DEMOCRITUS UNIVERSITY OF THRACE

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

SOFTWARE AND APPLICATION DEVELOPMENT SECTOR

ACCURATE ESTIMATION OF END-TO-END DELIVERY DELAY IN SPACE INTERNETS: PROTOCOL DESIGN AND IMPLEMENTATION

Nikolaos Bezirgiannidis

PhD Thesis

Xanthi, July 2015

Acknowledgements

With the completion of the present thesis, I would like to thank all the people that stood beside me throughout this period.

First and foremost I offer my sincerest gratitude to my supervisor, Prof. Vassilis Tsaoussidis, for his support, guidance, and plenty of opportunities he gave me in all the time of research and writing of this thesis. His positive attitude and moral integrity have greatly inspired me and assisted me in both a professional and personal level.

I would like also to thank the other members of my committee, Prof. Pavlos Efraimidis and Prof. Nikolaos Avouris for the fruitful comments and suggestions in improving this thesis.

I would like to express my gratitude to Mr. Scott Burleigh for the productive collaboration during my research visit in NASA's Jet Propulsion Laboratory. I am grateful to have worked with such a great researcher with unique engineering skills. I would like to thank Dr. Felix Flentge for his guidance and supervision in my research visit in ESA's European Space Operations Center. I am also thankful to Prof. Carlo Caini for the great teamwork during our recent collaboration.

During this journey, I have been blessed with a friendly and helpful group of colleagues, who have assisted me in many different ways throughout this period. My sincerest appreciation goes to Giorgos Papastergiou, Sotiris Diamantopoulos, Ioannis Komnios, Fani Tsapeli, Sotiris Lenas, Ioannis Alexiadis, Christos Samaras, Christina Malliou, and Agapi Papakonstantinou; I couldn't have asked for a better working environment.

Finally, I would like to express a deep sense of gratitude to my family, for their constant and unconditional support throughout my PhD studies and life in general.

Abstract

In this thesis, we study the issue of estimating end-to-end delivery delays for data transmissions in space internets. We provide solutions that deal with the challenging nature of space communications and improve the delay estimation for different network conditions. To achieve that, we leverage Delay/Disruption Tolerant Networking (DTN), an architecture that has recently emerged to interconnect space assets into the Interplanetary Internet (IPN). By exploiting the basic principles of the DTN architecture and studying the challenged space network conditions, we model the different components of the end-to-end latency, and advance the awareness of the network as well as its inherent ability to accurately predict them. In particular, we deal not only with the deterministic components of total latency (e.g., signal propagation delay for a given pair of space assets, transmission delays through links with given data rates, waiting times for scheduled communication establishment, etc.), but also with the probabilistic parameters that pertain to the data transmissions (e.g., queueing delays induced by cross-traffic data backlog, retransmission delays due to lost or corrupted data, etc.). We then exploit the improved latency prediction functions in real-time network operations: we enhance network routing ability to capture the end-to-end path that data items will follow from source to destination, and, by developing algorithms and protocols that improve routing capability in estimating delivery times at destination, we boost overall routing efficiency. Furthermore, we leverage the achieved accuracy in computing round-trip times and the corresponding maximum limits, for computing dynamic retransmission timeout intervals for end-to-end transport protocols.

Initially, we study in an analytical way the plausible delivery times of a data unit at destination. We design a technique that leverages management statistics to construct time series on the error rates, and uses a forecasting procedure to predict future error rates. Based on the extracted forecasts and the protocol retransmission procedures, we provide analytical methods to obtain the retransmission probabilities and corresponding delays, accordingly. We also exploit information on the network connectivity, as well as links' and data units' parameters, to extract, for a given data unit, a probabilistic delivery latency profile, which comprises a list of possible arrival times at destination along with the corresponding probabilities. We implement the Bundle Delivery Time Estimation (BDTE) tool that realizes the proposed analytical methods, and incorporate it into ION implementation. Validation experiments show that it can efficiently provide delivery latency profiles, in an accurate way, and thus constitutes a useful tool for administrative purposes.

