TCP-Planet: A Reliable Transport Protocol for InterPlaNetary Internet

Ian F. Akyildiz Özgür B. Akan Jian Fang

Broadband & Wireless Networking Laboratory
School of Electrical & Computer Engineering
Georgia Institute of Technology, Atlanta, GA 30332
Tel: (404) 894-5141 Fax: (404) 894-7883
Email:{akan,jfang,ian}@ece.gatech.edu

***Abstract*— The space exploration missions are crucial for acquisition of information about the space and the universe. The entire success of a mission is directly related to the satisfaction of its communications needs. For this goal, the challenges posed by the InterPlaNetary Internet need to be addressed. Current TCP protocols have very poor performance in the InterPlaNetary Internet which is characterized by extremely high propagation delays, link errors, asymmetrical bandwidth and blackouts. The window-based congestion control, which injects a new packet into the network upon an ACK reception, is responsible for such performance degradation due to high propagation delay. Slow start algorithms of the existing TCP protocols further contribute to the performance degradation by wasting long time periods to reach the actual data rate. Moreover, wireless link errors amplify the problem by misleading the TCP source to unnecessarily throttle the congestion window. The recovery from erroneous window decrease takes certain time, which is proportional to the round-trip time (RTT) and further decreases the network performance.**

In this paper, a reliable transport protocol, TCP-Planet, is presented for data traffic in the InterPlaNetary Internet. It is intended to address the challenges and achieve high throughput performance and reliable data transmission on deep space links of the InterPlaNetary Internet. TCP-Planet deploys an end-to-end rate-based additive-increase multiplicative-decrease (AIMD) congestion control, whose AIMD parameters are tuned to help avoid throughput degradation. TCP-Planet replaces the inefficient slow start algorithms with a novel Initial State algorithm which allows to capture link resources in a very fast and controlled manner. A new congestion detection and control mechanism, which decouples congestion decision from single packet loss, is developed to avoid the erroneous congestion decisions due to high link errors. In order to reduce the effects of blackout conditions on the throughput performance, TCP-Planet incorporates Blackout State procedure into the protocol operation. Bandwidth asymmetry problem is addressed by the adoption of delayed SACK. Simulation experiments show that TCP-Planet significantly improves the throughput performance and addresses the challenges posed by the InterPlaNetary Internet.

***Index Terms*— Reliable Transport Protocol, InterPlaNetary Internet, High Propagation Delay, Bandwidth Asymmetry, Blackouts.**

I. Introduction

THE developments in the space technologies in the last decade have enabled the realization of deep space scientific missions such as Mars exploration. These missions produce significant amount of scientific data to be delivered to the Earth. For successful transfer of scientific data and reliable navigational communications, NASA enterprises have outlined significant challenges for development of next-generation space network architectures. The next generation deep space networks are expected to provide communication services for scientific data delivery and navigation services for the explorer spacecrafts and orbiters [6]. The next step in the design and development of deep space networks is expected to be the Internet of the deep space planetary networks and defined as InterPlaNetary (IPN) Internet [25].

A typical deep space network architecture shown in Fig. 1 is proposed for the Mars Exploration mission [7]. The architectural elements of the proposed infrastructure can be summarized as follows:

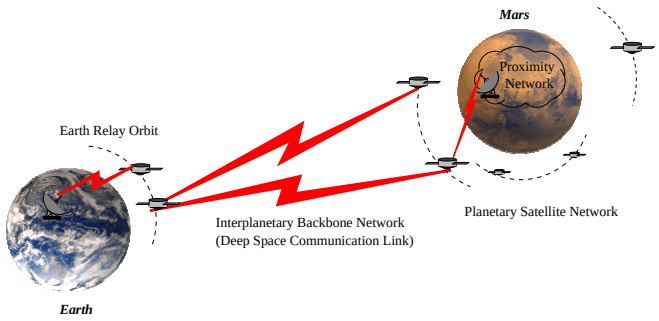

Fig. 1. Deep space network architecture for the Mars Exploration mission.

- **Interplanetary Backbone Network**: It includes the direct link or multi-hop paths between the outer-space planets and the Earth as well as the Earth-based infrastructure elements such as ground station for deep space network.
- **Planetary Satellite Network**: It consists of the satellites orbiting the planets to provide communication relay and navigation services to the surface elements.
- **Planetary Proximity Network**: It provides the communication links between diverse range of surface elements, which are often spread out to form an ad-hoc network.

Among the above architectural elements of the InterPlaNetary Internet, we mainly focus on the Interplanetary Backbone Network where the source and sink end-points are basically

relay satellites orbiting around the planets. This is because the Interplanetary Backbone Network plays a significant role for the performance of entire deep space communication. The most important characteristics and the challenges posed by the Interplanetary Backbone Network are listed as follows:

- **Very long propagation delays**: The deep space communication links may have extremely long propagation delays. For example, the end-to-end round trip time for the Mars-Earth communication network varies from 8.5 minutes to 40 minutes according to the orbital location of the planets [12].
- **High link error rates**: The bit error rates on the deep space links are very high usually on the order of 10^{-1} [12].
- **Blackouts**: Periodic link outages may occur due to orbital obscuration with the loss of line-of-sight because of moving planetary bodies, the interference of an asteroid or a spacecraft [6].
- **Bandwidth asymmetry**: The asymmetry in the bandwidth capacity of forward and reverse channels is typically on the order of $1000:1$ in space missions [12].

These challenges need to be addressed in order to meet the communication requirements of deep space missions. However, the existing TCP variants [16], [17], [13], [20], [8], [21], [9], [3] have been shown to achieve very poor performance in deep space communication networks [1]. The dominant factor in this performance degradation is the extremely high propagation delay in deep space links [1]. This is solely due to the window-based mechanism used by the current TCP protocols during slow start and congestion avoidance algorithms. In the slow start algorithm, the congestion window size (W) is increased by one packet per received ACK until the slow start threshold (W_{ss}) is reached, i.e., $W < W_{ss}$. However, this approach wastes the link resources for a very long duration which is proportional to the propagation delay. For $W_{ss} = 20$ and $RTT = 20$ minutes, it is shown in [1] that the slow start algorithm cannot utilize the link resources for approximately 120 minutes in deep space links.

The inefficiency in link utilization due to window-based mechanisms also exists during the congestion avoidance phase, i.e., $W \geq W_{ss}$, where the TCP source increments the congestion window size by roughly one at each RTT. The performance evaluation study in [1] shows that window-based TCP protocols achieve throughput of approximately 10 bytes/s for the link capacity of 1 Mb/s, packet loss probability of $p = 10^{-3}$ and $RTT = 40$ minutes. In other words, the entire deep space link remains almost unutilized during the entire connection period. Note that $RTT = 40$ minutes is within the RTT range for communication links between Mars and Earth, i.e., 8.5 to 40 minutes based on the orbital position [12].

Furthermore, the current TCP protocols are designed for wired links, which are reasonably assumed to have negligible bit error rates. Therefore, they invoke congestion control mechanisms in case of a single packet loss. However, this assumption does not hold in deep space communication links. Consequently, the packet loss based congestion detection mechanism results in unnecessary rate throttle and leads to severe throughput degradation. Much research has been performed in recent years in order to address the throughput degradation due to wireless link errors [5]. However, these solutions cannot be directly applied to Interplanetary Backbone Network because of the amplifying effects of the extremely high propagation delay and the other abovementioned characteristics on the problem.

The TCP performance on the links with high bandwidth-delay products and errors is analyzed in [18]. Many transport protocols [15], [2], [3] are proposed for satellite links, which are also characterized by high bandwidth-delay products and high bit error rates. Nevertheless, these studies mostly refer to Geo-stationary Earth Orbit (GEO) satellite links with typical RTT values around 550 ms, which are very low compared to RTTs in deep space communication links. Moreover, packet losses due the blackout conditions may also mislead the congestion control mechanisms based on packet losses. In [14], an enhancement for TCP is developed to address signal loss conditions due to mobility. However, the blackout situations in deep space links are much more complicated due to extremely high propagation delay and hence solutions as in [14] cannot be applied directly.

There are more challenges which need to be addressed by the new transport protocols in Interplanetary Backbone Network. These challenges are consequences of the characteristics of the deep space links, and can be summarized as follows:

- **Delayed Feedback**: TCP is expected to respond to network state. This expectation creates problems in long-delay environments, since TCP uses end-to-end signaling for its control loops. The higher RTT is experienced, the older information about link conditions is received at the source. Thus, the congestion control decision based on such past information might not lead to proper action. Therefore congestion control schemes, which react to instantaneous packet loss situations, do not yield proper response on the links with high propagation delay.
- **Buffer Size**: In order to assure 100% reliable transport, retransmission mechanism is inevitable. However, this brings considerable amount of memory requirement. For example, the transport protocol source should maintain 1.2 GB buffer size for $RTT = 20$ minutes and the average data transmission rate of 1MB/s.

