This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE Globecom 2011 proceedings.

Analysis of Proximity-1 Space Link Interleaved Time Synchronization (PITS) Protocol

Simon S. Woo
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, USA
University of Southern California
Los Angeles, USA
Email: sungilu@jpl.nasa.gov

Abstract—**To synchronize clocks between spacecraft in proximity, the Proximity-1 Space Link Interleaved Time Synchronization (PITS) Protocol has been proposed. PITS is based on the NTP Interleaved On-Wire Protocol and is capable of being adapted and integrated into CCSDS Proximity-1 Space Link Protocol with minimal modifications. In this work, we will discuss the correctness and liveness of PITS. Further, we analyze and evaluate the performance of time synchronization latency with various channel error rates in different PITS operational modes.**

I. Introduction

Spacecraft at great distances from Earth require high-precision time information for accurate navigation and communications. Typically, an on-board computer carries a mission-elapsed time counter or space clock (SCLK)[1]; however, the SCLK is susceptible to reset, skew and drift due to external environmental effects such as radiation, temperature fluctuation, relativistic effects, as well as software updates or reboots. Inaccurate and misaligned timing can cause other catastrophic subsystem failures and jeopardize critical mission objectives. In addition, accurate timing knowledge would be required for enabling future multiple access communications between orbiter and rovers operations.

The Proximity-1 Space Link Interleaved Time Synchronization (PITS) Protocol was first designed and developed for synchronizing spacecraft that are in *proximity*, where *proximity* means less than 100,000 km apart. One particular application of PITS is time synchronization between a Mars orbiter and rover. In addition, PITS is capable of providing time synchronization and distribution services in more general situations, such as when multiple flying entities need to achieve time synchronization using a single point-to-point link. This is particularly attractive in deep space environments, where direct Earth-to-spacecraft contact is intermittent, and propagation delay is long such that direct time synchronization from Earth to spacecraft is extremely difficult to achieve. In this case, PITS can provide the local time synchronization and the distribution service until the link from Earth is available. For example, in a Mars network, an orbiter can function as a *time server* and a rover can function as a *time client,* thus exchanging time information with each other. For missions to the outer planets such as the Europa Jupiter System Mission (EJSM), a mothership can function as a *time server*, and a probe can function as a *time client* such that a mothership can provide the time distribution service to a probe.

So far, most time synchronization research has primarily focused on measuring errors in timing accuracy and offset, while paying little attention toward analyzing the underlying time synchronization protocol operations with link errors. In particular, there is no framework for analyzing the performance of space time synchronization protocols.

In this work, we provide the sketch of proof for the liveness and correctness properties of PITS, which are critical for guaranteeing the correct operations of the protocol. Also, we analyze the expected number of transmissions required to achieve synchronization with link errors under different PITS operation modes.

II. Background

A. Time Synchronization

Typically, time synchronization involving two peers can be achieved by exchanging three time information packets[2] as shown in Fig. 1, where a peer that has accurate time information is called the *time server,* and the other peer who wishes to be synchronized to the time server is called the *time client*. In Fig 1, peer A is the time server and peer B is the time client. The One Way Time (*OWT*) is defined to be half of the round-trip time (*RTT*). We denote t_1 as the transmit-timestamp of peer A, which is the packet transmission time at A. Also, we define t_2 as the receive-timestamp of peer B, when B receives the packet from A. Similarly, t_3 and t_4 are defined to be the transmit-timestamp at peer B and receive-timestamp at peer A respectively. The offset is defined as the relative time difference between nodes A and B. Once each peer obtains four consecutive timestamps as shown below in Fig. 1, the *RTT* delay and offset can be calculated as follows:

At peer A,

$$\text{offset} = \tfrac{1}{2}((t_2 - t_1) - (t_4 - t_3))$$
$$\text{RTT delay} = (t_4 - t_1) - (t_3 - t_2)$$

At peer B,

The original PITS research described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology and was sponsored by the National Aeronautics and Space Administration.

$$\text{offset} = \tfrac{1}{2}((t_4 - t_3) - (t_6 - t_5))$$
$$\text{RTT delay} = (t_6 - t_3) - (t_5 - t_4)$$

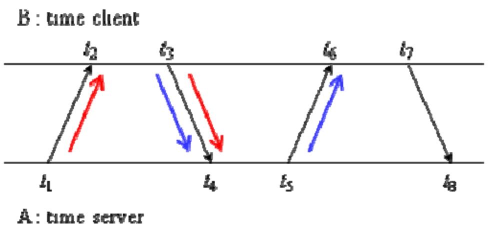

Fig 1. Illustration of Time Synchronization Process that starts at t_1 and ends at t_6

In general, the accuracy of time synchronization depends on the several factors including clock drift, skew, hardware processing delay, timestamping error, and propagation delay[5]. In this work, we primarily focus on analyzing and examining the PITS protocol interactions and provide the latency performance in terms of the number of time information exchanges required with different channel conditions.