We then focus on the queueing component of the total delivery delay, and introduce two different approaches to estimate it. In the first approach, we encode the queueing delay

component and incorporate it as a distinct element of the network connectivity plan, which we name Earliest Transmission Opportunity (ETO). In order to exploit the obtained information, we propose an enhancement to the Contact Graph Routing (CGR) algorithm, namely CGR-ETO, to incorporate backlog information into routing decisions. We pair the introduced algorithm with an update protocol, namely Contact Plan Update Protocol (CPUP), which implements the dissemination of contact plan changes. This way, information on increased queue backlogs is disseminated through the network with CPUP messages, and, hence, network nodes' inherent capability to calculate the corresponding delays for data transmissions is enhanced. In the second approach, we propose a proactive framework for estimating queueing delays through network statistics procedures and time series forecasting. We propose that network nodes extract queueing rate measurements in regular time intervals, and disseminate them to other network nodes using the CPUP dissemination mechanism. The obtained measurements are then stored in the contact plan, composing different time series between each pair of network nodes. The available time series information are then used to forecast future queueing rates, and the predictions are combined with the contact plan schedules to estimate the queueing delay for the data units to be transmitted. This way, the proposed estimation of the overall, end-to-end delivery delay incorporates the obtained forecasts of future queueing delays, and therefore can more accurately match the actual delays experienced in a congested network. Evaluation shows that both approaches can efficiently estimate the queueing delays and, hence, can provide more accurate predictions of total end-to-end delays. Furthermore, we observe, through both simulation and emulation studies, that the proposed CGR-ETO routing algorithm, based in its improved delay estimations, can improve routing decisions, and provide basic functionality of load balancing, as well as a way of proactively controlling the congestion that is observed with the capacity exhaustion of transmission windows.

In the last part of this thesis, we exploit the introduced analytical methods and algorithms to improve the transport layer's capability of estimating RTTs, and to enhance the efficiency of the end-to-end retransmission mechanism. To this end, we develop a novel, dynamic end-to-end retransmission framework that takes into account cross-layer information to estimate the major latency components, and combines them to calculate efficient retransmission timeout intervals using the maximum -within some boundaries- expected end-to-end delay, based on the worst-case network conditions that may be experienced on the routing paths. We develop the introduced framework as extension of Delay-Tolerant Payload Conditioning (DTPC) transport protocol, and incorporate it into ION DTN implementation. Emulation studies show that the advanced, more accurate RTT estimator provides faster retransmission of lost data, and significant reduction in the overall data transmission time, while keeping at the same time the overhead (due to duplicate transmissions) minimum. Finally, by achieving faster retransmission of lost data, the dynamic framework provides great reduction of the storage occupancy and

utilization, primarily at destination node, when the in-order delivery feature of DTPC protocol is applied.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ACF	Auto-Correlation Function
ACK	Acknowledgement
ADU	Application Data Unit
AMS	Asynchronous Message Service
ANP	Average Number of Packets
AOS	Advanced Orbiting Systems
ARIMA	Autoregressive Integrated Moving Average
ARMA	Autoregressive Moving Average
ARQ	Automatic Repeat reQuest
ASL	Aggregation Size Limit
ATL	Aggregation Time Limit
ATR	Average Transmission Rounds
BDD	Bundle Delivery Delay
BDDPredAcc	Bundle Delivery Delay Prediction Accuracy
BDDPredErr	Bundle Delivery Delay Prediction Error
BDT	Bundle Delivery Time
BDTE	Bundle Delivery Time Estimation
BER	Bit Error Rate
BP	Bundle Protocol
BSP	Bundle Security Protocol
BSS	Bundle Streaming Service
CARPOOL	Connectivity Plan Routing Protocol
CCSDS	Consultative Committee for Space Data Systems
CDF	Cumulative Distribution Function
CFDP	CCSDS File Delivery Protocol
CGR	Contact Graph Routing
CGR-ETO	Contact Graph Routing with Earliest Transmission Opportunity
CL	Convergence Layer
CP	Checkpoint
CPUP	Contact Plan Update Protocol
CSMA-CD	Carrier Sense Multiple Access with Collision Detection
DB	Database
DCCP	Datagram Congestion Control Protocol