There already exists an active research on transport layer protocols for space-based communication networks. Space Communications Protocol Standards-Transport Protocol (SCPS-TP) [11], [10] is a set of TCP extensions developed by Consultative Committee for Space Data Systems (CCSDS) for space communications. SCPS-TP is designed to support current communication environments and those of upcoming space missions [10]. SCPS-TP is developed based on the existing TCP protocols with some modifications and extensions to address the challenges posed by space-based systems such as link errors, bandwidth asymmetry, and link outages. It can provide full, best-effort and minimal reliability according to the mission specific communication requirements. The capabilities of the SCPS-TP are basically a combination of existing TCP protocols, which are shown to be inadequate in

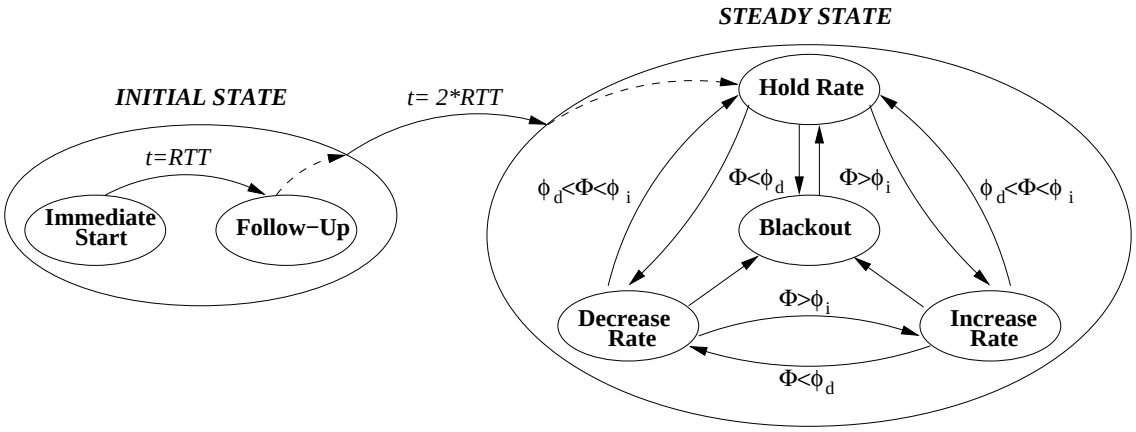

Fig. 2. TCP-Planet protocol operation state diagram including substates and the state transitions based on congestion control decision mechanism.

addressing the challenges in Interplanetary Backbone Network [1]. For example, SCPS-TP with Vegas congestion control uses window-based scheme and adopts slow-start algorithm. Although the rate-based version of SCPS-TP is under development, it disables congestion control mechanism and performs transmission with user selected fixed rate [24]. On the other hand, for example, SCPS-TP uses TCP-Vegas [8] congestion decision mechanism based on the RTT variation. However, since the window-based nature of TCP-Vegas cannot fully utilize the link, it is not even possible for it to experience the congestion and hence variation in RTT. Therefore, congestion decision based on RTT variation does not provide proper congestion control functionality. Furthermore, due to the extremely high propagation delay, the variation in RTT may not be measured accurately such that the resultant congestion control behavior may also not be accurate.

As pointed out in [1], there exists an urgent need for reliable transport protocol for InterPlaNetary Internet. In this paper, a reliable transport protocol, TCP-Planet, for InterPlaNetary Internet (IPN) is presented. The objective of TCP-Planet is to achieve high throughput performance and reliable data transmission by addressing the challenges in the IPN. In order to address the challenges due to extremely high propagation delay, TCP-Planet deploys a newly developed end-to-end rate-based additive-increase multiplicative-decrease (AIMD) congestion control, whose AIMD parameters are adjusted to compensate for the throughput degradation. Two novel algorithms, i.e., Initial State and Steady State, constitute the structure of the TCP-Planet protocol. Initial State algorithm replaces the inefficient slow start algorithms in order to capture link resources in a very fast and controlled manner. In Steady State, a new congestion detection and control mechanism is deployed to minimize the erroneous congestion decisions due to high link errors. In order to reduce the effects of blackout conditions on the throughput performance, TCP-Planet incorporates Blackout State procedure into the protocol operation. Bandwidth asymmetry problem is addressed by the adoption of delayed SACK options. Performance evaluation via simulation experiments reveals that TCP-Planet significantly achieves high throughput performance and addresses the challenges in deep space communication networks.

The remainder of the paper is organized as follows. TCP-Planet protocol overview along with the detailed operation of the Initial State algorithm is presented in Section II. In Section III, Steady State algorithm including the new rate-based AIMD congestion control and Blackout State behavior are explained. Performance evaluation presented in Section IV is followed by the concluding remarks stated in Section V.

II. TCP-PLANET: INITIAL STATE

TCP-Planet source starts connection in the *Initial State* at $t = 0$ by calling `Initial_State()` algorithm as shown in Fig. 2. TCP-Planet source then goes to the *Steady State* by calling `Steady_State()` algorithm at $t = 2 \cdot RTT$ as shown in Fig. 2.

The slow start algorithm used in the existing TCP protocols is shown to be inefficient on deep space links of the InterPlaNetary Internet [1]. In order to avoid performance degradation due to Slow Start algorithm, TCP-Planet deploys `Initial_State()` algorithm, which achieves to capture link resources in a very fast and controlled manner. The algorithm is composed of two main procedures, i.e., *Immediate Start* and *Follow-Up*.

A. Immediate Start

TCP-Planet starts a connection in the *Immediate Start* state at $t = 0$, where the actual RTT is divided into time intervals of size T. TCP-Planet then emulates Slow Start and Congestion Avoidance algorithms of the conventional TCP protocols by treating time intervals of size T as the RTT of the emulated connection. The objective of the Immediate Start phase is to probe the network in a fast and controlled manner so that the transmission rate can be increased quickly according to the feedback from the sink. The determination of the interval size T is presented in Section II-C. Here, we present the protocol operation in the Immediate Start phase in two parts, i.e., *Emulated Slow Start* and *Emulated Congestion Avoidance*.

1) Emulated Slow Start: As shown in Fig. 3, the first RTT is divided into time intervals of size T. This is done in order to have more fine-grained time unit than the extremely high RTT of the deep space link. The number of data packets

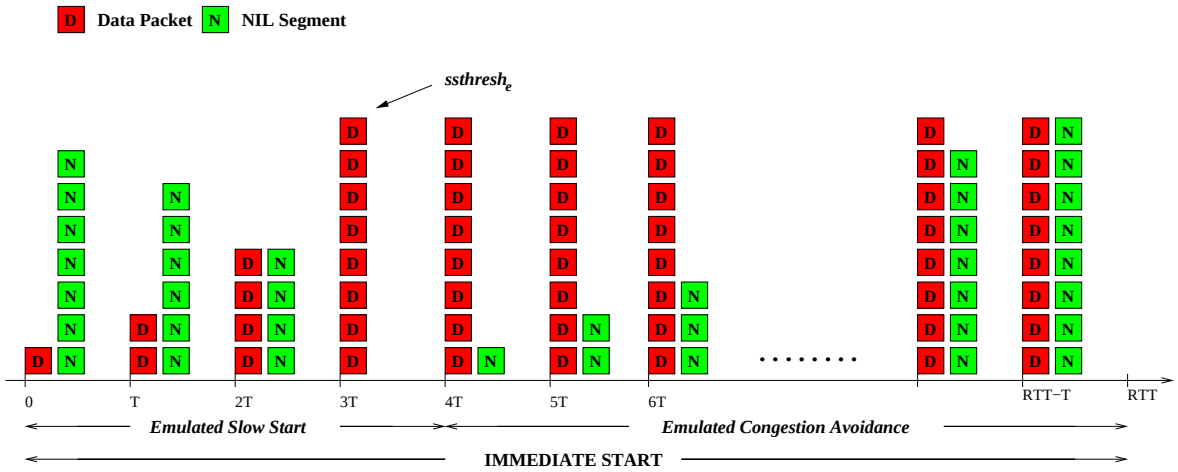

Fig. 3. An example illustration of time framing mechanism in the Immediate Start phase for $ssthresh_e = 8$.

transmitted in each interval, $cwnd$, is increased geometrically in each interval as in Slow Start algorithm of the classical TCP protocols. This increase continues until the emulated slow start threshold, $ssthresh_e$, is reached, i.e., $cwnd \leq ssthresh_e$.

In addition to data packets, *NIL segments* [3] are also sent in each time interval during the Immediate Start phase. The objective of NIL segment transmission is to probe actual link resource status in the beginning of the connection. NIL segments are chosen from the unacknowledged outstanding data packets to be used for packet loss recovery at the receiver. NIL segments are encapsulated by low priority IP packets. Note that the *type of service* (TOS) field of the IP packet header can be used for this purpose. Hence, assuming the relay satellites serving as routers along the Interplanetary Backbone link as shown in Fig. 1 have priority-queuing capability, low priority NIL segments are discarded first in case of congestion. Therefore NIL segment transmission does not affect the throughput of the actual data packet transmission. If there is no congestion, they are received and ACKed back by the sink, which reveals that there are still unutilized link resources along the path. NIL segments are created with `Generate_Nil_Segment()` algorithm shown in Fig. 4, where Q is the length of the queue of unacknowledged outstanding data packets and i is a counter. Further details of the NIL segment generation can be found in [3].

In the Emulated Slow Start phase, the number ($cwnd_N$) of NIL segments transmitted in each interval is determined such that the total number of packets transmitted does not exceed the emulated slow start threshold, i.e., $cwnd + cwnd_N \leq ssthresh_e$.

In Fig. 3, we assume the emulated slow start threshold is 8 packets, i.e., $ssthresh_e = 8$. Therefore, the number of data and NIL segments transmitted at each time interval during the Emulated Slow Start phase can be summarized as follows

- 1^{st} T $cwnd = 1$ and $cwnd_N = 7$.
- 2^{nd} T $cwnd = 2$ and $cwnd_N = 6$.
- 3^{rd} T $cwnd = 4$ and $cwnd_N = 4$.
- 4^{th} T $cwnd = 8$ and $cwnd_N = 0$.