B. *PITS Protocol*

PITS can be described as a simple request and reply protocol. As soon as the receiver receives the message from the sender, it immediately replies back with its own message. This message includes peer time information and further serves as an acknowledgement to the received message. PITS is a stateful protocol which keeps track of t_{org}, t_{rec}, and t_{xmt} time information in the payload in the SpaceNTP (SpNTP) packet [2-5] as shown in Table 1. In particular, each t_{rec}, and t_{xmt} is the received and transmitted time of SpNTP packet at each peer and t_{org} holds the transmitted time of SpNTP packet in the previous time slot. Time information t_{org}, t_{rec}, and t_{xmt} are updated when the new SpNTP packet is created. The actual SpNTP packet can be implemented and integrated into the CCSDS Proximity-1 Timestamp SPDU[6] where additional details on the SpNTP packet format and fields are described in [3,4]. Further, each peer has the *rec*, *dst*, *org*, *Aorg*, and *Borg* local state variables in its memory to store time information, t_{org}, t_{rec}, and t_{xmt} to calculate the RTT delay and offset.

Table 1. Timestamp and state variables used in PITS

	Descriptions
t_{org}	origin-timestamp
t_{rec}	receive-timestamp
t_{xmt}	transmit-timestamp
rec	receive timestamp state variable
dst	destination timestamp state variable
org	origin timestamp state variable used in BSM
Aorg, Borg	alternate origin timestamp state variables used in ISM

PITS can be operated in Basic Symmetric Mode (BSM), Interleaved Symmetric Mode (ISM), and Broadcasting Mode (BM)[2]. In this work, we mainly concentrate on BSM and ISM modes since we believe that the analysis of BM can be readily extendable based on this work.

First, in BSM, time server A initiates the synchronization by sending a SpNTP packet to B. Upon reception of a SpNTP packet at B, B updates its state variables *rec, dst*, and *org* and transmits a SpNTP packet with new time information. The packet exchange interactions in BSM are exactly the same as one shown in Fig. 1. The example of the PITS algorithm showing the SpNTP packet payload as well as the corresponding state variables in each peer at each time instant are explained in [2-5]. The details of state variables and packet payload updates occur according to the *Transmit* and *Receive process* presented in [2-5].

On the other hand, ISM operates in a slightly different manner than BSM. In ISM, time server A captures and records the timetag immediately after SpNTP packet is actually transmitted. Hence, a more accurate transmit-timestamp is obtained by capturing and timetagging the packet as close to the physical layer as possible and the transmit-timestamp is not included at the time of SpNTP packet transmissions, however, transmit-timestamp is included in the following SpNTP packet transmission. The obtained transmit-timestamp is encapsulated and sent in the following SpNTP packet transmission as shown in Fig 2.

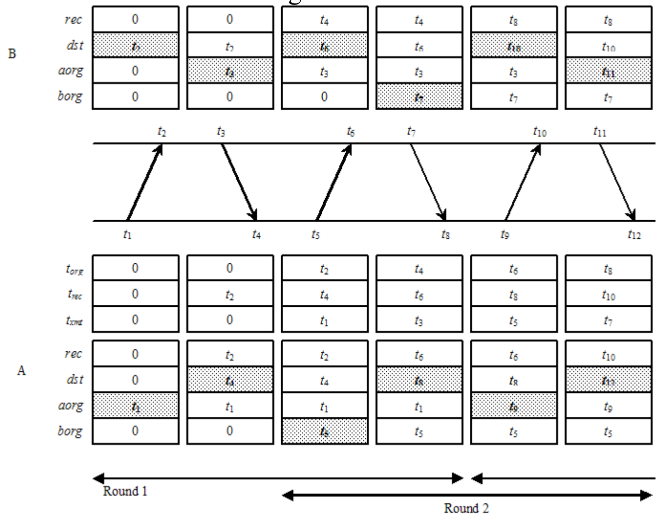

Fig 2. SpNTP packet Exchange with ISM[2-5]

On the other hand, the *Receive Process* in ISM operates in the same way as in BSM. Therefore, ISM requires more SpNTP packet transmissions than BSM because the transmit-timestamp must be transmitted in the following round. Also, instead of single origin timestamp *org*, two alternating *Arog* and *Borg* timestamps are used for storing t_{org} in ISM. At the expense of extra time exchanges, more accurate timing information can be captured in ISM mode. However, the link utilization of BSM and ISM are the same because both modes deliver the same number of timestamps in one RTT, where the link utilization is defined as a fraction of time that the link has been used. This interleaved feature in particular has been tested in the NTP On-Wire Protocol since 2005 and more details can be found in [2,5,7].

