Advanced Transport Protocols for Space Communications

A Thesis
Presented to
The Academic Faculty
by
Jian Fang

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in Electrical and Computer Engineering

School of Electrical and Computer Engineering
Georgia Institute of Technology
November 2004

Copyright © 2004 by Jian Fang
Advanced Transport Protocols for Space Communications

Approved by:

Professor Ian F. Akyildiz, Committee Chair
Professor Ragupathy Sivakumar
Professor Chuanyi Ji

Professor Gordon L. Stüber
Professor Mostafa Ammar

Date Approved: November 19, 2004
To my family.
ACKNOWLEDGEMENTS

I am truly grateful to everyone who has supported and helped me complete this Ph.D. dissertations. First of all, I would like to thanks my advisor, Professor Ian F. Akyildiz, for giving me a chance to work with him in the Broadband and Wireless Networking Laboratory at Georgia Institute of Technology about four years ago. He has been constant in his valuable guidance and support in my research. He offered many insightful feedback on my research. This thesis would not have been possible without his support and guidance. Professor Akyildiz also helped me a lot in other ways that could lead me to the success in my future career.

I would like to thank my dissertation committee member, Professor Raghupathy Sivakumar and Professor Chuanyi Ji, for their comments and suggestions on my research proposal and the draft of my dissertations. I would like to thank Professor Paul G. Steffes for his comments on my research proposal. I would also like to thank Professor Gordon L. Stüber and Professor Mostafa Ammar for serving on my dissertation defense committee and sharing their expertise in networking research.

I would also like to thank all the members in the Broadband and Wireless Networking Laboratory at Georgia Institute of Technology for their help, encouragement, and friendship. Especially thank Dr. Caterina Scoglio, Dr. Xudong Wang, Dr. Jiang Xie, Shantidev Mohanty, Dr. Ozgur Baris Akan, Chao Chen, and Mehmet Can Vuran for their generous help during my hardest time.

Last but not the least, I thank my parents, Xiuzhen Liu and Zhihua Fang, and my wife, Xin Fu, for their constant love and support, which have always been the endless source of my strength.
# TABLE OF CONTENTS

DEDICATION ............................................................... iii
ACKNOWLEDGEMENTS ...................................................... iv
LIST OF TABLES ............................................................. x
LIST OF FIGURES ............................................................ xi
SUMMARY ................................................................. xiv

CHAPTER I  INTRODUCTION ................................................ 1
  1.1 Transport Protocols for Satellite IP Networks .................. 4
      1.1.1 Unicast Transport Protocols ............................... 4
      1.1.2 Multicast Transport Protocols ............................ 5
  1.2 Transport Protocols for the InterPlaNetary Internet .......... 7
      1.2.1 Reliable Data Transport Protocols ........................ 8
      1.2.2 Multimedia Transport Protocols .......................... 9
  1.3 Research Objectives and Solutions .............................. 10
      1.3.1 Unicast Transport Protocol for Satellite IP Networks ... 11
      1.3.2 Reliable Data Multicast Transport for Satellite IP Networks 11
      1.3.3 Reliable Data Transport Protocol for the InterPlaNetary Internet ........................................ 12
      1.3.4 Multimedia Transport Protocol for the InterPlaNetary Internet 12
  1.4 Thesis Outline ................................................... 13

CHAPTER II  UNICAST TRANSPORT PROTOCOLS FOR SATELLITE IP NETWORKS ........................................ 14
  2.1 Problem and Solution ............................................. 14
  2.2 Congestion Control in TCP-Peach++ ............................. 15
      2.2.1 NIL Segments .............................................. 15
      2.2.2 The Jump Start Algorithm ................................. 17
      2.2.3 Quick Recovery Algorithm ............................... 19
3.4.1 Link Error Properties ...................... 53
3.4.2 Logical Hierarchical Multicast Groups ........ 55
3.4.3 Throughput Performance .................... 59
3.4.4 Overhead .................................. 60
3.4.5 Fairness to Unicast TCP Traffic ............ 61
3.4.6 Unbalanced Situation ....................... 63
3.4.7 Rain Fade ................................ 64
3.4.8 Bandwidth Asymmetry ....................... 65
3.4.9 Scalability ................................ 66

CHAPTER IV RELIABLE DATA TRANSPORT PROTOCOLS FOR
INTERPLANETARY INTERNET ....................... 68

4.1 Problem and Solution ......................... 68
4.2 TP-Planet: Initial State ....................... 74
  4.2.1 Immediate Start .......................... 75
  4.2.2 The Follow-Up Phase ..................... 77
  4.2.3 Determination of the Time Interval $T$ .......... 78
  4.2.4 The Connection Establishment ............. 82
4.3 TP-Planet: Steady State ....................... 82
  4.3.1 Congestion Control ...................... 83
  4.3.2 The New Rate-Based AIMD Scheme ............ 86
  4.3.3 The Blackout State Behavior .............. 88
  4.3.4 The Delayed SACK ....................... 91
4.4 Performance Evaluation ....................... 91
  4.4.1 Initial State Performance .................. 92
  4.4.2 Throughput Performance .................. 95
  4.4.3 Overhead ................................ 97
  4.4.4 Blackout Conditions ...................... 99
  4.4.5 Bandwidth Asymmetry .................... 100
6.2.1 Application-Level Multicast Protocol Built on Peer-to-Peer Overlays .................................. 145

6.2.2 Cooperative Problem-Solving Framework for Delay-Tolerant Networks .................................. 145

6.2.3 Cross Layer Optimization .................................. 146

REFERENCES .................................................. 147

VITA .................................................. 155
LIST OF TABLES

Table 1  The Resulting Logical Hierarchical Groups for $N = 200$ and $M = 10$. 56
Table 2  The Resulting Hierarchical Groups for $N = 200$ and $TRTT = 30ms$. 58
Table 3  Protocol Parameters Used in Experiments. . . . . . . . . . . . . . 96
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The architecture of space communications.</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Research topics on space communications.</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td>TCP-peach++: Flow Chart.</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>The NIL Segment Generating Algorithm.</td>
<td>18</td>
</tr>
<tr>
<td>5</td>
<td>The Jump Start Algorithm.</td>
<td>18</td>
</tr>
<tr>
<td>6</td>
<td>The Quick Recovery Algorithm.</td>
<td>21</td>
</tr>
<tr>
<td>7</td>
<td>The throughput performance comparison of TCP-Peach++ and TCP-Peach for different values of $P_{Loss}$.</td>
<td>24</td>
</tr>
<tr>
<td>8</td>
<td>The overhead comparison of TCP-Peach++ and TCP-Peach for different values of $P_{Loss}$.</td>
<td>25</td>
</tr>
<tr>
<td>9</td>
<td>The throughput vs. rain fade period.</td>
<td>27</td>
</tr>
<tr>
<td>10</td>
<td>The throughput vs. delay factor.</td>
<td>28</td>
</tr>
<tr>
<td>11</td>
<td>The Satellite Multicast Scenario.</td>
<td>33</td>
</tr>
<tr>
<td>12</td>
<td>A typical B+ tree</td>
<td>36</td>
</tr>
<tr>
<td>13</td>
<td>The Random Selection Algorithm.</td>
<td>38</td>
</tr>
<tr>
<td>14</td>
<td>The procedure for members to join the multicast group.</td>
<td>40</td>
</tr>
<tr>
<td>15</td>
<td>The procedure for members to leave the multicast group.</td>
<td>41</td>
</tr>
<tr>
<td>16</td>
<td>The NAK Fusion Algorithm.</td>
<td>48</td>
</tr>
<tr>
<td>17</td>
<td>The ACK Fusion Algorithm.</td>
<td>49</td>
</tr>
<tr>
<td>18</td>
<td>The Logical Hierarchical Structure for TCP-Peachtree.</td>
<td>52</td>
</tr>
<tr>
<td>19</td>
<td>The Aggregated Packet Loss Rate from the satellite to the receivers vs. $p$ for $TRTT = 30ms$ and $M = 10$.</td>
<td>55</td>
</tr>
<tr>
<td>20</td>
<td>Average RTT vs. $TRTT$ for $N = 200$, $M = 10$ and $p = 10^{-3}$.</td>
<td>57</td>
</tr>
<tr>
<td>21</td>
<td>RTT vs. $M$ for $N = 200$, $TRTT = 30ms$ and $p = 10^{-3}$.</td>
<td>58</td>
</tr>
<tr>
<td>22</td>
<td>Throughput Performance Comparison of TCP-Peachtree and TCP-NewReno for different values of $p$ for $TRTT = 30ms$ and $M = 10$.</td>
<td>59</td>
</tr>
<tr>
<td>23</td>
<td>The overhead vs. $p$ for $TRTT = 30ms$ and $M = 10$.</td>
<td>61</td>
</tr>
<tr>
<td>24</td>
<td>The fairness vs. $p$ for $TRTT = 30ms$ and $M = 10$.</td>
<td>62</td>
</tr>
</tbody>
</table>
Figure 25  The throughput comparison for balanced and unbalanced packet loss rates. ........................................ 63
Figure 26  The throughput vs. rain fade period. ........................................ 65
Figure 27  The throughput vs. delay factor. ........................................ 66
Figure 28  The scalability for $TRIT = 30ms$ and $M = 10$. ...................... 67
Figure 29  Deep space network architecture for the Mars Exploration mission. 69
Figure 30  TP-Planet protocol operation state diagram including substates and
the state transitions based on congestion control decision mechanism. 72
Figure 31  An example illustration of time framing mechanism in the Immediate
Start phase for $s\text{thresh}_e = 8$. ........................................ 74
Figure 32  The InitialState() algorithm. ........................................ 79
Figure 33  The SteadyState() algorithm. ........................................ 85
Figure 34  The blackout condition observed from the TP-Planet source for $L < 2x$. 89
Figure 35  The blackout condition observed from the TP-Planet source for $L \geq
2x$. ........................................ 90
Figure 36  The transmission rate change in the Initial State of the TP-Planet
source for $RTT = 600$ seconds. ........................................ 92
Figure 37  The congestion window size change during the Jump Start algorithm
of TCP-Peach+ and the Slow Start algorithm of TCP-NewReno and
for $RTT = 600$ seconds. ........................................ 93
Figure 38  The throughput for changing $p$ and file size with $RTT = 600$ seconds. 94
Figure 39  The throughput for changing $RTT$ for file size of 50MB and $p = 10^{-5}$. 96
Figure 40  The overhead resulting from the transmission of NIL and NIX segments
for changing file size and $p$ with $RTT = 600$ seconds. 98
Figure 41  The throughput for changing blackout duration with $RTT = 120$
seconds and $p = 10^{-5}$ ........................................ 99
Figure 42  The throughput for changing delayed SACK factor $d$ with $RTT =
120$ seconds and $p = 10^{-4}$. ........................................ 100
Figure 43  Throughput performance of existing multimedia rate control schemes
for very high RTT ranges. ........................................ 107
Figure 44  The RCP-Planet operation state diagram. ............................. 109
Figure 45  The Structure of Packet Level FEC. ................................. 111
Figure 46  The overhead of FEC vs. packet loss rate $p$. ........................ 113
SUMMARY

Satellite IP networks are characterized by high bit error rates, long propagation delays, low bandwidth feedback links, and persistent fades resulting from varying weather patterns. A new unicast transport protocol is designed to address all the above challenges. Two new algorithms, Jump Start and Quick Recovery, are presented to replace the traditional Slow Start algorithm and to recover rapidly from multiple segment losses within one window of data. Furthermore, a Hold State is developed to address persistent fades. The characteristics of satellite IP networks also distinguish satellite multicasting from multicasting in terrestrial wirelined networks. A reliable data multicast transport protocol, TCP-Peachtree, is proposed. TCP-Peachtree uses a modified B+ tree-like hierarchical multicast group to solve the acknowledgment implosion and scalability problems in satellite IP networks.

Developments in space technology are enabling the realization of deep space missions. The scientific data from these missions need to be delivered to the Earth successfully. To achieve this goal, the InterPlaNetary Internet is proposed as the Internet of the deep space planetary networks, which is characterized by extremely high propagation delays, high link errors, asymmetrical bandwidth, and blackouts. A reliable transport protocol, TP-Planet, is proposed for data traffic in the InterPlaNetary Internet. TP-Planet deploys rate-based additive-increase multiplicative-decrease (AIMD) congestion control and replaces the inefficient slow start algorithm with a novel Initial State algorithm that allows the capture of link resources in a very fast and controlled manner. A new congestion detection and control mechanism is developed and a Blackout State is incorporated into the protocol operation. Multimedia traffic is also one part of the aggregate traffic over InterPlaNetary Internet backbone links.
and it has additional requirements such as minimum bandwidth, smooth traffic, and error control. To address all the above challenges, RCP-Planet is proposed. RCP-Planet consists of two novel algorithms, i.e., Begin State and Operational State. The protocol is based on a novel rate probing mechanism and a new rate control scheme to update the media rate smoothly based on the observed rate for the probing sequence. Tornado codes are used for packet level FEC because of their fast encoding and decoding speed.
CHAPTER I

INTRODUCTION

The advent of the “Space Age” has revolutionized the field of telecommunications and the space communication becomes one of the most active areas of technology development of our time. This development is driven by the demand for global coverage and the interest in deep space explorations. The architecture of space communications is shown in Figure 1. Usually it includes the near Earth communication networks, such as satellite communication networks, and deep space networks.

![Diagram of space communications](image)

**Figure 1**: The architecture of space communications.

The technology of space communications is also highly innovative with a large range of actual and potential applications. For example, the satellite communication system can be used for carrying a wide range of communication services, including Internet access, television and telephony. For the broadband Internet, usually GEO satellites are used and they provide the users with a downlink. A dial-up modem is the most common form of back channel. Thus, satellite networks offer Internet access to remote and widely scattered locations. Satellite networks are capable of sustaining high bandwidth levels and supporting flexible and scalable network configurations and can also be used as a backup for existing networks in case of congestions or link
failures.

Satellite IP networks are characterized by high bit error rate (BER), long propagation delays, low bandwidth feedback link, and persistent fades because of varying weather patterns [6, 8, 55]. Thus, the space environment poses challenges in providing reliable and end-to-end data communication. Traditional TCP schemes usually perform poorly in these situations [60] and thus new transport protocols must be designed to address all the above challenges.

Satellite networks are broadcast in nature and definitely have a role in providing the multicast service to complement terrestrial networks. There are a number of issues that prevent Internet Service Providers (ISPs) from deploying and operating wide scale multicast networks in terrestrial networks. The issues include the difficulties in providing the multicast service across a wide geographical area and the difficulties in upgrading the existing deployed network routers. On the other hand, Satellite multicast allow the same data to be received by any receiver within the coverage area of the satellite. Thus, it minimizes the number of multicast-capable routers required in the network and simplifies the deployment, operations, and maintenance. Satellite can be use to distribute software products, multimedia content, images, and other kind of data to a large number of receivers in a wide geographical area. Satellite multicast is the most efficient way to reach the largest user in the most cost-effective manner.

Satellite networks have high bit error rate (BER), long propagation delays, low bandwidth feedback link, and asymmetrical bandwidth [4]. These characteristics distinguish satellite multicasting from multicasting in terrestrial wirelined networks. In addition to the acknowledgment implosion and scalability problems in terrestrial wirelined networks, satellite multicasting has a different multicast topology. However, relatively little research has been conducted on reliable multicast protocols [53, 55, 56]. Therefore, there is an urgent need for new satellite multicast protocols.
The developments in space technologies are enabling the realization of deep space missions. Recently, research interest in deep space have been increasing rapidly, including scientific spacecraft traveling, Mars exploration, radio and radar astronomy observations of the solar system, and the universe. Future space missions to the deep space require communications among planets, moons, satellites, asteroids, robotics spacecrafts, and crewed vehicles.

The scientific data from these missions need to be delivered to the Earth successfully. To achieve this goal, InterPlaNetary Internet is proposed as the Internet of deep space planetary networks [7, 87]. As shown in Figure 1, a typical InterPlaNetary Internet architecture includes InterPlaNetary Backbone Network, InterPlaNetary External Network, and Planetary Network.

- **InterPlaNetary Backbone Network**: It provides a common infrastructure for communications among the Earth, outer-space planets, moons, satellite relays, etc. It includes the data links (direct link or multi-hop paths) between elements with long-haul capabilities.

- **InterPlaNetary External Network**: It consists of spacecrafts flying in groups in deep space among planets, clusters of sensor nodes, and groups of space stations, etc. Some nodes also have long-haul communication capabilities.

- **Planetary Network**: This architecture can be implemented at any outer-space planet, providing interconnection and cooperation among the satellites and surface elements on a planet.

The possible applications of the InterPlaNetary Internet include time-insensitive data delivery, time-sensitive scientific data delivery, mission status telemetry, and command and control. For the Mars-Earth communications, the transport layer functionalities are necessary for both the reliable transfer of the scientific data and the
timely delivery of the multimedia information in InterPlaNetary Internet. Multimedia transport protocols should be able to address the additional challenges resulting from the unique requirements of multimedia applications, i.e., smooth traffic, and error control [37].

1.1 Transport Protocols for Satellite IP Networks

1.1.1 Unicast Transport Protocols

TCP has been heavily used in the Internet because of its ability to probe for unused network bandwidth and the back-off mechanism upon detection of congestions in the network. TCP maintains a congestion window, which presents the amount of outstanding data at the current time, to determine its sending rate. During the slow start, the congestion window is initialized to be one segment and it is doubled for each ACK that the sender received until a threshold is met and the congestion avoidance phase is entered. During this phase, the congestion window is increased at most one segment per RTT. Upon detection of congestion, the congestion window is halved. TCP also adopts a timeout mechanism if any retransmission is lost, which usually indicates a severe congestion. In such a case, the congestion window is reduced to one segment and the slow start is entered.

The space environment poses challenges for reliable data communications. Traditional congestion control schemes perform poorly in satellite networks because of high bit error rates (BER) and long propagation delays [4]. One reason is that TCP protocols were initially designed to work in terrestrial networks with low link error rates. All packet losses were mostly resulting from network congestions. As a result, the TCP sender decreases its transmission rate even though the packet losses are due to link error. However, this causes unnecessary throughput degradation. Another reason is that the long propagation delay cause longer duration of the slow start phase and thus the bandwidth utilization is degraded.
To address the challenges in satellite IP networks, the transport protocol needs to identify the source of packet losses and react appropriately. Moreover, the congestion control algorithms need to address the long duration of the slow start. These problems were addressed in TCP Peach [5]. In TCP-Peach, dummy segments are used to probe the availability of network resources to improve the performance when segment losses are due to link errors instead of congestion. However, TCP-Peach cannot recover multiple packet losses in one window of data. Furthermore, satellite IP networks are also characterized by low bandwidth feedback link and persistent fades, which are not addressed in TCP-Peach. As a result, new unicast transport protocols need to be proposed to address all the above problems.

1.1.2 Multicast Transport Protocols

IP Multicasting provides a way to the same packet simultaneously to a large number of users. In contrast to broadcast, multicast is selective and a user only accepts a stream of packets by joining a specific multicast group address. Usually, multicast-capable routers construct a multicast tree with the sender sitting at the root node and the receivers locating at the leaf nodes. Intermediate routers along the tree duplicate the incoming packets and send one copy to each branch where there are downstream receivers. IP multicasting provides an efficient way to disseminate data from a sender to a group of receivers. However, there are still challenges for the large scale deployment of multicast services over the terrestrial networks, such as router and software upgrades. Furthermore, the issues of scalability, network stability, and fairness to unicast TCP traffic prevent ISPs to offer multicast service to users.

Satellite offers a natural way of providing multicast service to a large number of users over large geographical areas and allows a single-hop access to the Internet, thus, it bypasses congested multiple hops over terrestrial networks. On the other hand, satellite networks have high bit error rate (BER), long propagation delays, and
asymmetrical bandwidth [4]. These characteristics distinguish satellite multicasting from multicasting in terrestrial wirelined networks. In addition to the acknowledgment implosion and scalability problems in terrestrial wirelined networks, satellite multicasting has a different multicast topology because all receivers receive packets directly from the satellite and there are no intermediate routers between the satellite and the receivers. On the other hand, the receivers send acknowledgments directly to the satellite and there is no physical hierarchy between the satellite and the receivers. Consequently, the multicast schemes [40, 49, 59, 71, 76, 91, 92] that are based on multicast trees and are developed for terrestrial multicast networks cannot be applied in satellite multicasting.

Reliable satellite multicast protocols must be able to address the following challenges:

- **Reliability:** For reliable data transmissions, receivers may experience packet losses resulting from either link errors or congestions. The receivers must acknowledge the receipt of data to guarantee reliability.

- **Feedback Suppression:** In satellite multicast applications, the acknowledgments from the receivers may result in a feedback implosion at the sender. A feedback suppression scheme must be adopted to reduce the number of acknowledgments from the receivers.

- **Congestion Control:** Congestion control algorithms must be designed to prevent overwhelming the network with data and also be fair to other traffic in the network.

- **Scalability:** In satellite multicast applications, there may be thousands, or tens of thousands of receivers in the multicast session. The multicast protocols must be able to support such a large number of receivers and also to address heterogeneity of receivers with different link error rates.
1.2 Transport Protocols for the InterPlaNetary Internet

Deep space scientific missions such as Mars exploration produce a significant amount of scientific data to be delivered to the Earth. The successful mission of Mars exploration requires a robust, dependable, high capacity space communication infrastructure over the vast distance, heterogeneous, and extreme environment. The InterPlaNetary Backbone Network plays a significant role in the performance of the entire deep space communication. The characteristics of the InterPlaNetary Backbone Network can be summarized as follows:

- **Extremely Long Propagation Delays:** The InterPlaNetary backbone links usually 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 [30].

- **High Link Error Rates:** The bit error rates on the deep space links are very high, usually on the order of $10^{-1}$ [30].

- **Blackout:** Periodic link outages may occur because of orbital obscuration with the loss of line-of-sight resulting from moving planetary bodies or the interference of an asteroid or a spacecraft [15].

- **Bandwidth Asymmetry:** The asymmetry in the bandwidth capacity of forward and reverse channels is typically on the order of 1000 : 1 in space missions [30].

To successfully deliver the scientific data to the Earth, an urgent need exists for transport layer protocols for the InterPlaNetary Internet.
1.2.1 Reliable Data Transport Protocols

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 because of high propagation delays. Slow start algorithms of 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.

Many transport protocols [5, 6, 47] are proposed for satellite links, which are also characterized by high bandwidth-delay products and high BERs. 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 resulting from blackout conditions may also mislead the congestion control mechanisms based on packet losses.

Space Communications Protocol Standards-Transport Protocol (SCPS-TP) [24, 29] is a set of TCP extensions developed by the 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 [24]. SCPS-TP was developed based on 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. The capabilities of the SCPS-TP are basically a combination of existing TCP protocols, which are shown to be inadequate in addressing the challenges in the Interplanetary Backbone Network [2].

1.2.2 Multimedia Transport Protocols

Multimedia traffic including planet images and data from scientific observations will be transmitted over the deep space communication links. Multimedia transport protocols need to address the challenges posed by the InterPlanetary Internet Backbone Network, i.e., extremely long propagation delays, high link errors, asymmetrical bandwidth, and blackouts. In addition to that, they should be able to address the additional challenges because of the unique requirements of multimedia applications, such as minimum bandwidth, smooth traffic, and error control.

Many multimedia transport protocols are proposed to control the flow of multimedia traffic in terrestrial networks [25, 45, 66, 74, 84]. These proposed protocols can be mainly categorized into two types of rate control schemes, i.e., additive increase multiplicative decrease (AIMD-based) and equation-based.

AIMD-based rate control schemes are TCP-compatible, i.e., they compete reasonably fairly with the existing TCP by following TCP behavior to conservatively update the sending rate based on feedback information [25, 74, 84]. Streaming Control Protocol (SCP) [25] is a modified version of TCP that performs TCP-Vegas-like rate adjustment. Rate adaptation protocol (RAP) [74] is a rate-based congestion control mechanism for wired and short distance networks. Rate control scheme (RCS) [84] is a rate control scheme for real-time traffic in networks with high bandwidth-delay products and lossy links. However, all of these existing AIMD-based rate control schemes [25, 74, 84] were developed based on the assumption that the propagation delay is relatively short, which does not hold in the InterPlaNetary Backbone Network links. Moreover, the AIMD schemes cause abrupt and frequent fluctuations in
the media rate in the form of a saw-tooth pattern, which is not suitable for most multimedia applications.

The equation-based rate control schemes [45, 66] are proposed to provide relatively smooth congestion control for multimedia traffic in the terrestrial networks. The idea of equation-based congestion control is to adjust the transmission rate no more than the estimated throughput of the corresponding TCP counterpart experiencing the same packet loss rate, round trip time, and packet size. Although the use of the TCP response function ensures that equation-based control schemes compete fairly with TCP over long time scales, the steady-state throughput model of TCP source is highly sensitive to RTT values. Therefore, the equation-based rate control schemes cannot achieve high link utilization and hence are not promising solutions for the InterPlaNetary Backbone Network links with extremely high propagation delays.