In general, the number of data packets and NIL segments transmitted in the i^{th} time interval during the Emulated Slow Start is given by

$$cwnd = 2^{i-1} \quad (1)$$

$$cwnd_N = ssthresh_e - 2^{i-1} \quad (2)$$

Generate_NIL_Segment()
```
  Add unACKed packets into the queue;
  i = 0;
  if (NIL_Segment is needed)
    q = i mod Q;
    i = i + 1;
    NIL_Segment = q^th packet in queue;
  end;
  if (j^th packet in the queue is ACKed)
    Remove the packet from Q;
    if (j < i and i > 0)
      i = i − 1;
    end;
  end;
  if (New packet is added to Q)
    Add the packet to the tail;
    Q = Q + 1;
  end;
  Return NIL_Segment;
end;
```

Fig. 4. The NIL Segment Generating Algorithm

2) Emulated Congestion Avoidance: The Emulated Slow Start is left for Emulated Congestion Avoidance phase when $cwnd = ssthresh_e$. Since there is no feedback information about the link resources as yet, the TCP-Planet source does not increase the number $cwnd$ of data packets transmitted in each time interval. Therefore, $cwnd = ssthresh_e$ until the end of the emulated congestion avoidance phase. However, for further probing the link status, the source increases $cwnd_N$ additively by one NIL segment per T interval emulating congestion avoidance algorithm of the classical TCP protocols. As in Fig. 3, during this phase $cwnd$ is kept constant and $cwnd_N$ is increased by one segment in each T until the end of the Immediate Start, when $cwnd_N$ becomes equal to $cwnd$. The transmission of NIL segments is terminated at the end of the first RTT and TCP-Planet source goes to the *Follow-Up* state

of the Initial State algorithm as shown in Fig. 2.

B. The Follow-Up Phase

During this phase, the feedback for the packets transmitted in the Immediate Start phase are started to be received by the TCP-Planet source. In order to save scarce reverse channel resources of the bandwidth asymmetrical deep space link, the sink does not ACK back all the packets it receives . Several data packets are ACKed by a delayed SACK, whose details are explained in Section III-D. Since NIL segments are transmitted to probe the link status, each received NIL segment indicates the existence of unutilized link resources. Hence, the TCP-Planet sink counts the total number (N) of packets received in every T period and sends this information back to the source. This information is carried in NIL ACKs. TCP-Planet source transmits $ssthresh_e$ packets in $RTT \leq t \leq RTT + T$ until the first NIL ACK received at $t = RTT + T$. Then, TCP-Planet source adjusts its transmission rate (S) by using the information carried in NIL ACKs, i.e., $S = N/T$, until the end of $2 \cdot RTT$. Therefore, the transmission rate S at the end of the Initial State depends on the number of ACKs received in the last time interval of the Follow-Up phase.

Let N_{ACK} be the number of packets received by the TCP-Planet sink during $RTT - T \leq t \leq RTT$. Since the total number of packets sent in the last interval of the Immediate Start phase is $2 \cdot sstresh_e$, the data transmission rate S at the end of the Initial State is expressed by

$$S = \frac{\min\{N_{ACK}, 2 \cdot ssthresh_e\}}{T} \quad (3)$$

In addition to the data packets, TCP-Planet source also starts sending *NIX segments* during the Follow-Up phase. The NIX segments are much smaller than the data packets, i.e., 40 bytes. They are carried in both low and high priority IP packets. During the Follow-Up phase, low and high priority NIX segments are transmitted with the transmission rate S_{Nix} which is equal to the transmission rate of the data packets, i.e., $S_{Nix} = S$. The objective of the NIX segment transmission is to capture congestions and make decisions accordingly in the Steady State. The details of the congestion detection mechanism based on NIX segments are explained in Section III. The Follow-Up phase is over at $t = 2 \cdot RTT$ and the source leaves Initial State for the Steady State as shown in Fig. 2.

Consequently, TCP-Planet source can capture the link resources as soon as $t \geq RTT + T$ based on the number of ACKs it receives from the sink. It can achieve this without leading to congestion by controlling the number of probe packets injected into the network.

C. Determination of the Time Interval T

In the Initial State, the extremely high actual RTT is divided into time intervals of size T and the conventional TCP behavior is emulated to capture the available link resources in a controlled manner. The performance of the Initial State depends on the size of the T intervals. While the smaller T increases link utilization, it also increases overhead incurred by NIL segment transmission.

```
Initial_State()
  Send connection request CONN_REQUEST;
  Set T & ssthresh_e by (8) & (9)
  n = 1;
  cwnd = 1;
  While (t ≤ RTT)
  /* Immediate Start */
    While (cwnd ≤ ssthresh_e)
    /* Emulated Slow Start */
      If ((n − 1)T ≤ t ≤ nT)
        Send (cwnd) DATA pkts;
        Send (ssthresh_e − cwnd) NIL pkts;
      end;
      n = n + 1;
      cwnd = 2^(n−1);
    end;
    /* Emulated Congestion Avoidance*/
    cwnd = ssthresh_e;
    cwnd_N = 1;
    While ((n − 1)T ≤ t ≤ nT ≤ RTT)
      Send (cwnd) DATA pkts;
      Send (cwnd_N) NIL pkts;
      cwnd_N = cwnd_N + 1;
      n = n + 1;
    end;
  end;
  While (RTT ≤ t ≤ 2 · RTT)
  /* Follow-Up */
    If (NIL_ACK_RECEIVED)
      Set data rate S = N/T;
      Set NIX rate S_Nix = S;
    end;
  end;
  Steady_State();
end;
```

Fig. 5. The `Initial_State()` algorithm

The objective of the Initial State is to reach a certain data transmission rate, S, as soon as possible without leading to a congestion. During the Follow-Up phase, TCP-Planet source adjusts its transmission rate S according to the feedback received from the sink every T period via NIL ACKs. Since $ssthresh_e$ is the maximum number of packets transmitted in one interval during the Emulated Slow Start phase, the transmission rate reached in the Follow-Up phase is dependent on $ssthresh_e$. Here, we assume that TCP-Planet source has a given target minimum transmission rate requirement, B. This requirement can be mostly due to two main reasons:

- **Application**: The minimum transmission rate is required in order to meet specific application needs such as transmission of scientific multimedia data which requires 100% reliability and a minimum bound on the transmission rate.
- **Latency**: The maximum allowed latency can be advertised by the specific space mission requirements as the maximum duration for the transmission of certain amount of data. This determines the minimum target transmission rate requirement for a given connection.

Consequently, in order to achieve the target transmission rate of B packets/s at $t = RTT + T$, the corresponding $sshtresh_e$ is calculated as

$$ssthresh_e = B \cdot T \quad (4)$$

The Emulated Slow Start phase of the Immediate Start terminates when the number of data packets transmitted in one time interval reaches the emulated slow start threshold, i.e., $cwnd = ssthresh_e$. Therefore, from (2) it follows that the duration of the Emulated Slow Start phase (T_{ess}) is given by

$$T_{ess} = (\log_2 ssthresh_e + 1) \cdot T \quad (5)$$

During the Emulated Congestion Avoidance phase, the number of data packets per interval, $cwnd$, is kept constant. Moreover, the number of NIL segments is increased by one per interval until the end of the Immediate Start, i.e., $t = RTT$. In order to probe the network resources in the range of $[B, 2B]$, the source increases the number of NIL segments transmitted in each interval until it is equal to the number of data packets, i.e., $cwnd_N = ssthresh_e$. The Immediate Start is over once $cwnd_N$ reaches $ssthresh_e$ at $t = RTT$. Therefore, the duration of the Emulated Congestion Avoidance phase (T_{eca}) can be expressed by

$$T_{eca} = ssthresh_e \cdot T \quad (6)$$

Since the total duration of the Immediate Start is $T_{ess}+T_{eca} = RTT$, from (5) and (6) it follows that

$$(\log_2 ssthresh_e + 1 + ssthresh_e) \cdot T = RTT \quad (7)$$

As $(\log_2 ssthresh_e + 1)$ is negligible compared to $ssthresh_e$ itself, from (4) and (7) the size of the time interval T can be calculated by

$$T = \sqrt{\frac{RTT}{B}}, \quad (8)$$

where B is the target transmission rate in packets/s. At the beginning of a connection, RTT is also not known to the TCP-Planet source. Therefore, TCP-Planet source also caches the RTT of the past connections and uses that value in order to calculate the time interval size T by (8) during the Initial State.

Consequently, the emulated slow start threshold can be expressed by using (4) and (8) as follows

$$ssthresh_e = \sqrt{RTT \cdot B} \quad (9)$$

Thus, at the beginning of the connection, T and $ssthresh_e$ are set by (8) and (9). The first RTT period is then divided into time intervals of size T. The Emulated Slow Start and Emulated Congestion Avoidance phases are performed in the first RTT in order to perform resource probing. In the second RTT, the transmission rate is increased by sending a new data packet for each received ACK until $t = 2 \cdot RTT$. By this way, TCP-Planet source can increase its transmission rate very quickly and efficiently utilize the resources during the Initial State without leading to any congestion.

D. The Connection Establishment

The conventional TCP protocols perform three-way handshake for connection establishment. The existing protocols send connection request segment at the beginning of the connection and do not transmit any data segments until the connection ACK is received from the receiver. This approach results in waste of huge bandwidth for at least a duration of one RTT in deep space links. In order to avoid this inefficiency, TCP-Planet source does not wait for the ACK for the connection request and starts data transmission in the Initial State as if the session request is granted. If the request is rejected, then the connection is terminated after one RTT.

As a result, TCP-Planet achieves better utilization of link resources in the early phases of the connection by the Initial State algorithm. The overall Initial State algorithm of the TCP-Planet is summarized in Fig. 5.