DEN	DTN Engineering Network
DINET	Deep Impact Network Experiment
DNS	Domain Name System
DRTS	Data Relay Test Satellite
DSN	Deep Space Network
DTLSR	Delay-Tolerant Link State Routing
DTN	Delay/Disruption Tolerant Networking
DTNMP	Delay/Disruption Tolerant Network Management Protocol
DTNRG	Delay Tolerant Networking Research Group
DTNWG	Delay Tolerant Networking Working Group
DTPC	Delay Tolerant Payload Conditioning
DTPC-dRTO	DTPC with dynamic Retransmission Timeout
DTPC-sRTO	DTPC with static Retransmission Timeout
ECC	Estimated Capacity Consumption
ECGR	Enhanced Contact Graph Routing
ECTP	Erasure-Coding Transport Protocol
EID	Endpoint Identifier
EMA	Exponential Moving Average
EOB	End of Block
ESA	European Space Agency
ETO	Earliest Transmission Opportunity
FEC	Forward Error Correction
GEO	Geostationary Earth Orbit
GEO CC	Geostationary Earth Orbit Control Center
GS	Ground Station
IETF	Internet Engineering Task Force
IOAG	Interagency Operations Advisory Group
ION	Interplanetary Overlay Network
IP	Internet Protocol
IPN	Interplanetary Internet
IRTF	Internet Research Task Force
ISS	International Space Station
JAXA	Japan Aerospace Exploration Agency
JPL	Jet Propulsion Laboratory
LDPC	Low-Density Parity-Check
LEO	Low Earth Orbit

LEO CC	Low Earth Orbit Control Center
LOS	Line-of-Sight
LTP	Licklider Transmission Protocol
MBS	Mean Block Size
MEO	Medium Earth Orbit
METERON	Multi-purpose, End-To-End Robotic Operations Network
MOC	Mission Operations Center
MPL	Mean Packet Length
MRO	Mars Reconnaissance Orbiter
NAK	Negative Acknowledgement
NASA	National Aeronautics and Space Administration
NetEm	Network Emulator
ONE	Opportunistic Network Environment
OWLT	One-Way-Light-Time
PDU	Protocol Data Unit
PER	Packet Error Rate
PRoPHET	Probabilistic Routing Protocol using. History of Encounters and Transitivity
PSS	Portable Satellite Simulator
QoS	Quality of Service
RA	Report Acknowledgement
RS	Report Segment
RTO	Retransmission Timeout
RTT	Round-Trip Time
SCPS-TP	Space Communications Protocol Standards-Transport Protocol
SDNV	Self-Delimiting Numeric Value
SISG	Space Internetworking Strategy Group
SPICE	Space Internetworking Center
SPP	Space Packet Protocol
SSI	Solar System Internet
STK	Satellite Toolkit
TC	Telecommand
TCP	Transmission Control Protocol
TM	Telemetry
TPList	Time-Probability List
TTL	Time-to-Live
UCL	Underwater Convergence Layer

UDP	User Datagram Protocol
UK-DMC	United Kingdom Disaster Monitoring Constellation
URI	Uniform Resource Identifier