The SCPS Rate-based protocol is proposed for space communications [86], but without a congestion control algorithm. The transmission rate in the SCPS rate-based protocol is defined by the user and also constrained by the receiver buffer size. In other words, the SCPS rate-based protocol does not adapt its transmission rate to the network conditions. Thus, it may cause congestion for InterPlaNetary Internet backbone links if its transmission rate is higher than the available bandwidth.

1.3 Research Objectives and Solutions

The above characteristics of satellite networks and the InterPlaNetary Internet lead to different research challenges. In this thesis, four research topics are investigated, as shown in Figure 2. One of the important issues is to design the unicast and multicast transport protocols for satellite networks. For Mars-Earth communications, reliable data transport protocols and multimedia transport protocols need to be proposed.
Figure 2: Research topics on space communications.

1.3.1 Unicast Transport Protocol for Satellite IP Networks

A new congestion control protocol, TCP-Peach++, was proposed for satellite IP networks [6, 35]. A new type of low priority segment, NIL segment, is used to probe the availability of network resources as well as error recovery. Two new algorithms, Jump Start and Quick Recovery, are adopted in TCP-Peach++ to recover rapidly from multiple segment losses within one window of data. The delayed SACK scheme is adopted to address the bandwidth asymmetry problems and a Hold State is developed to address persistent fades.

1.3.2 Reliable Data Multicast Transport for Satellite IP Networks

A reliable multicast transport protocol, TCP-Peachtree, was proposed for satellite IP networks [8]. TCP-Peachtree uses a modified B+ tree-like hierarchical multicast group to solve the acknowledgment implosion and scalability problems in reliable IP multicast applications. The two new congestion control algorithms are adopted, i.e., Jump Start and Quick Recovery, so that TCP-Peachtree is suitable for satellite IP networks with long propagation delays and high bit error rates. NIL segments are used to exploit the availability of network resources and recover lost packets on
the receiver side. The delayed SACK scheme is adopted to address the bandwidth asymmetry problems. Furthermore, a Hold State is developed to address persistent fades.

1.3.3 Reliable Data Transport Protocol for the InterPlaNetary Internet

A reliable data transport protocol, TP-Planet, was proposed for InterPlaNetary Internet [3]. TP-Planet replaces the inefficient slow start algorithms with a novel Initial State algorithm, which captures link resources in a very fast and controlled manner. To address the challenges because of extremely high propagation delay, TP-Planet deploys 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 decisions from single packet loss events, is developed to minimize the erroneous congestion decisions because of high link errors. To reduce the effects of blackout conditions on the throughput performance, TP-Planet incorporates Blackout State behavior into the protocol operation. The bandwidth asymmetry problem is addressed by the adoption of delayed SACK options.

1.3.4 Multimedia Transport Protocol for the InterPlaNetary Internet

A multimedia transport protocol, RCP-Planet, was proposed for InterPlaNetary Internet [36]. Two novel algorithms, Begin State and Operational State, were designed. A novel rate probing mechanism is proposed to capture the available bandwidth. Based on the rate probing mechanism, the new rate control scheme updates the media rate smoothly and conservatively in the Operational State. To recover packet losses because of link errors and congestions, Tornado codes are used for packet level FEC because of their very fast encoding and decoding times. The FEC block length is chosen appropriately to minimize the FEC overhead. Furthermore, FEC block level ACKs are used to address bandwidth asymmetry problems. Apart from that,
the blackout state is incorporated into RCP-Planet to improve the performance in blackout conditions.

1.4 Thesis Outline

In this thesis, the above research challenges are addressed and the solutions are proposed. A new unicast transport protocol is described in Chapter 2. The reliable data multicast transport protocols for satellite IP networks is presented in Chapter 3. In Chapter 4, a new reliable data transport protocol for the Mars-Earth communications is discussed. The multimedia transport protocol for Mars-Earth communications is provided in Chapter 5 and is followed by the conclusions and future research work in Chapter 6.
CHAPTER II

UNICAST TRANSPORT PROTOCOLS FOR SATELLITE IP NETWORKS

2.1 Problem and Solution

Satellite networks will play a crucial role in the global infrastructure of the Internet. They do not only provide global coverage, but also are capable of sustaining high bandwidth levels and supporting flexible and scalable network configurations. Satellite networks can also be used as a backup for existing networks, e.g., in case of congestions or link failures, traffic can be rerouted through satellites. Satellite networks have high bit error rate (BER) and long propagation delays[6]. Traditional TCP schemes usually perform poor in these situations [60] and thus, TCP-Peach [5] is proposed to address these problems. Although it solves the problem of throughput degradation in satellite networks over Fast Recovery [51], when the link error is high and multiple segment losses occur within one window of data, the throughput degradation is still large. Moreover, satellite IP networks are also characterized by the following problems, which are not addressed in TCP-Peach [5],

- Low Bandwidth Feedback Link

  A terrestrial feedback link to the sender has been proposed in [53]. However, this is difficult to implement in some cases, especially for mobile receivers, which need to use low-bandwidth uplink channel as the feedback link. The feedback link is usually not faster than several hundred Kbps for small satellite terminals and a few Mbps for larger satellite terminals [47].

- Persistent Fades
Satellite link channels can experience persistent link fades because of varying weather patterns [8, 55]. The congestion control algorithms should be able to address persistent fades to reduce performance degradation and unnecessary transmissions when the links are unavailable.

Considering the challenges, we propose a unicast transport protocols for satellite IP networks, called TCP-Peach++, to improve the throughput performance for satellite IP networks with asymmetrical bandwidth and persistent fades. The delayed SACK scheme is adopted to address the bandwidth asymmetry problems and a Hold State is incorporated into the congestion control in TCP-Peach++ to address the persistent fades.

The remainder of the chapter is organized as follows. The congestion control algorithms in TCP-Peach++ are presented in Section 2.2 and simulation results are given in Section 2.3.

2.2 Congestion Control in TCP-Peach++

TCP-Peach++ includes the following new schemes: Jump Start and Quick Recovery, and two classical algorithms, Congestion Avoidance, and Fast Retransmit. The TCP SACK option [61] is also adopted in TCP Peach++ to improve the performance when multiple segments are lost from one window of data. To address the persistent fades resulting from varying weather patterns, such as rain fades, a Hold State is incorporated into the congestion control in TCP-peach++. The flow chart of TCP-Peach++ congestion control algorithm is illustrated in Figure 3.

2.2.1 NIL Segments

In TCP-Peach [5], the dummy segments, are used to probe the availability of network resources to improve the performance when segment losses are due to link error instead of congestion. In TCP-Peach++, we introduce low priority segments, called NIL
Figure 3: TCP-peach++: Flow Chart.

segments to probe the availability of network resources as well as error recovery. There are two main differences between dummy segments and NIL segments. First, dummy segments are used only to probe the availability of network resources, while NIL segments are used to probe the availability of network resources and also for error recovery. Another difference is the way how the low priority segments are generated. In TCP-Peach, dummy segments are generated by the sender as a copy of the last transmitted data segment. While NIL segments are generated using the NIL Segment Generating Algorithm as shown in Figure 4 to carry unacknowledged packets, which can be used by the receivers to recover missing packets. Since the probability for a packet to be lost somewhere in the satellite multicast network is rather high, using NIL segments to recover errors is advantageous.

Low-priority segments are also used in [68] to improve TCP Slow Start Algorithm. However, the low-priority segments in [68] are different from NIL segments in that:

- They are not used to probe the availability of network resources. In fact, their objective is to carry information to the receiver more rapidly without harming other flows.
- Since they carry new information to the receiver, they are still data segments, and if they are lost, then they must be recovered.

- They are used only in the beginning of a new connection.

NIL segments are low priority segments that do not carry any new information. If a router on the connection path is congested, then it discards the NIL segments first. Consequently, the transmission of NIL segments does not cause a decrease of throughput of actual data segments. If the routers are not congested, then the NIL segments can reach the receiver. The sender sets one or more of the six reserved bits in the TCP header to distinguish NIL segments from data segments. Therefore, the receiver can recognize the NIL segments and for each of them transmits an ACK back to the sender. The ACKs for NIL segments are also marked using one or more of the six reserved bits of the TCP header and are carried by low priority IP segments. Upon receiving ACKs for NIL segments, the sender interprets those ACKs as the evidence that there are unused resources in the network and can increase its transmission rate accordingly. NIL segments are only transmitted when the sender is in the Jump Start phase or when packet losses are first found during the Quick Recovery phase.

Let $Q$ be the queue length and $i$ be a counter, then the NIL segments are created as shown in Figure 4.

2.2.2 The Jump Start Algorithm

In satellite networks, long propagation delays cause TCP Slow Start to be inefficient since its $cwnd$ increases slowly. In TCP-Peach [5], a new scheme called TCP Sudden Start is proposed to improve TCP Slow Start. Dummy segments are used to probe the availability of network resources and increase $cwnd$ quickly. To further improve the Slow Start Algorithm, in TCP-Peach++, we propose a Jump Start algorithm using NIL segments which is described in 2.2.1 to probe the availability of network resources and recover errors at the same time. The algorithm is shown in Fig 5.
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;

Figure 4: The NIL Segment Generating Algorithm.

Jump_Start()
    cwnd = 1;
    \tau = RTT/rwnd;
    send(Data_Segment);
    for (i = 1 to rwnd - 1)
        wait(\tau);
        send(NIL_Segment);
    end;
end;

Figure 5: The Jump Start Algorithm.

In Jump Start, the TCP sender sets the congestion window, cwnd, to 1. After
sending the first data segment, it transmits (rwnd - 1) NIL segments generated as
in Fig 4 every \( \tau = RTT/rwnd \), where rwnd is the receiver window size. As a result,
after one round trip time, the congestion window size cwnd increases very quickly as
the ACKs for NIL segments arrive at the sender.
2.2.3 Quick Recovery Algorithm

In TCP-Peach, dummy segments are transmitted in Rapid Recovery to resume $cwnd$ rapidly from the decrement resulting from segment losses caused by link errors. Although it solves the problem of throughput degradation in satellite networks over Fast Recovery [51], when the link error is high and multiple segment losses occur within one window of data, the throughput degradation is still large. So we propose Quick Recovery to recover from high link errors.

As the TCP SACK proposed in [33], we adopt the SACK option in TCP-Peach++ to avoid retransmitting out of order segments received at the destination. At the sender, a data structure called scoreboard is maintained to update information about cumulatively ACKed and SACKed segments. During Quick Recovery, a variable called pipe that represents the estimated number of segments outstanding in the network is maintained. The variable pipe is incremented by one when the sender either sends a new segment or retransmits an old segment. It is decreased by one when the sender receives an ACK that reports the new data has been received at the receiver and left the pipe. Whenever a SACK arrives, the retransmit timer is also reset. When the ACK for the segment arrives which is transmitted right before Quick Recovery is entered, which ACKs all data that is outstanding before Quick Recovery, the sender exits Quick Recovery and begins congestion avoidance normally.

The parameters used in Quick Recovery are:

- **HighAck** is the sequence number of the highest cumulative ACK received at a given point.

- **HighData** is the highest sequence number transmitted just before Quick Recovery begins.

- a Duplicate ACK is an ACK whose cumulative ACK is equal to **HighAck** and conveys new SACK information.
• a Partial ACK is an ACK that increases the \textit{HighAck} value, but does not ACK all of the data up to \textit{HighData}.

• a Recover ACK is the ACK for all data up to \textit{HighData}.

• \textit{ndupacks} is the number of duplicate packets which trigger the TCP to Fast Retransmit and Quick Recovery phases, which is chosen to be 3.

• \textit{maxburst} limits the number of packets that can be sent in response to a single incoming ACK. It is chosen to be 4 as proposed in [33].

• \textit{amountacked} is the number of data segments that ACKed by the receiver when an ACK arrives.

• \textit{adsn} is the number of NIL segments that the TCP sender is allowed to inject into the network.

• \textit{adps} is the number of allowed data segments to be sent.

• \textit{nps} is the number of data segments which have been really sent.

The details of the Quick Recovery algorithm is shown in Fig 6. When the sender has received \textit{ndupacks} SACKs, which indicates segment loss, the Quick Recovery then behaves conservatively after Fast Retransmit: the sender halves its congestion window \textit{cwnd}. It means that approximately half of the original window size of data will be transmitted during Quick Recovery phase. The variable \textit{pipe} is initialized as $2 \times \textit{cwnd} - \textit{ndupacks} + 1$ since the original congestion window is $2 \times \textit{cwnd}$ and the sender has received \textit{ndupacks} SACKs right before Quick Recovery, each of which reports a new data segment left the pipe. The sender also retransmits a lost segment using the Fast Retransmit algorithm. The number of NIL segments sent during the Quick Recovery, \textit{adsn}, is equal to \textit{cwnd}. This is because Quick Recovery will last roughly one RTT time. The ACKs for NIL segments sent during Quick Recovery will
Quick_Recovery()
  cwnd = cwnd/2;
  adsn = cwnd;
  adps = 0;
  END = 0;
  pipe = HighData - HighAck - msacked;
  while (END = 0)
    if (ACK.ARRIVAL)
      if (Duplicate ACK or Partial ACK)
        pipe = pipe - nsacked;
      if (NAK )
        pipe = pipe - 1
      end;
    if (NIL ACK)
      cwnd = cwnd + 1;
      adsn = adsn - 1;
    end;
    if (ACK which acknowledges all data up to and cover HighData)
      update HighAck;
      END = 1;
    end;
    update scoreboard;
    adps = cwnd - pipe;
    nps = min (maxburst, adps)
    if (nps > 0 )
      send nps missing or new packets;
      pipe = pipe + nps;
    end;
    else if (adsn > 0)
      send a NIL packet;
      adsn = adsn - 1;
    end;
  end;
end;

---

Figure 6: The Quick Recovery Algorithm.

be received in the Congestion Avoidance phase. If the segment loss is due to link error, cwnd will be increased by one for each NIL ACK arrival. Thus, the congestion window reaches its value before the data segment loss was detected rapidly.

During the Quick Recovery phase, upon a duplicate ACK arrives, which indicates one segment left the network, pipe is reduced by one. When a partial ACK arrives, pipe is reduced by amountacked and HighAck is also updated. Amountacked is
the exact number of segments acknowledged by a partial ACK [10]. It is used to accurately estimate the number of segments ACKed either by data ACKs or NIL ACKs. The arrival of a NIL ACK indicates that there are still unused resources in the network. Therefore, $cwnd$ is increased by one and $adsn$ is reduced by one. Finally, if a cumulative ACK acknowledges all data up to and cover $HighData$, the Quick Recovery phase is terminated.

When an ACK is received during the Quick Recovery, the scoreboard is updated to keep the latest information about missing segments. The scoreboard is cleared when Quick Recovery is exited. The number of segments which can be sent to the network, $adps$, is $cwnd – pipe$. To avoid injecting bursty segments into the network, $maxburst$ is set. When data segments are allowed to be sent, the missing segments are always transmitted before new data segments. If no data segments are allowed to be sent, a NIL segment is transmitted as long as $adsn > 0$.

2.2.4 The Delayed SACK

TCP-Peach++ use selective acknowledgment (SACK) options for assurance of reliable data segment transmission. TCP-Peach++ sink continuously sends SACK back to the source for each data packet it receives. If the data packet size is 1KB and the SACK size is 40 Bytes, then the ratio of the traffic in the forward and feedback channels is 25 : 1, i.e., no congestion will occur in the feedback channel only if the bandwidth asymmetry does not exceed 25 : 1. However, satellite IP networks have low bandwidth feedback links usually not faster than several hundred Kbps for small satellite terminals and a few Mbps for larger satellite terminals [47], i.e., the bandwidth asymmetry may exceed 25 : 1. Thus, sending one SACK for each data packet can cause congestions in the feedback channel.

To reduce the number of SACKs in the feedback channel, the delayed SACK scheme is adopted here. The TCP-Peach++ sink maintains a delayed-SACK factor,
and sends one SACK for every $d$ data packets received if the sequence numbers of the received packets are increased accumulatively. Otherwise, it sends the SACK to the satellite immediately. As a result, the amount of traffic in the feedback channel is controlled by adjusting the delayed-SACK factor $d$, if $d$ is large enough, no congestion will occur in the feedback channel.

### 2.2.5 The Hold State

The sender receives SACKs for reliability control purposes. If the sender does not receive any SACK the receiver for a certain period of time $T_f$, it infers this condition as rain fade and goes to the Hold State. In the Hold State, the sender first records the current congestion window $cwnd$ in the variable $rf_{cwnd}$. Then it freezes all retransmission timers and starts to send probing segments to the receiver periodically. Upon receiving a probing segment, the receiver sends a SACK immediately to the sender to report its current buffer status.

If the sender receives the SACKs for the probing segments, it infers that the rain fade is over and resumes sending data packets. If $rf_{cwnd}=1$, it goes to the Jump Start State, otherwise, let $cwnd=rf_{cwnd}$. In the latter case, the sender first transmits $rwnd-cwnd$ NIL segments to probe the available bandwidth and then enters the Quick Recovery State to recover the missing packets.

### 2.3 Performance Evaluation

To evaluate the performance of TCP-Peach++, we simulate a GEO satellite system with $N$ senders transmit data to $N$ receivers. The $N$ streams are multiplexed in the earth station with buffer size of $K$. As in [5], we assume $K=50$ segments, $rwnd=64$ segments, the link capacity $c=1300$ segments/s which is approximately 10 Mb/s for TCP segments of 1000 bytes. The round trip time $RTT=550$ms.

The performances for the throughput, overhead, rain fades, and bandwidth asymmetry are illustrated in Section 2.3.1, 2.3.2, 2.3.3, and Section 2.3.4, respectively.
2.3.1 Throughput

we assume $N = 20$ and the simulation time is $550 s$. The throughput performance comparison of TCP-Peach++ with TCP-Peach for different values of segment loss probability $P_{\text{Loss}}$ is shown in Fig 7, where the throughput performance of TCP-Peach++ is better than TCP-Peach. This is due to Jump Start as well as Quick Recovery. When the loss probability is low, the throughput improvement is mostly because of Jump Start since the segment losses are mostly due to network congestion instead of link errors. In this case, Nil segments in TCP-Peach++ carry unacknowledged segments, which make the connections to recover from lost segments more rapidly than dummy segments in TCP-Peach. When the loss probability is high, the Quick Recovery contributes to the throughput improvement. It avoids halving the $cwnd$ multiple times resulting from multiple losses of segments within one window of data as in Rapid Recovery. In Fig 7, when $P_{\text{Loss}} = 5 \times 10^{-2}$, the throughput improvement of TCP-Peach++ over TCP-Peach is 30.24%.

![Figure 7: The throughput performance comparison of TCP-Peach++ and TCP-Peach for different values of $P_{\text{Loss}}$.](image)

24
2.3.2 Overhead

As dummy segments in TCP-Peach, NIL segments in TCP-Peach++ do not carry any new information. Thus, they introduce overhead in the network [5]. Use the same assumptions in Section 2.3.1, the overhead comparison of TCP-Peach++ vs. TCP-Peach is depicted in Fig 8. For most $P_{Loss}$ values, the overhead ratio of TCP-Peach++ to TCP-Peach is less than one. This is because in the Quick Recovery phase of TCP-Peach++, one NIL segment is sent when no data is allowed to be transmitted while in the Rapid Recovery phase of TCP-Peach, two dummy segments are sent. For large $P_{Loss}$ values which cause multiple segment losses within one window of data, cwnd in Quick Recovery is reduced more slowly than in Rapid Recovery. Therefore, more NIL segments are allowed to be sent to probe network resources and recover errors. When $P_{Loss}=5 \times 10^{-2}$, the ratio is the maximum value, which indicates that the overhead of TCP-Peach++ is 18.03% higher than TCP-Peach. However, we already see that the throughput improvement in this case is 30.24%.

![Graph showing overhead ratio](image)

**Figure 8**: The overhead comparison of TCP-Peach++ and TCP-Peach for different values of $P_{Loss}$. 

25
Besides the throughput, we evaluate another important metric fairness of TCP-Peach++. We assume 10 connections $N = 10$ and $P_{\text{Loss}} = 0$. The simulation results we obtained are consistent with the fairness evaluations in [5]. As a result, TCP-Peach++ provides a fair share of the network resources both in homogeneous and heterogeneous scenarios.

### 2.3.3 Rain Fade

We assume $N = 10$ and assume there are 11 unicast TCP-NewReno connections from the gateway to the receivers. For those connections, we assume $rwnd = 64$ segments. Assume rain fades last for a short duration of not more than 60 secs similar to the assumption made in [55] and rain fade occurs at time $t=20$ secs. The timeout threshold $T_f$ for detecting rain fades is chosen to be the same as the retransmission timeout threshold. The sender may go to the Jump Start State before it can infer rain fade, thus, it always records the current congestion window $cwnd$ into the variable $r_f.cwnd$ before it goes to the Jump Start State. During the Hold State, the sender stops sending any data packets and freezes all retransmission timers to avoid unnecessary transmission. However, it sends probing packets periodically to detect when rain fade is over. To investigate the performance improvement by the Hold State, we also consider TCP-Peastree without the Hold State, where the sender keeps going into the Jump Start State when the transmission timer expires. The simulation time is 100 secs. The throughput performance for TCP-Peastree with and without the Hold State is shown in Figure 9.

The throughput with the Hold State is always higher than that without the Hold State and the throughput difference is approximately constant for different rain fade periods. The reasons that TCP-Peach++ with the Hold State achieves higher throughput are as follows:
Figure 9: The throughput vs. rain fade period.

- In the Hold State, the sender uses probing segments to obtain the exact information about rain fade. On the other hand, for TCP-Peach++ without the Hold State, the sender has to wait for the retransmission timer to expire and then goes to the Jump Start State to send packets to the receiver.

- The sender records its current congestion window when rain fade occurs and keeps this congestion window after rain fade is over. While for TCP-Peach++ without the Hold State, the sender always goes to the Jump Start State and the congestion window is set to 1.

2.3.4 Bandwidth Asymmetry

We assume \( N = 10 \) and assume there are 11 unicast TCP-NewReno connections from the gateway to the receivers. For those connections, we assume \( rwnd = 64 \) segments. Since the feedback link capacity is usually very low for satellite networks, the delayed SACK scheme discussed in Section 2.2.4 is adopted to address the bandwidth asymmetry problem, i.e., the receiver sends one SACK for a given number of received data
packets. This number is called the delay factor. We investigate 3 cases of the bandwidth asymmetry problems, i.e., the feedback link bandwidth is 1 Mbps, 100 Kbps and 64 Kbps, respectively. Also assume the link capacity from the sender to the receiver is 10 Mbps. The simulation time is 100 secs. The throughput performance with respect to different delay factors is shown in Figure 10.

![Graph showing throughput vs. delay factor.](image)

**Figure 10:** The throughput vs. delay factor.

When the feedback link bandwidth is 1 Mbps, the throughput performance is not degraded for this bandwidth asymmetry ratio. The reason is that the ACK size is usually about 40 Bytes, which is much smaller than the data packet size 1 KB, thus, the feedback link is not congested. However, the throughput performance decreases with increasing bandwidth asymmetry ratio. For example, the throughput drops to 73.81 from 103.31 when the feedback link bandwidth is 100 Kbps and the delayed SACK scheme is not used. The throughput increases with increasing delay factor and reaches the highest value when delay factor is 3, which is close to the throughput for 1 Mbps. Since TCP relies on the ACK clock to transmit packets, the delay factor can not be too large, thus, the throughput performance can be degraded for very high
bandwidth asymmetry ratios. For example, the throughput decreases approximately 20% when the feedback link bandwidth is 64 Kbps and the delay factor is 4.
CHAPTER III

RELIABLE DATA MULTICAST TRANSPORT
PROTOCOLS FOR SATELLITE IP NETWORKS

3.1 Problem and Solution

Satellite networks will play a crucial role in the global infrastructure of the Internet. They do not only provide global coverage, but also are capable of sustaining high bandwidth levels and supporting flexible and scalable network configurations. Currently, two thirds of the world still does not have a wired network infrastructure. Locally built networks or individual hosts can be connected to the rest of the world via satellites by simply installing satellite interfaces. Satellite networks can also be used as a backup for existing networks, e.g., in case of congestions or link failures, traffic can be rerouted through satellites.

Multicasting provides an efficient way of disseminating data from a sender to a group of receivers. Instead of sending a separate copy of the data to each individual receiver, the sender just transmits a single copy to all receivers. In this case, a multicast tree is set up in the network where the sender is the root and the receivers are the leaf nodes. Data generated by the sender flows through the multicast tree, traversing each tree edge exactly once [71].