III. TCP-PLANET: STEADY STATE

TCP-Planet source leaves Initial State for *Steady State* at $t = 2 \cdot RTT$ and remains in the Steady State until the connection is terminated. During the Steady State operation, TCP-Planet source can be in one of the four states, i.e., *Increase Rate*, *Decrease Rate*, *Hold Rate* and *Blackout* as shown in Fig. 2. In the beginning of the Steady State, the source goes to Hold Rate state, where no transmission rate change is performed. During Steady State operation, TCP-Planet deploys a new congestion control scheme. Hence the transitions between these states in the Steady State is decided based on this congestion control scheme. Therefore, the data transmission rate S can be *increased*, *decreased* or *hold* according to the current state.

A. Congestion Control

TCP-Planet deploys a new congestion control scheme in order to address the challenges due to high link error rates in Interplanetary Backbone Network. The objective of this method is to decouple network congestions and packet losses due to errors.

During the Steady State, TCP-Planet source transmits low and high priority *NIX segments* continuously for congestion detection purposes. NIX segments are carried by IP packets, which are marked as *high* and *low* priority using TOS field in the IP packet header. NIX segments differ from NIL segments used in Initial State in terms of their size and functionality. Unlike NIL, NIX segments are much smaller compared to data segments, i.e., 40 bytes. They do not carry any information and thus, cannot be used for error recovery purposes.

Low and high priority NIX segments are transmitted simultaneously with the same rate (S_{Nix}) equal to the data transmission rate (S), i.e., $S_{Nix} = S$. The objective of this is to obtain congestion decision support via comparison between the reception statistics of both low and high priority NIX segments. Since low and high priority NIX segments are equal in size and are transmitted with the same rate S_{Nix}, they experience the same packet loss rate due to space link errors. However, assuming the relay satellites serving as routers along the Interplanetary Backbone link as shown in Fig. 1 have priority-queuing capability, low priority NIX segments are discarded first in case of a congestion. The only reason for low and high priority NIX segments to have different packet loss rates is the additional loss experienced by low priority segments due to congestion. This reasoning constitutes the basis for our congestion detection method via NIX segment transmission.

TCP-Planet sink counts the number of received low (N_{Low}) and high (N_{High}) priority NIX segments in a sliding time window of T_w. The received NIX segments are not ACKed back to the sender. Note that this also avoids an overhead in the reverse channel, which can also become a bottleneck due to bandwidth asymmetry in deep space links. Only the reception statistics within a time window of T_w are returned to the sender every τ period. This information is carried by *NIX ACKs*. TCP-Planet source receives NIX ACKs carrying (N_{Low}, N_{High}) at the end of each measurement period τ.

Let Φ be the ratio of the number of received low and high priority NIX segments, i.e.,

$$\Phi = \frac{N_{Low}}{N_{High}} \tag{10}$$

Assuming that the low and high priority NIX packets experience same packet loss rate due to the link errors within a measurement period τ, since the only reason for $N_{Low} < N_{High}$ is the congestion along the path, TCP-Planet source infers that a congestion exists if $\Phi < 1$.

The entire congestion control of the TCP-Planet source is determined according to the value of Φ. The transitions between *Increase*, *Decrease* and *Hold* Rate states in Fig. 2 are performed based on Φ. In order to avoid unnecessary state transitions, decision is made via comparison of Φ with preset rate decrease, ϕ_d, and the rate increase thresholds, ϕ_i, as shown in Fig. 2. These thresholds are protocol parameters, whose selection is an implementation issue. The values of ϕ_d and ϕ_i, which yield highest protocol performance, are determined during simulation experiments and are given in Section IV.

The summary of the congestion control mechanism is given as follows:

1) $\mathbf{\Phi < \phi_d}$: In this case, TCP-Planet source infers that congestion is experienced along the path. Thus, the source goes to the *Decrease Rate* state where the transmission rate S is decreased multiplicatively, i.e., $S = S \cdot \zeta$.
2) $\mathbf{\phi_d \leq \Phi \leq \phi_i}$: In this case, the data transmission rate S is kept unchanged until next feedback is received from the sink.
3) $\mathbf{\Phi > \phi_i}$: TCP-Planet source infers that no congestion is experienced. Consequently, it increases the data transmission rate additively, i.e., $S = S + \delta$.

The additive-increase (δ) and multiplicative-decrease (ζ) parameters are used to perform AIMD rate control every τ period. The selection of these AIMD parameters are explained in Section III-B. The steps of the Steady State algorithm of TCP-Planet protocol is summarized in Fig. 6.

```
Steady_State()
  Set ξ;
  Set α by (12);
  Set ζ & δ by (13) & (14);
  Send DATA pkts with rate S;
  Send Low pri. NIX pkts with rate S;
  Send High pri. NIX pkts with rate S;
  If (NIX_ACK_RECEIVED)
  Congestion Decision:
    If (Φ < φ_d)
    /* Decrease Rate */
      S = S · ζ;
    else if (φ_d ≤ Φ ≤ φ_i)
    /* Hold Rate */
      S = S;
    else if (Φ > φ_i)
    /* Increase Rate */
      S = S + δ;
    end;
  end;
  If (ZERO_NIX_ACK_RECEIVED)
    /* After Blackout State */
       Goto Hold State;
  end;
  If (NO_ACK_in_T_w)
    /* Blackout State */
    Send Low pri. NIX pkts with rate S;
    Send High pri. NIX pkts with rate S;
    Retransmit TIMEOUT pkts;
    If (NIX_ACK_RECEIVED)
      If (ZERO_NIX_ACK)
        /* L ≥ 2x */
        Goto Hold State;
      else
        /* L < 2x */
        Goto Congestion Decision;
      end;
    If (DATA_ACK_RECEIVED)
      /* L < 2x */
      Goto Congestion Decision;
    end;
  end;
end;
```

Fig. 6. The `Steady_State()` algorithm.

B. The New Rate-Based AIMD Scheme

As mentioned before, current TCP protocols achieve very poor performance on the links with extremely high propagation delay mostly due to their window-based operation [1]. The throughput of the window-based TCP protocols and rate-based schemes are inversely proportional to the RTT [22] and the square-root of RTT (see the Appendix), respectively. Thus, the rate-based congestion control schemes are more robust to excessive propagation delays than the window-based mechanisms. Hence, in order to address the adverse effects of extremely high propagation delay on the throughput performance, TCP-Planet deploys rate-based additive-increase multiplicative-decrease (AIMD) congestion control. The steady state throughput of the rate-based AIMD scheme is derived in the Appendix and expressed by

$$T = \frac{\alpha}{4 \cdot (1-\xi)} \left[1 + \xi + \sqrt{(3-\xi)^2 + \frac{8 \cdot (1-\xi^2)}{\alpha \cdot RTT \cdot p}} \right] \tag{11}$$

where α and ξ are the additive-increase and the multiplicative-decrease factors, respectively and p is the packet loss probability. It is observed from (11) that the throughput of the rate-based AIMD scheme depends on the values of α and ξ. Therefore, TCP-Planet adapts its AIMD parameters to the link conditions, i.e., RTT and packet loss rate, such that their adverse effect on the performance are compensated and a given target throughput is achieved.

The additive-increase parameter (α) of the rate-based AIMD scheme to achieve a target throughput of B packets/s can be obtained from (11) as follows

$$\alpha = \frac{(1+\xi)}{2}\left(B + \frac{1}{RTT \cdot p}\right)\left[\sqrt{1 + \frac{8B^2(1-\xi)}{\left(B+\frac{1}{RTT\cdot p}\right)^2(1+\xi)^2}} - 1\right] \quad (12)$$

where ξ is the multiplicative-decrease factor and p is the packet loss probability, respectively. Here, the target throughput B can be defined as the average data rate required to transmit certain amount of information within the certain delay bound as in Section II-C.

These AIMD parameters, α and ξ, in (12) are the rate change parameters to be used to control the rate with period RTT. However, TCP-Planet source performs rate control with the feedback received from the sink every τ period, where $\tau \ll RTT$. Thus, the rate control parameters α and ξ in (12) cannot be directly used for TCP-Planet. These parameters, instead, represent the upper bound for the rate change of the TCP-Planet source within one RTT period. Hence, the AIMD parameters to be used by the source with period τ can be derived from α and ξ.

Let τ be the NIX segment reception statistics feedback period. Thus, TCP-Planet source performs at most RTT/τ number of rate changes within one RTT period.

Let δ and ζ be additive-increase and multiplicative-decrease parameters to be used by TCP-Planet for rate control with period τ. Then ζ can be calculated as

$$\zeta = \xi^{(\tau/RTT)} \quad (13)$$

By the same reasoning, the additive increase factor to be used by TCP-Planet source can also be calculated as

$$\delta = \frac{\alpha \cdot \tau}{RTT} \quad (14)$$

TCP-Planet source uses AIMD scheme during the Steady State operation as shown in Fig. 2. At the end of the Initial State, i.e., $t = 2 \cdot RTT$, it calculates the packet loss rate p by using the number of transmitted and ACKed data packets and NIL segments. TCP-Planet source then uses (12), (13) and (14) to calculate its AIMD parameters, i.e., ζ, δ. Consequently, according to the result of the congestion decision mechanism presented in Section III-A, the transmission rate S is multiplicatively decreased or additively increased with ζ and δ, respectively.

C. The Blackout State Behavior

Link outages due to loss of line-of-sight by orbital obscurations lead to burst packet losses and decrease in the throughput. In order to provide reliable transport, SACK options [20] are adopted by TCP-Planet to address burst losses. Due to possible inadequacy of the number of SACK blocks in the SACK option field for very long blackout durations, timeout mechanism is also included in TCP-Planet. In order to reduce the throughput loss due to blackouts, *Blackout State* is developed and incorporated into the Steady State.