Chapter 1 Introduction

1.1 Thesis Description

In this thesis, we study the issue of estimating end-to-end delivery delay for data transmissions in space internets. We provide solutions that deal with the challenging nature of space communications and improve the delay estimation for different network conditions. To achieve that, we leverage Delay/Disruption Tolerant Networking (DTN), an architecture that has recently emerged to interconnect space assets into the Interplanetary Internet (IPN). By exploiting the basic principles of the DTN architecture and studying the challenged network conditions in space, we model the different components of the end-to-end latency, and advance the awareness of the network as well as its inherent ability to accurately predict them. In particular, we deal not only with the deterministic components of total latency (e.g., signal propagation delay for a given pair of space assets, transmission delays through links with given data rates, waiting times for scheduled communication establishment, etc.), but also with the probabilistic parameters that pertain to the data transmissions (e.g., queueing delays induced by cross-traffic data backlog, retransmission delays due to lost or corrupted data, etc.). We then exploit the improved latency prediction functions in real-time network operations: we enhance network routing ability to capture the end-to-end path that data items will follow from source to destination, and, by developing algorithms and protocols that improve routing capability to estimate delivery times at destination, we boost overall routing efficacy. Finally, we leverage the achieved accuracy in computing round-trip times (RTTs) and the corresponding maximum limits for computing dynamic retransmission timeout (RTO) intervals for end-to-end transport protocols, hence providing an efficient, dynamic retransmission framework for the transport layer over intermittent and scheduled networks such as the IPN.

Our ultimate intention is to enhance the space networking efficiency in estimating the end-to-end path that a data unit is expected to follow from source to destination node and the corresponding delays that it may encounter during the transmission over that path, and to provide the network with an inherent functionality to forecast the overall time interval required for that data unit to reach its destination.

1.2 Context and Motivation

Our research focuses on the upcoming era of space communications, which will signify the transition from static and segregated mission communications, to a more dynamic, unified, and

internetworked model. The importance of space internetworking is twofold. Firstly, it allows for better exploitation of network resources, which in turn allows engineers to communicate with space assets in an easier and safer way. Secondly, it designates a new paradigm in space communications, where interoperability, interagency communication, and unification of space and terrestrial networking are feasible. Along these lines, the IPN concept was introduced [1] and a research group was established [2] to work towards its realization. The concept was embraced by the majority of space agencies, and, consequently, the Space Internetworking Strategy Group (SISG) was formed [3] to work towards a network-centric Solar System Internet (SSI) [4], which will connect spacecraft from different missions that belong to different agencies or space operators, and will interoperate with terrestrial networks and planetary internets.

In this context, DTN architecture [5] provides an ideal networking paradigm to interconnect diverse environments and thus realize the IPN concept. DTN has gradually evolved since it first appeared: a variety of protocols have been proposed to deal with challenging network conditions (e.g., long delays, disruptions, data losses, etc.), in different layers of the DTN protocol stack. Bundle Protocol (BP [6]) has been adopted as the overlay network protocol, with the potential to unify different internetworks under a global layer, in the same way IP connects Internet regions across the globe. BP interfaces with different underlying Convergence Layer (CL) protocols, in order to transfer data at each hop from source to destination. Different CL protocols include traditional Internet protocols like Transmission Control Protocol (TCP), User Datagram Protocol (UDP), Ethernet, Bluetooth, as well as protocols designed to operate in specific network conditions, such as the Licklider Transmission Protocol (LTP [7]), which was primarily designed to support data transmissions over long-haul space links. BP and the underlying CL protocols establish an architectural design model of hop-by-hop data transmissions, principally originated by the disruptive nature of DTN communications. Delay Tolerant Payload Conditioning (DTPC) [8] was lately introduced as an end-to-end protocol to support end-to-end services and functionalities missing from the aforementioned DTN architectural model, including end-to-end application-layer reliability as a safety-net for the underlying layers' reliability, in-order data delivery, duplicate suppression, etc., and thus complement the network services offered by BP, in an end-to-end, transport-layer fashion.

Along these lines, the proper coordination of the DTN protocol stack, as well as the required network functions and planning operations in the SSI, necessitate a set of requirements; among those requirements, the operations concept defines two time restrictions as necessary elements for the SSI functionality, namely *Timeliness* and *Predictability* [4]: The former objective states that the network shall allow timely delivery of data, as a user requirement, and that users will need to know the predicted epoch by which a given forward product will reach the destination node. The latter principle represents the ability to identify all components' latency and the

resulting earliest as well as latest physical delivery times under normal conditions of SSI network operation. Practically, those objectives call for methods and protocols to accurately and efficiently estimate delivery paths and delivery timeline and complement the operation of the aforementioned DTN protocols. Our work is motivated by those principles: we introduce techniques and protocols for accurate predictions of the destination delivery times of data units; we enhance network routing's ability to capture the data path to destination; and we provide the framework to calculate delivery intervals constrained within some minimum and maximum limits, in various network conditions.