A large variety of reliable multicast protocols [40, 49, 59, 71, 76, 91, 92] have been proposed for the terrestrial wirelined networks where packet losses occur mostly resulting from congestions. These protocols are usually classified into two groups: sender-initiated protocols, which use ACKs, and receiver-initiated protocols, which use NAKs. By this classification, RMTP [71] is a typical sender-initiated protocol,
in which receivers form local groups for *local error recovery*. SRM [40] is a typical receiver-initiated protocol, retransmission is performed in the entire group. TMTP [92] uses both ACKs and NAKs. MTCP [76] uses error bitmaps in ACKs, which are similar to TCP SACK options. However, MTCP introduces high overhead for a large congestion window.

Satellite networks have high *bit error rate* (BER), *long propagation delays* and *asymmetrical bandwidth* [4]. These characteristics distinguish satellite multicasting from multicasting in terrestrial wirelined networks. In addition to the *acknowledgment implosion* and *scalability problems* in terrestrial wirelined networks, satellite multicasting has the following additional problems:

- **Different Multicast Topology**

  In terrestrial multicast networks, the receivers may be located in different places, connected by intermediate routers, and may be several hops distant from the sender. Usually, a multicast tree is created with the help of intermediate routers. While in satellite multicast networks, all receivers receive packets directly from the satellite and there are no intermediate routers between the satellite and the receivers. On the other hand, the receivers send acknowledgments directly to the satellite. In other words, the receivers are only one hop away from the satellite. Therefore, there is no physical hierarchy between the satellite and the receivers. Consequently, the multicast schemes [40, 49, 59, 71, 76, 91, 92] that are based on multicast trees and are developed for terrestrial multicast networks cannot be applied in satellite multicasting. Furthermore, each receiver may experience different data losses for satellite multicasting and there is no guarantee that some receivers may always have better channel conditions. As a result, some mostly used terrestrial multicast schemes such as selecting some special servers or some intermediate routers as repairers [49, 59] may not be used here.
• **Congestion Control Problems**

TCP problems in satellite IP networks and related research issues are discussed in [4]. As pointed out in [4], the congestion control for satellite IP networks is difficult because of long propagation delays, high bit error rate (BERs) and asymmetrical bandwidth for unicast applications. For satellite multicasting, these problems become more complicated. Traditional TCP schemes usually perform poor in these situations [60]. Furthermore, satellite link channels can experience persistent link fades because of varying weather patterns [55]. The congestion control algorithms should be able to address the persistent fades to reduce the performance degradation and unnecessary transmissions when the links are unavailable.

• **Low Bandwidth Feedback Link**

A terrestrial feedback link to the sender has been proposed in [53, 56]. However, this is difficult to implement in some cases, especially for mobile receivers, which need to use a low-bandwidth uplink channel as the feedback link. The feedback link is usually not faster than several hundred Kbps for small satellite terminals and a few Mbps for larger satellite terminals [47].

In the satellite domain, relatively few research has been performed on reliable multicast protocols [53, 55, 56]. Some key issues to design satellite multicasting are reviewed in [56] and it is shown in [53] that the introduction of a feedback channel is the key to realize bandwidth-efficient, robust, and fully reliable multicast communication via satellites.

In this chapter, we consider the satellite multicast scenario as shown in Figure 11. Here, the sender transmits packets to the satellite through a gateway, then the satellite multicasts packets directly to all receivers. Some receivers may have terrestrial connection among them. Note that not all receivers necessarily need to be connected
by terrestrial networks. The receivers use the satellite uplink channel as the feedback link to the sender. Usually, the receiver is a User Earth Station with two way satellite connectivity and terrestrial multicast connectivity. The User Earth Station can work as a proxy for all the terrestrial users connected to it and will be treated as only one receiver in this chapter.

![Diagram of Satellite Multicast Scenario](image)

**Figure 11:** The Satellite Multicast Scenario.

Considering the challenges in satellite multicasting, we propose a new reliable multicast transport protocol, TCP-Peachtree, for satellite IP networks. Our contributions are

- *Hierarchical Logical Groups on the Reverse Path:* Because of the special topology of satellite multicasting, all receivers are only one hop away from the satellite and hence there is no physical hierarchy. It is not possible to generate a multicast tree for satellite multicasting as in the case of wirelined networks. The satellite transmits packets directly to all receivers. If all receivers send acknowledgments directly to the satellite, the acknowledgment implosion problem becomes very challenging. This is based on the fact that the huge number of receivers are only one hop away from the satellite and no intermediate routers can be used to aggregate the ACKs. Furthermore, the low bandwidth for the
feedback link makes the problem worse. To address the acknowledgment implosion and the low bandwidth feedback link problems, we propose the so-called logical hierarchical groups on the reverse path. Based on the logical hierarchical groups, we propose ACK aggregation, local error recovery, and local relay schemes to suppress the acknowledgments from the receivers and to reduce retransmissions from the sender.

- **New Window-Based Congestion Control Scheme:** As mentioned before, in GEO satellite multicasting, all receivers have approximately the same RTT values. Thus, we propose a new window-based congestion control scheme for reliable multicasting in the satellite networks. The new window-based congestion control scheme uses the so-called NIL segments. Unlike the dummy segments proposed in [5], NIL segments are not only used to probe the available network resources, but also to recover packet losses in the receivers. The delayed SACK scheme is adopted to address the bandwidth asymmetry problems. Two new algorithms: Jump Start and Quick Recovery are also proposed to address long propagation delays and high bit error rates (BERs) in the satellite networks. Furthermore, a Hold State is introduced to address the persistent fades.

The remainder of the chapter is organized as follows. The multicast procedures in TCP-Peachtree are presented in Section 3.2. Then the congestion control problem is discussed in Section 3.3. Simulation results are given in Section 3.4.

### 3.2 Logical Hierarchical Groups

#### 3.2.1 The B+ Tree Hierarchy

For GEO satellite multicasting, the satellite sends packets directly to all receivers and there is no physical hierarchy. To suppress the number of ACKs from the receivers and to reduce retransmissions from the sender, we propose logical hierarchical groups on the reverse path. As some receivers may have terrestrial connections among them,
they may form one or more logical hierarchical groups so that each group only needs
to send one ACK for each received packet. On the other hand, local recovery and local
relay schemes can be used for local error recovery to reduce retransmissions from the
sender. The satellite still sends packets directly to all receivers, the logical hierarchical
group is only formed on the reverse path such that the packet transmissions from the
satellite to the receivers are not affected. Note that, “reorganizing” receivers in the
logical hierarchy does not lead to high overhead and is not error prone.

The logical hierarchy is constructed in a way similar to the B+ tree [32]. B+ tree
is a data structure typically used for searching data with data pointers stored only
at the leaf nodes of the tree. The main benefit of B+ tree is that it is balanced. A
B+ tree of order $M$ has the following characteristics:

- Each internal node is of the form
  $$< P_1, K_1, P_2, K_2, \ldots, P_{q-1}, K_{q-1}, P_q >$$
  where $q \leq M$ and each $P_i$ is a tree pointer and $K_i$ is the key value for searching.

- Within each internal node, $K_1 < K_2 < \ldots < K_{q-1}$.

- For all search field values $X$ in the subtree pointed at by $P_i$, we have $K_{i-1} <
  X \leq K_i$ for $1 < i < q$; $X \leq K_i$ for $i = 1$; and $K_{i-1} < X$ for $i = q$.

- Each internal node has at most $M$ tree pointers

- Each internal node, except the root, has at least $\lceil M/2 \rceil$ tree pointers. The root
  node has at least two tree pointers if it is an internal node.

- An internal node with $q$ pointers, $q \leq M$, has $q - 1$ search field values.

A typical B+ tree is illustrated in Figure 12, where 6 numbers: 1, 3, 5, 7, 8 and 12
are stored in a B+ tree of order 2. All numbers are stored in the leaf nodes. The key
values 3, 5, 8 are used for key searching. The leaf nodes of the B+ tree are usually linked together to provide ordered access to the records.

Here, we use the B+ tree to update the hierarchical structure automatically instead of searching for a key. So we modify the traditional B+ tree [32] as follows:

- The modified B+ tree forms a multicast group.
- Each leaf node corresponds to a multicast subgroup which consists of physical receivers.
- Each internal node corresponds to a logical subgroup, which is created by choosing the Designated Receiver (DR) from each of its child subgroups. The Designated Receiver is a node in a subgroup that is responsible for ACK aggregation, local error recovery, and exchanging information with members in its parent subgroup on behalf of all members in its subgroup.
- In a subgroup, one member is selected as the DR.
- Each node (i.e., a multicast group) has at most $M$ members.
- Each node (i.e., a multicast group), except the root, has at least $\lceil M/2 \rceil$ members. The root node has at least two members, if it is an internal node.

The join and split procedures for the standard B+ tree are also modified to update the modified B+ tree dynamically when a member joins or leaves the multicast
group. Using the modified B+ tree we create a dynamic hierarchical structure for each multicast group.

3.2.2 Selecting the Designated Receiver

Any member can start a multicast group by sending a signaling message to the satellite via the uplink channel. This member becomes the DR initially and it also gets a group ID from the sender, which is used for later transmissions. When members join or leave a multicast group, subgroups may be split or joined accordingly and consequently new DRs can be selected. A DR is selected according to the packet loss statistics. The DR Selection procedures are as follows:

- Each receiver calculates the packet loss statistics and sends it to the DR periodically. Let $L_i^j$ be the number of losses from the sender observed by receiver $i$ in epoch $j$ and let $L_i$ be the packet loss statistics from the sender, then $L_i$ is updated as follows:

\[
L_i = \eta L_{i-1} + L_i^j
\]

where $j \geq 0$ and $0 < \eta \leq 1$. If $\eta = 1$, $L_i$ is the sum of missing packets from the sender. Usually, $\eta < 1$.

- The DR then ranks all members in an increasing order according to their packet loss statistics $L_i$ from the sender.

- The probability $P_i$ for the receiver $i$ to be selected as a DR is calculated from

\[
P_i = \frac{1/r_i}{\sum_{i=1}^{M} 1/r_i}
\]

where $r_i$ is the rank of the multicast receiver $i$ and $M$ is the number of members in the multicast subgroup.
• Choose a random value \( r \) \((0 < r < 1)\).

• The random selection algorithm is as shown in Figure 13 to select the DR (assuming \( P \) and \( K \) are variables):

\[
\text{Random Selection()}
\]
\[
P = 0;
K = 1;
\text{for } i = 1 \text{ to } M
\]
\[
\text{if } (r \geq P)
\]
\[
P = P + P_i;
K = K + 1;
\]
\[
\text{else}
\]
\[
\text{break;}
\]
\[
\text{end;}
\]
\[
\text{end;}
\]

**Figure 13:** The Random Selection Algorithm.

As a result, the receiver \( K \) is chosen as the DR for this multicast group.

The advantage of the random selection procedure is to choose the DR according to its packet loss statistics. The receiver with lower packet loss rate has a higher chance to become a DR. The random selection algorithm can also address the dynamic nature of the network. A receiver with very low packet loss rate does not necessarily mean that it maintains that state forever.

The DR in each subgroup has the following functions:

• Buffer packets received from the sender and use them for local error recovery.

• Aggregate ACKs for a packet in this subgroup into one ACK and pass it to the DR in its parent logical subgroup.

• Perform local retransmission.

• Forward retransmission requests to its parent logical subgroup.
• Work as a relay for receivers which have very bad channel conditions.

• Manage member joining or leaving.

• Calculate and select a new DR for a new subgroup which is split from the current subgroup.

3.2.3 Members Joining a Multicast Group

A new member sends a particular \textit{LOOK\_FOR\_DR} packet with time-to-live (TTL) parameter equal to 1. Upon receiving this packet, the DRs send \textit{IM\_DR} type of ACK to the new member, which then chooses the DR with the smallest RTT value. If no reply comes back from any DR, the TTL value is increased by one and this procedure is repeated until a DR is found with the RTT value satisfying: \(RTT/2 < TRTT\), where \(TRTT\) is a threshold and is discussed and defined in the performance evaluation, Section 3.4. Then, the new member joins that multicast group. If no DR can be found with a very large TTL value, it means that either no terrestrial connection exists from the new member to other DRs, or all DRs are too far away. If the DRs are too far away, it takes a long time to do retransmissions and ACK aggregations when the new member joins that group and hence affects the response time of that group to the sender and consequently the overall performance is degraded. In this case, the new member must initiate a new multicast group by sending a signaling message to the sender and becomes the DR of this multicast group initially. When a new member joins a multicast group, it must first join a subgroup at the leaf in the logical hierarchical structure. If the number of the members in a multicast subgroup exceeds a given threshold \(M\), then we split the group into two subgroups, with one subgroup keeping the current DR while the other subgroup selecting a new DR. As a result, a new DR in the upper logical multicast group is selected. We repeat this procedure in the upper logical multicast groups if necessary until the highest logical multicast group is reached.
The member joining procedure is illustrated by the following example. Assume 6 receivers, 1, 2, 3, 4, 5, 6, which are connected together, join a multicast group one by one. Assume also that the maximum member value in a subgroup is equal to 2. The procedure is illustrated in Figure 14. First, receiver 1 initiates the multicast group and is selected as the DR, then receiver 2 joins the multicast group as shown in Figure 14(A). When receiver 3 joins the multicast group, the member number exceeds 2, so the multicast group is split into two subgroups: 1 and 2 forming a subgroup with 1 as the DR, and 3 is in another subgroup and as the DR in that subgroup. Since two subgroups are created, the DRs 1 and 3 form a logical subgroup. Suppose receiver 4 chooses 3 as its DR and joins that subgroup. The resulting structure is shown in Figure 14(B). Receiver 5 and 6 join the multicast group in a similar way and the final structure is shown in Figure 14(C).

![Diagram](image)

**Figure 14:** The procedure for members to join the multicast group.

### 3.2.4 Members Leaving a Multicast Group

When a member leaves a multicast group and the number of group members becomes less than \(\lceil M/2 \rceil\), then the group members are redistributed to a neighboring group, or two groups are merged into one. In the latter case, a DR is identified for the new subgroup and the according DR is removed from the upper logical multicast group. This procedure is repeated if necessary until the highest logical multicast group is reached. If the last member in the multicast group is removed, it sends a release message to inform the sender that this multicast group is removed.

The member leaving procedure is illustrated by the following example. Assume
receiver 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 and 13 form a logical hierarchical multicast tree as shown in Figure 15(A) with the maximum member in a subgroup equal to 4. Thus, the minimum member number, except the root, in a subgroup is 2. Suppose receiver 13 leaves the multicast group 12, 13. The member number in this group becomes smaller than 2. Thus, 12 joins the subgroup \{9, 10, 11\} and the logical member 13 in subgroup \{9, 13\} is removed. Finally, 9 joins the subgroup \{1, 4, 7\}. As a result, the logical hierarchical level is reduced from 3 to 2 as shown in Figure 15(B).

\[\text{Figure 15: The procedure for members to leave the multicast group.}\]

### 3.2.5 Logical Tree Setup Overhead

To setup the logical tree, new members need to send \textit{LOOK\_FOR\_DR} packets to the DRs and the DRs send \textit{IM\_DR} type of ACKs back to the new members. We call the \textit{LOOK\_FOR\_DR} and \textit{IM\_DR} packets \textit{G\_JOIN} messages. The number of \textit{G\_JOIN} messages used to form logical subgroups is a measurement of logical tree setup overhead [26]. Assume a logical tree has \(l\) levels of hierarchies and the average number of \textit{G\_JOIN} messages for a group member to join a subgroup is \(J\). To calculate the total number of \textit{G\_JOIN} messages for a logical tree, we consider two extreme cases:

- \textbf{The Best Case:} The number of members in each subgroup is \(M\), hence the maximum number of receivers in this group is \(M^l\). The total number of members \(S\) in the logical tree, including physical receivers and DRs, is
\[ S = \sum_{i=1}^{l} M^i = \frac{M(M^l - 1)}{M - 1} \] (3)

Obviously, the total number of \(G\text{.JOIN}\) messages is \(S \times J\) and the number of \(G\text{.JOIN}\) messages for the physical receivers to join the multicast group is \(M^l \times J\), hence the number of \(G\text{.JOIN}\) messages used to form logical subgroups is \((S - M^l)J\). As a result, the logical tree setup overhead \(O_s\) is

\[ O_s = \frac{(S - M^l)J}{M^l J} = \frac{M(M^l - 1)}{(M - 1)M^l - 1} \] (4)

Since the total number of receivers in a multicast group, i.e., \(M^l\), is very large, \(M^l - 1 \approx M^l\), we get

\[ O_s = \frac{1}{M - 1} \] (5)

- The Worst Case: The root has 2 members and the other subgroups have \(\left\lceil \frac{M}{2} \right\rceil\) members. Thus, the minimum number of receivers is 1 for \(l = 1\) and \(2 \left\lceil \frac{M}{2} \right\rceil^{l-1}\) for \(l > 1\). The total number of members \(S\) in the logical tree, including physical receivers and DRs, is

\[ S = \sum_{i=0}^{l-1} 2 \left\lceil \frac{M}{2} \right\rceil^i = \frac{2 \left( \left\lceil \frac{M}{2} \right\rceil^l - 1 \right)}{\left\lceil \frac{M}{2} \right\rceil - 1} \] (6)

Similarly, we get the logical tree setup overhead

\[ O_s = \frac{1}{\left\lceil \frac{M}{2} \right\rceil - 1} \] (7)

Consequently, the logical tree setup overhead is

\[ \frac{1}{M - 1} \leq O_s \leq \frac{1}{\left\lceil \frac{M}{2} \right\rceil - 1} \] (8)
Usually, $M$ is very large and hence the logical tree setup overhead is very small. For example, if $M=20$, the logical tree setup overhead $O_s$ is in the range of 5% to 11%. If $M$ is larger, $O_s$ becomes much smaller.

### 3.2.6 ACK Aggregation

Received packets are reported in ACKs. The DR in a logical multicast group receives ACKs from members in its group. After it receives all ACKs for a packet from all members in its group, the DR sends an ACK to the upper logical subgroup. The DR in the upper logical group also receives ACKs from its members and sends ACKs to its parent logical group. The DR in the highest logical group then sends an ACK to the satellite via the uplink channel. Thus, for each packet only one ACK is sent to the sender and hence the feedback implosion problem is solved. Upon receiving ACKs, the sender also aggregates the ACKs from different multicast groups.

To further reduce the number of ACKs from this logical hierarchical group, the delayed SACK scheme is adopted here. The DR in the highest level sends one SACK for a given number of data packets received if the sequence numbers of the received packets are increased accumulatively. Otherwise, the DR sends the SACK to the satellite immediately.

### 3.2.7 Local Error Recovery

For satellite multicasting, different receivers may have uncorrelated packet losses. Although FEC is very efficient to recover uncorrelated packet losses [78], it also introduces high overhead and requires additional software or hardware processing capability at the receiver side. Here local error recovery is used to address uncorrelated packet losses. Local error recovery may also suffer from 'single point of failure' problem when a DR fails, this problem is solved by reassigning another DR to take it over in TCP-Peachtree.

*Local error recovery* solution is also presented in [71, 76, 92], however, they are
based on the terrestrial multicast tree. In TCP-Peachtree, local error recovery is based on the B+ tree hierarchical structure introduced in Section 3.2.1.

Each DR in a subgroup maintains a buffer to hold a number of packets for local retransmissions. If a receiver is DR in several different logical hierarchical groups, it shares the same buffer for different hierarchical logical levels. The advantage is that this receiver can obtain lost packets from different logical levels to increase the chance of local recovery.

The missing packets are reported in NAKs. When a member does not receive a packet correctly, it sends a NAK to the DR in its subgroup. If the lost packet is in the DR's buffer, the DR retransmits the lost packet to the receiver immediately. If not, the DR first multicasts a NAK to all members in its subgroup. Any member having the correct packet in its buffer can unicast that packet to the DR. On the other hand, the DR sets a timer for the lost packet. If the lost packet is received from other members in this subgroup, the DR unicasts the lost packet to the corresponding receiver if only one NAK is received for the lost packet, or multicasts the lost packet to its entire subgroup if multiple NAKs for this lost packet are received. If a timeout occurs, the DR sends a NAK to the DR in its parent logical group and will try to get the lost packet from its upper logical subgroup. This procedure is repeated until the highest logical subgroup is reached. If no success, one NAK is sent to the satellite via the uplink channel. The sender then multicasts the missed packets to all receivers.

3.2.8 Local Relay

Because of very bad channel conditions, some receivers may have relatively much higher packet loss rates than some other receivers. As a result, the overall performance is degraded since the sender needs to make reliable transmissions to those receivers. To improve the overall performance, a local relay scheme is designed, which is used if the packet loss rate \( k_i \) from the sender of member \( i \) is higher than a given threshold
\( \theta (0 < \theta < 1) \), i.e.,

\[ k_i \geq \theta \quad (9) \]

In this scheme, the DR in the corresponding subgroup forwards every packet it receives from the satellite to member \( i \) to reduce retransmission requests from member \( i \). \( k_i \) is updated and reported to the DR periodically. If \( k_i \) is less than the given threshold \( \theta \), the DR stops forwarding packets to member \( i \).

### 3.2.9 DR Handoff

When a DR leaves a multicast group, or when a new DR is selected during group join or merge, handoff is performed between two DRs. The new DR needs to create or update the data structure to store its group member information in each logical hierarchical level in which it is a DR. The new DR only needs to get the information of aggregated ACKs from the current DR or its DR in its parent logical subgroup. The current DR also needs to inform group members to change their DR. After the new DR is ready to work, it sends a message to the current DR and also all members in each logical subgroup in which it is a DR. Upon receiving this message, the subgroup members send ACKs to the new DR. The current DR forwards all ACKs it receives to the new DR. ACKs may get lost during DRs handoff. To prevent those losses, the new DR multicasts a message to all members in its group to resend ACKs to it. Upon receiving such message, all members send the ACKs to the DR immediately.

### 3.2.10 DR Failure

If a DR fails, a new DR is selected based on the procedure described in Section 3.2.1 and the B+ tree type hierarchical structure is updated accordingly. There are two ways to detect a DR failure. In the first case, a DR detects the failure of a member, which is a DR in its child subgroup, by keeping track of the ACKs from that member. If the DR does not get any message from a member for a certain time period, it sends
a poll message to this member. Upon receiving this message, the member should send an ACK for this message immediately. If such an ACK is not received during a certain time period, the DR assumes this member as non-existing and it acts as a DR for the malfunctioning subgroup temporarily until a new DR is selected from that subgroup.

For the DR in the highest logical subgroup, the above method does not work. Thus, the DR in the highest logical subgroup multicasts a poll message to its member periodically. If a member does not receive such a message within a certain time period, it sends the poll message to the DR. If it does not get an ACK for this poll message from the DR within a time period, it assumes that the DR has failed. Thus, it acts as a DR temporarily until a new DR is selected.

We could also use another method to cope with the DR failure. A special receiver is selected by the DR as a backup in each subgroup. The DR sends updated information to the backup receiver periodically. This receiver sends a poll message to the DR periodically to probe the up-to-date status of the DR. If the DR fails, the backup receiver takes over the control and works as a new DR.

3.3 Congestion Control

TCP-Peachtree congestion control contains two new algorithms: Jump Start and Quick Recovery, which are discussed in Chapter 2, as well as the two traditional TCP algorithms, Congestion Avoidance and Fast Retransmit. The TCP SACK options are also adopted. Moreover, a Hold State is introduced to address persistent fades. The flow chart of TCP-Peachtree congestion control algorithm is the same as that in Figure 3.

3.3.1 Selective Acknowledgment Scheme

Traditional TCP Schemes like TCP-Reno have problems when multiple packets are lost in a window, the TCP SACK option is proposed to solve these problems for
different applications [9, 12, 33, 82]. Hoe [48] proposed changes to the Fast Retransmit algorithm so that it can quickly recover from multiple packet losses without waiting unnecessarily for the timeout. In TCP-Peachtree, TCP SACK options are adopted to recover multiple packet losses in one round-trip time.

There are two types of selective ACKs, i.e., positive ACKs (ACKs) and negative ACKs (NAKs). Upon receiving a packet, an ACK is sent immediately to the DR. If the DR received all ACKs for this packet from all its members, an ACK is sent to the DR in its parent logical group until the sender is reached. While one or more missing packets are detected by the receiver, a NAK is sent to the DR for retransmission. If the DR does not have the missing packets in its buffer, it sends a NAK to its parent logical group until the NAK reaches the sender.