TCP-Planet source receives data ACKs for reliability control purposes and NIX ACKs for NIX segment reception statistics. If the source does not receive any type of ACK for a certain period of time T_w, it infers this condition as blackout and goes to the *Blackout State* as shown in Fig. 2. The objective of the Blackout State procedure is to reduce the throughput degradation due to the blackout situation.

During blackout, TCP-Planet source keeps sending low and high priority NIX packets without changing its transmission rate. The same blackout event is also detected by the TCP-Planet sink if no packet is received within T_w period. Although the sink does not receive any low and high priority NIX packets during the blackout, it keeps sending NIX ACKs with (N_{Low}, N_{High}) as (0,0). These ACK packets are called *Zero NIX ACKs*. The objective of Zero NIX ACKs is to help TCP-Planet source to capture accurate information regarding the blackout situation and act accordingly.

Since RTT is very high, the effect of blackout on the performance changes with its relative location of blackout occurrence with respect to the sink. Let $t = t_0$ be the time when blackout occurs and L is the duration of the blackout. Assume that the blackout occurs at a position x seconds away from the TCP-Planet sink. For $rtt = RTT/2$, there are two distinct cases according to the duration of the blackout and its relative distance to the TCP-Planet sink in time:

1) $L < 2x$: After $rtt - x$ from t_0, i.e., at $t_1 = t_0 + rtt - x$, TCP-Planet source detects the period without ACKs. If the duration of the period with no ACK takes more than T_w, then the source moves to Blackout State at $t = t_1$, as in Fig. 7.

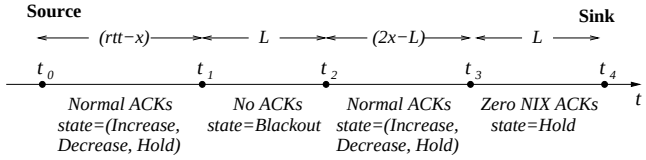

Fig. 7. Blackout condition observed from TCP-Planet source for $L < 2x$.

In this case, TCP-Planet source does not send any new data packets but sends low and high priority NIX segments without changing their transmission rate. It also does not invoke congestion control mechanism. It starts retransmission of the packets whose retransmission timer is expired. At $t_2 = t_1 + L$, TCP-Planet source receives normal ACKs for duration of $2x - L$. Therefore, TCP-Planet source infers that blackout is over and goes to any of the Increase, Decrease and Hold states according to the information it receives in the first NIX ACK as shown in Fig. 2. At $t_3 = t_2 + 2x - L$, the source starts to receive Zero NIX ACKs for duration of L. These Zero NIX ACKs, are in fact, transmitted by the receiver when it detects the same blackout condition. Therefore, TCP-Planet does not go to Blackout State instead it goes to Hold State, where it keeps sending new data packets with the same transmission rate again as shown in Fig. 2. Consequently, TCP-Planet reduces the effect of blackout on the performance by not wasting the link resources for a duration of L.

2) $L \geq 2x$: In this case, TCP-Planet source detects no ACK period and goes to Blackout State at $t_1 = t_0 + rtt - x$

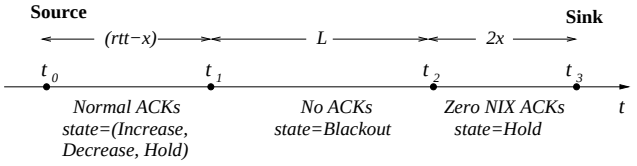

Fig. 8. Blackout condition observed from TCP-Planet source for $L \geq 2x$.

for the blackout which occurred at $t = t_0$. It again does not change its transmission rate and does not send any new data packet. At $t_2 = t_1 + L$, the source starts to receive zero NIX ACKs for duration $2x$ as shown in Fig. 8. This reveals that the blackout is over for TCP-Planet source and then the source leaves Blackout State for Hold State as shown in Fig. 2. It stays in Hold State and keeps sending data packets with the same transmission rate before the blackout was detected. At $t_3 = t_2 + 2x$, Zero NIX ACKs period is over and according to the information received in the first NIX ACK the source performs state transition. Hence, the duration of $2x$ is efficiently utilized by the help of Zero NIX ACK mechanism.

Consequently, the Blackout State reduces the throughput degradation due to blackout conditions and improves the link utilization for duration of L or $2x$ in the cases $L < 2x$ and $L \geq 2x$, respectively.

D. The Delayed SACK

TCP-Planet uses selective acknowledgement (SACK) [20] options for assurance of reliable data segment transmission. TCP-Planet sink continuously sends SACK back to the source for each data packet it receives. Given that the data packets are 1KB and SACKs are 40B, then the ratio of the traffic in the forward and reverse channels is 25:1, i.e., 1KB/40B. Thus, the bandwidth asymmetry up to 25:1 cause no congestion in the reverse link. However, the bandwidth asymmetry in the space links is usually on the order of 1000:1 [12]. Thus, sending one SACK for each data packet can cause reverse channel to be congested resulting in packet losses in the reverse link.

In order to avoid this problem, TCP-Planet deploys SACK congestion control by delaying the SACKs. TCP-Planet sink maintains delayed-SACK factor, d, and sends one SACK for every d data packets received. If there is no packet loss and hence no change in the SACK blocks, then TCP-Planet sink keeps delaying SACKs with a delayed-SACK factor of d. Otherwise, it sends a new SACK with an updated block immediately. Therefore, the amount of traffic on the reverse channel is controlled by adjusting the delayed-SACK factor d. Effects of the blackout on throughput and the improvement achieved by delayed-SACK is evaluated in Section IV-D.

IV. Performance Evaluation

In order to investigate the performance of the TCP-Planet, we conducted extensive simulation experiments. The improvement in the initial phase of the connection achieved by the Initial State algorithm is evaluated in Section IV-A. Throughput performance of TCP-Planet is analyzed in Section IV-B along with the overhead introduced. Effects of blackout conditions on the performance and the performance of TCP-Planet with the improvement by Blackout State are investigated in IV-D. TCP-Planet performance on deep space links with asymmetrical bandwidth and the improvement with delayed SACK are explored in Section IV-E.

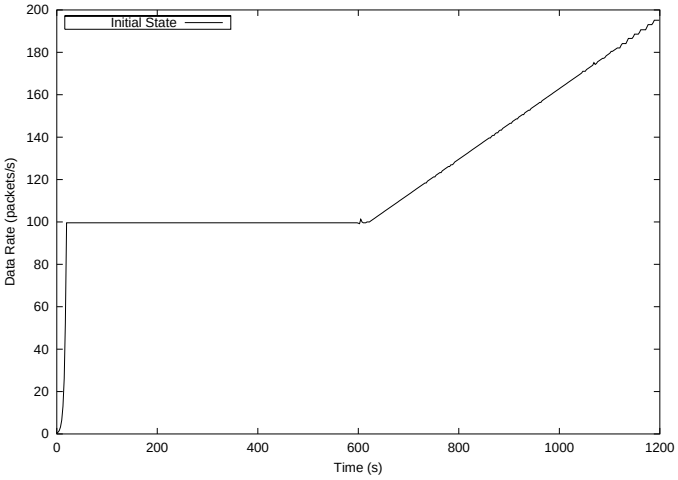

Fig. 9. Transmission rate change in the Initial State of TCP-Planet source for $RTT = 600$ seconds.

A. Initial State Performance

In order to avoid performance degradation due to inefficient connection starting behavior, TCP-Planet deploys `Initial_State()` algorithm in Fig. 2 and Fig. 5. Here, we simulate a topology where the source and sink are connected through a deep space link with $RTT = 600$ seconds and $p = 10^{-5}$. We perform same experiment with TCP-Planet, TCP-Peach+ [3] and TCP-NewReno. The reason is that most of the current TCP protocols deploy initial phase behavior based on the Slow Start algorithm, which is also used by TCP-NewReno.

In Fig. 9, the transmission rate change in the Initial State of TCP-Planet source is plotted, where the target transmission rate of TCP-Planet source is assumed to be 100 packets/s. Because of the very high propagation delay and 100% reliability requirement, the amount of buffer required for retransmission mechanism is proportional to the data transmission rate. Therefore, for very high data rates, this brings considerable amount of memory requirement. For example, the source needs to maintain 0.6 GB of buffer for $RTT = 10$ minutes and the average data transmission rate of 1MB/s. Thus, we set target data rate to 100 packets/s in order to maintain practicality in terms of memory requirements.

As explained before, TCP-Planet source performs Emulated Slow Start phase, which ends once the number of data packets transmitted in one time interval is equal to $ssthresh_e$. As seen in Fig. 9, the Emulated Slow Start phase lasts for approximately $T_{ess} = 19.58$ seconds and the TCP-Planet source reaches 100 packets/s data rate by then. Emulated Congestion Avoidance phase then starts and lasts until the end of the first RTT of the connection period. During this

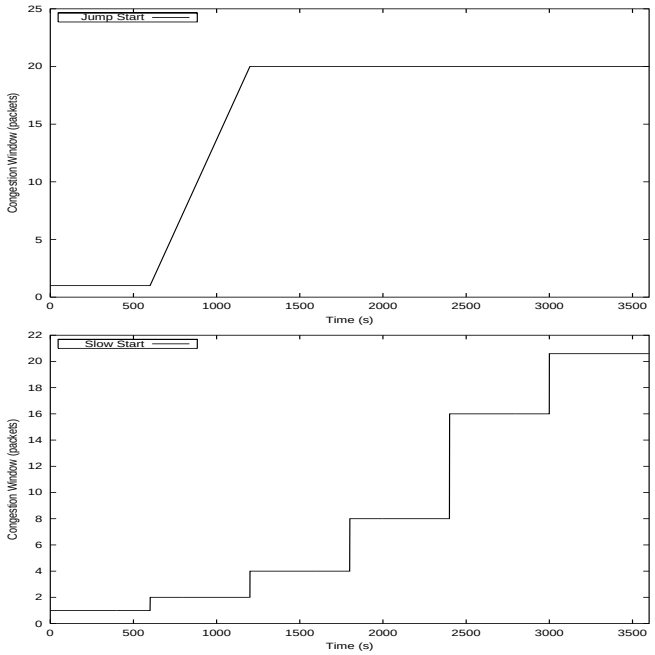

Fig. 10. Congestion window size change during Jump Start algorithm of TCP-Peach+and Slow Start algorithm of TCP-NewReno and for $RTT = 600$ seconds.

