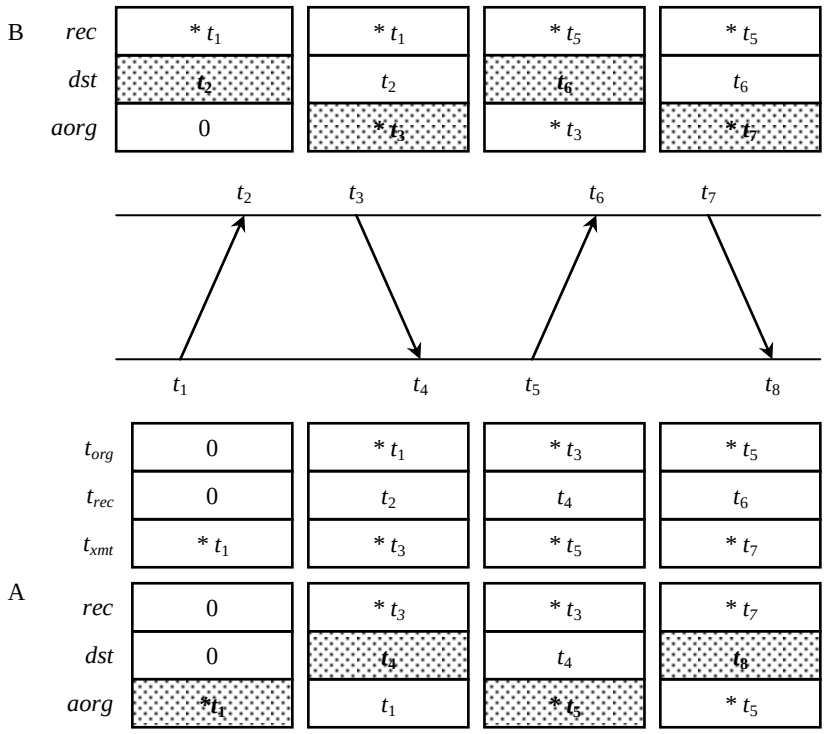

Figure 3. Illustration of Basic Mode showing SpNTP packet and state variables[4]

A server A initiates the synchronization process by sending a SpNTP packet to B as shown in Fig. 3, where this is defined as *Transmit Process* and pseudo codes[4] are given in Fig. 4. The flow chart in Fig. 6. is provided to pictorially illustrate the *Transmit Process* in Fig. 4, where SpNTP and peer variable in the flow chart indicate the content of SpNTP packet and the local state variables. In Fig. 4, each t_{org} and t_{rec} stores receive state variable, *rec*, and destination timestamp state variables, *dst*, respectively as shown in Fig. 4. Just before transmitting an origin

timestamp, state variable *aorg* stores the current time, *clock*. 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 4. Pseudo codes of Transmit Process for 1) Basic and 2) Interleaved Mode[4]

```
rec = t_xmt
dst = clock
T1 = t_org
T2 = t_rec
T3 = t_xmt
T4 = dst
If (b != 0)
   rec = t_rec
If (T1 == 0 || T2 == 0)
   Not Synchronized
else if (T1 != aorg)
   Out-of-order packet
```

```
Offset between A and B
= ½ [(T2 – T1) + (T3 – T4)]
RTT Delay between A and B
= (T4 – T1) – (T3 - T2)
```

Figure 5. Pseudo codes of Basic Receive Process[4]

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. 5 and Fig. 7, respectively. After receiving a SpNTP packet from A, *rec* state variable in B stores t_{xmt} timestamp. 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 RTT delay according to in Eqs. (1) through (4) as shown in Fig. 5. This received process is also described in the flow chart in Fig 7.

Hence, the *Transmit Process* occurs just before transmitting SpNTP packets and the *Receive Process* occurs right after receiving SpNTP packets[4]. This captures the fundamental difference between the Basic and Interleaved Mode and this operational difference is further explained in the subsequent section.