The PITS protocol operates in a symmetric manner, where any involved peers can initiate the time synchronization process providing flexibility and efficiency of operations. In addition, we also consider the server-client based operation where the orbiter functions as a time server and the rover functions as a time client. This mode would be applicable for the deep space mission where an orbiter or mothership typically carries a more accurate clock than the one in a rover, providing the time services to a rover.

C. Assumptions

We make the following assumptions throughout this work: each SpNTP packet has a fixed packet size, which is known among peers a priori. Also, we assume that the *OWT* from A to B and B to A does not change during the time synchronization process. Therefore, the effect of relative velocity changes due to spacecraft movements is not included in the proposed work. Although the relative distance changes, different internal hardware processing delay, asymmetrical bandwidth, and channel conditions can cause different *OWT*, we believe that the several SpNTP packets can be rapidly exchanged in a short period of time over the link in *proximity*. Therefore, we focus our analysis on capturing the theoretical performance and the underlying PITS protocol behavior rather than capturing the performance under specific hardware and trajectory constraints. In addition, we assume that there are i.i.d. SpNTP packet errors on each link, where the probability of packet error is never equal to one so that there is always a small probability of success on SpNTP packet transmissions by persistently sending packets. We do not consider the packet loss during the time synchronization. Throughout this work, we assume that the underlying CCSDS Proximity-1 Space Link Protocol provides the perfect CRC check.

III. CORRECTNESS OF THE PITS PROTOCOL

The correctness guarantees the integrity of the protocol operations. Furthermore, the correctness property provides correct functioning of the designed protocol. The correctness proof requires proving that packets are delivered to the higher layer of the receiver in order, and without deletion or repetition. Protocol correctness is established by showing the 1) liveness and 2) safety properties[8]. The liveness property guarantees that the protocol will continue to produce results. The safety condition guarantees that the protocol will never produce an incorrect result. In other words, once a packet has been received, packets are delivered error-free and in proper order to higher layer application for the offset and RTT delay calculation. First we will describe the liveness and then show the safety property of PITS.

A. Liveness of PITS Protocol

The protocol liveness property guarantees that the protocol never falls in the deadlock conditions. We need to show that peers will eventually collect a total 4 consecutive timestamps to achieve synchronization in BSM. We prove the liveness using a contradiction as follows:

(Proof) Suppose there is a deadlock and the currently received transmit-timestamp is equal or less than the previously received timestamp:

$$t_{xmt} \leq xmt,$$

where *xmt* is previously received timestamp.

In this case, the PITS protocol will not move forward simply ignoring the received packet. Let $xmt = i^*$ at time t^* for all $t^* \leq t$, where i^* is previously received transmit-timestamp, and let $\tilde{t}$ be some average time after t. Further, if we assume the timeout period to be constant, $T < \infty$; which is the time that a time server has to wait to send next SpNTP packet if SpNTP packet does not sent back from the other peer. With above fact, we need to consider the following different cases: $t_{xmt} > xmt$, $t_{xmt} = xmt$, and $t_{xmt} < xmt$.

Case 1: $t_{xmt} > xmt$

This is no deadlock condition. The current transmit timestamp received from a sender is greater than the previously received transmit timestamp from a sender due to causality of timestamps. Therefore, in this case, PITS protocol moves forward, sending its SpNTP packet back to the other peer.

Case 2: $t_{xmt} = \tilde{t} = xmt$

If $t_{xmt} = xmt$, the peer will persistently send a new SpNTP packet after every T, if it does not hear SpNTP packet from the other peer. After some time later persistently sending packets, t_{xmt} becomes

$$t_{xmt} = \tilde{t} + nT > xmt,$$

where n is an integer. Therefore, a protocol moves forward since

$$t_{xmt} > xmt.$$

Case 3: $t_{xmt} = \tilde{t} < xmt$

The same reasoning applies to case 3 as case in 2. After some number of timeout period, a peer transmits SpNTP packets persistently and t_{xmt} will eventually become greater than *xmt* at some time instance,

$$t_{xmt} = \tilde{t} + mT > xmt,$$

where m is an integer.