Notwithstanding the fact that delivery time estimation is a core requirement for the space communications operations, it is a challenging task in a space internet that has not been tackled efficiently yet. In contrast to the traditional space data transmission model, data in the SSI are not forwarded through a single, dedicated communication channel. The intermission and interagency operational concept will provide different routes for data delivery that are dynamically selected and might include multipath, parallel transmissions, employing a functionality similar to the Internet, albeit in a less-escalated factor (i.e., with fewer network nodes). That said, there are some features that differentiate the IPN from a terrestrial internet, within the context of delivery delay, and make delay estimation a demanding task:

First, planet trajectories and spacecraft movement frequently disrupt the line-of-sight (LOS) between communicating antennas, and, hence, data propagation is suspended for potentially long timespans. Consequently, the calculation of data transmission intervals in a multi-hop space network, characterized by intermittent rather than continuous links, is an intrinsically complex task.

Second, the signal propagation times are higher, reaching the scales of minutes or hours, for deep-space communication. Long propagation delays, along with intermittent connectivity, hinder the timely transmission of network state updates. Therefore, the majority of calculations including delivery time estimations are constrained by lack of knowledge of the current network state, and, thus, need to be performed in advance of the data transmission procedure, rather than reactively, without abandoning the dynamic scope of a delay calculation framework.

Third, the deterministic nature of planetary and spacecraft movement requires a different way of connectivity planning: link availability is typically known a priori, and is not decided via dialogue. Thus, network nodes are assumed to have perfect knowledge of anticipated changes in network connectivity, in a time-ordered list of scheduled topology changes, which constitutes the *contact plan*. Contact plan knowledge can be beneficial to the communication protocols in a variety of ways, including routing decisions, as well as delivery time estimations. Despite the deterministic nature of scheduled interplanetary communications, space environments are also characterized by events such as solar activities and varying space weather that occur in a stochastic manner and may cause high error rates and unexpected disruptions.

In parallel, the internetworking capabilities of communication assets in the SSI introduce other dynamic parameters of probabilistic nature, such as cross-traffic queue backlogs. As a result, those events may perturb the scheduled communication plan in a stochastic, hard-to-predict way. A proper approach of estimating all delay components, consequently, requires methods and protocols that cover different probabilistic conditions and have the potential to respond to various unanticipated events.

The research performed in this thesis is motivated by those exact observations. We attempt to cover the SSI requirements for accurately and dynamically predicting the end-to-end path of data units, as well as the total data delivery latency, in automated ways. In this context, we provide algorithms, protocols and mechanisms that deal with all aforementioned challenges imposed by the special nature of space networking environments, within a variety of space network scenarios and for different network conditions.

1.3 Thesis Contributions

In order to fulfill the operational requirements of the IPN and achieve our intention for accurate latency estimation, we focus on the main contributing factors of the end-to-end delay, and analyze various network conditions that pertain to the particularities of space Internetworking and affect the interval required for data to reach destination. We provide applications and tools that facilitate different network functions, and thus can be exploited in various operational processes, including administrative and management procedures, the data routing facility, as well as end-to-end service and transport-layer operations.