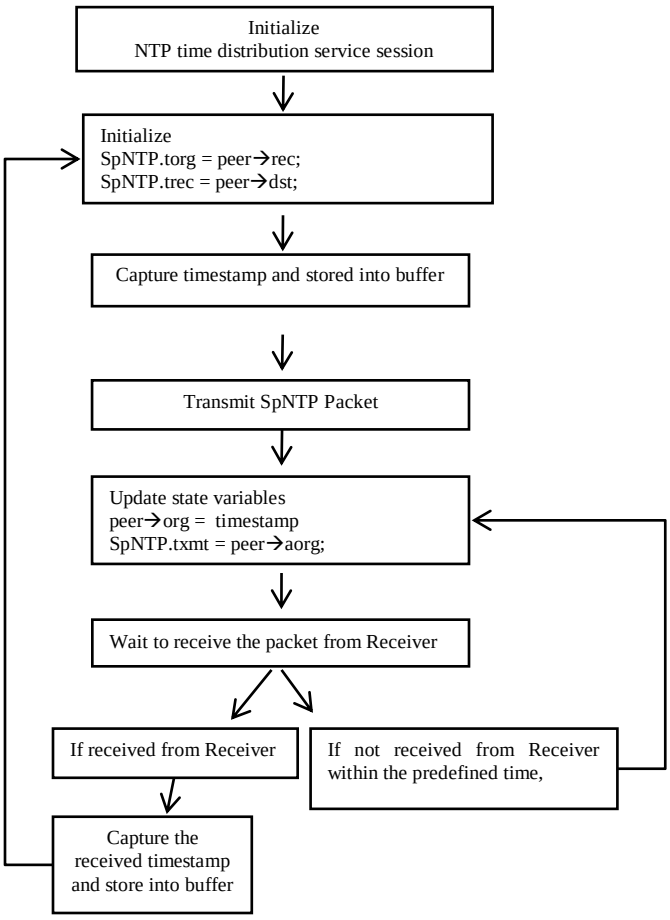

Figure 6. Flow Chart of Basic Transmit Process

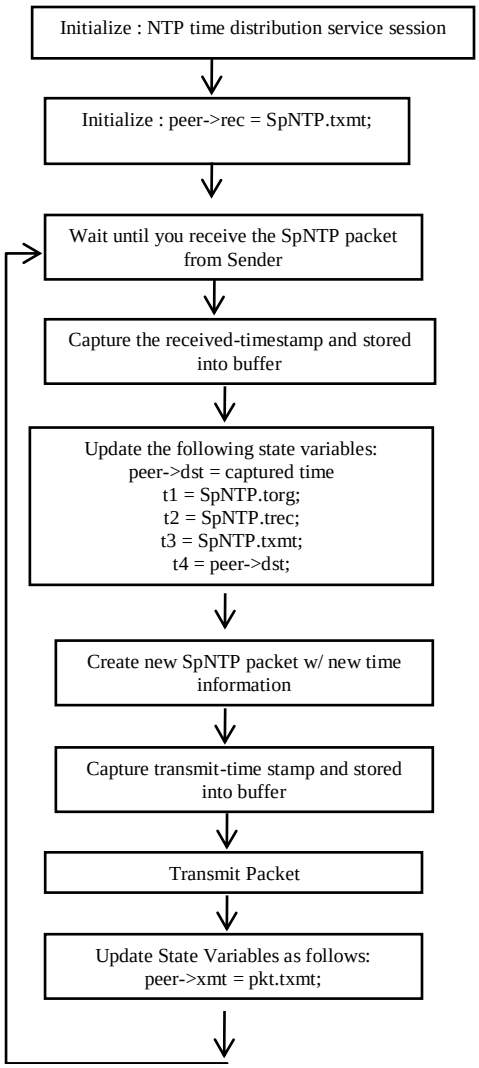

Figure 7. Flow Chart of Basic Receive Process

B. PITS 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 and interoperability 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. 8.

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 following SpNTP packet transmission, whereas the receive-transmit timestamp is immediately available as soon as a packet has been received. Figure 8 provides the example of exchanging SpNTP packets in the Interleaved Mode. In Fig. 8, 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 available and sent in the following SpNTP packet transmission unlike Basic Mode. Therefore, the receive-timestamp and the transmit-timestamp information from the other side are interleaved and available at every other round.