3.3.2 The ACK Filter

An ACK Fusion scheme was proposed in [43] by using intermediate routers. But only positive ACKs are considered. In TCP-Peachtree, there are three different kinds of ACKs which should be considered in different ways. Furthermore, because of the special topology of satellite multicasting, this method cannot be simply adopted. As a result, a different ACK fusion scheme is proposed. ACK Fusion in TCP-Peachtree is performed by each DRs and the sender and three different types of ACKs are handled using different ways.

The ACK filter is designed to aggregate ACKs from multiple return paths into one ACK. Whenever an ACK is received by the sender, it first goes through the ACK filter to eliminate duplicate ACKs from different return paths. The sender keeps the multicast group IDs in memory and used them for ACK filtering. There are three types of ACKs, i.e., NAK, ACK and NIL ACK, which are treated differently. For a NAK, the filter works as shown in Figure 16.

The ACK filter uses a matrix $M$ to keep track of the packets ACKed by different

47
\begin{verbatim}
NAK_Fusion()
    if a NAK is received
        if the group id is different from the latest NAK processed
            if the NAK contains different packet loss information from
            the last sent NAK
                Construct a new NAK with different missing packets;
                Process the NAK;
                Update the last sent NAK by adding the new missing packets
                into it
            end;
        else
            This is a new NAK, process the NAK;
            Replace the last processed NAK with the new NAK;
        end;
    end;
end;
\end{verbatim}

\textbf{Figure 16}: The NAK Fusion Algorithm.

For an ACK of a NIL segment, the ACK filter counts the number of ACKs for
these NIL segments. If the percentage of reception for these NIL packets exceeds
a given threshold $T_{nil}$, where $0 < T_{nil} \leq 1$, the ACK filter generates a NIL ACK for
processing.

\subsection{3.3.3 NIL Segments}

NIL segments introduced in Chapter 2 Section 2.2.1 are used to probe the availability
of network resources as well as error recovery. NIL segments are low priority segments.
If a router on the connection path is congested, then the NIL segments are discarded
first. Thus, the transmission of NIL segments does not affect data segments. For each
NIL segment, a NIL ACK carried by a low priority IP packet is transmitted back to
the sender. The arrival of NIL ACKs at the senders are indications that there are
unused resources in the network. Hence, the senders increase their transmission rate
**ACK Fusion()**

Upon receiving an ACK, create a new ACK;
Check the SACK option field of the ACK packet;
for each ACKed packet, check if the corresponding element
$m_{ij}$ is in Matrix $M$
if this packet is not presented in the matrix $M$ and its
sequence number is larger than the highest ACKed sequence number
which the sender has received all packets up to
Add a new column to the matrix $M$ and set the corresponding
element $m_{ij} = 1$;
else
if this bit is already set to 1
 Do nothing;
else
 $m_{ij} = 1$;
end;
if $m_{ij} = 1$ for all $j$
 Set the corresponding bit to 1 in the new ACK;
 Remove this column from the matrix $M$;
end;
end;
end;
if the new ACK contains some acknowledged packet
 Process the new ACK;
end;
end;

---

**Figure 17**: The ACK Fusion Algorithm.

accordingly to make full utilization of network resources. NIL segments are only transmitted when the sender is in the Jump Start phase or when packet losses are first found during the Quick Recovery phase.

Upon receiving NIL segments at receivers, they can use them for error recovery and send a NIL ACK to the DR. The DR counts the number of ACKs for a NIL segment, if the percentage of reception for this NIL segment exceeds a given threshold $T_{\text{nil}}$, where $0 < T_{\text{nil}} \leq 1$, then the DR sends a NIL ACK to the DR in the parent logical group or to the sender.
3.3.4 The Jump Start Algorithm

The Slow Start algorithm is disadvantageous in networks with long propagation delays. Several schemes have been proposed to improve the Slow Start algorithm, TCP Fast Start [68] transmits low priority packets during the Fast Start phase. Since these packets carry new information, they must be recovered if they are lost. In TCP-Pech [5], dummy segments, a type of low priority segments, are used to probe the availability of network resources. In TCP-Peachtree, the Jump Start algorithm described in Chapter 2 Section 2.2.2 is adopted to replace the Slow Start algorithm.

The basic idea of the Jump Start algorithm is that in the beginning of a connection, the TCP sender sets the congestion window, $cwnd$, to 1 and after the first data segment, it transmits $(rwnd - 1)$ NIL segments every $\tau = RTT/rwnd$. For each received ACK of a data packet or a NIL segment, the congestion window size is increased by one. As a result, after one round trip time, the congestion window size $cwnd$ increases very quickly. Note that the TCP sender can estimate RTT during the connection setup phase. Here $RTT$ is the largest value the sender gets from the receivers.

3.3.5 The Quick Recovery Algorithm

The Quick Recovery discussed in Chapter 2 Section 2.2.3 substitutes the classical Fast Recovery algorithm [50] with the objective of recovering multiple losses in a window and solving the throughput degradation problem because of link errors. As shown in Figure 3, when one or more segment losses are detected by either duplicate SACKs or a NAK, we use the Fast Retransmit Algorithm presented in [50] to transmit a missing packet. After completing the Fast Retransmit algorithm we apply the Quick Recovery algorithm.

Similar to the TCP SACK proposed in [33], the Quick Recovery maintains a data structure called scoreboard to update information about missing packets and a
variable called pipe that represents the estimated number of packets outstanding in the path. The variable pipe is incremented by one when the sender either sends a new packet or retransmits an old packet. It is decremented by one when the sender receives a duplicate ACK packet with a SACK option reporting the new data is received at the receiver. The sender always sends the missing packets first. If no missing packet exists, the sender sends a new packet. Whenever a SACK is accepted, the retransmit timer is also reset. The sender exits the Quick Recovery when a recovery ACK is received ACKing all data that were outstanding when the Quick Recovery was entered.

In TCP-Peachtree, when a retransmission timeout occurs, the sender checks whether it has received any ACK for that packet from any multicast group. If it did receive some ACKs for that packet from some multicast groups, the timeout is due to link error. This is based on the fact that no multicast group can receive a packet if it is lost in the gateway resulting from congestion. In this case, the sender retransmits that packet and stays in the Quick Recovery phase. On the other hand, if there is no ACK for that packet from any multicast group, the timeout is due to congestion, the sender terminates the Quick Recovery and executes the Jump Start.

3.3.6 The Hold State

To address the persistent fades resulting from varying weather patterns, such as rain fades, the Hold State described in Chapter 2 Section 2.2.5 is adopted in TCP-Peachtree. The sender receivers ACKs for reliability control purposes. If the sender does not receive any ACK from a multicast group or all multicast groups for a certain period of time $T_f$, it infers this condition as rain fade and goes to the Hold State. In the Hold State, the sender first records the current congestion window $cwnd$ in the variable $rf_{cwnd}$. Then it freezes all retransmission timers and starts to send probing segments to the receiver periodically. Upon receiving a probing segment, the DR in
the highest hierarchical level sends an ACK immediately to the sender to report its current buffer status.

If the sender receives ACKs for the probing segments, it infers that the rain fade is over and resumes sending data packets. If $r f \_ cwnd=1$, it goes to the Jump Start State, otherwise, $cwnd=r f \_ cwnd$. The sender first transmits $rwnd-cwnd$ NIL segments to probe the available bandwidth and then enters the Quick Recovery State to recover the missing packets.

### 3.4 Performance Evaluation

![Figure 18: The Logical Hierarchical Structure for TCP-Peachtree.](image)

We developed our own simulation model. The physical structure for satellite multicasting is shown in Figure 11 and the logical hierarchical structure is shown in Figure 18. The sender sends packets to the satellite through a gateway, then the satellite multicasts packets to all receivers. Some receivers may have terrestrial connection among them, but not all of them need to be connected by terrestrial networks. The receivers use the satellite uplink channel as the feedback link to the sender. Since the GEO satellite network is considered, the RTT values ($550ms$) from the sender to all receivers are approximately the same.
The measured packet loss probability resulting from link errors in the channel varies from $10^{-6}$ to $10^{-2}$. Furthermore, we assume the gateway buffer length to be 50 segments and $rwnd = 64$ segments. We also assume that the link capacity is $c = 1300$ segments/sec which is approximately 10 Mb/sec for TCP segments of 1000 bytes. Moreover, there are 11 unicast TCP-NewReno connections from the gateway to the receivers. For those connections, we assume $rwnd = 64$ segments. Also assume the application is reliable data distribution and simulation time is 100 secs.

Link error properties are first investigated in Section 3.4.1 and is followed by logical hierarchical multicast groups in Section 3.4.2. To illustrate the basic properties of TCP-Peachtree, i.e., throughput performance, overhead, and fairness with respect to packet loss rates because of channel errors, we first assume all receivers have the same packet loss rate, i.e., $p_i = p$. The unbalanced situation where different receivers have different packet loss rates is discussed in Section 3.4.6. The performances for rain fades and bandwidth asymmetry are illustrated in Section 3.4.7 and Section 3.4.8, respectively. Finally, the scalability is discussed in Section 3.4.9.

### 3.4.1 Link Error Properties

In our simulation, we assume data, NIL and ACK segments experience losses because of link errors. In Figure 18, there are two types of link errors. One is the shared link error, i.e., error occurs in the channel between satellite and gateway. The other is the multicast link error, i.e., link error occurs between satellite and the receivers.

- **The Shared Link Errors**

  The shared link errors include data/NIL segment losses from the gateway to the satellite and ACK segment losses from the satellite to the gateway. If a data segment is lost, every receiver cannot receive this packet, so it is retransmitted and recovered by either a data segment or a NIL segment. There is no way to recover this type of errors locally on the receiver side. Once a SACK
segment is lost, it can be recovered by subsequent SACK segments because of
the redundancy in SACK options.

- The Multicast Link Errors

The Multicast Link Errors include segment losses from the satellite to the re-
ceivers and ACKs from the DRs to the satellite. In satellite multicasting, there
are multiple paths to the receivers via the downlink channel. Multiple paths can
cause the loss path multiplicity problem [16]. Because of the loss path multi-
plicity problem, the aggregated segment loss rate from satellite to the receivers
is rather large even if the packet loss rate \( p_i \) for each individual return path \( i \) is
small. Fortunately, most of the lost segments from the satellite can be locally
recovered within the logical hierarchical groups. The SACK segments sent to
the satellite may also experience packet loss and they can also be recovered by
subsequent SACKs. If a NAK is lost, the sender eventually receives duplicate
ACKs.

Here we assume all \( p_i \)'s are equal, \( p_i = p \) and also assume the packet loss rate
from the gateway to the satellite and from the satellite to the the gateway is also \( p \).

The aggregated packet loss rate from the satellite to the receivers is obtained as
shown in Figure 19 with \( N=100, 200, 500 \) and 1000, respectively, where \( N \) is the total
number of receivers in the multicast session.

Now, we observe that:

- When \( p \) increases, the aggregated packet loss rate also increases.

- If \( N \) increases, the aggregated packet loss rate also increases. When \( N \) is large
  enough, it becomes approximately 1.

Figure 19 clearly demonstrates the loss path multiplicity problem, i.e., even if the
packet loss rate \( p \) from satellite to a single receiver is rather small, with increasing
Figure 19: The Aggregated Packet Loss Rate from the satellite to the receivers vs. $p$ for $TRTT = 30ms$ and $M = 10$.

With a large number of receivers, the packet loss rate is approaching 1. Although the aggregated packet loss rate is rather high, the multicast link errors have the property that different receivers may have different channel conditions so that some receivers receive the packet while the others do not. In other words, the lost packet may be received by some receivers. This makes it possible to use the local recovery scheme to solve the loss path multiplicity problem. In TCP Peachtree, a logical hierarchical multicast group is designed to recover this type of errors locally.

Here we emphasize the loss path multiplicity problem to show how severe it can be for satellite IP networks because of a huge number of receivers and high bit error rate (BER) in satellite networks.

3.4.2 Logical Hierarchical Multicast Groups

Logical hierarchical groups are formed using the modified B+ tree presented in Section 3.2. The group formation is affected by the following factors:
• **Topology of Receivers**: Receivers form clusters and there is no terrestrial connection among different clusters. Thus, each cluster forms one or more separate multicast groups.

• **RTT Threshold**: TRTT is the threshold used when a receiver selects a DR to join. Assume the RTT value from a new receiver to a DR is RTT, if $RTT/2 < TRTT$, the new receiver can join that subgroup. If the RTT value from any DR to the new receiver is larger than $2 \times TRTT$, the new receiver needs to initiate a new multicast group.

• **Maximum Number of Members Allowed in a Subgroup**: M is the maximum member number allowed in a subgroup. If the member number in a subgroup is larger than M, this subgroup is split into two subgroups. After a receiver leaves a multicast group, if the member number in this subgroup is smaller than $M/2$, this subgroup is merged with a neighboring subgroup to form a new subgroup.

Assume $N = 200$, $M = 10$, and all receivers are in one cluster. The resulting logical hierarchical multicast groups are shown in Table 1.

**Table 1**: The Resulting Logical Hierarchical Groups for $N = 200$ and $M = 10$.

<table>
<thead>
<tr>
<th>TRTT (ms)</th>
<th>15</th>
<th>20</th>
<th>25</th>
<th>30</th>
<th>35</th>
<th>40</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group Number</td>
<td>21</td>
<td>10</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Highest Level</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

For a given number of receivers, with $TRTT$ increasing, the number of groups decreases, but the highest hierarchical level in a group may increase. If TRTT is too small, there are too many multicast groups.

Although the RTT value from one receiver to the sender is already known. Because of logical hierarchical groups, ACKs are aggregated level by level by the DRs in a group and then sent to the sender, the actual RTT value for one multicast group
to the sender should be larger than the given RTT value, i.e., the RTT value for a multicast group should be larger than 550ms. The actual RTT values are shown in Figure 20, where it is demonstrated that:

- As TRTT increases, the RTT value also increases, thus, the TRTT value should not be very high.

- Usually, if the highest logical hierarchical level increases, the RTT value also increases, but usually 3 levels of hierarchy are sufficient for most multicast cases.

- The reason why the RTT value increases is that ACKs need to pass along DRs from low hierarchical level to the highest hierarchical level. Another reason is the local error recovery discussed in Section 3.2.7, the ACK for a packet is sent to the sender only after the packet is received by all receivers in this group, which may also introduce some delays.

![Graph showing RTT vs. TRTT](image)

**Figure 20:** Average RTT vs. TRTT for $N = 200$, $M = 10$ and $p = 10^{-3}$.

Now we consider how the value of $M$ affects the formation of the logical hierarchical multicast groups. Let $TRTT = 30ms$. In a similar way, we get the hierarchical
groups as shown in Table 2. Obviously, if $M$ decreases, the highest level in a hierarchical group might increase, which is property of the B+ tree. $M$ can also affect how many groups are formed. One interesting observation is that the number of groups formed does not necessarily decrease for increasing $M$.

**Table 2:** The Resulting Hierarchical Groups for $N = 200$ and $TRTT = 30ms$.

<table>
<thead>
<tr>
<th>M</th>
<th>5</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>25</th>
<th>30</th>
<th>35</th>
<th>40</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group Number</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>6</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Highest Level</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

As shown in Figure 21, the effect of $M$ on the RTT value is random and the variations of the RTT are not large.

![RTT vs. M](image)

**Figure 21:** RTT vs. $M$ for $N = 200$, $TRTT = 30ms$ and $p = 10^{-3}$.

Obviously, the aggregated RTT value has very limited sensitivity to the parameter $M$. Thus, we can choose $M$ sufficiently large to put more receivers in one group so as to reduce the number of ACKs from the receivers and the number of retransmissions from the sender.
3.4.3 Throughput Performance

TCP-NewReno [39] is a TCP unicast algorithm, the most important feature of TCP-NewReno is that it can recover multiple missing packets in one window of data. TCP-Peachtree congestion algorithm discussed in Section 3.3 is also designed to recover multiple packets in one window of data. To evaluate TCP-Peachtree congestion algorithms, we use TCP-NewReno only for congestion control in our simulations and keep the logical hierarchical groups as we introduced in Sections 3.2. Thus, we can compare the throughput performance between TCP-Peachtree and TCP-NewReno. For multicast applications, we assume $N = 200$, $TRTT = 30ms$ and $M = 10$, the resulting throughputs are shown in Figure 22 for different values of $p$.

![Throughput Performance](image)

**Figure 22:** Throughput Performance Comparison of TCP-Peachtree and TCP-NewReno for different values of $p$ for $TRTT = 30ms$ and $M = 10$.

Figure 22 shows that the throughput performance of TCP-Peachtree is much better than the TCP-NewReno, especially when $p$ is large, i.e., TCP-Peachtree is more suitable for the reliable satellite multicasting. The reasons are obvious:

- In satellite IP multicasting, because of the loss path multiplicity problem, the
aggregated link error probability is rather high.

- Using the Jump Start algorithm, TCP-Peachtree can reach $rwnd$ much faster than TCP-NewReno, which uses the Slow Start algorithm. It is a critical factor in satellite IP networks [4], because the RTT value in satellite IP networks is rather high.

- NIL segments are used to exploit the network resources and also to recover lost packets on the receiver side. Because of the loss path multiplicity problem, if the receiver number is large, the NIL recovery can have a high advantage over TCP-NewReno.

- In the Quick Recovery, the information contained in the SACK option is used to improve the performance of TCP-Peachtree.

More detailed performance comparisons of the proposed congestion scheme and other TCP protocols can be found in [2, 6].

### 3.4.4 Overhead

In TCP-Peachtree, overhead as shown in Figure 23 is introduced by NIL segments sent during the Jump Start and Quick Recovery states. The overhead is measured by dividing the number of NIL segments by the number of total segments sent at the sender.

In general, the overhead decreases with decreasing $p$ from $10^{-2}$ to $10^{-6}$. For $N = 1000$ and $p = 10^{-2}$, the overhead is maximum at 29%. The reason for high overhead is that the aggregated link error probability is very large at this point and the loss path multiplicity problem makes it worse. Considering these factors, the overhead here is reasonable. Moreover, the practical channel conditions are below $p \leq 10^{-3}$ where the overhead drops below 10%.
3.4.5 Fairness to Unicast TCP Traffic

Usually, the TCP multicast traffic must co-exist with other existing TCP unicast traffic. This also applies to satellite IP networks. In order not to overtake bandwidth and starve the unicast traffic, TCP-Peachtree should be fair to unicast traffic.

As described in [76], the fairness index, based on throughput, for a bottleneck link is defined as:

$$FI = \frac{[\sum_{i=1}^{K} T(x)]^2}{N \sum_{i=1}^{K} T(x)^2}$$

(10)

where \(T(x)\) is the throughput of the \(x\)-th flow, and \(K\) is the number of flows sharing the resource. \(FI\) always lies between \(1/K\) (indicating one of them gets all the bandwidth and all others starve) and 1 (indicating all get an equal share of the bandwidth). Assume there are 11 TCP-NewReno unicast connections from the gateway to the receivers with \(rwnd = 64\), they compete for the bandwidth with TCP-Peachtree. The resulting fairness is shown in Figure 24.
For \( p = 10^{-2} \), the fairness is not good. However, it does not mean that the TCP-Peachtree is unfair to the unicast TCP-NewReno traffic. As the link error probability is very large, TCP-NewReno cannot fully utilize the bandwidth, while TCP-Peachtree can utilize the bandwidth more efficiently. The reason for the result is based on the fact that the bandwidth is not fully utilized by TCP-NewReno at this moment.

For decreasing \( p \), the fairness also increases. For \( p \leq 10^{-3} \), fairness becomes larger than 0.9. Moreover, fairness keeps increasing with decreasing \( p \) and eventually it reaches 1. As a result, the TCP-Peachtree is fair to TCP unicast traffic. The reason is that TCP-Peachtree only uses NIL segments to probe the network resources. When congestion occurs, NIL segments are usually dropped first. Consequently, TCP-Peachtree increases its sending rate much slower. In the worse case, it acts very similar to TCP-NewReno.

![Figure 24](image)

**Figure 24:** The fairness vs. \( p \) for \( TRIT = 30ms \) and \( M = 10 \).
3.4.6 Unbalanced Situation

For practical satellite multicast applications, different receivers may experience different channel conditions, thus, they may have different packet loss rates because of channel errors. Assume that the packet loss rate $p_i$ because of channel errors for receiver $i$ is randomly chosen in the range of $p_l$ to $p_h$. Here $p_l$ is chosen as $10^{-6}$ and $p_h=p$ increases from $10^{-6}$ to $10^{-2}$, the throughput performance with respect to $p_h$ is shown in Figure 25. The throughput of the balanced situation, i.e., all $p_i=p$, is also shown in Figure 25.

![Figure 25: The throughput comparison for balanced and unbalanced packet loss rates.](image)

The throughput for the unbalanced situation is always higher than that for the balanced situation. Obviously, the throughput difference is quite small when $p_h$ is close to $p_l$, i.e., $10^{-6}$, but it increases when $p_h$ increases from $10^{-6}$ to $10^{-3}$ and it reaches the highest value at $p_h=10^{-3}$. The reason is that local error recovery is very helpful to recover lost packets for the unbalanced situation. However, the throughput difference drops dramatically at $p_h=10^{-2}$. This is because most receivers are in worse
channel conditions and the local error recovery is not efficient in such situations, thus, the throughput performance is degraded greatly.

3.4.7 Rain Fade

Assume rain fades last for a short duration of not more than 60 secs similar to the assumption made in [55] and rain fade occurs at time $t=20$ secs. The timeout threshold $T_f$ for detecting rain fades is chosen to be the same as the retransmission timeout threshold. The sender may go to the Jump Start State before it can infer rain fade, thus, it always records the current congestion window $cwnd$ into the variable $r_f.cwnd$ before it goes to the Jump Start State. During the Hold State, the sender stops sending any data packets and freezes all retransmission timers to avoid unnecessary transmissions. However, it sends probing packets periodically to detect when the rain fade is over. To investigate the performance improvement by the Hold State, we also consider TCP-Peachtree without the Hold State, where the sender keeps going into the Jump Start State when the transmission timer expires. The throughput performance for TCP-Peachtree with and without the Hold State is shown in Figure 26.

The throughput with the Hold State is always higher than that without the Hold State and the throughput difference is approximately constant for different rain fade periods. The reasons that TCP-Peachtree with the Hold State achieves higher throughput are as follows,

- In the Hold State, the sender uses probing segments to obtain the exact information about the rain fade. On the other hand, for TCP-Peachtree without the Hold State, the sender has to wait for the retransmission timer to expire and then goes to the Jump Start State to send packets to the receiver.

- The sender records its current congestion window when the rain fade occurs
Figure 26: The throughput vs. rain fade period.

and keeps this congestion window after the rain fade is over. While for TCP-Peachtree without the Hold State, the sender always goes to the Jump Start State and the congestion window is set to 1.

3.4.8 Bandwidth Asymmetry

Since the feedback link capacity is usually very low for satellite networks, the delayed SACK scheme discussed in Section 3.2.6 is adopted to address the bandwidth asymmetry problem, i.e., the DR in the highest level sends one SACK for a given number of received data packets. This number is called the delay factor. We investigate 3 cases of the bandwidth asymmetry problems, i.e., the feedback link bandwidth is 1 Mbps, 100 Kbps, and 64 Kbps, respectively. Also assume the link capacity from the sender to the receivers is 10 Mbps. The throughput performance with respect to different delay factors is shown in Figure 27.

When the feedback link bandwidth is 1 Mbps, the throughput performance is not degraded for this bandwidth asymmetry ratio. The reason is that the ACK size is usually about 40 Bytes, which is much smaller than the data packet size 1 KB, thus,
the feedback link is not congested. However, the throughput performance decreases with increasing bandwidth asymmetry ratio. For example, the throughput drops to 73.81 from 103.31 when the feedback link bandwidth is 100 Kbps and the delayed SACK scheme is not used. The throughput increases with increasing delay factor and reaches the highest value when delay factor is 3, which is close to the throughput for 1 Mbps. Since TCP relies on the ACK clock to transmit packets, the delay factor can not be too large, thus, the throughput performance can be degraded for very high bandwidth asymmetry ratios. For example, the throughput decreases approximately 20% when the feedback link bandwidth is 64 Kbps and the delay factor is 4.

### 3.4.9 Scalability

The scalability is one of the most important metrics of IP multicast schemes. Usually, the scalability is measured by the performance over different numbers of receivers in a multicast session. Schemes with good scalability can be applied to very large size of receivers. Here we use the throughput as the performance measurement. For $TRTT = 30ms$ and $M = 10$, the resulting scalability is shown in Figure 28, where

---

**Figure 27:** The throughput vs. delay factor.
we observe that the throughput is constant and does not decrease with $N$ increasing from 50 to 1000. We can conclude that TCP-Peachtree has good scalability. The reasons are based on the facts:

- The acknowledgment implosion problem is solved by the ACK fusion procedure performed by the DRs of the hierarchical multicast groups and the sender.
- The loss path multiplicity problem is solved by the local error recovery and the NIL recovery.
- The congestion control algorithm in TCP-Peachtree can recover multiple packet losses in one window of data so that TCP-Peachtree is very suitable to work in applications with high packet loss rates because of link errors.