We begin our research by providing an analytical measurement of the end-to-end delivery delay of data units at an administrative basis. Due to the stochastic delay components mentioned in the previous subsection, a purely deterministic approach for delivery delay predictions is rather impractical. Instead, an analysis of the likelihood for each data unit to follow some path, which incorporates both transmission latencies and retention latencies (contact interruption intervals) and considers the most plausible retransmission scenarios, allows for a weighted probabilistic delivery latency profile to be computed. Our approach departs from this observation and we introduce a novel method for estimating the Bit Error Rate (BER) on each link. To this end, the proposed method uses recent network processing statistics to calculate the mean expected number of retransmissions of lost data on each segment of the end-to-end path and a binary search algorithm to estimate the expected BER. Based on the extracted forecasts and the protocol retransmission procedures, we provide analytical methods to obtain the retransmission probabilities and corresponding delays, accordingly. We also exploit

information on the contact plan, the network links (e.g., protocol parameters, data rates, etc.), and data unit parameters (e.g., size, lifetime, sender node, destination node, etc.), to extract a probabilistic delivery latency profile, which comprises a list of possible arrival times at destination along with the corresponding probabilities. We implement the Bundle Delivery Time Estimation tool (BDTE) that realizes the proposed analytical methods and has the potential to provide multiple administrative services to mission operators: it can estimate the earliest and latest plausible arrival times; it can provide a Quality of Service (QoS) equivalent service for space communications, calculating the maximum time interval that a specific delivery is guaranteed within some confidence level; using a specific future time as application input, it extracts the probability that data will reach the destination before that time.

We note that the BDTE application and its analysis are based on the assumption that the input data items are of high priority, and, therefore, face negligible queueing delays on the end-to-end path. To complement this service, we study the intriguing task of calculating the queueing latency component of the delivery delay, which is in essence the waiting timespan until all data ahead of the current data unit is forwarded. For this purpose, we propose two different approaches for computing queueing delay: the reactive calculation based on network update messages, and the proactive estimation based on network statistics measurements and forecasting procedures.

In the former approach, we encode the queueing delay component and incorporate it as a distinct element of the network contact plan. That is, queueing delay information becomes an explicit parameter of each contact, which we name Earliest Transmission Opportunity (ETO). In order to exploit the obtained information, we propose an update to the Contact Graph Routing (CGR [9]) algorithm, namely Contact Graph Routing with Earliest Transmission Opportunity (CGR-ETO), which incorporates backlog information into routing decisions. We pair the introduced algorithm with an update protocol, namely Contact Plan Update Protocol (CPUP), which implements the diffusion of knowledge that pertains to transmission opportunity updates, dynamic network features, and contacts parameter changes, and we implement the dissemination mechanism of CPUP, accordingly. Therefore, CPUP provides the network with a dynamic framework to effectively propagate any changes in the contact plan through the network. This way, information on increased queue backlogs is disseminated with update messages, and, hence, the inherent capability of network nodes to calculate the corresponding delays for data transmissions is enhanced. All in all, the combined contact plan update framework makes network awareness and routing process more robust against delays imposed by cross-traffic that may significantly modify the predetermined transmission path and the overall delivery delay, respectively.

In the latter approach, we propose a proactive framework for estimating queueing delays through network statistics procedures and time series forecasting. The rationale for this

alternative is that the space internet will be characterized by periodicity (due to orbital movements of planets and spacecraft), repetitiveness, and, to a great extent, predictability of data production and delivery rates. In this context, we introduce a novel method where network nodes extract queueing rate measurements and disseminate them to other network nodes using the CPUP dissemination mechanism. The obtained measurements are then stored in the contact plan, composing different time series between each pair of network nodes. The available time series information are then used to forecast future queueing rates, and the predictions are combined with the contact plan schedules to estimate the queueing delay for the data units to be transmitted. This way, the proposed estimation of the overall, end-to-end delivery delay incorporates the obtained forecasts of future queueing delays, and can, therefore, match the actual delays experienced in a congested network more accurately.