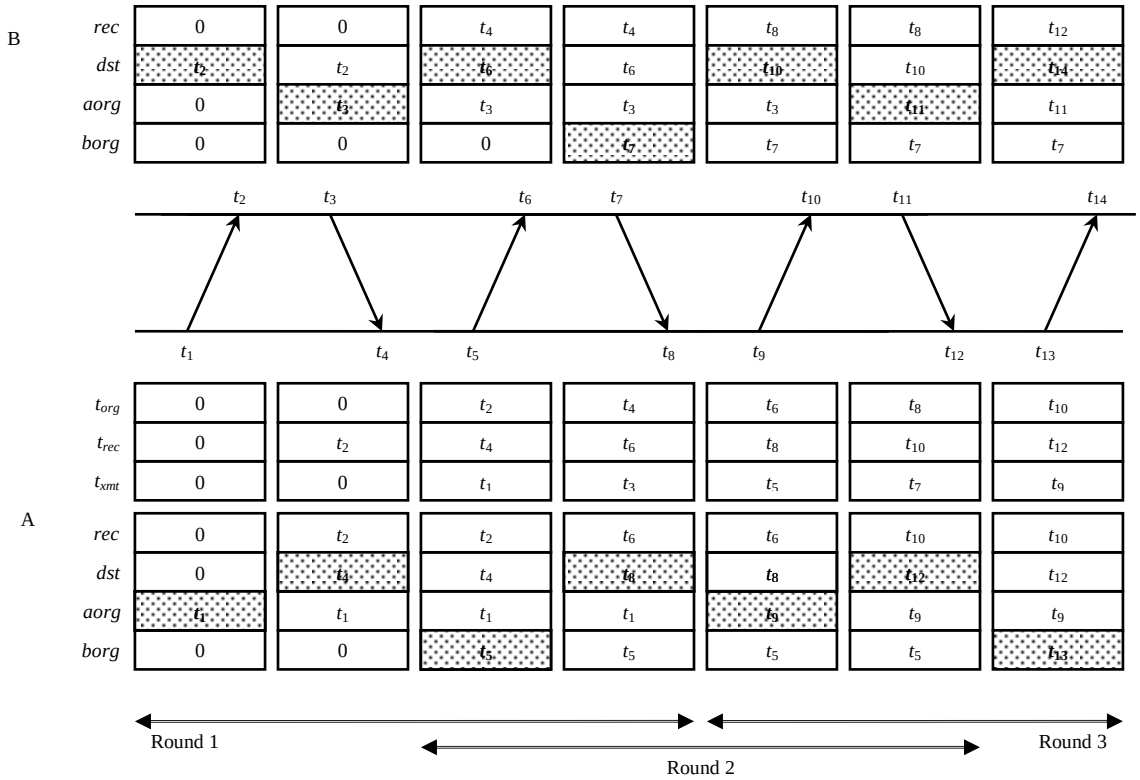

Figure 8. Illustration of Interleaved Mode showing SpNTP packet and state variables[4]

```
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)
{
   Not Sychronized
} else if (torg != 0 && torg != T4)
   Out of order packet
}
```

Figure 9. Pseudo codes of Interleaved Receive Process[4]

Therefore, 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 SpNTP 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.

Comparison between PITS and CCSDS Time Collection Services

The existing time collection method in CCSDS Proximity-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 as shown in Fig. 5 and 9.

The additional key difference between the proposed PITS algorithm and the conventional CCSDS Proximty-1 Time Collection services is described in Fig. 10, in terms of protocol interactions and time packet exchanges. As shown in the following Fig. 10.(b), in the existing time collection service, a time server keeps sending SPDU frames until the requested time interval or to collect the specified number of samples, where A*i* and B*j* indicates the time packet sent from A to B at *i*th transmission and the time packet sent from B to A at *j*th transmission respectively . In Fig. 10.(b), a time client B passively timetags the received SPDU frames. On the other hand, in PITS shown in Fig. 10.(a), when a time client sends back 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 samples 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 SpNTP packet. PITS time collection occurs concurrently whereas, in the existing approach, one waits until the completion of the other. PITS will require additional latency to send SPDUs back and forth to collect time samples.