![Figure 28](image-url)  
**Figure 28:** The scalability for $TRT = 30ms$ and $M = 10.$
CHAPTER IV

RELIABLE DATA TRANSPORT PROTOCOLS
FOR INTERPLANETARY INTERNET

4.1 Problem and Solution

The developments in space technologies in the last decade have enabled the realization
of deep space scientific missions such as Mars exploration. These missions produce
a 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 the development of the 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 [15]. The next step in the design and develop-
ment of deep space networks is expected to be the Internet of deep space planetary
networks, defined as the InterPlaNetary (IPN) Internet [87].

A typical deep space network architecture shown in Figure 29 is proposed for the
Mars Exploration mission [14]. The architectural elements of the proposed infra-
structure can be summarized as follows:

- **Interplanetary Backbone Network**: It includes the direct link or multi-hop
  paths between the outer-space planet and the Earth as well as the Earth-based
  infrastructure elements such as ground stations for deep space networks. It may
  also include relay stations placed at gravitationally stable Lagrangian points of
  outer-space planets such as Jupiter and Pluto [14].

- **Planetary Satellite Network**: It consists of the satellites orbiting the planets
Figure 29: Deep space network architecture for the Mars Exploration mission.

to provide communication relay and navigation services to the surface elements.

- **Planetary Proximity Network:** It provides communication links between a 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 the ground station at the Earth and the planetary gateway connected to the relay satellites orbiting around the outer-space planets, as shown in Figure 29. This is because the Interplanetary Backbone Network plays a significant role in the performance of the entire deep space communication. The most important characteristics and the challenges posed by the Interplanetary Backbone Network are as follows:

- **Extremely 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 [30].

- **High link error rates:** The bit error rates on the deep space links are very high, usually on the order of 10^{-1} [30].
• **Blackouts**: Periodic link outages may occur because of orbital obscuration with the loss of line-of-sight resulting from moving planetary bodies or the interference of an asteroid or a spacecraft [15].

• **Bandwidth asymmetry**: The asymmetry in the bandwidth capacity of forward and reverse channels is typically on the order of 1000 : 1 in space missions [30].

These challenges need to be addressed to meet the communication requirements of deep space missions. However, the existing TCP variants[6], [19], [23], [39], [50], [51], [61], [62] have been shown to achieve very poor performance in deep space communication networks [2]. The dominant factor in this performance degradation is the extremely high propagation delay in deep space links [2]. 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 that is proportional to the propagation delay. For \(W_{ss} = 20\) and \(RTT = 20\) minutes, it is shown in [2] that the slow start algorithm cannot utilize link resources for approximately 120 minutes in deep space links.

The inefficiency in link utilization resulting from the window-based mechanism 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 [2] shows that the 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 [30].

Furthermore, the current TCP protocols are designed for wired links, which are reasonably assumed to have negligible bit error rates. Therefore, TCP protocols invoke congestion control mechanisms in the 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 to address the throughput degradation resulting from wireless link errors [13]. However, these solutions cannot be directly applied to the Interplanetary Backbone Network because of the amplifying effects of the extremely high propagation delay and the other abovementioned characteristics of the problem.

The TCP performance on the links with high bandwidth-delay products and errors is analyzed in [58]. Many transport protocols [5], [6], [47] 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 resulting from blackout conditions may also mislead the congestion control mechanisms based on packet losses. In [44], an enhancement for TCP is developed to address signal loss conditions resulting from mobility. However, the blackout situations in deep space links are much more complicated because of extremely high propagation delays and hence solutions as in [44] cannot be applied directly.

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

- **Delayed Feedback**: TCP is expected to respond to network states. This expectation creates problems in long-delay environments, since TCP uses the
Figure 30: TP-Planet protocol operation state diagram including substates and the state transitions based on congestion control decision mechanism.

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 actions. Therefore, congestion control schemes, which react to instantaneous packet loss situations, do not yield proper responses on the links with high propagation delays.

- **Buffer Size**: To assure 100% reliable transport, the retransmission mechanism is inevitable. However, this requires a 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.

Active research already exists on transport layer protocols for space-based communication networks. Space Communications Protocol Standards-Transport Protocol (SCPS-TP) [24], [29] is a set of TCP extensions developed by the Consultative Committee for Space Data Systems (CCSDS) for space communications. SCPS-TP was designed to support current communication environments and those of upcoming space missions [24]. SCPS-TP was developed based on 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 addressing the challenges in the Interplanetary Backbone Network [2]. For example, SCPS-TP with Vegas congestion control uses the window-based scheme and adopts the slow-start algorithm. Although the rate-based version of SCPS-TP is under development, it disables the congestion control mechanism and performs transmissions with user selected fixed rates [86]. On the other hand, for example, SCPS-TP uses the TCP-Vegas [19] 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 the variation in RTT. Therefore, the congestion decision based on the RTT variation does not provide proper congestion control functionality. Furthermore, the variation in RTT may not be measured accurately because of the extremely high propagation delay such that the resulting congestion control behavior may also not be accurate.

As pointed out in [2], an urgent need exists for reliable transport protocol for the InterPlaNetary Internet. In this chapter, a reliable transport protocol, TP-Planet, for the InterPlaNetary Internet (IPN) is presented. TP-Planet is mainly implemented at Interplanetary Backbone Network nodes, i.e., the TP-Planet source and sink are the ground station at the Earth and the planetary gateway connected to the relay satellites orbiting around outer-space planets. The objective of TP-Planet is to achieve high throughput performance and reliable data transmission by addressing the challenges in the InterPlaNetary Backbone Network. To address the challenges resulting from extremely high propagation delays, TP-Planet deploys 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., the Initial State and the Steady State, constitute the structure of the TP-Planet protocol. The Initial State algorithm replaces the inefficient
slow start algorithm to capture link resources in a very fast and controlled manner. In the Steady State, a new congestion detection and control mechanism is deployed to minimize erroneous congestion decisions resulting from high link errors. To reduce the effects of blackout conditions on the throughput performance, TP-Planet incorporates the Blackout State procedure into the protocol operation. The bandwidth asymmetry problem is addressed by the adoption of delayed SACK options.

The remainder of the chapter is organized as follows. The TP-Planet protocol overview along with the detailed operation of the Initial State algorithm are presented in Section 4.2. In Section 4.3, the Steady State algorithm, including the new rate-based AIMD congestion control and the Blackout State behavior, is explained. Performance evaluation is presented in Section 4.4.

![Time Framing Mechanism](image)

**Figure 31:** An example illustration of time framing mechanism in the Immediate Start phase for $ssthresh_c = 8$.

### 4.2 TP-Planet: Initial State

The TP-Planet source starts a connection in the *Initial State* at $t = 0$ by calling the `Initial_State()` algorithm, as shown in Figure 30. The TP-Planet source then goes to the *Steady State* by calling the `Steady_State()` algorithm at $t = 2 \cdot RTT$, as shown in Figure 30.

The slow start algorithm used in existing TCP protocols is shown to be inefficient on deep space links of the InterPlanetary Internet [2]. To avoid the performance degradation resulting from the Slow Start algorithm, TP-Planet deploys the
**Initial State** algorithm, which captures link resources in a very fast and controlled manner. The algorithm is composed of two main procedures, i.e., **Immediate Start** and **Follow-Up**.

### 4.2.1 Immediate Start

TP-Planet starts a connection in the **Immediate Start** state at \( t = 0 \), where the actual RTT is divided into time intervals of size \( T \). TP-Planet then emulates Slow Start and Congestion Avoidance algorithms of the conventional window-based 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 4.2.3. Here, we present the protocol operation in the Immediate Start phase in two parts, i.e., **Emulated Slow Start** and **Emulated Congestion Avoidance**.

#### 4.2.1.1 Emulated Slow Start

As shown in Figure 31, the first RTT is divided into time intervals of size \( T \). This is done to have more fine-grained time units than the extremely high RTT of the deep space link. The number of data packets transmitted in each interval, \( cwnd \), is increased geometrically in each interval by emulating the window-based behavior of the Slow Start algorithm of 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** [6] are also sent in each time interval during the Immediate Start phase. The objective of the NIL segment transmission is to probe the 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 Figure 29, have priority-queueing capability, low priority NIL segments are discarded first in the case of congestions. Therefore, the 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 the Generate_Nil_Segment() algorithm shown in Figure 4, where $Q$ is the length of the queue of unacknowledged outstanding data packets and $i$ is a counter.

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 Figure 31, we assume the emulated slow start threshold is eight 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:

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

$$cwnd = 2^{i-1}$$

(11)

$$cwnd_N = ssthresh_e - 2^{i-1}$$

(12)
4.2.1.2 Emulated Congestion Avoidance

The Emulated Slow Start is left for the Emulated Congestion Avoidance phase when $cwnd = ssthresh_e$. Since there is no feedback information about link resources as yet, the TP-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 the congestion avoidance algorithm of classical TCP protocols. As in Figure 31, 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 the TP-Planet source goes to the Follow-Up state of the Initial State algorithm, as shown in Figure 30.

4.2.2 The Follow-Up Phase

During this phase, the feedback for the packets transmitted in the Immediate Start phase starts to be received by the TP-Planet source. 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 4.3.4. Since NIL segments are transmitted to probe the link status, each received NIL segment indicates the existence of unutilized link resources. Hence, the TP-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. The TP-Planet source transmits $ssthresh_e$ packets in $RTT \leq t \leq RTT + T$ until the first NIL ACK received at $t = RTT + T$. Then, the TP-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 TP-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 $2 \cdot ssthresh_e$, the data transmission rate $S$ at the end of the Initial State is expressed by

$$S = \frac{\min\{N_{ACK}, 2 \cdot ssthresh_e\}}{T} \quad (13)$$

In addition to data packets, the TP-Planet source also starts sending NIX segments during the Follow-Up phase. NIX segments are much smaller than 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 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 4.3. The Follow-Up phase is over at $t = 2 \cdot RTT$ and the source leaves the Initial State for the Steady State, as shown in Figure 30.

Consequently, the TP-Planet source can capture 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 congestions by controlling the number of probe packets injected into the network.

### 4.2.3 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 smaller $T$ increases link utilization, it also increases
Initial_State()
Send connection request CONN_REQUEST;
Set $T$ & $ssthresh_c$ by (18) & (19)
$n = 1$;
cwnd = 1;
While ($t \leq RTT$)
/* Immediate Start */
While (cwnd $\leq ssthresh_c$)
/* Emulated Slow Start */
If ($(n - 1)T \leq t \leq nT$)
  Send (cwnd) DATA pkts;
  Send ($ssthresh_c - cwnd$) NIL pkts;
end;
n = n + 1;
cwnd = 2^{n-1};
end;
/* Emulated Congestion Avoidance*/
cwnd = ssthresh_c;
cwnd_N = 1;
While ($(n - 1)T \leq t \leq nT \leq RTT$)
  Send (cwnd) DATA pkts;
  Send (cwnd_N) NIL pkts;
cwnd_N = cwnd_N + 1;
n = n + 1;
end;
end;
While ($RTT \leq t \leq 2 \cdot RTT$)
/* Follow-Up */
If (NILACK.RECEIVED)
  Set data rate $S = N/T$;
  Set NIX rate $S_{Nix} = S$;
end;
end;
Steady_State();
end;

Figure 32: The Initial_State() algorithm.
the overhead incurred by the NIL segment transmission.

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, the TP-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 the TP-Planet source has a given target minimum transmission rate requirement, $B$. This requirement is mostly due to two main reasons:

- **Application**: The minimum transmission rate is required to meet specific application needs such as the 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 specific space mission requirements as the maximum duration for the transmission of a 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 $ssthresh_e$ is calculated as

$$ssthresh_e = B \cdot T$$

(14)

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 (12) it follows that the duration of the Emulated Slow Start phase ($T_{ess}$) is given by

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

(15)
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., \( t = RTT \). 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., \( cwnd_N = ssthresh_c \). The Immediate Start is over when \( cwnd_N \) reaches \( ssthresh_c \) at \( t = RTT \). Therefore, the duration of the Emulated Congestion Avoidance phase \( T_{eca} \) can be expressed by

\[
T_{eca} = ssthresh_c \cdot T
\]  

(16)

Since the total duration of the Immediate Start is \( T_{ess} + T_{eca} = RTT \), from (15) and (16) it follows that

\[
(\log_2 ssthresh_c + 1 + ssthresh_c) \cdot T = RTT
\]  

(17)

As \( \log_2 ssthresh_c + 1 \) is negligible compared to \( ssthresh_c \) itself, from (14) and (17) the size of the time interval \( T \) can be calculated by

\[
T = \sqrt{\frac{RTT}{B}},
\]  

(18)

where \( B \) is the target transmission rate in packets/s. At the beginning of a connection, \( RTT \) is also not known to the TP-Planet source. Therefore, the TP-Planet source also caches the \( RTT \) of the past connections and uses that value to calculate the time interval size \( T \) by (18) during the Initial State.

Consequently, the emulated slow start threshold can be expressed by using (14) and (18) as follows

\[
ssthresh_c = \sqrt{RTT \cdot B}
\]  

(19)
Thus, at the beginning of the connection, \( T \) and \( ssthresh_e \) are set by (18) and (19). The first RTT period is then divided into time intervals of size \( T \). The Emulated Slow Start and the Emulated Congestion Avoidance are performed in the first RTT to perform resource probing. In the second RTT, the transmission rate is increased by sending a new data packet for each received ACK until \( t = 2 \cdot RTT \). In this way, the TP-Planet source can increase its transmission rate very quickly and can efficiently utilize resources during the Initial State without leading to congestion.

### 4.2.4 The Connection Establishment

The conventional TCP protocols perform the three-way handshake for connection establishment. Existing protocols send the 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 the waste of huge bandwidth for at least a duration of one \( RTT \) in deep space links. To avoid this inefficiency, the TP-Planet source does not wait for the ACK for the connection request and starts the data transmission in the Initial State as if the session request was granted. If the request is rejected, then the connection is terminated after one \( RTT \).

As a result, TP-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 TP-Planet is summarized in Figure 32.

### 4.3 TP-Planet: Steady State

The TP-Planet source leaves the Initial State for the Steady State at \( t = 2 \cdot RTT \) and remains in the Steady State until the connection is terminated. During the Steady State operation, the TP-Planet source can be in one of four states, i.e., Increase Rate, Decrease Rate, Hold Rate, and Blackout, as shown in Figure 30. In the beginning of the Steady State, the source goes to the Hold Rate state, where no transmission rate change is performed. During the Steady State operation, TP-Planet deploys a
new congestion control scheme. Hence, the transition 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 held according to the current state.

4.3.1 Congestion Control

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

During the Steady State, the TP-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 the TOS field in the IP packet header. NIX segments differ from NIL segments used in the 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 resulting from space link errors. However, assuming the relay satellites serving as routers along the Interplanetary Backbone link, as shown in Figure 29, have priority-queuing capability, low priority NIX segments are discarded first in the case of 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 resulting from congestions. This reasoning constitutes the basis for our congestion detection method via the NIX segment transmission.

The TP-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 because of the bandwidth asymmetry in deep space links. Only the reception statistics within a time window of \(T_w\) are returned to the sender every \(\tau\) period. This information is carried by NIX ACKs. The TP-Planet source receives NIX ACKs carrying \((N_{Low}, N_{High})\) at the end of each measurement period \(\tau\).

Let \(\Phi\) be the ratio of the number of received low and high priority NIX segments, i.e.,

\[
\Phi = \frac{N_{Low}}{N_{High}}
\]  

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

The entire congestion control of the TP-Planet source is determined according to the value of \(\Phi\). The transitions among Increase, Decrease, and Hold rate states in Figure 30 are performed based on \(\Phi\). To avoid unnecessary state transitions, the decision is made via comparing \(\Phi\) with preset rate decrease, \(\phi_d\), and the rate increase threshold, \(\phi_t\), as shown in Figure 30. These thresholds are protocol parameters, whose selection is an implementation issue. The values of \(\phi_d\) and \(\phi_t\), which yield highest protocol performance, are determined during simulation experiments and are given in Section 4.4.

A summary of the congestion control mechanism is given as follows:
Steady_State()
    Set $\xi$;
    Set $\alpha$ by (22);
    Set $\zeta$ & $\delta$ by (23) & (24);
    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 ($\Phi < \phi_d$)
                /* Decrease Rate */
                $S = S \cdot \zeta$;
            else if ($\phi_d \leq \Phi \leq \phi_l$)
                /* Hold Rate */
                $S = S$;
            else if ($\Phi > \phi_l$)
                /* Increase Rate */
                $S = S + \delta$;
            end;
        end;
    If (ZERO_NIX_ACK_RECEIVED)
        /* After Blackout State */
        Goto Hold State;
    end;
    If (NO_ACK_in_Tw)
        /* 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)
                /* $L \geq 2x$ */
                Goto Hold State;
            else
                /* $L < 2x$ */
                Goto Congestion Decision;
            end;
        end;
        If (DATA_ACK_RECEIVED)
            /* $L < 2x$ */
            Goto Congestion Decision;
        end;
    end;
end;

Figure 33: The Steady_State() algorithm.
1. \( \Phi < \phi_d \): In this case, the TP-Planet source infers that a 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. \( \phi_d \leq \Phi \leq \phi_i \): In this case, the data transmission rate \( S \) is kept unchanged until next feedback is received from the sink.

3. \( \Phi > \phi_i \): the TP-Planet source infers that no congestion is experienced. Consequently, it increases the data transmission rate additively, i.e., \( S = S + \delta \).

The additive-increase (\( \delta \)) and multiplicative-decrease (\( \zeta \)) parameters are used to perform AIMD rate control every \( \tau \) period. The selection of these AIMD parameters is explained in Section 4.3.2. The procedure for the Steady State algorithm of the TP-Planet protocol is summarized in Figure 33.

### 4.3.2 The New Rate-Based AIMD Scheme

As mentioned before, current TCP protocols achieve very poor performance on links with extremely high propagation delays. The throughput of window-based TCP protocols and rate-based schemes is inversely proportional to the RTT [67] and the square-root of RTT, respectively. Thus, rate-based congestion control schemes are more robust to excessive propagation delays than window-based mechanisms. Hence, to address the adverse effects of extremely high propagation delays on throughput performance, TP-Planet deploys rate-based additive-increase multiplicative-decrease (AIMD) congestion control. The steady state throughput of the rate-based AIMD scheme is expressed by [1]

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

(21)

where \( \alpha \) and \( \xi \) are the additive-increase and the multiplicative-decrease factors, respectively, and \( p \) is the packet loss probability. It is observed from (21) that the
throughput of the rate-based AIMD scheme depends on the values of $\alpha$ and $\xi$. Therefore, TP-Planet adapts its AIMD parameters to link conditions, i.e., RTT and packet loss rate, such that their adverse effects on the performance are compensated and a given target throughput is achieved.

The additive-increase parameter ($\alpha$) of the rate-based AIMD scheme to achieve a target throughput of $B$ packets/s can be obtained from (21) as follows

$$\alpha = \frac{(1+\xi)}{2} \left( B + \frac{1}{RTT \cdot p} \right) \left[ \sqrt{1 + \frac{8B^2(1-\xi)}{(B+1/RRT \cdot p)^2(1+\xi)^2} - 1} \right]$$  \hspace{1cm} (22)

where $\xi$ 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 a certain amount of information within a certain delay bound as described in Section 4.2.3.

These AIMD parameters, $\alpha$ and $\xi$, in (22) are the rate change parameters to be used to control the rate with period RTT. However, the TP-Planet source performs rate control with the feedback received from the sink every $\tau$ period, where $\tau \ll RTT$. Thus, the rate control parameters $\alpha$ and $\xi$ in (22) cannot be directly used for TP-Planet. Instead, these parameters represent the upper bound for the rate change of the TP-Planet source within one RTT period. Hence, the AIMD parameters to be used by the source with period $\tau$ can be derived from $\alpha$ and $\xi$.

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

Let $\delta$ and $\zeta$ be additive-increase and multiplicative-decrease parameters to be used by TP-Planet for rate control with period $\tau$. Then, $\zeta$ can be calculated as

$$\zeta = \xi^{(\tau/RTT)}$$  \hspace{1cm} (23)

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

\[ \delta = \frac{\alpha \cdot \tau}{RTT} \]  

(24)

The TP-Planet source uses the AIMD scheme during the Steady State operation, as shown in Figure 30. 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. The TP-Planet source then uses (22), (23), and (24) to calculate its AIMD parameters, i.e., \( \zeta \) and \( \delta \). Consequently, according to the result of the congestion decision mechanism presented in Section 4.3.1, the transmission rate \( S \) is multiplicatively decreased or additively increased with \( \zeta \) and \( \delta \), respectively.

### 4.3.3 The Blackout State Behavior

Link outages resulting from loss of line-of-sight by orbital obscurations lead to burst packet losses and a decrease in the throughput. To provide reliable transport, SACK options [61] are adopted by TP-Planet to address burst losses. A timeout mechanism is also included in TP-Planet because of the possible inadequacy of the number of SACK blocks in the SACK option field for very long blackout durations. To reduce the throughput loss resulting from blackouts, the Blackout State is developed and incorporated into the Steady State.

The TP-Planet source receives data ACKs for reliability control purposes and NIX ACKs for the NIX segment reception statistics. If the source does not receive any type of ACKs for a certain period of time \( T_w \), it infers that this condition is a blackout and goes to the Blackout State, as shown in Figure 30. The objective of the Blackout State procedure is to reduce the throughput degradation caused by the blackout situation.

During the blackout, the TP-Planet source keeps sending low and high priority NIX packets without changing its transmission rate. The same blackout event is also detected by the TP-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 \((N_{\text{Low}}, N_{\text{High}})\) as \((0,0)\). These ACK packets are called \textit{Zero NIX ACKs}. The objective of Zero NIX ACKs is to help the TP-Planet source to capture accurate information regarding the blackout situation and act accordingly.

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

1. \(L < 2x\): After \(rtt - x\) from \(t_0\), i.e., at \(t_1 = t_0 + rtt - x\), the TP-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 the Blackout State at \(t = t_1\), as in Figure 34.

![Diagram](image)

\textbf{Figure 34}: The blackout condition observed from the TP-Planet source for \(L < 2x\).

In this case, the TP-Planet source does not send any new data packets, but sends low and high priority NIX segments without changing their transmission rates. It also does not invoke the congestion control mechanism. It starts retransmitting the packets whose retransmission timer is expired. At \(t_2 = t_1 + L\), the TP-Planet source receives normal ACKs for a duration of \(2x - L\). Therefore, the TP-Planet source infers that the blackout is over and goes to any of the Increase, Decrease, or Hold states according to the information it receives in the first NIX ACK, as shown in Figure 30. At \(t_3 = t_2 + 2x - L\), the source
starts to receive Zero NIX ACKs for a duration of \( L \). These Zero NIX ACKs are transmitted by the receiver when it detects the same blackout condition. Therefore, TP-Planet does not go to the Blackout State; instead it goes to the Hold State, where it keeps sending new data packets with the same transmission rate again, as shown in Figure 30. Consequently, TP-Planet reduces the effect of the blackout on the performance by not wasting link resources for a duration of \( L \).

![Figure 35: The blackout condition observed from the TP-Planet source for \( L \geq 2x \).](image)

2. \( L \geq 2x \): In this case, the TP-Planet source detects no ACK period and goes to the Blackout State at \( t_1 = t_0 + rtt - x \) for the blackout that 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 a duration of \( 2x \), as shown in Figure 35. This reveals that the blackout is over for the TP-Planet source and then the source leaves the Blackout State for the Hold State, as shown in Figure 30. It stays in the Hold State and keeps sending data packets with the same transmission rate before the blackout was detected. At \( t_3 = t_2 + 2x \), the Zero NIX ACK period is over and the source performs state transition according to the information received in the first NIX ACK. Hence, the duration of \( 2x \) is efficiently utilized with the help of the Zero NIX ACK mechanism.

Consequently, the Blackout State reduces the throughput degradation caused by blackout conditions and improves the link utilization for a duration of \( L \) or \( 2x \) in the cases \( L < 2x \) and \( L \geq 2x \), respectively.
4.3.4 The Delayed SACK

TP-Planet uses selective acknowledgment (SACK) [61] options to assure reliable data segment transmission. The TP-Planet sink continuously sends SACKs back to the source for each data packet it receives. Given that data packets are 1 KB and SACKs are 40 B, then the ratio of the traffic in the forward and reverse channels is 25 : 1, i.e., 1 KB/40 B. Thus, the bandwidth asymmetry up to 25 : 1 causes no congestion in the reverse link. However, the bandwidth asymmetry in the space links is usually on the order of 1000 : 1 [30]. Thus, sending one SACK for each data packet can cause the reverse channel to be congested, resulting in packet losses in the reverse link.