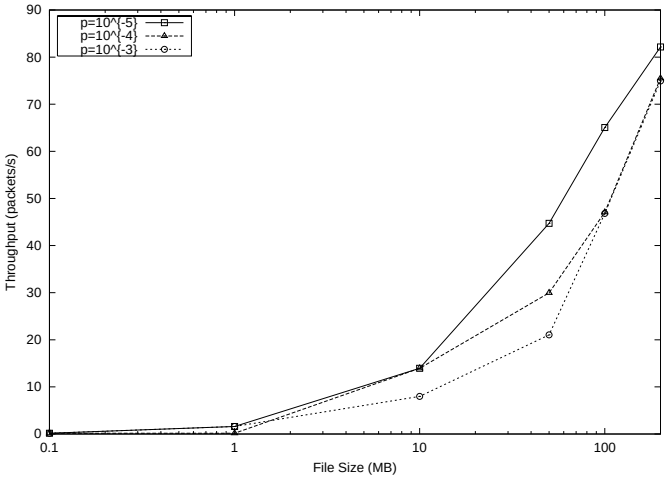

Fig. 11. Throughput for changing p and file size with $RTT = 600$ seconds.

phase, the data transmission rate is not increased instead NIL segment transmission rate is increased linearly to further probe the link resources. At $t \approx 600$, the source goes to Follow-Up period and increases its transmission rate based on the feedback received from the sink every T period. At $t = 1200$ seconds, the transmission rate reaches 200 packets/s.

TCP-Peach+ [3] deploys *Jump Start* algorithm in order to capture link resources very quickly. In Jump Start, the sender sets the congestion window, $cwnd$, to 1. After sending the first data segment, it transmits *NIL segments* every $RTT/rwnd$, where $rwnd$ is the receiver window size. As a result, after one round trip time, the congestion window size increases very quickly as the ACKs for NIL segments arrive at the sender. The performance of Jump Start is shown in Fig. 10. The congestion window of TCP-Peach+ reaches 20 packets

TABLE I

PROTOCOL PARAMETERS USED IN EXPERIMENTS

Parameter	Value	Definition
ϕ_i	0.8	Rate increase threshold
ϕ_d	0.2	Rate decrease threshold
τ	5	NIX ACK period in *seconds*
T_w	20	Sliding window size in *seconds*
ξ	0.5	Rate decrease parameter for RTT
d	5	Delayed-SACK factor in *number of packets*

in at most $2 \cdot RTT$ duration. It is, however, still very low compared to the performance of Initial State of TCP-Planet, which reaches 200 packets/s data rate at the end of $2 \cdot RTT$.

The slow start performance is, however, not even close to what Initial State of TCP-Planet achieves in deep space links of the IPN. In Fig. 10, the congestion window evolution dependent on time is shown during the slow start. The slow start period lasts for approximately $6 \cdot RTT$ for threshold window size of 20 packets. Therefore, the link is not efficiently utilized for 60 minutes due to unsuitability of the slow start algorithm to extremely high propagation delays in deep space links.

B. Throughput Performance

In order to show the throughput performance of TCP-Planet in deep space links, we perform several experiments by varying packet loss probability p and the size of the data to be transmitted. We assume 1Mb/s as the capacity of the link and $RTT = 600$ seconds. The target transmission rate B is set to be 100 packets/s, i.e., 100 KB/s for data packets of size 1KB. The protocol parameters given in Table I are used during simulation experiments unless otherwise stated. The investigation of the effects of these parameters on the protocol performance are left for future study.

In [1], existing TCP protocols including TCP-Vegas, which is adopted by SCPS-TP protocol as a congestion control scheme for space-based communications [10], have been shown to achieve very poor performance in deep space links. For $RTT = 600$ seconds and $p = 10^{-3}$, the throughput achieved by TCP protocols is approximately around 30 B/s and hence almost the entire link remains almost unutilized [1]. Although TCP-Peach+ has significantly outperformed other TCP schemes for the same link, the performance degradation was yet too serious that it could achieve throughput around 93 B/s. Thus, the same experiments are not repeated here and the details can be found in [1].

TCP-Planet simulation experiments are performed for transmission of varying file sizes between 100KB and 200MB and for varying packet loss probability p of 10^{-5}, 10^{-4}, 10^{-3}. As shown in Fig. 11, the throughput increases with increasing file size. This is because the larger the file size is the longer TCP-Planet stays in the Steady State. As a result, the link utilization is increased. Although throughput decreases for increasing packet loss probability, this degradation also decreases for increasing file size. This is mostly because throughput improvement by new congestion control scheme is higher when the protocol stays in Steady State for longer.

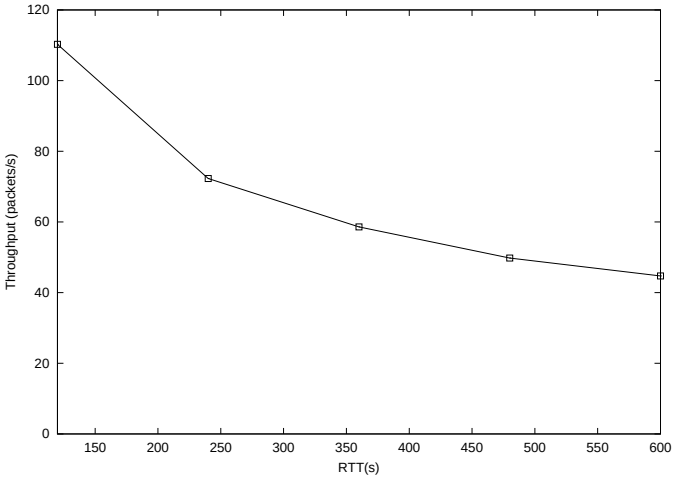

Fig. 12. Throughput for changing RTT for file size of 50MB and $p = 10^{-5}$.

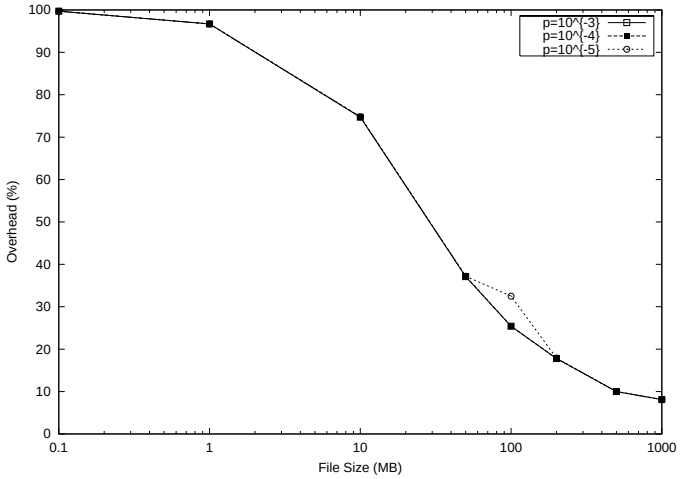

Fig. 13. Overhead due to transmission of NIL and NIX segments for changing file size and p with $RTT = 600$ seconds.

For transmission of 200MB and $p = 10^{-5}$, TCP-Planet throughput increases up to 82.2 packets/s approaching its target throughput value, i.e., $B = 100$ packets/s. Hence, TCP-Planet outperforms existing TCP protocols by approximately 10^3 times in terms of throughput.

In Fig. 12, the effect of RTT on the throughput performance is shown. The experiments are performed for the transmission of a 50MB file. The increase in RTT leads to throughput degradation as shown in Fig. 12. However, the throughput degradation due to high propagation delay is not as severe as it is for conventional TCP protocols [1]. This is because of the new rate-based congestion control scheme used in Steady State of the TCP-Planet. At $RTT = 600$ seconds, TCP-Planet achieves 44.72 KB/s throughput. Note that this value can further increase with increasing file size as in Fig. 11.

C. Overhead

During the connection period, TCP-Planet source brings overhead to the deep space link. This overhead is due to NIL segment transmission to probe the link resource in the Initial State and low and high priority NIX packets transmission for congestion decision support in the Steady State. In this subsection, we investigate the overhead caused by the injection of NIL and NIX segments into the network.

In Fig. 13, the overhead incurred by TCP-Planet is shown for transmission of files with varying sizes and for different packet loss rates, i.e., $p = 10^{-5}, 10^{-4}, 10^{-3}$. For very small files, i.e., $< 10MB$ the overhead is relatively high. Because, in this case, significant portion of the connection period is spent in the Initial State, where the number of NIL segments transmitted is high compared to that of data packets. As the file size increases, the overhead decreases as in Fig. 13. This is because the overhead in the Steady State is due to the transmission of small sized (40 bytes) NIX segments and hence is much lower compared to Initial State. Therefore, as the file size increases, the time spent in Steady State also increases, which in turn decreases the overall overhead in the connection period. As a matter of fact, the scientific data delivered during space exploration missions are significantly high [6], i.e., in the order of gigabytes, which leads to very low overhead.