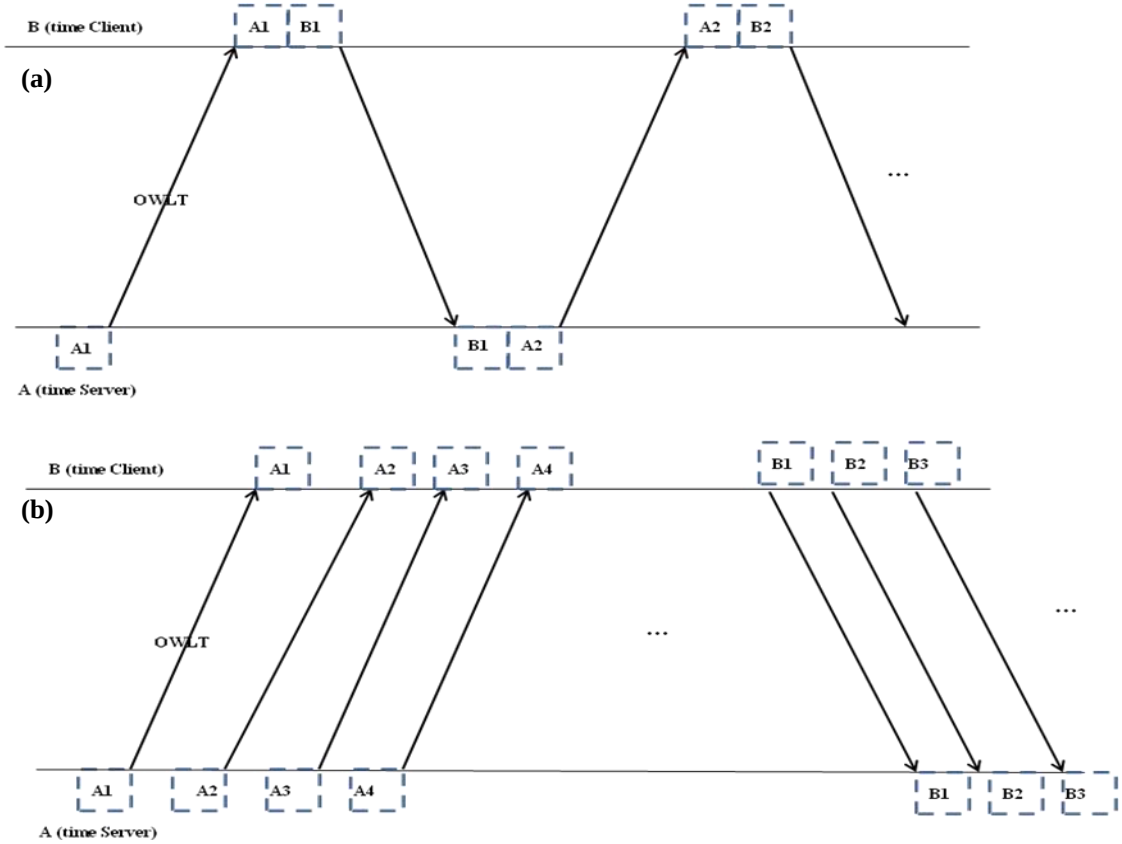

Figure 10. A side-by-side comparison between the (a) proposed PITS algorithm and the (b) conventional CCSDS Proximty-1 Time Collection Services

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, C&S Subayer, and Physical Sublayer. The MAC Sublayer has a capability to instruct the C&S Sublayer to intercept a 24-bit ASM used to delimit C&S frames, check CRC-32, captures a time tag from SCLK, and store the frame type and sequence

number from the frame header. Therefore, PITS can simply reuse the existing CCSDS Proximity-1 time tagging capabilities.

First, MAC Sublayer creates a SpNTP packet and encapsulates this into a SPDU frame and transmits and receives to and from the other remote spacecraft. 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-time stamp followed by the SPDU header. After that, the transmit time tag buffer is read, converted to coordinate time such as ephemeris time (ET) and saved in transmit state variables for the next message. Therefore, the state variables can be located in the MAC Sublayer to store the received time information and used for RTT delay and offset calculation.