Thus, these prove the liveness of PITS. This proof can be easily extended for ISM mode.

B. Safety of PITS protocol

Since the space environment is harsh and errors can occur during packet transmissions, the safety property is critical and must be maintained in order to provide correct results under severe link errors. In general, safety means the protocol does not deliver erroneous packets nor packets with the incorrect time information to the higher layer. The safety condition should properly handle the erroneous as well as out-of-order, and duplicate packets. The safety is examined in the following three cases:

1) **Erroneous packet**: We assume that the underlying error correcting codes and CRC can detect errors. Thus PITS will not forward erroneous packets to the higher layer. Based on our assumption this is trivially true. In this work, we further explore the exceptional case where a receiver can make a use of the received time of erroneous packet even though PITS cannot decode the content of the erroneous packet. We denote this as an error-torrent timestamp. Therefore, PITS can only make a use of the received timestamp information for the offset calculation at the physical layer and does not forward, discarding the content of the erroneous packets. Although this

scenario is somewhat an unrealistic operational case, it can be realizable only if we have dedicated time synchronization sessions only allowing the exchanging of SpNTP packets during this period. This is further explored in later sections.

2) **Duplicate packet**: The protocol can detect a *duplicate packet* if $t_{xmt} = xmt$, where *xmt* is a local variable that stores the t_{xmt} value from the previous packet reception.

(Proof) If t_{xmt} in the currently received transmit-timestamp is the same as the previously received timestamp, *xmt* , then the current packet must be the same as the previous packet. Packet retransmissions may occur due to packet loss or malicious replay of the same packet. Therefore, the protocol simply ignores the duplicate without updating any state variables.

3) **Out of order packet**: The protocol can detect the *out of order packet* if t_{org} is different from the *dst* state variable.

(Proof) Each *org, Aorg* or *Borg* state variable is not changed during one RTT as shown in Fig 2. and is only updated after successful reception of a packet from the other side. Therefore, if *org*, *Aorg* or *Borg* is different from t_{org}, then it must be the out of sequence packet. In this case, PITS simply ignores the received packet but updates the state variables accordingly. Therefore, PITS provides the additional protocol level protection achieving a high level of safety conditions.

IV. Analysis and Evaluation

We first consider the server-client operation mode where time synchronization is always started from the time sever A. Next, we analyze the peer-to-peer operation when any nodes can be a time server and initiate the time synchronization process. We evaluate the total latency required until successful synchronization between two nodes. Since the PITS time synchronization performance is essentially equivalent to the number of SpNTP packet exchanges, we define the random variable $\boldsymbol{X}$ to represent the total time incurred until the first successful time synchronization. The random variable $\boldsymbol{X}$ is measured as a multiple of *OWT*. Further, we define timeout *T* as a multiple of *OWT,* such as $T = n{\cdot}OWT$, when *n* is an integer.

In particular, we denote $\boldsymbol{X_{BSM}}$ and $\boldsymbol{X_{ISM}}$ for BSM and ISM mode respectively. We assume that the path characteristics between A and B are statistically equivalent during the time synchronization. Therefore, we consider the i.i.d. channel error rate *p* in the link. In addition, we specifically consider four cases based on the location of error occurrence during the SpNTP packet transmission and different operational modes as follows: 1) Packet Error in Client-Server mode, 2) Packet Error with valid Receive-Timestamp in Client-Server mode, 3) Packet Error in Peer-To-Peer Mode, and 4) Packet Error in Peer-to-Peer Mode with valid Receive-Timestamp.

For each of these four cases, we derive the expected latency required between two nodes A and B and denote the random variable $\boldsymbol{X_{BSM}}$ and $\boldsymbol{X_{ISM}}$ for case 1), $\boldsymbol{X_{BSM}^{Partial}}$ and $\boldsymbol{X_{ISM}^{Partial}}$ for case 2), $\boldsymbol{X_{BSM}^{Peer}}$ and $\boldsymbol{X_{ISM}^{Peer}}$ for case 3) and $\boldsymbol{X_{BSM}^{Peer\text{-}Partial}}$ and $\boldsymbol{X_{ISM}^{Peer\text{-}Partial}}$ for case 4), respectively.