To avoid this problem, TP-Planet deploys SACK congestion control by delaying the SACKs. The TP-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, the TP-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 are evaluated in Section 4.4.4.

4.4 Performance Evaluation

To investigate the performance of TP-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 4.4.1. Throughput performance of TP-Planet is analyzed in Section 4.4.2, along with the overhead introduced. The Effects of blackout conditions on the performance and the performance of TP-Planet with the improvement by the Blackout State are investigated in 4.4.4. TP-Planet performance on deep space links with asymmetrical bandwidth and the improvement with delayed SACKs are explored in Section 4.4.5.
Figure 36: The transmission rate change in the Initial State of the TP-Planet source for $RTT = 600$ seconds.

4.4.1 Initial State Performance

To avoid performance degradation resulting from inefficient connection starting behavior, TP-Planet deploys the Initial State() algorithm in Figure 30 and Figure 32. 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 the same experiments with TP-Planet, TCP-Peach+ [6], and TCP-NewReno. The reason is that the initial phase behavior of most current TCP protocols is based on the Slow Start algorithm, which is also used by TCP-NewReno.

In Figure 36, the transmission rate change in the Initial State of the TP-Planet source is plotted, where the target transmission rate of the TP-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 the retransmission mechanism is proportional to the data transmission rate. Therefore, very high data rates require considerable memory. For example, the source needs to maintain 0.6 GB of buffer for $RTT = 10$ minutes and the average data transmission rate of 1 MB/s.
Thus, we set the target data rate to 100 packets/s to maintain practicality in terms of memory requirements.

As explained before, the TP-Planet source performs the 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 Figure 36, the Emulated Slow Start phase lasts for approximately $T_{ess} = 19.58$ seconds and the TP-Planet source reaches 100 packets/s
Figure 38: The throughput for changing $p$ and file size with $RTT = 600$ seconds.

data rate by then. The emulated Congestion Avoidance phase then starts and lasts until the end of the first RTT of the connection period. During this phase, the data transmission rate is not increased. Instead, the NIL segment transmission rate is increased linearly to further probe link resources. At $t \approx 600$, the source goes to the 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+ [6] deploys the Jump Start algorithm to capture link resources very quickly. In the Jump Start algorithm, 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 the Jump Start algorithm is shown in Figure 37. The congestion window of TCP-Peach+ reaches 20 packets in at most $2 \cdot RTT$ duration. It is, however, still very low compared to the performance of the Initial State of TP-Planet, which reaches 200 packets/s data rate at the end of $2 \cdot RTT$. 

94
The slow start performance, however, is not even close to what the Initial State of TP-Planet achieves in deep space links of the InterPlaNetary Internet. In Figure 37, 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 because of the unsuitability of the slow start algorithm to extremely high propagation delays in deep space links. Recall that for higher threshold windows size values, the link is not efficiently utilized for a longer period of time [2].

### 4.4.2 Throughput Performance

To show the throughput performance of TP-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 3 are used during simulation experiments unless otherwise stated. The investigation of the effects of these parameters on the protocol performance is left for future study.

In [2], existing TCP protocols including TCP-Vegas, which is adopted by the SCPS-TP protocol as a congestion control scheme for space-based communications [24], 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 30 B/s and hence the entire link remains almost unutilized [2]. Although TCP-Peach+ has significantly outperformed other TCP schemes for the same link, the performance degradation was still too high that it only achieves throughput around 93 B/s. Thus, the same experiments are not repeated here and the details can be found in [2].

TP-Planet simulation experiments are performed for the transmission of varying
Table 3: Protocol Parameters Used in Experiments.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi_i$</td>
<td>0.8</td>
<td>Rate increase threshold</td>
</tr>
<tr>
<td>$\phi_d$</td>
<td>0.2</td>
<td>Rate decrease threshold</td>
</tr>
<tr>
<td>$\tau$</td>
<td>5</td>
<td>NIX ACK period in seconds</td>
</tr>
<tr>
<td>$T_w$</td>
<td>20</td>
<td>Sliding window size in seconds</td>
</tr>
<tr>
<td>$\xi$</td>
<td>0.5</td>
<td>Rate decrease parameter for RTT</td>
</tr>
<tr>
<td>$d$</td>
<td>5</td>
<td>Delayed-SACK factor in number of packets</td>
</tr>
</tbody>
</table>

Figure 39: The throughput for changing RTT for file size of 50MB and $p = 10^{-5}$. File sizes between 100 KB and 200 MB and for varying packet loss probability $p$ of $10^{-5}, 10^{-4}, 10^{-3}$. As shown in Figure 38, the throughput increases with increasing file size. This is because the larger the file size is, the longer TP-Planet stays in the Steady State. As a result, the link utilization is increased. Although the throughput decreases for increasing packet loss probability, this degradation also decreases for increasing file size. This is mostly because the throughput improvement by the new congestion control scheme is higher when the protocol stays longer in the Steady State. For the transmission of a 200 MB file and $p = 10^{-5}$, the throughput increases up to 82.2 packets/s, approaching its target throughput value, i.e., $B = 100$ packets/s.
Hence, TP-Planet outperforms existing TCP protocols by approximately $10^3$ times in terms of throughput.

In Figure 39, the effect of the $RTT$ on the throughput performance is shown. The experiments are performed for the transmission of a 50 MB file. The increase in $RTT$ leads to the throughput degradation, as shown in Figure 39. However, the throughput degradation resulting from the high propagation delay is not as severe as it is for conventional TCP protocols [2]. This is because of the new rate-based congestion control scheme used in the Steady State of TP-Planet. At $RTT = 600$ seconds, TP-Planet achieves 44.72 KB/s throughput. Note that this value can further increase with increasing file size, as in Figure 38.

4.4.3 Overhead

During the connection period, the TP-Planet source brings overhead to the deep space link. This overhead is caused by the NIL segment transmission to probe the link resource in the Initial State, and low and high priority NIX packets transmission for the 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 Figure 40, the overhead incurred by TP-Planet is shown for the 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, a 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 Figure 40. This is because the overhead in the Steady State is caused by the transmission of small sized (40 bytes) NIX segments and hence is much lower compared to the Initial State. Therefore, as the file size increases, the time spent in the Steady State also increases, which in turn decreases the overall overhead in the connection period. As a matter of fact, the
**Figure 40:** The overhead resulting from the transmission of NIL and NIX segments for changing file size and \( p \) with \( RTT = 600 \) seconds.

Scientific data delivered during space exploration missions are significantly high [15], i.e., on 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 number 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 the Initial State brings overhead, it also leads to significant performance improvement as observed in Section 4.4.1. As the NIX transmission rate is equal to data rate and the congestion control is robust to link errors, the overhead resulting from the NIX transmission is also independent of \( p \). The overhead resulting the NIX transmission exists after the Initial State until the end of the connection. However, the congestion control decision support provided by NIX segments leads to significant throughput performance improvement. For 1 GB 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 \( 10^3 \).
Figure 41: The throughput for changing blackout duration with $RTT = 120$ seconds and $p = 10^{-5}$.

4.4.4 Blackout Conditions

When a blackout is detected, TP-Planet moves to the Blackout State as shown in Figure 30 to reduce its effect on the throughput performance as explained in Section 4.3.3. The throughput achieved by TP-Planet for different blackout durations is given in Figure 41. 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 Figure 41, the throughput decreases with increasing blackout duration as expected. This decrease is observed in both of the curves representing the TP-Planet operation with and without the Blackout State procedure. However, the 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, TP-Planet achieves 36.41 packets/s throughput with the help of the Blackout State behavior. This corresponds to approximately a 14% performance improvement over the case without the Blackout procedure.
Figure 42: The throughput for changing delayed SACK factor $d$ with $RTT = 120$ seconds and $p = 10^{-4}$.

4.4.5 Bandwidth Asymmetry

To address bandwidth asymmetry problems, TP-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 SACKs. 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 Figure 42, two cases with different bandwidth asymmetry ratios are investigated:

1. **100:1.** In this case, the 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 the asymmetrical channel capacity of the deep space link. As shown in Figure 42, 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, the reverse channel is congested. On the
other hand, if $d$ is too large, the source receives fewer SACKs than it expects, which leads to performance degradation. Thus, at $d = 15$, the throughput achieved increases up to 52.02 packets/s.

2. **1000:1.** In this experiment, the bandwidth asymmetry on the order of 1000 : 1 is simulated by setting the 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, the throughput improvement with the delayed SACK method is more significant for higher bandwidth asymmetry. For the case without delayed SACKs, the throughput decreases up to 1.99 KB/s. As in Figure 42, the throughput increases with an 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 are required to be transmitted back to the source immediately. For a delayed SACK factor of 15, the throughput increases up to 21.37 KB/s, which corresponds to a 10 times increase in the throughput performance compared to the case without delayed SACKs, i.e., $d = 1$ in Figure 42.
CHAPTER V

MULTIMEDIA TRANSPORT PROTOCOLS FOR INTERPLANETARY INTERNET

5.1 Problem and Solution

Recently the research interests in deep space are arising rapidly, which include scientific spacecraft traveling, Mars exploration, radio and radar astronomy observations of the solar system and the universe. The future space missions to deep space require communication among planets, moons, satellites, asteroids, robotics spacecrafts, and crewed vehicles. The scientific data from these missions need to be delivered to the Earth successfully. To achieve this goal, InterPlaNetary Internet is proposed to be the Internet of the deep space planetary networks [7, 87].

As discussed in Chapter 5 Section 4.1, a typical InterPlaNetary Internet architecture is proposed for the Mars Exploration Mission [15], which includes InterPlaNetary Backbone Network, Planetary Satellite Network, and Planetary Proximity Network.

Multimedia traffic is a part of the aggregate traffic over the deep space communication links [15]. Some audio and visual information including planet images and data from scientific observations will be transmitted via these links. This type of traffic does not require 100% reliability and mostly has strict requirements on jitter bound, minimum bandwidth, and smooth traffic [46, 75]. The multimedia applications usually are classified into two classes: streaming of stored or live multimedia and real-time interactive multimedia. Obviously, real-time interactive multimedia is not applicable over InterPlaNetary Internet backbone links because of the extremely long propagation delays. However, live or stored media streaming will be a part of
the traffic carried over deep space links. The control for multimedia traffic is an important problem, because uncontrolled multimedia traffic can not only congest the network, but can also cause unfairness and starvation for data traffic.

We focus on the Mars-Earth communication. Multimedia data is first transmitted from the ground station on the Mars surface, which is the source end-point, to Mars orbiters, Mars orbiters then send the data to the satellites orbiting Earth either directly or through the relay station near Mars. The Earth satellites then deliver the multimedia data to the ground station on Earth, which is the sink end-point. Along the communication path, the InterPlaNetary Backbone Network plays a significant role for the performance of the entire deep space communication.

Multimedia transport protocols need to address the challenges posed by the Inter-Planetary Internet Backbone Network, i.e., *extremely long propagation delays, high link errors, asymmetrical bandwidth*, and *blackouts*. In addition to that, they should be able to address the additional challenges because of the unique requirements of multimedia applications, which are summarized as follows:

- **Minimum Bandwidth**: Most multimedia applications require minimum bandwidth to maintain minimum media perception quality. The received media cannot be perceived properly if the bandwidth drops below this threshold.

- **Smooth Traffic**: Abrupt and frequent fluctuations in the media rate can cause significant quality degradation in the received media quality. Consequently, the primary goal of multimedia transport protocols is not to aggressively find and use the available bandwidth, but to maintain a relatively steady media rate while still being responsive to congestions. On the other hand, the InterPlaNetary Internet also requires smooth traffic. The source cannot get the feedback until one RTT later. Because of the very long propagation delay, the congestion control decision based on such past information might not lead to proper
actions. As a result, rapid changes in the transmission rate may lead to serious congestions and packet losses.

- **Error Control**: The multimedia traffic over the InterPlaNetary Internet can be coded in MPEG, motion JPEG, or H.26x. Even though error resilience techniques are adopted in coded video [57], compressed video is still highly sensitive to data loss. The quality of other types of multimedia can also be degraded dramatically if the packet loss rate is high. As a result, error control mechanism must be designed to deal with the packet losses resulting from link errors or congestions in the InterPlaNetary Internet.

Many multimedia transport protocols are proposed to control the flow of multimedia traffic in terrestrial networks [25, 41, 45, 66, 74, 77, 83, 84]. These proposed protocols can be mainly categorized into two types of rate control schemes, i.e., AIMD-based (Additive Increase Multiplicative Decrease) and equation-based.

AIMD-based rate control schemes are TCP-compatible, i.e., they compete fairly with existing TCP by changing the sending rate in such a way similar to that of TCP [25, 74, 84]. The existing AIMD-based rate control schemes [25, 83, 77, 84, 74] are developed based on the assumption that the propagation delay is relatively short. SCTP (Stream Control Transmission Protocol) [83] implements TCP-like mechanisms, such as slowstart, fast retransmit, and fast recovery. SCP (Streaming Control Protocol) [25] is a modified version of TCP that performs TCP-Vegas-like rate adjustment. TEAR (TCP Emulation at Receiver) [77] determines the receiving rates at the receiver based on signals, such as packet arrivals, packet losses, and timeouts. Using these signals, TEAR emulates the TCP flow control functions at the receiver including slow start, fast recovery, and congestion avoidance. RAP (Rate Adaptation Protocol) [74] is a rate-based congestion control mechanism for wired and short distance
networks. RCS (Rate Control Scheme) [84] is a rate control scheme for real-time traffic in networks with high bandwidth-delay products and lossy links. The traditional AIMD-based rate control schemes perform rate increase additively at each RTT and halve the transmission rate in case of packet losses. Since the space links have very long propagation delays, the link may not be fully utilized during additive transmission rate increase with RTT-granularity. Moreover, the traditional AIMD schemes cause abrupt and frequent fluctuations in the media rate in the form of a saw-tooth pattern which is not suitable for most multimedia applications.

The equation-based rate control schemes [41, 45, 66] are proposed to provide relatively smooth congestion control for multimedia traffic in the terrestrial networks. The idea of the equation-based congestion control is to adjust the transmission rate no more than the throughput of the corresponding TCP counterpart with the same packet loss rate, round-trip time, and packet size. TFRC (TCP Friendly Rate Control) [45] is an equation-based rate control scheme which adopts a simple TCP throughput model in its congestion control mechanism. MPEG-TFRCP (TCP Friendly Rate Control Protocol for MPEG-2 Video Transfer) [66] is another equation-based rate control scheme designed for transporting MPEG-2 video in a TCP-friendly manner. Unlike TFRC, TFRCP takes video characteristics into consideration while adjusting its media rate. Although the use of TCP response function ensures that equation-based control schemes competes fairly with TCP over long time scales, the steady-state throughput model of TCP source is highly sensitive to RTT values. Therefore, the equation-based rate control schemes cannot achieve high link utilization and hence are not promising solutions for InterPlaNetary Internet with extremely long propagation delay links.

The throughput performance of SCTP [83], RAP [74], TFRC [45], and RCS [84] are shown in Figure 43 over a 10 Mb/s InterPlaNetary backbone link. The RTT value ranges from 100 secs to 2400 secs including the RTT range for Mars-Earth
communications, i.e., 8.5 to 40 minutes based on the orbital location of the planets. Note that the throughput axis is plotted in logarithmic scale to be able to include the throughput curves of all protocols. The throughput achieved by TFRC [45] and SCTP [83] is below 100 B/s and it is 237 B/s for RAP [74]. Obviously, the throughput is very low and the entire link remains almost unutilized. Although RCS [84] outperforms other schemes, it can only utilize 41 KB/s of the 10 Mb/s link for 40 minutes round-trip time, thus, the performance degradation is still very serious. More detailed performance evaluation is presented in [37]. The reason behind is that SCTP, RAP, TFRC and RCS are sensitive to propagation delays. SCTP is window-based and adopt TCP-like mechanisms. Since TCP protocols perform very badly for very long propagation delays [2], thus, the throughput SCTP achieved is very low. TFRC adopts the steady-state model of TCP, which is very sensitive to the propagation delay, and RAP's slope of linear increase of the transmission rate is inversely related to the propagation delay [74], for the similar reason, they cannot achieve high throughput in InterPlaNetary Internet. For RCS, its initial state cannot address long propagation delay and also it adjusts its transmission rate in RTT-grain, although it outperforms other schemes, but still cannot utilize the link bandwidth.

SCPS Rate-based protocol is proposed for space communication [86], but without a congestion control algorithm. The transmission rate in SCPS Rate-based protocol is defined by the user and also constrained by the receiver buffer size. In other words, SCPS Rate-based protocol does not adapt its transmission rate to the network conditions. Thus, it may cause congestion for InterPlaNetary Internet backbone links if its transmission rate is higher than the available bandwidth.

Besides the rate control schemes mentioned above, layered approaches [75] are proposed for terrestrial networks to minimize the variations in video quality. Many commonly used compression standards, such as MPEG-2, MPEG-4, and H.263 have
extensions for layered coding. Using hierarchical encoding, the source maintains a layered encoded stream, i.e., a base layer and multiple enhanced layers. The rate control is performed by adding or dropping enhancement layers. If more bandwidth becomes available, more enhancement layers are added to improve the video quality. On the other hand, if the available bandwidth decreases, some enhancement layers have to be dropped. In layered approaches, to decode an enhancement layer, it requires that all the lower quality layers have been received successfully. For InterPlaNetary Internet channels, such requirement usually cannot be guaranteed. Furthermore, layered approaches also incur a significant compression penalty as compared to non-layered approaches. Because of this reason, layered approaches are not widely used even in terrestrial networks. Consequently, the layered approaches might not be suitable for InterPlaNetary Internet.

Consequently, the existing rate control schemes cannot address the challenges posed by the InterPlaNetary Internet Backbone Network. New multimedia transport protocols should be proposed in InterPlaNetary Internet to address all the discussed

Figure 43: Throughput performance of existing multimedia rate control schemes for very high RTT ranges.
challenges. In this Chapter, we introduce a rate control protocol, RCP-Planet, for the multimedia traffic in the InterPlaNetary Internet. The objective of RCP-Planet is to address the challenges in the InterPlaNetary Internet to achieve high throughput and satisfy multimedia requirements. To address extremely long propagation delays, RCP-Planet deploys a Begin State algorithm in the first RTT and an Operational State algorithm after one RTT to control the multimedia traffic. A novel rate probing mechanism is proposed to capture the available network bandwidth. Based on the observed rate for a probing sequence, a new rate control scheme is designed to update the media rate smoothly and conservatively to meet multimedia requirements and to reduce the chances of congestions. To address the error control problem for multimedia applications, packet level FEC is adopted to recover lost packets for the multimedia traffic. Moreover, FEC block level ACKs are used to solve the bandwidth asymmetry problem. To reduce the performance degradation caused by blackout conditions, a Blackout State is incorporated into RCP-Planet.

The remainder of the chapter is organized as follows. First, the overview of the RCP-Planet protocol and the Begin State algorithm are presented in Section 5.2, which includes packet level FEC, the rate probing mechanism, and the Begin State algorithm. The Operational State algorithms, including the new rate control scheme, the Blackout State behavior, and bandwidth asymmetry, are explained in Section 5.3.

5.2 RCP-Planet: Begin State

RCP-Planet consists of two states, i.e., Begin State and Operational State, as shown in Figure 44.

RCP-Planet starts a connection in the Begin State at time $t=0$ by calling the Begin_State() algorithm as shown in Figure 49 and goes to the Operational State by calling the Operational_State() algorithm at $t=RTT$, as shown in Figure 50.

During the first RTT, no knowledge of the network is available. In order not to
waste bandwidth for a duration of one RTT, which is extremely long in the InterPlaNetary Backbone link, RCP-Planet determines the initial source sending rate in a conservative and controlled manner so as to reduce the chances of congestions.

Because of the extremely long propagation delay in the InterPlaNetary Backbone Network, retransmissions of multimedia data packets are impossible. To address the error control problem for multimedia applications, the packet level FEC is used for forward error correction. However, the packet loss rate is also unknown in the Begin State. Thus, the most recent history value of the packet loss rate is used to determine the amount of redundancy and extra redundancy are added to address the possible worse network conditions.

To capture the available network bandwidth and increase the transmission rate fast and smoothly, a new rate probing mechanism is introduced in RCP-Planet and is used in both the Begin State and the Operational State. The basic idea is to use the probing sequence to capture the available bandwidth. The length of the probing sequence is chosen appropriately and the rate to send the probing sequence is updated adaptively based on network conditions.

5.2.1 Packet Level FEC

Although multimedia flows are inherently loss-tolerant, error control mechanisms are still necessary to maintain a certain level of success rate in the InterPlaNetary Internet. Typical error control mechanisms fall into two classes, i.e., Automatic Repeat
reQuest (ARQ) and Forward Error Correction (FEC) [72]. ARQ mechanisms are based on the retransmission of lost packets, thus they are not applicable in the Inter-PlaNetary Internet because of extremely long propagation delays. FEC mechanisms are based on the transmission of redundant information along with the original information so that the lost original data can be recovered from the redundant information. Although FEC is an effective mechanism for error control without retransmission, it also incurs overhead.

The packet level FEC has been widely used for communication networks [64, 69]. An important factor for the packet level FEC is the encoding and decoding times. Traditional FEC schemes such as Reed-Solomon codes have rather slow encoding and decoding times, which limit the FEC block size to a very small number [21] and hence result in high FEC overhead. On the other hand, Tornado codes [21] are based on random bipartite graphs and exclusive-or operations, thus, they are orders of magnitude faster than the standard erasure codes, which makes Tornado codes appropriate for the packet level FEC with large FEC block sizes. For example, for data size of 250KB, the encoding and decoding times of Tornado codes can be as small as 0.06 seconds. Although Tornado codes require slightly more encoding packets to reconstruct the original data, this disadvantage is compensated by the larger FEC block size, hence the lower FEC overhead. Furthermore, Tornado codes are simple to implement in practice because they use only exclusive-or operations. Consequently, Tornado codes are adopted for the packet level FEC in RCP-Planet.

The encoding and decoding times for Tornado codes are proportional to \((d + l)ln(1/\epsilon)S\) [22], while for Reed-Solomon, the times are \(dlS\), where \(d\) is the number of original data packets in a FEC block and \(l\) is the number of redundant packets, \(S\) is the packet size, and \(\epsilon\) is the so-called reception overhead. Tornado codes require \(k\) packets to recover the FEC block, where \(k\) is defined as:
\[ k = (1 + \epsilon)d \] (25)

\( \epsilon \) is a very small number around 0.05 [21].

The packet level FEC is used in both the Begin State and the Operational State. The structure of the packet level FEC in RCP-Planet is shown in Figure 45. At the sender side, the data rate set for the application is \( r_m \), which can be obtained either by setting the encoding rate of the encoder or by regulating the output of the encoder and is called the media rate. The data is then encoded into FEC blocks and is transmitted using the source sending rate \( r_s \).

![Figure 45: The Structure of Packet Level FEC.](image)

Obviously, the media rate \( r_m \) and the source sending rate \( r_s \) have the following relationship,

\[ r_s = r_m \frac{n}{d} \] (26)

where \( n \) is the FEC block length and \( d \) is the original data length. If one of these rates is determined, the other can be calculated accordingly.

Whenever the destination receives an original data packet, it passes it to the application layer and keeps one copy in the buffer for decoding in case of losses.
least \( k \) out of \( n \) packets are received, the FEC block can be recovered successfully, thus, the lost original packets can be reconstructed and passed to the application layer. If fewer than \( k \) packets are received, the lost original packets cannot be reconstructed and hence the received media quality is degraded. Furthermore, if the original packets in the FEC block arrive or are reconstructed later than the jitter bound, they are also discarded.

Let us define the probability of receiving at least \( k \) packets out of a group of \( n \) packets as \( P(n,k) \), then,

\[
P(n,k) = \sum_{i=k}^{n} \binom{n}{i} (1 - p)^i p^{n-i}
\]  

(27)

If a FEC block can be recovered successfully, it must receive at least \( k \) out of \( n \) packets, where \( p \) is the packet loss rate. For a given packet loss rate \( p \), to recover a FEC block successfully, the FEC block length \( n \) must satisfy:

\[
P(n,k) > D
\]

(28)

where \( D \) is a constant smaller than but close to 1. Here we choose \( D = 0.999 \). \( n \) can be calculated online by the recurrence relation:

\[
P(n,k) = P(n - 1, k) + \binom{n-1}{k-1} (1 - p)^k p^{n-k}
\]

(29)

with the initial condition

\[
P(k,k) = (1 - p)^k
\]

(30)

For a given \( n \), the overhead \( h \) for the packet level FEC is:

\[
h = \frac{n - d}{n}
\]

(31)
The FEC block length should be chosen appropriately to minimize the FEC overhead $h$. The FEC overhead $h$ for packet loss rates range from $10^{-5}$ to $10^{-1}$ is shown in Figure 46.