In the current CCSDS Proximity-1 implementation, SPDU does not carry FSN and is not time-tagged. However, time-tagged U-frame that has FSN is used for time sample collection services. In the PITS design, we will time tag timestamp SPDUs and use Timestamp SPDUs to carry SpNTP packets instead of U-frames as shown below:

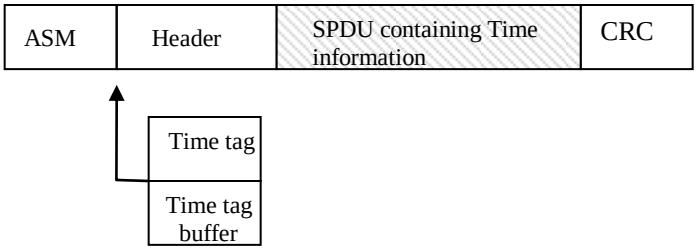

Figure 11. The proposed CCSDS Proximiy-1 Timestamp SPDU Frame including SpNTP packet

Therefore, on transmission, the C&S Sublayer recognizes the Timestamp SPDU, captures a time tag from SCLK and saves it in a buffer. On reception, the C&S Sublayer recognizes this SPDU, captures a time tag from SCLK and saves it in a buffer. Therefore, no sequence numbers are necessary and capture does not need to be enabled since it is always enabled for the Timestamp SPDU.

In the current CCSDS Time Code Formats[5], eight-octet data fields are used to represent time. The current variable-length SPDU has only for two timestamp fields for storing transmit and receive time tag. Therefore, we will need a minor change in the SPDU format to include original timestamp, having a total of three time fields.

If we consider the bandwidth consumption, Time Collection and Distribution service in general 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 Proximity-1 Space Link Protocol will potentially require a large on board buffer space to store the time samples which need to be predetermined in advance. 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 protocol 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. We used UDP packets for testing SpNTP packet interactions in PITS algorithm. In particular, for simplification, we used the software level clock, soft timestamping, (or network layer timestamping) using the OS Kernel system call method to time tag receive and transmit UDP packets. 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:

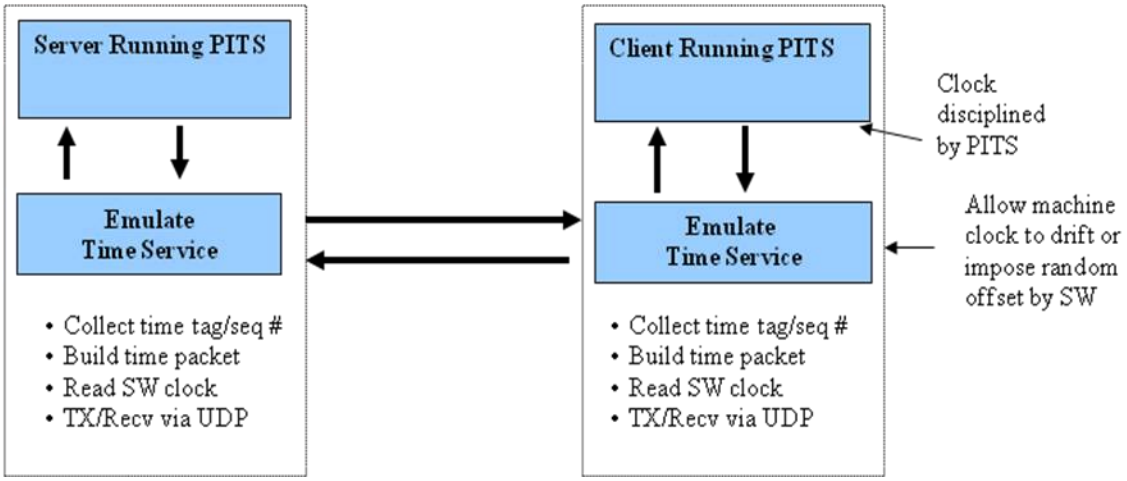

Figure 12. Simulation Setup between two Linux machines

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 time stamping strategy. However, we believe that software implementation is adequate enough to capture the packet exchange behaviors disciplined by the PITS algorithm and offset and RTT delay 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. With these assumptions and slight variations in the measurement, we have implemented the various subroutine calls needed for *Transmit Process* and *Receive Process* in Fig. 4 and 9. Developed subroutines are called in the following order and these procedures are briefly summarized: at a receiver, a packet arrives at the host the kernel and this will trigger the driver interrupt by calling *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* subroutine. Then at the protocol's discretion, *transmit packet to peer* subroutine 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 in order to generate intentional time deviations for from the time server. Therefore, we made that the client's clock ran slower than the time server's clock. This allows us observing the RTT delay and offset estimate difference at each time server and client.