1) **Packet Error in Client-Server mode**: This is the nominal client-server data transmission scenario. If the packet is received with an error at node, node simply ignores and drops the erroneous packet and does nothing until sender times out and resends a new SpNTP packet. In this case, all three consecutive transmissions need to be successful from time server A to time client B to collect four consecutive time stamps. Whenever there are packet errors from A to B, the packet is retransmitted from A after *T*. Hence,

$$E[X_{BSM}] = \{3 + \frac{pn}{(1-p)^2(1-p)} + \frac{2p}{(1-p)(1-p)} + \frac{(2+n)p}{(1-p)}\}OWT. \quad (1)$$

Similarly, five consecutive transmissions initiated from A need to be successful in ISM to collect as follows,

$$E[X_{ISM}] = \{5 + \frac{pn}{(1-p)^3(1-p)^2} + \frac{p(2+n)}{(1-p)^2(1-p)} + \frac{4p}{(1-p)(1-p)} + \frac{(4+n)p}{(1-p)}\}OWT. \quad (2)$$

2) **Packet Error with valid Receive-Timestamp in Client-Server mode**: In this case, we consider that node still makes a use of the received-time of erroneous packet (error-tolerant timestamp) and transmits a new packet back to the other peer. Therefore, we allow PITS protocol to collect the error-tolerant timestamp from the reception of erroneous packet, even though it cannot make a use of the payload of the SpNTP packet.

By examining protocol interactions and state variables more carefully in Fig 2, we can observe that the first SpNTP packet from A to B does not contain timestamp needed for B's offset or round trip calculation. Also, A already has timestamp information before transmitting to B. Therefore, the first data transmission can be error-tolerant and have errors from A to B without impacting the correct operation of PITS. However, follow-on packet content from second (B$\rightarrow$A) and third (A$\rightarrow$B) transmissions are necessarily for calculating offset and delay at A and B. Since receiver sends immediately its own packet upon the reception of erroneous packet, there are no timeout at the sender. Therefore,

$$E[X_{BSM}^{Partial}] = \{3 + \frac{2p}{(1-p)(1-p)} + \frac{4p}{(1-p)}\}OWT. \quad (3)$$

Similarly, in ISM, the first transmission can be error-tolerant with the same reason as above. Regardless of the success of the first transmission, next 4 consecutive transmissions must be error-free. Hence,

$$E[X_{ISM}^{Partial}] = \{5 + \frac{2p}{(1-p)^2(1-p)^2} + \frac{4p}{(1-p)^2(1-p)} + \frac{4p}{(1-p)(1-p)} + \frac{6p}{(1-p)}\}OWT. \quad (4)$$

3) **Packet Error in Peer-To-Peer Mode**: If there are errors in this mode, time client and time server peer can be immediately switched. Upon reception of erroneous packet, time client peer can restart the time synchronization process immediately becoming a new time server peer. Therefore, it can save timeout period. This case can be modeled as mean time for the specific patterns to occur as discussed in [9], since time synchronization can be dynamically started from any peers during the time synchronization process. If we

enumerate transmission success as "1" and failure as "0" in any link, then the problem is equivalent to obtaining the following three consecutive ones "111" for BSM and five consecutive ones "11111" for ISM mode. The mean time to obtain the pattern "111" regardless of which node initiated is:

$$E[X_{BSM}{}^{Peer}] = E[X_{111}] = \frac{1}{1-p} + \frac{1}{(1-p)^2} + \frac{1}{(1-p)^3}. \quad (5)$$

Similarly, in ISM

$$\begin{aligned} &E[X_{ISM}{}^{Peer}] = E[X_{11111}] \\ &= \frac{1}{1-p} + \frac{1}{(1-p)^2} + \frac{1}{(1-p)^3} + \frac{1}{(1-p)^4} + \frac{1}{(1-p)^5}. \end{aligned} \quad (6)$$

4) **Packet Error in Peer-to-Peer Mode with valid Receive-Timestamp**: We allow each peer can make a use of receive-timestamp of erroneous SpNTP packets as described in above case 2). In this case, first transmission can be either "0" or "1". Therefore, we are looking for the pattern "*x*11" two consecutive 1's followed by "*x*" which can be either "0" or "1". Therefore,

$$E[X_{BSM}{}^{Peer-Partial}] = 1 + E[X_{11}] = 1 + \frac{1}{1-p} + \frac{1}{(1-p)^2}. \quad (7)$$

In ISM, we have

$$\begin{aligned} E[X_{ISM}{}^{Peer-Partial}] &= 1 + E[X_{1111}] = 1 + \frac{1}{1-p} + \frac{1}{(1-p)^2} \\ &+ \frac{1}{(1-p)^3} + \frac{1}{(1-p)^4}. \end{aligned} \quad (8)$$

Next, we evaluated the various expected values of ***X*** from Eqs (1) – (8) with *p* from 0 to 0.25 to evaluate the PITS performance in the various channel conditions. For convenience, we used $T = 2.5OWT$ assuming $OWT = 1$ time unit in the simulation. In Fig. 3, the X-axis is the channel error probability *p* and the Y-axis is the expected latency required for the successful time synchronization measured in *OWT*.