On the other hand, the overhead does not significantly vary for different p. As a matter of fact, the amount of NIL segments transmitted in the Initial State depends only on the target throughput and the RTT and thus it is not dependent on p. This overhead exists only in the beginning of the connection. Although the NIL segment transmission during Initial State brings overhead, it also leads to significant performance improvement as observed in Section IV-A. As the NIX transmission rate is equal to data rate and the congestion control is robust to link errors, the overhead due to NIX transmission is also independent of p. The overhead due to NIX transmission exists after the Initial State until the end of the connection. However, the congestion control decision support provided by NIX segments lead to significant throughput performance improvement. For 1GB file size and $p = 10^{-3}$, the overhead becomes as low as 8.1 %, which is quite low compared to throughput improvement with a factor more than 10^3.

D. Blackout Conditions

When a blackout is detected, TCP-Planet moves to Blackout State as shown in Fig. 2 in order to reduce its effect on the throughput performance as explained in Section III-C. Throughput achieved by TCP-Planet for different blackout durations is given in Fig. 14. Here, $RTT = 120$ seconds, $p = 10^{-5}$ and the target data rate is assumed to be 50 KB/s. The simulations are performed for a duration of 600 seconds, where the blackout occurs at $t = 250$ seconds.

As in Fig. 14, throughput decreases with increasing blackout duration as expected. This decrease is observed in both of the curves representing the TCP-Planet operation with and without Blackout State procedure. However, Blackout State procedure improves the performance for long blackout durations. For even a blackout of 150 seconds, which is $1/4$ of the entire simulation time, TCP-Planet achieves 36.41 packets/s throughput with the help of Blackout State behavior. This corresponds to

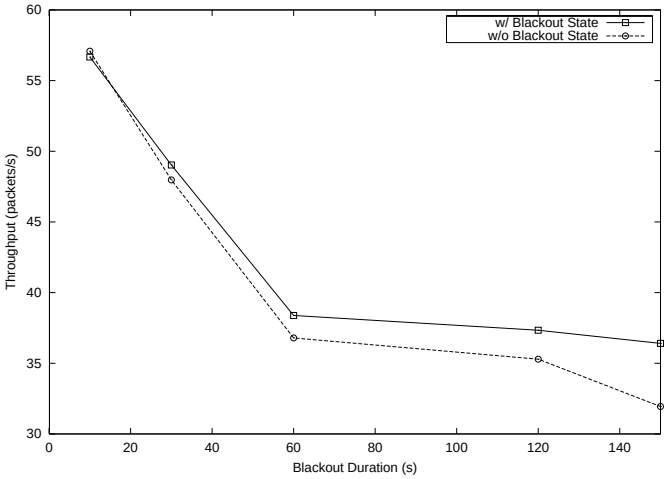

Fig. 14. Throughput for changing blackout duration with $RTT = 120$ seconds and $p = 10^{-5}$.

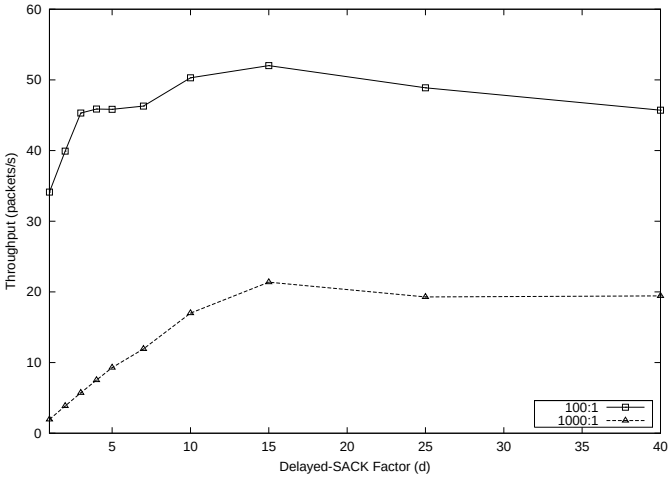

Fig. 15. Throughput for changing delayed SACK factor d with $RTT = 120$ seconds and $p = 10^{-4}$.

approximately 14% performance improvement over the case without Blackout procedure.

E. Bandwidth Asymmetry

In order to address bandwidth asymmetry problems, TCP-Planet delays SACKs with a certain delayed SACK factor. Simulation experiments are performed to show the effect of bandwidth asymmetry on the performance and the improvement achieved by delayed SACK. Here, $RTT = 120$ seconds $p = 10^{-4}$ and the simulation time is assumed to be 1200 seconds. The target rate is set to 50 KB/s. In Fig. 15, two cases with different bandwidth asymmetry ratio are investigated:

1) **100:1.** In this case, forward and reverse link capacities are 100 KB/s and 1 KB/s, respectively. Since the ratio of the size of data packets and SACK packets is 25:1, the actual asymmetry ratio is 4:1 in this case. Therefore, the throughput is not significantly degraded by asymmetrical channel capacity of the deep space link. As shown in Fig. 15, an increase in the delayed SACK factor also leads to an increase in the throughput. However, throughput decreases for both cases with either too high or too small delayed SACK factor of d. This is because if it is too small, reverse channel is congested. On the other hand, if d is too large, the source receives less number of SACKs than it expects which leads to performance degradation. Thus, at $d = 15$, throughput achieved increases up to 52.02 packets/s.
2) **1000:1.** In this experiment, bandwidth asymmetry on the order of 1000:1 is simulated by setting forward and reverse link capacities to 100 KB/s and 0.1 KB/s, respectively. Thus, the throughput is dramatically affected by the bandwidth asymmetry in this case, where the actual bandwidth asymmetry is 40:1. Therefore, throughput improvement with delayed SACK method is more significant for higher bandwidth asymmetry. For the case without delayed SACK, throughput decreases to up to 1.99 KB/s. As in Fig. 15, throughput increases with increasing number of delayed SACKs. However, this pattern does not last long since very high delayed SACK factors cannot be reached. This is because a data packet loss yields new SACK blocks, which requires to be transmitted back to the source immediately. For delayed SACK factor of 15, throughput increases up to 21.37 KB/s which corresponds to 10 times increase in the throughput performance compared to the case without delayed SACK, i.e., $d = 1$ in Fig. 15.

V. CONCLUSIONS

The inadequacy of the current TCP protocols in Interplanetary Backbone Network has already been known and the need for new transport protocol has been pointed out in [1]. In order to address this need, a new reliable transport protocol, TCP-Planet, is developed in this paper. The objective of TCP-Planet is to address the challenges posed by the InterPlaNetary Internet for reliable data delivery and achieve high throughput performance. TCP-Planet deploys a rate-based additive-increase multiplicative-decrease (AIMD) congestion control scheme. It runs on top of Internet Protocol (IP) layer and does not require any specific modification to the lower layers in the current TCP/IP suite. Performance evaluation via simulation experiments revealed that TCP-Planet significantly improves the throughput performance and addresses the challenges posed by deep space communication networks.

TCP-Planet replaces the inefficient slow start algorithms with a novel Initial State algorithm, which captures link resources in a very fast and controlled manner. Simulation experiments showed that TCP-Planet can reach high initial data rates very quickly. In order to address the challenges due to extremely high propagation delay, TCP-Planet deploys an end-to-end rate-based additive-increase multiplicative-decrease (AIMD) congestion control, whose AIMD parameters are tuned to help avoid throughput degradation. A new congestion control mechanism, which decouples congestion decision from single packet loss events, is developed to minimize the erroneous congestion decisions due to high link errors. Consequently, TCP-Planet improves throughput with a factor more than 10^3 compared to the current TCP protocols. In order

to reduce the effects of blackout conditions on the throughput performance, TCP-Planet incorporates Blackout State behavior into the protocol operation. By this way, it achieves up to 14% performance improvement in blackout conditions. The bandwidth asymmetry problem is addressed by the adoption of delayed SACK options. As a result, TCP-Planet is a reliable transport protocol equipped with diverse set of algorithms and functionalities, which can address the requirements of the InterPlaNetary Internet.

APPENDIX

The time-dependency of the transmission rate, $R(t)$, is shown in Fig. 16. The rate-based generic AIMD scheme is assumed to increase the rate, R, additively with α at each RTT, i.e., $R = R + \alpha$. It throttles the transmission rate, R, multiplicatively by ξ if the congestion is detected, i.e., $R = \xi \cdot R$.

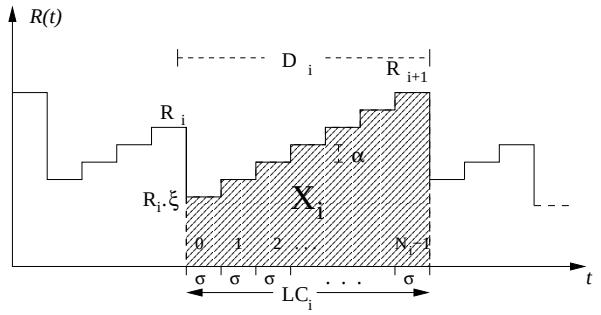

Fig. 16. Data rate change of rate-base generic AIMD congestion control.