Following is the simulation results obtained from the several experiments, where the blue lines in Fig. 13 and 14 indicate data collected from the time server and the red lines capture data from the time client. The unit in Y-axis is sec and the X-axis indicates the number of complete PITS rounds. Offsets are computed at each side using Eq (1) and (3). As we can observe, the computed offsets have almost same magnitude with the opposite sign. Since time server's clock ran faster than the client's clock, offset calculated at the server is positive. On the other hand, the sign of estimated offsets from the client is negative because client's clock ran slower. Hence, we can verify that PITS protocol correctly computes the offset. Also, each A and B correctly calculated the RTT delays using Eq (2) and (4).

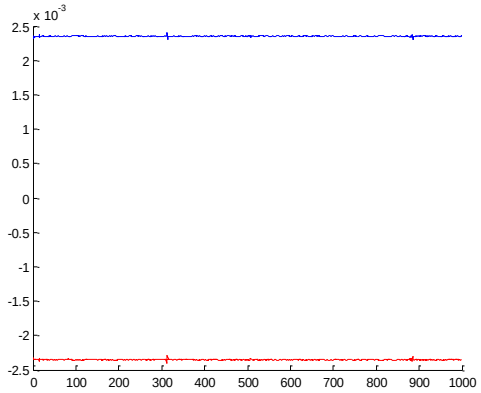

Figure 13. Offset results obtained between two Linux machines

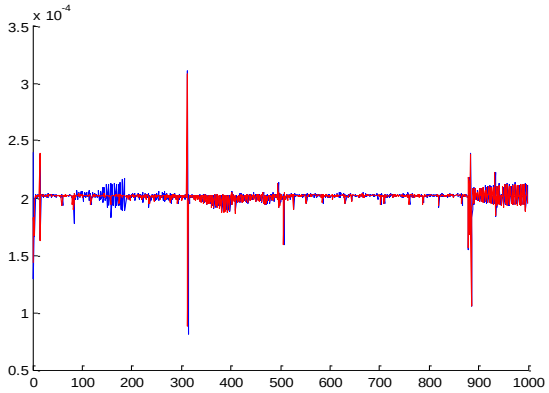

Figure 14. RTT delay obtained between two Linux machines

However, there were some spikes or outliers in this experiment as shown in Fig. 14 in the computed RTT delays. We believe that there were some other services such as OS kernel processes might cause some of spikes in Fig. 14. Also, we were not able to completely isolate network traffic that were generated from other machines in the network while taking measurements. Therefore, additional network traffic made RTT delays fluctuate.

Overall, the results accurately estimate the offsets and RTT delays. We have validated that the PITS algorithm and protocol is working properly between two Linux machines testing.

VII. Future Work and Conclusion

In this work, we described the NTP based PITS protocol to provide time services in Mars proximity domain. We presented the framework to provide the onboard time distribution capability at the spacecraft which would be a key component for *In Situ Time Distribution Service*. We verified the correctness of the PITS algorithm, validated the design, and provided the preliminary 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 through proximity links. Further prototype development and hardware integration and testing will show the timing accuracy improvement by exploiting the benefit of Interleaved Mode. This will further mature the implementation and resolve design issues arising from integrating PITS with [1]. Also, future work may be necessary for time attack or the incorporation with security.

In particular, the proposed PITS can be extended to multi-spacecraft formation flying missions to provide accurate and periodic time updates. Further, PITS can be considered as a time transfer option in the Space-Based Positioning, Navigation, and Timing (PNT) system[6] to enable integrated communications. Therefore, PITS is expected to significantly reduce the amount time and resources required for accurate time synchronization and distribution services for future NASA missions.

Acknowledgments

The authors 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]Time Synchronization for Space Data Links, http://www.cis.udel.edu/~mills/proximity.html
[3]The Network Time Protocol (NTP), http://www.ntp.org/
[4]NTP Interleaved On-Wire Protocol, http://www.cis.udel.edu/~mills/onwire.html
[5]CCSDS Time Code Formats, 301.0-B-3: https://ccsds.org
[6]The Positioning, Navigation and Timing (PNT) Overview, https://www.spacecomm.nasa.gov/spacecomm/programs/system_planning/PNT/default.cfm