![Figure 46: The overhead of FEC vs. packet loss rate $p$.](image)

Obviously, the overhead $h$ decreases with increasing original data length $d$, but $h$ increases with increasing packet loss rate $p$. However, $d$ should not be too large, because larger $d$ results in longer encoding and decoding times. Moreover, $d$ is also constrained by the jitter bound defined multimedia applications. The lost original packets can be recovered and then passed to the application layer only after the entire FEC block is received. If the jitter bound is smaller than the time to transmit the FEC block, part of the data packets in the FEC block will exceed the jitter bound and will be discarded. As a result, $d$ should be chosen appropriately. In our simulations, $d$ is chosen to be 86 packets as discussed in Section 5.4.1.
5.2.2 The Rate Probing Mechanism

Rate probing is a mechanism to measure the observed rate at the receiver to determine the available bandwidth. The probing techniques include one-packet, packet-pair, and multi-packet methods [70]. WTCP [82] sends two back-to-back packets only during connection establishment and use their inter-packet delay as an approximate estimate of the transmission rate. However, two back-to-back packets may not be accurate enough to measure the available bandwidth. This problem can be worse in the backbone links of the InterPlaNetary Internet, because a long period of congestion can occur if the transmission rate, which is set based on the inaccurate information, is higher than the actual available bandwidth. TCP-Real [89] transmits packets in waves, i.e., a number of packets are sent back-to-back. The observed rate at the receiver side is compared with the previous observed rate to update the transmission rate by changing the number of packets in a wave. Since all packets are transmitted in the pattern of waves, i.e., back-to-back, this method might keep creating instantaneous bursty traffic in the network as the wave size increases and hence congest other traffic in the InterPlaNetary Internet. Multimedia traffic usually has a relatively high transmission rate, which leads to a very large wave size and makes the problem worse. As a result, TCP-Real [89] is not suitable for the InterPlaNetary Internet. The TOPP methods [65] extend the packet pair probing technique by sending carefully spaced probing packets rather than back-to-back packets.

Overall, all these methods do not take multimedia requirements into consideration and do not adjust their probing rates based on network conditions, hence, they are not suitable for multimedia traffic in the InterPlaNetary Internet.

In RCP-Planet, we propose a novel rate probing scheme to capture the available network bandwidth. Our rate probing scheme is performed in each FEC block, i.e., for each FEC block, we first send a fixed number of packets, called a probing sequence, at a high rate so-called the probing rate $r_p$. The remaining packets in the FEC block
are sent using the current source sending rate $r_s$, as shown in Figure 47.

![Figure 47: The rate probing scheme.](image)

Probing packets may experience different end-to-end delays and some probing packets can be dropped by the gateway because of the limit of the network bandwidth. As a result, the observed rate $r_o$ for the probing sequence at the receiver is assumed to be the available bandwidth. $r_o$ can be calculated by dividing the number of received probing packets by the duration of the probing sequence measured at the receiver. The receiver reports this information by sending a message packet back to the sender. For convenience, such message packet is also called an ACK here, but it is different from the ACK in TCP protocols. The message packet is the same size as the ACK packet in TCP protocols and it carries the value of $r_o$ and current packet loss rate in the body of the packet.

The probing packet is also a data packet indicated by a flag set in its header. Since the probing packets also carry new information, they need to be recovered if they are dropped. Thus, extra redundant packets are needed to recover the lost probing packet. We assume the probing sequence length, i.e., the number of packets in the probing sequence, is $L$. The number of additional redundant packets is chosen to be $L$.

The probing sequence length $L$ and the probing rate $r_p$ are two design parameters for the rate probing mechanism. If $L$ is too small, the observed rate $r_o$ might not be accurate enough to capture the available bandwidth. On the other hand, if $L$ is too
large, the probing sequence might affect other traffic because it is transmitted usually at a rate higher than the source sending rate. How to choose the appropriate value of \( L \) is discussed in Section 5.4.2.

The probing rate \( r_p \) determines the amount of network bandwidth and it should be chosen appropriately to capture the available bandwidth as fast as possible while avoiding to create burst traffic in the network. In the Begin State, no knowledge of network condition is available, thus, \( r_p \) is set in such a way that it can capture the available bandwidth very fast. If \( r_{m,\max} \) is the maximum media rate defined by the application, the corresponding source sending rate \( r_{s,\max} \) can be obtained from equation (26). Since the multimedia application cannot utilize network bandwidth higher than \( r_{s,\max} \), \( r_p \) is set to be \( r_{s,\max} \) so that it can capture the network bandwidth up to \( r_{s,\max} \).

In the Operational State, the highest probing rate is also set to be \( r_{s,\max} \) so that RCP-Planet only probes the available bandwidth up to \( r_{s,\max} \), which is the highest amount of network bandwidth required by the application. However, the probing rate should be decreased during congestions.

The sender obtains the observed rate \( r_o \) from ACKs. Since the probing sequence length \( L \) is rather small, the random packet losses resulting from link errors only affect the observed rate \( r_o \) slightly. Most probing packet losses are due to congestions. If the available bandwidth is equal to or larger than \( r_p \), no probing packets will be dropped and \( r_o \) only fluctuates slightly because of different queuing delays. During congestions, the gateway drops the packets when the buffer overflows. Because the probing sequence length is relatively short, \( r_o \) fluctuates more dramatically with increasing degree of congestion.

The variance of the observed rate is a good measure of the degree of the congestion and can be used to update the probing rate. Assume \( r_{o,i+1} \) is the current observed rate, its mean value \( E_{o,i+1} \) and the variance \( V_{o,i+1} \) can be estimated recursively,
\[ E_{o,i+1} = \gamma E_{o,i} + (1 - \gamma)r_{o,i+1} \]  
\[ V_{o,i+1}^2 = \gamma V_{o,i}^2 + (1 - \gamma)(r_{o,i+1} - E_{o,i+1})^2 \]

where \( \gamma \) is the forgetting factor and is chosen as 0.95 [93].

To update the probing rate adaptively based on the network condition, the probing rate is decreased from \( r_{s,\text{max}} \) by the amount proportional to \( V_{o,i+1}/E_{o,i+1} \),

\[ \frac{r_{s,\text{max}} - r_{p,i+1}}{r_{s,\text{max}}} = \frac{V_{o,i+1}}{E_{o,i+1}} \]  

Since \( r_{s,\text{max}} = r_{m,\text{max}} n/d \), we have

\[ r_{p,i+1} = r_{m,\text{max}} \frac{n}{d} \left(1 - \frac{V_{o,i+1}}{E_{o,i+1}}\right) \]  

Obviously, when \( V_{o,i+1} \) increases, i.e., the congestion becomes heavier, the probing rate \( r_{p,i+1} \) will be decreased according to equation (35). On the other hand, if there is no congestion, \( V_{o,i+1} \) is close to zero, the probing rate is then set to be \( r_{s,\text{max}} \) so that the media rate can reach the maximum media rate \( r_{m,\text{max}} \).

5.2.3 The Begin State Algorithm

Since no network information is available in the Begin State, it is difficult to determine the initial transmission rate and the number of redundancy for the FEC block. Conservatively, we set the initial media rate \( r_{m,\text{init}} \) to be the minimum media rate \( r_{m,\text{min}} \) required by the application in order not to inject too many packets into the network, because a very long period of congestion can be incurred if the initial transmission rate is higher than the available bandwidth. Usually \( r_{m,\text{min}} \) is very low compared with the maximum media rate, i.e., target media rate, \( r_{m,\text{min}} \) and does not affect other traffic too much. 

117
The packet loss rate is also unknown in the Begin State. The most recent history value $p_h$, which is an approximation of the current packet loss rate, is first used to determine the FEC block length $n$. The procedure to determine the FEC block length is described in Section 5.2.1. However, the actual packet loss rate might not be exactly the same as $p_h$. To address the worse network conditions, a much higher packet loss rate $p_l$ is conservatively chosen to calculate the actual number of redundancy. Assume the corresponding FEC block length is $n'$, then $n'$ is used as the actual FEC block length to encode the data, as shown in Figure 48.

![Figure 48: The FEC block in Begin State.](image)

After determining the FEC block length $n'$, the initial source sending rate $r_{s,\text{init}}$ can be obtained from

$$r_{s,\text{init}} = r_{\text{m, min}} \frac{n'}{d}$$

(36)

The $n' - n$ redundant packets are additional overhead to address the possible worse network condition and they are sent in low priority, because the low priority packets are dropped first during congestions and thus, they do not affect regular data traffic during congestions. The remaining $n$ packets are transmitted as regular data packets.

The low priority packet can be identified by one of the eight bits of the TOS field in the IP header and more recent IP versions, e.g., IPv6 [28], explicitly provide several priority levels [5, 6]. For Mars-Earth communications, the gateway on the Mars surface, Mars orbiters, Earth satellites, and the gateway on the Earth surface can have routing capability and they are assumed to be able to identify low priority packets. If they do not have such capability, the $n' - n$ redundant packets are treated
as regular data packets.

The rate probing mechanism introduced in Section 5.2.2 is used in the Begin State, i.e., a probing sequence is included in each FEC block. The initial probing rate \( r_{p,init} \) is set as \( r_{m, max} n' / d \) as discussed in Section 5.2.2 so as to capture the available bandwidth as fast as possible. The remaining \( n' - L \) packets are transmitted using the rate \( r_{s,init} \).

The Begin State algorithm is summarized in Figure 49.

---

**Figure 49:** The Begin State Algorithm.

### 5.3 RCP-Planet: Operational State

The sender leaves the Begin State for the Operational State at \( t=RTT \) and remains in the Operational State until the connection is terminated. The Operational State consists of three states, i.e., *Increase Rate*, *Decrease Rate*, and *Blackout*, as shown in Figure 44.

In the beginning of the Operational State, the sender goes to Increase Rate state to increase the rate from the minimum media rate set in the Begin State. Based on the rate probing mechanism discussed in Section 5.2.2, a new rate control scheme
is proposed in RCP-Planet to increase the media rate fast and smoothly to address the extremely long propagation delay in the InterPlaNetary backbone link and the smooth traffic requirement for multimedia applications.

During the Operational State operation, the RCP-Planet sender can go to Increase Rate or Decrease Rate state based on the new rate control scheme. RCP-Planet also incorporates the Blackout State into the protocol operation to reduce the throughput degradation resulting from blackouts. Moreover, the bandwidth asymmetry problem is addressed by FEC block level ACKs in the Operational State.

5.3.1 The New Rate Control Scheme

Packet level FEC discussed in Section 5.2.1 is also used in the Operational State to address the error control problem for multimedia applications. The media rate $r_m$ and the source sending rate $r_s$ have relationship defined by equation (26). If one is determined, the other can be calculated accordingly. Thus, there are two ways to update the rate, either first determine $r_m$ and calculate $r_s$ according to equation (26), or vice versa. The first method is preferred, because if we first determine $r_s$, then $r_m$ depends on both $r_s$ and the FEC block length $n$. Since $n$ may change dramatically with the fluctuating packet loss rate $p$, $r_m$ may also fluctuate with $n$ even though $r_s$ is smooth and thus, the smooth traffic requirement for multimedia applications cannot be satisfied. On the other hand, if we update $r_m$ smoothly and then determine $r_s$ accordingly, the received multimedia quality will not be affected even if $r_s$ fluctuates dramatically, because it is only determined by $r_m$. As a result, our rate control scheme is to update $r_m$ first and then calculate $r_s$ according to equation (26).

The RCP-Planet source uses the probing mechanism discussed in Section 5.2.2 to capture the available network bandwidth and updates the probing rate adaptively based on network conditions. Once received the probing sequence, the receiver reports the observed rate $r_o$ to the sender by putting this information into the ACK. Upon
receiving an ACK from the receiver, the sender obtains the current observed rate $r_{o,i+1}$, which reveals the available network bandwidth for RCP-Planet. Obviously, $r_{o,i+1}$ is the current upper bound for the source sending rate and the corresponding upper bound for the media rate, denoted by $r_{a,i+1}$, is calculated from equation (26). If $r_{a,i+1} > r_{m,max}$, then $r_{a,i+1}$ is set to be $r_{m,max}$, i.e.,

$$r_{a,i+1} = \min \left( r_{a,i+1}, \frac{d}{n}, r_{m,max} \right)$$ (37)

If $r_{a,i+1} \geq r_{m,i}$, where $r_{m,i}$ is the current media rate, the network bandwidth is not fully utilized and the media rate should be increased. However, the media rate should not be increased by the amount $(r_{a,i+1} - r_{m,i})$ at once. The reasons are as follows,

- The available bandwidth might be shared by multiple connections, one RCP-Planet connection should not be too aggressive to take all the available bandwidth.
- Multimedia traffic requires smooth change of the media rate.
- The feedback of the rate change is only available after one RTT. Because of the extremely long propagation delay, the media rate should be increased slowly to decrease the chances of congestions in the network.

Because of the above reasons, the extra amount $(r_{a,i+1} - r_{m,i})$ is increased in one RTT linearly with respect to time. Thus, the media rate for the next FEC block $r_{m,i+1}$ is,

$$r_{m,i+1} = r_{m,i} + \frac{r_{a,i+1} - r_{m,i}}{RTT} \Delta t$$ (38)

where $\Delta t$ is the time to transmit the next FEC block and it is equal to $n/r_{s,i+1}$. By combining equation (26) and (38), we have

121
\[ r_{m,i+1} = r_{m,i} + \frac{(r_{a,i+1} - r_{m,i})d}{r_{m,i+1}RTT} \]  

(39)

By solving equation (39), we get

\[ r_{m,i+1} = \frac{1}{2} \left[ r_{m,i} + \sqrt{r_{m,i}^2 + \frac{4d(r_{a,i+1} - r_{m,i})}{RTT}} \right] \]  

(40)

Obviously, \( r_{m,i+1} \) increases very fast, if \( (r_{a,i+1} - r_{m,i}) \) is large. However, \( r_{m,i+1} \) increases slowly, if \( r_{a,i+1} \) is close to \( r_{m,i} \). This is a conservative behavior to reach the available bandwidth. If \( r_{a,i+1} = r_{m,i} \), then \( r_{m,i+1} = r_{m,i} \), i.e., the sender holds its rate after it reaches the available bandwidth.

On the other hand, if \( r_{a,i+1} < r_{m,i} \), i.e., the current media rate is too high, the sender needs to back up and to decrease its media rate. The media rate is then decreased multiplicatively,

\[ r_{m,i+1} = \beta \cdot r_{m,i} \]  

(41)

where \( \beta \) is the rate decreasing factor and \( 0 < \beta < 1 \).

The Operational State algorithm is summarized in Figure 50,

5.3.2 The Blackout State Behavior

To reduce the throughput degradation resulting from blackouts, the Blackout State is developed and incorporated into the Operational State. The sender starts to receive ACKs from the receiver after one RTT time. If it does not receive any ACKs for a certain period \( T_w \), it infers this condition as a blackout and goes to the Blackout State. During blackout, the sender stops sending any packets because power efficiency is critical in the InterPlaNetary Internet.

The receiver also infers blackout after not receiving any packets from the sender for a certain period of time \( T_w \). Then it starts to transmit ACKs with \( r_o = 0 \) and \( p = 1 \) periodically, which are called Zero ACKs. The objective of Zero ACKs is to help
Operational State()
Encode or regulate media using media rate \( r_m \);
Encode FEC block using Tornado Codes;
For each FEC block, first send \( L \) probing packets;
Send the remaining packets using source sending rate \( r_s \);
If (ACK RECEIVED)
   Calculate FEC block length based on \( p \);
   Calculate \( r_a \) from \( r_o \) by (37);
Rate Control:
If \((r_a \geq r_m) / * Increase Rate */
   Update \( r_m \) by (40);
else / * Decrease Rate */
   Update \( r_m \) by (41);
end
Update \( r_s \) by (26);
Probing Rate Adaptation:
Calculate the mean and the variance of \( r_o \) by (32) and (33);
Update the probing rate \( r_p \) by (35);
If (NOACK in \( T_w \) / * Blackout State */
If (current sub-state = \( S_0 \))
   current sub-state= \( S_1 \);
   Stop sending packet;
end;
If (NORMAL ACK RECEIVED)
If (current sub-state = \( S_1 \) / * \( B < 2x \) */
   current sub-state= \( S_2 \);
   Update \( r_m \), \( r_s \) and \( r_p \);
   Send packets using \( r_s \);
end;
If (current sub-state = \( S_3 \))
   current sub-state= \( S_0 \), Blackout State is over;
   Update \( r_m \), \( r_s \) and \( r_p \);
   Send packets using \( r_s \);
end;
end;
If (ZERO ACK RECEIVED)
If (current sub-state = \( S_1 \) / * \( B \geq 2x \) */
   current sub-state= \( S_3 \);
   Send packets using \( r_s \);
end;
If (current sub-state = \( S_2 \) / * \( B > 2x \) */
   current sub-state= \( S_3 \);
   Send packets using \( r_s \);
end;
end;
end;

Figure 50: The Operational State Algorithm.
the sender to capture accurate information regarding the blackout situation and act accordingly. There are two distinct cases observed at the sender side according to the duration of the blackout and its relative distance to the receiver in time. The two cases are explained in Chapter 4 Section 4.3.3.

The operation state diagram of the Blackout state is shown in Figure 51, where state $S_0$ is the period of $t_0$ to $t_1$ in Figure 34 and 35. State $S_1$ is the blackout period and state $S_2$ corresponds to the period $t_2$ to $t_3$ in Figure 34. State $S_3$ is the Zero ACK period.

During $S_1$, the blackout is detected by the timeout mechanism and the sender stops sending packets. Then it goes to either $S_2$ upon receiving an ACK or to $S_3$ upon receiving a Zero ACK. In $S_2$ and $S_3$, the sender resumes sending packets. In $S_2$, the sender either increases or decreases its sending rate based on the feedback information from the ACKs. While in $S_3$, the sender holds its rate. Upon receiving an ACK in $S_3$, the sender goes to $S_0$ and the Blackout State is over.

Consequently, the Blackout State reduces the throughput degradation resulting
from blackout conditions and improves the link utilization for duration of $B$ or $2x$ in the cases $B < 2x$ and $B \geq 2x$, respectively. Furthermore, InterPlaNetary Backbone links usually have intermittent connectivity within a round-trip time period, which can also be addressed by the Blackout State. Since power efficiency is critical for InterPlaNetary Internet, especially on the Mars surface, the sender stops sending any packets after it infers blackout situation and it only resumes transmission after it receives ACKs from the receiver.

5.3.3 Bandwidth Asymmetry

The RCP-Planet receiver needs to send message packets, which include the observed rate for a probing sequence and the packet loss rate of a FEC block, back to the sender so that the sender can adjust its media rate and the amount of FEC redundancy accordingly. Since InterPlaNetary backbone links are usually asymmetrical on the order of 1000 : 1 or more [30], too many message packets can cause congestions in the reverse channel.

In RCP-Planet, the FEC block level ACK is used, i.e., only one ACK is sent for an entire FEC block, which includes the observed rate and the current packet loss rate. If the FEC block size is large enough, the bandwidth asymmetry problem can be solved by the FEC block level ACKs. Delayed ACKs can also be used to further reduce the number of ACKs in the reverse link, i.e., only sends one ACK for a certain number of FEC blocks. In this case, the observed rate and the current packet loss rate are the average values over multiple FEC blocks.

The bandwidth asymmetry factor $f$ is defined to measure the ratio of the traffic in the forward and reverse channels for a RCP-Planet connection, i.e.,

$$f = \frac{N_p S}{N_a A}$$  \hspace{1cm} (42)

where $N_p$ is the number of packets received at the receiver for a period of time, $N_a$
the number of ACKs sent by the receiver for the same duration, $S$ packet size and $A$ ACK size.

$f$ is a measure to illustrate the traffic ratio in the forward and reverse channels for RCP-Planet. If the bandwidth asymmetry is smaller than $f$, RCP-Planet will not cause congestions in the reverse link, i.e., the bandwidth asymmetry problem is solved for the bandwidth asymmetry ratio up to $f$.

### 5.4 Performance Evaluation

We conducted extensive simulation experiments to investigate the performance of RCP-Planet. Still image is considered as data traffic but with some degree of tolerance for packet losses. For audio, the main challenge is to minimize the impact of loss and jitter. Video is more complicated, which includes all the challenges of still image and audio with additional requirements such as smooth traffic. As a result, video traffic is mainly considered in the simulations.

MPEG-4 is used to provide high video quality at relatively low bit rates and has real-time adaptive encoding, which makes it possible to change the source sending rate in reaction to variable network conditions [63]. Hence the MPEG-4 source is chosen for our simulations. The MPEG-4 traffic model is based on the Transform Expand Sample Methodology [63].

How to choose the appropriate probing sequence length $L$ is discussed in Section 5.4.2. The media and source sending rate is illustrated in Section 5.4.3. Throughput performance of RCP-Planet is analyzed along with the overhead, FEC block recovery rate, and fairness in Section 5.4.4, 5.4.5, 5.4.6, and 5.4.7, respectively. Bandwidth asymmetry is discussed in Section 5.4.8. Finally, the blackout performance is analyzed in Section 5.4.9.
5.4.1 Simulation Scenario

The simulation scenario is shown in Figure 52. $N$ RCP-Planet senders on Mars transmit multimedia data to $N$ receivers on Earth. The multimedia data is first transmitted from Gateway B on the Mars surface to the Mars orbiter, then through the InterPlaNetary Backbone Link to the Earth satellite, and finally arrives at the gateway A on the Earth surface. The feedback message is transmitted on the reverse links. The $N$ multimedia data flows are multiplexed in gateway A on the Mars surface and the reverse data flows in gateway B on the Earth surface. Segments in the forward and reverse channels may get lost because of link errors with a probability $p_{\text{loss}}$. If not specified, we assume $N=10$, the gateway buffer size is 200 packets. We also assume that the link capacity is $c=1300$ packets/sec, which is approximately 10 Mb/s for a data packet of size 1 KB.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{simulation_scenario.png}
\caption{The simulation scenario.}
\end{figure}

Furthermore, the minimum media rate is 29.49 KB/s obtained from the MPEG-4 source used in the simulation and the maximum media rate is assumed to be 140 KB/s unless otherwise stated. The receiver buffering time is 10 secs and the jitter bound is 500 ms, which is chosen because the typical response time of a human to the video is 50 ms to 500 ms [64].

In the Begin State, we assume the history value of packet loss rate $p_h$ is $10^{-4}$ and the much larger packet loss rate $p_l=10^{-1}$.

As shown in Figure 46, $d=86$ packets is a suboptimal point, the variance of the FEC overhead for $86 \leq d < 150$ is only 0.7% for $p_l < 10^{-1}$ and 2.8% for $p_l = 10^{-1}$. Since usually $p_l < 10^{-1}$, the FEC overhead cannot be reduced greatly for $d > 86$. 

127
packets. On the other hand, higher $d$ incurs higher encoding and decoding times of
the FEC block and it also requires larger buffering time at the receiver. Considering
the parameters used in the simulations, $d=86$ packets requires at least 8 secs buffering
time at the receiver. If $d$ is larger, the buffering time must also be larger. As a result,
the original data length $d$ is chosen to be 86 packets in the simulations.

The blackout threshold $T_w$ is set to be $4d/r_{m, min}$. Since the end-to-end path for
the InterPlaNetary Internet is not always available, the communication time cannot
last for a very long time. As a result, the simulation time is 10 RTTs in most cases.
Because existing TCP and rate control protocols have very poor performance in the
InterPlaNetary Internet [2, 37], no TCP and other background traffic are included.

### 5.4.2 Probing Sequence Length

Rate probing mechanism presented in Section 5.2.2 is used to capture the available
bandwidth in RCP-Planet, the probing sequence $L$ should be chosen appropriately
to capture the available network bandwidth as accurate and fast as possible and to
reduce its effects on other traffic to the minimum degree.

In RCP-Planet, the observed rate is treated as invalid if the number of the received
probing packets in a probing sequence is less than 4 packets and it will not be used for
updating the media rate, because it is not accurate enough to capture the available
bandwidth. Similar to TCP-Real [89], we only consider $L \geq 4$ packets here.

We use the parameters defined in Section 5.4.1 to investigate how $L$ affects the
overhead, FEC block recovery rate, and throughput. Assume RTT=600 secs, the
packet loss rate because of link errors $p_{loss}=10^{-4}$, and the simulation time is 6000
secs. The results are shown in Figure 53.