Let $X(t)$ be the total number of packets transmitted in $[0, t)$, which can be calculated by

$$X(t) = \int_0^t R(\tau)\mathrm{d}\tau \tag{15}$$

For $T(t) = X(t)/t$ being the throughput achieved in $[0, t)$, the steady-state throughput achieved by the rate-based generic AIMD scheme is given by

$$T = \lim_{t\to\infty} \frac{1}{t} \int_0^t R(\tau)\mathrm{d}\tau \tag{16}$$

Let X_i be the number of packets transmitted in the i^{th} loss cycle, LC_i, which starts and ends with rate halving due to congestion decision. If the duration of LC_i is D_i, then the throughput achieved in LC_i is given by $T_i = X_i/D_i$. Assuming the evolution of the transmission rate R_i to be Markov Regenerative Process with rewards X_i [19], then the steady-state throughput can be calculated by

$$T = E[X]/E[D]\,, \tag{17}$$

where $E[X]$ and $E[D]$ are the means of X_i and D_i, respectively. The transmission rate change during LC_i, i.e., the value of the transmission rate at k^{th} RTT of the LC_i, is given by

$$R_{i,k} = R_i \cdot \xi + k \cdot \alpha \tag{18}$$

Given that N_i is the total number of RTTs in LC_i, i.e., the rate is throttled in $(N_i - 1)^{th}$ RTT, then the transmission rate at the end of the LC_i is expressed by $R_{i+1} = R_i \cdot \xi + N_i \cdot \alpha$. Hence, the expectation of the i.i.d random variable R denoting the transmission rate, i.e., $E[R]$, can be calculated as

$$E[R] = \frac{\alpha}{1-\xi} \cdot E[N]\,, \tag{19}$$

If LC_i lasts for $D_i = N_i \cdot \sigma$, where $\sigma = RTT$, then the total number of packets transmitted during LC_i can be calculated by

$$X_i = \int_0^{D_i} R_i(t) \cdot \mathrm{d}t \tag{20}$$

For rate-based AIMD scheme, whose rate change is performed with RTT granularity, this can be calculated by replacing the integration with the discrete summation and substituting (18) into (21) as follows

$$\begin{aligned} X_i &= \sum_{k=0}^{N_i-1} R_{i,k} \cdot \sigma \\ &= \sum_{k=0}^{N_i-1} (R_i \cdot \xi + k \cdot \alpha) \cdot \sigma \\ &= \frac{N_i}{2}\Big[2 \cdot \xi \cdot R_i + \alpha \cdot (N_i - 1)\Big] \cdot \sigma \end{aligned} \tag{21}$$

Thus for mutually independent random variables of N and R, the mean of X can be calculated by taking expectation of both sides of (21) and substituting (19) as follows

$$E[X] = \frac{E[N]}{2} \cdot \left[\left(\frac{1+\xi}{1-\xi}\right) \cdot E[N] - 1\right] \cdot \alpha \cdot \sigma \tag{22}$$

On the other hand, the total number of packets transmitted in loss cycle i can also be calculated by $n_i + R_{i,N_i-1} \cdot \sigma$, where n_i and $R_{i,N_i-1} \cdot \sigma$ are the number of packets transmitted until the dropped packet and in the last RTT, respectively. Therefore, $E[X]$ can also be calculated by

$$E[X] = E[n] + E[R] \cdot \sigma\,, \tag{23}$$

where the expectation of the random variable n is given by

$$\begin{aligned} E[n] &= \sum_k kP[n_i = k] \\ &= \sum_{k=0}^{\infty} k(1-p)^{k-1}p = \frac{1}{p}\,, \end{aligned} \tag{24}$$

where p is the packet loss probability, if loss-based congestion detection is used by the generic AIMD rate-based congestion control algorithm. By substituting (19) and (24) into (23), and equating it to (22), we solve for $E[N]$ and obtain it as follows

$$E[N] = \frac{3-\xi}{2 \cdot (1+\xi)}\left[1 + \sqrt{1 + \frac{8 \cdot (1-\xi^2)}{\alpha \cdot \sigma \cdot p \cdot (3-\xi)^2}}\right] \tag{25}$$

Thus, it follows from (17), (22) and (25) that the steady-state throughput of the rate-based generic AIMD congestion control scheme as a function of rate-increase and decrease

paratemeters, i.e., ξ and α, and round-trip time, σ, and the packet loss probability p, can be expressed as follows

$$T(\alpha,\xi,\sigma,p) = \frac{\alpha \cdot \left[1+\xi+\sqrt{(3-\xi)^2+\dfrac{8\cdot(1-\xi^2)}{\alpha\cdot\sigma\cdot p}}\right]}{4\cdot(1-\xi)} \quad (26)$$

ACKNOWLEDGMENT

This paper is dedicated to the memory of 7 astronauts on the Columbia space shuttle. One of their missions was to investigate InterPlaNetary Internet and to test a mobile Internet protocol in space.

REFERENCES

[1] O. B. Akan, J. Fang, I. F. Akyildiz, "Performance of TCP Protocols in Deep Space Communication Networks," *IEEE Commun. Lett.*, Vol. 6, No. 11, pp. 478-480, November 2002.

[2] I. F. Akyildiz, G. Morabito, S. Palazzo, "TCP-Peach: A New Congestion Control Scheme for Satellite IP Networks," *IEEE/ACM Trans. Networking*, Vol. 9, No. 3, pp. 307-321, June 2001.

[3] I. F. Akyildiz, X. Zhang, J. Fang, "TCP-Peach+: Enhancement of TCP-Peach for Satellite IP Networks," *IEEE Commun. Lett.*, Vol. 6, No. 7, pp. 303-305, July 2002.

[4] H. Balakrishnan, V. N. Padmanabhan, R. H. Katz, "The Effects of Asymmetry on TCP Performance," *Proc. ACM MOBICOM 1997*, pp. 77-89, Hungary, September 1997.

[5] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, R. H. Katz, "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links", *IEEE/ACM Trans. Networking*, Vol. 5, No. 6, pp. 756-769, December 1997.

[6] K. Bhasin, J. Hayden, J. R. Agre, L. P. Clare, T. Y. Yan, "Advanced Communication and Networking Technologies for Mars Exploration," *19th Annual AIAA International Communications Satellite Systems Conference*, Toulouse, France, April 2001.

[7] K. Bhasin, J. Hayden, "Space Internet Architectures and Technologies for NASA Enterprises," *Proc. IEEE Aerospace 2001*, Vol. 2, pp. 931-941, 2001.

[8] L. S. Brakmo, S. O Malley, L. L. Peterson, "TCP Vegas: New Techniques for Congestion Detection and Avoidance," *Proc. ACM SIGCOMM 1994*, pp. 24-35, October 1994.

[9] C. Casetti, M. Gerla, S. Mascolo, M. Y. Sanadidi, R. Wang, "TCP-Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links," *Proc. ACM MOBICOM 2001*, pp. 287-297, Rome, July 2001.

[10] Consultative Committee for Space Data Systems, "Space Communications Protocol Specification-Transport Protocol (SCPS-TP)", *Recommendation for Space Data Systems Standards, CCSDS 714.0-B-1.*, Blue Book. Issue 1, Washington, D.C.: CCSDS, May 1999.

[11] R. C. Durst, G. J. Miller, E. J. Travis, "TCP Extensions for Space Communications," *ACM/Kluwer Wireless Networks*, Vol. 3, No. 5, pp. 389-403, October 1997.

[12] R. C. Durst, P. D. Feighery, K. L. Scott, "Why not use the Standard Internet Suite for the Interplanetary Internet?," *http://www.ipnsig.org/techinfo.htm*.

[13] S. Floyd, T. Henderson, "The NewReno Modification to TCP's Fast Recovery Algorithm," *RFC 2585*, April 1999.

[14] T. Goff, J. Moronski, D. S. Phatak, V. Gupta, "Freeze-TCP: A True End-to-End TCP Enhancement Mechanism for Mobile Environments," *Proc. INFOCOM 2000*, Vol. 3 , pp. 1537-1545, Israel, 2000.

[15] T. R. Henderson, R. H. Katz, "Transport Protocols for Internet-Compatible Satellite Networks," *IEEE J. Select. Areas Commun.*, Vol. 17, No. 2, pp. 326-344, February 1999.

[16] V. Jacobson, "Congestion Avoidance and Control," *Proc. ACM SIGCOMM 1988*, pp. 314-329, Stanford, August 1988.

[17] V. Jacobson, "Berkeley TCP Evolution from 4.3-Tahoe to 4.3-Reno," *Proc. British Columbia Internet Engineering Task Force*, July 1990.

[18] T. V. Lakshman, U. Madhow, "The Performance of TCP/IP for Networks with High Bandwidth-Delay Products and Random Loss," *IEEE/ACM Trans. Networking*, Vol. 5, No. 3, pp. 336-350, June 1997.

[19] D. Logothetis, K. S. Trivedi, A. Puliafito, "Markov Regenerative Models", *Proc. Int. Computer Performance and Dependability Symp.*, pp. 134-143, Erlangen, Germany 1995.

[20] M. Mathis, J. Mahdavi, S. Floyd, A. Romanov, "TCP Selective Acknowledgement Options," *RFC 2018*, 1996.

[21] M. Mathis, J. Mahdavi, "Forward Acknowledgement: Refining TCP Congestion Control," *Proc. ACM SIGCOMM 1996* , pp. 281-292, Aug. 1996.

[22] J. Padhye, V. Firoio, D. Towsley, J. Kurose, "Modeling TCP Reno Performance: A Simple Model and Its Empirical Validation", *IEEE/ACM Trans. Networking*, Vol. 8, No. 2, pp. 133-145, April 2000.

[23] P. Sinha, N. Venkitaraman, R. Sivakumar, V. Bharghavan, "WTCP: A Reliable Transport Protocol For Wireless Wide-Area Networks," *Proc. ACM MOBICOM 1999*, pp. 231-241, Seattle, Washington, August 1999.

[24] D. T. Tran, F. J. Lawas-Grodek, R. P. Dimond, W. D. Ivancic, "SCPS-TP, TCP and Rate-Based Protocol Evaluation for High-Delay, Error-Prone Links," *SpaceOps 2002*, Houston, TX, October 2002.

[25] E. Travis, "The Interplanetary Internet: Architecture and Key Technical Concepts," *Internet Global Summit*, INET 2001, June 5, 2001.