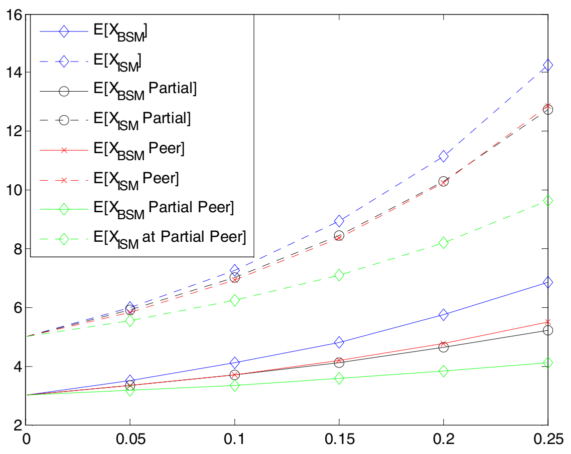

Fig 3. Expected Latency required (measured in *OWT*) vs. *p*

The dotted lines capture the results in ISM mode and solid lines are the results obtained in BSM mode. These provide several intuitions for PITS protocol operations. First, we can observe that BSM requires less time for synchronization than ISM mode and the differences increase as *p* increases. Secondly, the peer-to-peer mode performs better than the client-server approach as channel error increases regardless of including error-tolerant transmit-timestamp information. The gain is obtained because time synchronization process starts without timeout between peers. Further, if we make a use of available receive-timestamp, the performance gain can be achieved by 1*OWT* and 7*OWT* for BSM and ISM mode respectively at $p = 0.25$. This clearly shows the benefit of including error-tolerant timestamps. In summary, if we want the fast time synchronization, we can use the receive-timestamp of erroneous packets with peer-to-peer mode. If we want to obtain more accurate time synchronization performance based on only using error-free packets, then we can use the ISM mode with server-client mode.

V. Conclusion and Future work

In this work, we presented the correctness of PITS protocol and carefully investigated how packets received with errors can still benefit the time synchronization. The low-overhead time synchronization methods in PITS can operate under various conditions which provide the flexibility in time synchronization and distribution services. Future work will be necessary to more accurately characterize the performance and constraints in the operational environment by incorporating realistic spacecraft trajectories.

Acknowledgment

The authors would like to thank Dave Mills at Univ. of Delaware for developing original PITS protocol and other colleagues at JPL and anonymous reviewers for reviewing the initial manuscript of this work.

References

[1] "Basics of Space Flight." http://www2.jpl.nasa.gov/basics/bsf2-3.php.

[2] D. Mills. "Analysis and Simulation of the NTP On-Wire Protocols." http://www.eecis.udel.edu/~mills/onwire.html.

[3] S. S. Woo, J. Gao, and D. Mills. "Space Time Distribution and Synchronization Protocol Development for Mars Proximity Link," AIAA SpaceOps 2010.

[4] S. S. Woo, J. Gao, and D. Mills. "Space Time Distribution and Synchronization Protocol Development for Mars Proximity Link" in *Progress in Astronautics and Aeronautics, Space Operations:Exploration, Scientific Utilization & Technology Development*, *AIAA,* 2011.

[5] Mills, D.L. *Network time synchronization –The Network Time Protocol on Earth and in Space*, Second Edition. CRC Press, 2010

[6] *Proximity-1 Space Link Protocol—Data Link Layer*, Recommendation for Space Data System Standards, CCSDS 211.1-B-3., Blue Book, Issue 3, Washington, DC, CCSDS, March 2006.

[7] Mills, D.L, J. Martin, E., et al. "Network Time Protocol Verstion 4: Protocol and Algorithm Specification", RFC 5905, IETF, June 2010.

[8] Nancy A. Lynch, Distributed Algorithms, Morgan Kaufmann Publishers Inc., San Francisco, CA, 1996 .

[9] S. M. Ross, Introduction to Probability Models, 10th Ed. Academic Press, 2010.