The FEC recovery rate $R_{blk}$ is the percentage of FEC blocks that are recovered
successfully and it is defined as

$$R_{blk} = \frac{N_r}{N_t}$$  \hspace{1cm} (43)
Figure 53: Choose the probing sequence length $L$ based on the overhead, FEC block recovery rate, and throughput.

where $N_r$ is the number of FEC blocks recovered successfully and $N_t$ is the total number of received FEC blocks. If a FEC block cannot be recovered successfully, the lost original packets can not be reconstructed and thus, lead to the degradation of the received media quality. As a result, RCP-Planet tries to achieve as high $R_{blk}$ as possible. When $L$ increases from 4 packets, $R_{blk}$ increases from 95% at $L=4$ packets to 99.2% at $L=12$ packets. When $L$ is in the range of 12 to 25 packets, $R_{blk}$ does not increase much because it has already approached 100%.

When $L$ is in the range of 4 to 12 packets, the throughput is approximately constant around 77.3 KBps. For $L > 12$ packets, the throughput starts to decrease because a longer probing sequence is more likely to affect other traffic and may cause congestions, hence, results in lower throughput.

After the FEC block length $n$ is calculated from the current packet loss rate $p$ by equation (29) as discussed in Section 5.2.1, we conservatively add $L$ extra redundant packets to recover the lost probing packets. As a result, the overhead increases with
increasing $L$ from 0.113 at $L=4$ to 0.177 at $L=12$ packets. For $L=30$ packets, the overhead is very high and reaches 0.289.

To achieve good performance, the probing sequence length $L$ should be appropriately chosen such that RCP-Planet has a high FEC block recovery rate, high throughput, and low overhead. We choose $L=12$ packets for all the subsequent simulations. For $L=12$ packets, RCP-Planet has high throughput at 77.255, a high FEC recovery rate at 99.2% and relatively low overhead at 0.177. The probing sequence length 12 matches one of the wave lengths used in the Wave & Probe Communication Mechanism [88].

5.4.3 The Media and Source Sending Rate

We assume the video is encoded in MPEG4 format in reaction to network conditions, i.e., the encoder adaptively set its rate factor so that the media rate $r_m$ is in the range of $r_{m,\text{min}}$ and $r_{m,\text{max}}$, where $r_{m,\text{min}}$ and $r_{m,\text{max}}$ are the minimum and maximum data rates required by the application, respectively. The rate factor for the encoder is set as $r_m / r_{m,\text{min}}$. A video GOP (Group of Pictures) consists of I (intra-coded), P (predictively coded), and B (bidirectionally predictively coded) frames. I, P, B have difference frame sizes and also the frame size of each type is based on the video content. As a result, the actual video output is busty.

To show the behavior of the video output rate, the media rate $r_m$ and the source sending rate $r_s$, we assume RTT=600 secs, the packet loss rate because of link errors $p_{\text{loss}}=10^{-4}$, and the simulation time $= 12000$ secs. Other parameters are the same as defined in Section 5.4.1. The resulting video output, media, and source sending rate are illustrated in Figure 54.

In the Begin State, we conservatively set the media rate as the minimum media rate because no knowledge of the network is available. Thus, during the first RTT period, the media rate $r_m$ and the source sending rate $r_s$ keep constant. After one
Figure 54: The video, media, and source sending rate vs. time.

RTT, the observed rate $r_o$ and the current packet loss rate $p$ become available. Since the upper bound of the media rate $r_a > r_m$, the media rate is increased by equation (40). $r_m$ and $r_s$ increase very fast and $r_m$ reaches a relatively high rate around 80 KBps at $t=2000$, then increases slowly and reaches the highest rate around $t=2400$, i.e., 4 RTTs. Since each RCP-Planet connection increases its media rate during the initial phase of the connection, $r_a$ keeps decreasing. Hence, the speed to increase the media rate also decreases according to equation (40). However, the RCP-Planet still reaches the highest available media rate very fast, i.e., in 3 RTTs, from a relatively low media rate 29 KBps to a much higher rate 117 KBps. As a result, RCP-Planet performs very well in increasing its media rate in the initial phase of the connection.

Since 10 RCP-Planet connections compete for the bandwidth, congestions occur after $t=2400$ secs. The RCP-Planet sender backs up and decreases its media rate during congestions. For multimedia applications, we choose the rate decrease factor to be 0.9 so that the rate will not drop too sharply. After the congestion is over, it starts to increase the media rate again. Consequently, the media rate and the source sending rate are updated according to the network condition, but the media rate still
keeps in the range of 80 KBps to 110 KBps and the corresponding source sending rate is close to the equal share of the bandwidth for each connection, i.e., 130 KBps for a link capacity \( c = 1300 \) KBps shared by 10 RCP-Planet connections. Since \( r_m \) is used to set the rate factor for the MPEG4 encoder, the magnitude of the actual video output rate also varies adaptively with \( r_m \).

Another observation is that the media rate \( r_m \) changes much more smoothly than the source sending rate \( r_s \). This is because \( r_m \) is updated smoothly first in RCP-Planet and then \( r_s \) is calculated by equation (26). Thus, \( r_s \) depends on both \( r_m \) and the FEC block length \( n \). Since \( n \) is updated adaptively based on the current packet loss rate for each FEC block, the degree of fluctuation in \( r_s \) is much higher than that in \( r_m \). However, the fluctuation of \( r_s \) does not affect the received media quality.

5.4.4 Throughput Performance

The Mean Opinion Score (MOS) has been used for years to measure the perceptual quality [54], which relies on the judgement of a group of human observers. On the other hand, the throughput is also a good measure of the multimedia quality. Higher throughput usually means higher media rate \( r_m \), i.e., higher rate factor for the media encoder, and hence higher quality.

We use parameters defined in Section 5.4.1 and assume RTT=300, 600, 1200 secs, respectively. The packet loss rate because of link errors \( p_{\text{loss}} \) is \( 10^{-5} \) to \( 10^{-1} \). The simulation lasts for 10 RTTs, the throughput performance is illustrated in Figure 55. Note that the throughput is calculated only for the original data packets from the application, FEC redundant packets are not included.

RCP-Planet achieves high throughput for different RTT values and packet loss rates because of link errors. The throughput for one individual RCP-Planet connection is around 82 KBps for \( RTT = 300 \) secs, 77 KBps for \( RTT = 600 \) secs, and 73 KBps for \( RTT = 1200 \) secs. Considering FEC redundancy is not counted and the simulation
Figure 55: Throughput vs. packet loss rate resulting from link errors.

time is relatively short because of network constraints, the achieved throughput is high. The reasons that RCP-Planet achieves high throughput can be summarized as follows:

- The rate probing mechanism discussed in Section 5.2.2 is used to capture the available bandwidth.

- The new rate control scheme updates the media rate smoothly and conservatively to address the extremely long propagation delay and the smooth traffic requirement for multimedia applications.

For a given RTT, the throughput decreases slightly for $p_{loss}$ in the range of $10^{-5}$ to $10^{-2}$. For $p_{loss} > 10^{-2}$, the throughput decreases much faster, because more packets are dropped because of link errors. The throughput also decreases when RTT increases, but the degree of degradation is slight. For example, the throughput decreases around 6% when RTT increases from 300 secs to 600 secs, and around 5% from 600 secs to 1200 secs, i.e., the throughput degradation is only about 6% for
increasing RTT twice. This reveals that RCP-Planet is delay-tolerant.

5.4.5 Overhead

We use the same parameters as in Section 5.4.4 for our simulation. The resulting overhead vs. packet loss rate because of link errors is shown in Figure 56. Here the overhead includes all the redundant packets sent in both high and low priority.

![Figure 56: The overhead vs. packet loss rate because of link errors.](image)

Overhead is introduced by the redundant packets to recover packet losses because of link errors and congestions. First, we observe that the overhead is approximately the same for different RTTs and fixed $p_{loss}$. It only increases very slightly for $p_{loss} < 10^{-2}$. On the other hand, the overhead increases with increasing $p_{loss}$. For RTT=600 secs, it increases from 0.171 for $p_{loss}=10^{-5}$ to 0.255 for $p_{loss}=10^{-1}$. The reason is that more redundancy is required to recover packet losses because of link errors.

Since packet level FEC is used in RCP-Planet for the multimedia traffic to recover packet losses because of link errors and congestions, the overhead is about 0.171 even when $p_{loss} = 10^{-5}$. This amount of overhead is mainly introduced by the following
factors:

- Tornado codes require slightly more packets to recover a FEC block.
- In the Begin State, a much higher packet loss rate $p_l$ is chosen to address the possible worse network condition. This conservative method can incur extra overhead if the channel is good.
- Additional redundancy is also introduced to recover packet losses in the probing sequence.

However, this amount of redundancy is quite reasonable for the packet level FEC and is also compensated by the high throughput as discussed in Section 5.4.4 and the high FEC block recovery rate as discussed in Section 5.4.6.

5.4.6 FEC Block Recovery Rate

We use the same parameters as in Section 5.4.4 for our simulation. The resulting FEC block recovery rate vs. packet loss rate because of link errors is shown in Figure 57.

As discussed in Section 5.4.2, RCP-Planet tries to achieve as high $R_{blk}$ as possible. For RTT=600, 1200 secs, $R_{blk}$ is 1 for $p_{loss} \leq 10^{-2}$, but it drops to 0.954 for $p_{loss} = 10^{-1}$ because of high link errors. The high FEC block recovery rates that RCP-Planet achieves are mainly resulting from two reasons, the first one is the smooth update of the media rate, which leads to less congestions and hence less packet losses, and the other is the way to choose the amount of FEC redundancy as discussed in Section 5.4.5. On the other hand, $R_{blk}$ almost does not change with RTT increasing from 600 secs to 1200 secs. This also illustrates that RCP-Planet is delay-tolerant.

5.4.7 Fairness

To the best of our knowledge, no existing rate control scheme has been proposed for multimedia applications in the InterPlaNetary Internet so far, thus, we only consider
Figure 57: The FEC block recovery rate vs. packet loss rate because of link errors.

the homogeneous fairness here, i.e., the fairness of the 10 RCP-Planet connections.

As described in [76], the fairness index based on throughput for a bottleneck link is defined in Equation 10. $FI$ always lies between $1/N$ (indicating one of them gets all the bandwidth and all others starve) and 1 (indicating all get an equal share of the bandwidth).

The same parameters in Section 5.4.4 are used in the simulation. $RTT = 600$ secs. To evaluate the fairness of RCP-Planet, we consider the following scenarios:

- **Case I:** For all connections, $r_{m,\text{min}} = 29.49$ KBps and $r_{m,\text{max}} = 130$ KBps. All connections start at time $t = 0$.

- **Case II:** For all connections, $r_{m,\text{min}} = 29.49$ KBps, but the target rate $r_{m,\text{max}}$ is in the range of 120 KBps to 210 KBps, i.e., for the $i$-th connection, $r_{m,\text{max}} = 120 + 10(i - 1)$ KBps, $i=1, 2, \ldots, 10$. All connections start at time $t = 0$.

- **Case III:** For all connections, $r_{m,\text{max}} = 130$ KBps, $r_{m,\text{min}} = 29.49$ KBps for 5
connections and $r_{m,\text{min}} = 58.98$ KBps for the other 5 connections. All connections start at time $t = 0$.

- **Case IV:** For all connections, $r_{m,\text{min}} = 29.49$ KBps and $r_{m,\text{max}} = 130$ KBps. 5 connections start at time $t = 0$ and the other 5 connections start at $t = 3\times RTT$, i.e., $t = 1800$ secs.

Case II and III are used to evaluate the fairness for RCP-Planet connections with different application requirements. Case IV is used to evaluate the fairness for late join RCP-Planet flows. In case IV, we calculate the throughput over the last 4 RTTs where all connections reach the steady state. The resulting fairness vs. packet loss rate because of link errors is shown in Figure 58.

![Figure 58: The fairness vs. packet loss rate because of link errors.](image)

The resulting fairness in Figure 58 shows that the fairness for case I, III, and IV is approximately 1 for different packet loss rates because of link errors and 0.98 for case II for different packet loss rates because of link errors. Consequently, RCP-Planet connection always shares the available network bandwidth equally for different application requirements and late join flows.
5.4.8 Bandwidth Asymmetry Factor

Bandwidth asymmetry factor $f$ as defined by equation (42) is introduced in Section 5.3.3 to measure the ratio of the traffic in the forward and reverse channels for a RCP-Planet connection. The bandwidth asymmetry problem is addressed by block level ACKs in RCP-Planet. If the bandwidth asymmetry is smaller than $f$, RCP-Planet will not cause congestions in the reverse link, i.e., the bandwidth asymmetry problem is solved for the bandwidth asymmetry ratio up to $f$.

The same parameters in Section 5.4.4 are also used in the simulation. The resulting bandwidth asymmetry factor vs. packet loss rate because of link errors is shown in Figure 59.

![Figure 59: The bandwidth asymmetry factor vs. packet loss rate because of link errors.](image)

For RTT=300, 600, 1200 secs, $f$ remains approximately constant at 2600 for different $p_{\text{loss}}$. Overall, $f$ is equal to or higher than 2600 for different RTTs and different $p_{\text{loss}}$ values. This means that RCP-Planet solves the bandwidth asymmetry up to 2600 : 1 by using FEC block level ACKs. Since the asymmetry in the bandwidth
capacity of forward and reverse channels is typically on the order of $1000 : 1$ in space missions [30], the bandwidth asymmetry ratio $2600 : 1$ is quite high for InterPlaNetary Internet links, hence, RCP-Planet works well in the InterPlaNetary Internet with high bandwidth asymmetry. Furthermore, delayed ACKs can also be used to further reduce the number of ACKs in the reverse link as discussed in Section 5.3.3.

5.4.9 The Blackout Performance

When a blackout is detected, RCP-Planet moves to the Blackout State as shown in Figure 44 to reduce its effect on the throughput performance as explained in Section 5.3.2. Throughput achieved by RCP-Planet for different blackout durations is shown in Figure 60, where RTT=600 secs, $p_{loss}=10^{-4}$, and the blackout occurs at a position 150 seconds away from the receiver at time $t=2000$ secs. Throughput is used to measure the performance of RCP-Planet in blackout conditions. To eliminate the effect of congestions on the throughput performance, only one RCP-Planet connection is used in the simulation. The simulation time is 4000 secs and the other parameters are the same as in Section 5.4.1.

![Figure 60: The throughput vs. blackout duration.](image-url)
To investigate RCP-Planet performance in blackout conditions, we consider two versions of the RCP-Planet:

- **RCP-Planet V1**: This version incorporates the Blackout State.

- **RCP-Planet V2**: This version does not incorporate the Blackout State. It uses the link-probing scheme introduced in SCPS-TP [81] to detect when the blackout is over.

In the link-probing scheme, when the sender detects that a blackout occurs, it sends link-probing segments periodically to the receiver. Upon receiving a link-probing segment, the receiver sends an ACK immediately back to the sender. Once received an ACK for the link-probing segment, the sender infers that the blackout is over and resumes the transmission.

Although the throughputs of RCP-Planet V1 and V2 both decrease with increasing blackout duration $B$, RCP-Planet V1 always outperforms RCP-Planet V2. For $B < 300$ secs, i.e., $B < 2x$, the throughput difference between RCP-Planet V1 and V2 increases with increasing $B$. For example, the throughput difference is 0.46 KBps at $B = 20$ secs, but goes up to 6.35 KBps at $B = 300$ secs, which agrees with the analysis in Section 5.3.2, i.e., the gain of the blackout state is $B$ for $B < 2x$. Since $B$ increases, the gain also increases.

For $B \geq 300$ secs, i.e., $B \geq 2x$, the throughput difference between RCP-Planet V1 and V2 remains approximately constant. For example, the throughput difference is 6.48 KBps at $B = 400$ secs, 6.59 KBps at $B = 500$ secs, and 6.77 KBps at $B = 600$ secs. This also matches our conclusion in Section 5.3.2, i.e., the gain of the blackout state is $2x$ for $B \geq 2x$. Thus, the gain remains the same for $B \geq 2x$. 

140
CHAPTER VI

CONCLUSIONS AND FUTURE RESEARCH WORK

6.1 Summary of Research Results

The research work in this thesis was focused on the development of transport protocols for satellite IP networks and the InterPlaNetary Internet. Research contributions have been made in the following areas:

- Unicast transport protocol for satellite IP networks
- Reliable multicast transport protocol for satellite IP networks
- Reliable data transport protocol for the InterPlaNetary Internet
- Multimedia transport protocol for the InterPlaNetary Internet

6.1.1 Unicast transport protocol for satellite IP networks

In Chapter 2, a new congestion control protocol, TCP-Peach++, was proposed for satellite IP networks [6, 35]. A new type of low priority segment, NIL segment, is used to probe the availability of network resources as well as error recovery. Two new algorithms, Jump Start and Quick Recovery, are adopted in TCP-Peach++ to recover rapidly from multiple segment losses within one window of data. The delayed SACK scheme is adopted to address the bandwidth asymmetry problems and a Hold State is developed to address persistent fades.

Simulation results show that the performance of TCP-Peach+ is better than TCP-Peach and TCP-Peach++ improves the throughput performance during rain fades and
addresses the bandwidth asymmetry problems.

6.1.2 Reliable multicast transport protocol for satellite IP networks

In Chapter 3, a reliable multicast transport protocol, TCP-Peachtree, was proposed for satellite IP networks [8]. In addition to the acknowledgment implosion and scalability problems in terrestrial wirelined networks, satellite multicasting has additional problems, i.e., different multicast topology, different type of congestion control problems, and low bandwidth feedback link. TCP-Peachtree used a modified B+ tree-like hierarchical multicast group to solve the acknowledgment implosion and scalability problems in reliable IP multicast applications. Local error recovery and ACK aggregations were performed within each subgroup and also via logical subgroups. To avoid the overall performance degradation caused by some worst receivers, a local relay scheme was designed. The two new congestion control algorithms were adopted, i.e., Jump Start and Quick Recovery, so that TCP-Peachtree was suitable for satellite IP networks with long propagation delays and high bit error rates. NIL segments were used to exploit the availability of network resources and recover lost packets on the receiver side. The delayed SACK scheme was adopted to address the bandwidth asymmetry problems. Furthermore, a Hold State was developed to address persistent fades.

Simulation results show that the congestion control algorithms in TCP-Peachtree perform better than that of the TCP-NewReno and improve the throughput performance during rain fades. It is also shown that TCP-Peachtree achieves fairness and has high scalability.

6.1.3 Reliable data transport protocol for the InterPlaNetary Internet

In Chapter 4, a reliable data transport protocol, TP-Planet, was proposed for Inter-PlaNetary Internet [3]. The objective of TP-Planet is to address the challenges posed by the InterPlaNetary Backbone Network for reliable data delivery and achieve high
throughput performance. TP-Planet replaces the inefficient slow start algorithms with a novel Initial State algorithm, which captures link resources in a very fast and controlled manner. To address the challenges because of extremely high propagation delay, TP-Planet deploys 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 decisions from single packet loss events, is developed to minimize the erroneous congestion decisions because of high link errors.

Consequently, TP-Planet improves the throughput with a factor more than $10^3$ compared to current TCP protocols. To reduce the effects of blackout conditions on the throughput performance, TP-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, TP-Planet is a reliable transport protocol equipped with diverse set of algorithms and functionalities, which can address the requirements of the InterPlaNetary Backbone Network.

Note that TP-Planet is mainly implemented at Interplanetary Backbone Network nodes, i.e., the TP-Planet source and sink are the ground station gateway at the Earth and the planetary gateway connected to the relay satellites orbiting around the outer-space planets. The end-to-end transport control can be achieved by using the existing transport protocols developed for terrestrial wireless networks on the PlaNetary Surface Networks in conjunction with TP-Planet on the InterPlaNetary Backbone Network. However, the detailed description of such cooperation and its performance evaluation are beyond the scope of this thesis and left for future study.
6.1.4 Multimedia transport protocol for the InterPlaNetary Internet

In Chapter 5, a multimedia transport protocol, RCP-Planet, was proposed for InterPlaNetary Internet [36]. Multimedia traffic is a part of the aggregate traffic over InterPlaNetary Internet backbone links, which includes planet images and multimedia data from some scientific observations. Existing rate control schemes cannot solve the rate control problem in InterPlaNetary Internet which is characterized by extremely long propagation delays, high link errors, asymmetrical bandwidth, and blackouts. Furthermore, multimedia traffic has additional requirements such as minimum bandwidth, smooth traffic and error control.

Two novel algorithms, Begin State and Operational State, were designed. A novel rate probing mechanism was proposed to capture the available bandwidth. Based on the rate probing mechanism, the new rate control scheme updated the media rate smoothly and conservatively in the Operational State. To recover packet losses because of link errors and congestions, Tornado codes were used for packet level FEC because of their very fast encoding and decoding times. The FEC block length was chosen appropriately to minimize the FEC overhead. Furthermore, FEC block level ACKs were used to address bandwidth asymmetry problems. Apart from that, the blackout state was incorporated into RCP-Planet to improve the performance in blackout conditions.

Simulation experiments show that RCP-Planet reaches the available media rate fast and smoothly using the rate probing mechanism and the new rate control scheme. It achieves high throughput and FEC block recovery rate with reasonable overhead. Multiple RCP-Planet connections can share the available bandwidth equally. The Blackout State in RCP-Planet always outperforms the link-probing scheme introduced in SCPS-TP [81]. Moreover, the simulation results also reveal that RCP-Planet is delay-tolerant. As a result, RCP-Planet is a multimedia transport protocol with
diverse set of algorithms and functionalities, which addresses the challenges of multimedia rate control in InterPlaNetary Internet.

6.2 Future Research Work

6.2.1 Application-Level Multicast Protocol Built on Peer-to-Peer Overlays

IP Multicasting provides an efficient way to disseminate data from a sender to a group of receivers. However, the difficulties in upgrading the existing network routers and software prevent Internet Service Providers (ISPs) from deploying and operating wide scale multicast networks in terrestrial networks. These difficulties can be overcome by using application-level multicast protocols built on Peer-to-Peer overlays since the protocols do not require the multicast capability in the routers. The user can use the multicast application software to join a multicast session. The upgrade of the software can be done by the users themselves.

Since users become online or off-line frequently in peer-to-peer networks. The multicast protocols that are based on a physical multicast tree will have difficulties to address the dynamic changes in such an environment. However, the hierarchical logical group introduced in Chapter 3 can be adopted into application-level multicast protocols to address these challenges. But further research must be done to make the DR selection procedure more robust to dynamic changes in peer-to-peer networks.

6.2.2 Cooperative Problem-Solving Framework for Delay-Tolerant Networks

Although the end-to-end path is possible for Mars-Earth communications, InterPlaNetary Internet is mainly characterized by intermittent connectivity and hence an end-to-end path may not exist for more general scenarios [20]. As a result, the bundling
protocols [31, 34, 80] are proposed to deliver information in the InterPlaNetary Internet. Even though the communication resources are highly scheduled in the InterPlaNetary Internet, congestion can still occur because of unexpected link failures and data bursts from opportunistic contacts. Hence congestion and flow control at the bundle layer are important issues for the InterPlaNetary Internet. The main challenge of congestion and flow control at the bundle layer is how to manage the buffer space in a node, since bundles received at a node consume permanent storage and generally cannot be discarded safely because of the natural of custodial transfer.

Because of the distributed nature of delay-tolerant networks, cooperative flow control schemes can be used to solve the flow control problem in the InterPlaNetary Internet. The local information at a node is attached into the bundle header and is passed to its neighboring nodes. Based on the partial global information, each node constructs its own partial global graph and a rolling horizon algorithm is adopted to schedule the flow in the partial global graph and to address the dynamic changes in the network. Neighboring nodes can work cooperatively to determine the actual flow over an edge.

6.2.3 Cross Layer Optimization

Because of the scarcity of the power and processing resources at the planetary distant communication nodes, the cross-layer optimization is an essential direction to pursue. The cross-layer optimization for transport layer protocols should be researched to achieve highest efficiency in resource utilization in the extreme networking environment such as in the InterPlaNetary Internet. For example, the link information, which is available to lower layers, should be exploited to the maximum to achieve resource-efficient reliable data and multimedia transport throughout the InterPlaNetary Internet.
REFERENCES


VITA

Jian Fang was born in Anhui province, P. R. China. He received the B.S., M.S., and Ph.D. degrees from Shanghai Jiao Tong University, Shanghai, China. He was a Research Associate in the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, in 1996. After that, he became a Lecturer at Shanghai Jiao Tong University. He was a Ph.D. student in the department of Computer Science and Engineering at the university of Texas at Arlington in 2000. He joined the Broadband and Wireless Networking Laboratory at Georgia Institute of Technology in 2001. He received the M.S. and Ph.D. degrees in May 2003 and December 2004, respectively, in the department of Electrical and Computer Engineering at Georgia Institute of Technology.