In the last part of our research, we exploit the introduced analytical methods and algorithms to improve the transport layer's capability of estimating RTTs, and to enhance the efficiency of the end-to-end retransmission mechanism of transport protocols that operate on top of space DTN architecture. In this context, we develop a novel, dynamic end-to-end retransmission framework for the transport layer of the DTN architecture that targets networks with intermittent and scheduled connectivity, such as space networks. The introduced framework takes into account cross-layer information to estimate the major latency components, and combines them to calculate efficient RTO intervals using the maximum -within some boundaries- expected end-to-end delay, based on the worst-case network conditions that may be experienced on the routing paths. In particular, the proposed framework:

- Gets feedback from the routing algorithm to predict the end-to-end path, i.e., the complete path from source to destination and back, for the data units and the corresponding acknowledgements.
- Groups protocol data units (PDUs) into blocks based on the anticipated end-to-end path.
- Estimates the worst-case end-to-end delivery delay per block by: (i) exploiting statistical network data and performance modeling of underlying CL protocols to calculate worst-case delivery delay for each hop of the predicted route, and (ii) leveraging network connectivity information and performing routing simulations based on the worst-case data arrival time at each intermediate node, through the predicted end-to-end route.
- Sets retransmission timers at block granularity, based on the worst-case estimated RTT.

The introduced retransmission framework is designed in a modular way to enable straightforward addition of other CL protocols and corresponding transmission policies, and adjustment to future modifications of the BP layer and routing algorithm. We design the main concepts of operation in a protocol-independent way, and then develop the operation algorithms within the technical context of DTPC protocol, which functions at the transport layer over the

DTN architecture. We accordingly implement the introduced framework, namely DTPC-dRTO (DTPC with dynamic Retransmission Timeout), as well as its algorithmic methods as DTPC extensions.

1.4 Evaluation Methodology

1.4.1 Evaluation Tools

In order to validate the applicability and assess the performance of the introduced research components of this thesis, we use different evaluation methods, i.e., simulation studies and emulation experiments, to complement and enhance the provided analytical studies.

In particular, we design and implement *SpaceDTNSim*, a Java-based, discrete-event simulator that targets space-oriented DTNs, to perform simulation studies for the queueing delay calculation methods. We incorporate different versions of the CGR routing algorithm in *SpaceDTNSim*, including the proposed CGR-ETO variant, as well as the CPUP protocol, to assess the efficiency of queueing delay estimation based on dissemination methods. We also use *SpaceDTNSim* to evaluate the proactive method of queueing delay prediction through time series forecasting, as well as conduct a comparison study between those two approaches.

Moreover, in order to evaluate the proposed protocols and methods in a realistic emulation networking environment, we deploy and exploit *SPICE DTN Testbed*, funded by European Space Agency (ESA) and FP7 Space Internetworking Center (SPICE) project [10] to accurately emulate space components and links, support diverse protocol stacks, and provide a realistic testing environment to evaluate, benchmark and optimize new protocols. SPICE DTN Testbed includes, among other software, National Aeronautics and Space Administration (NASA) Jet Propulsion Laboratory (JPL) implementation of the DTN protocol stack, namely Interplanetary Overlay Network (ION [11] [12]), which is provided as open source software. We develop BDTE, CGR-ETO, and DTPC-dRTO and incorporate them into ION, in order to conduct experiments in a real DTN implementation with emulated space conditions. We highlight that an initial version of CGR-ETO algorithm has been accepted and integrated into the standard distribution of ION (from v. 3.2.1), to enhance the core functionality of CGR algorithm.

1.4.2 Evaluation Scenarios

We design a set of evaluation scenarios to assess the performance of the proposed protocols and mechanisms. Our purpose is to capture typical use cases of space data transmissions, in a variety of space network conditions. In certain experiments, we stretch the scenario parameters to capture extreme network conditions, e.g., high error rates due to adverse weather conditions, large number of nodes producing cross-traffic data, etc. Our overall intention is to evaluate how the introduced methods enhance the network's ability to predict the end-to-end delivery delay of the transmitted data, to quantify the improvement in the delivery latency prediction, and assess the provided benefits of this improvement, as far as routing algorithm's efficiency and transport protocol's retransmission scheme are concerned.

We divide our evaluation in five different scenarios: In the first scenario, we showcase that the BDTE tool can provide detailed and accurate information of data delivery in an administrative way, for a space network with long-haul links and error rates that fluctuate (either following a pattern or in a random way) through time. In the second scenario, we measure the efficiency of CGR-ETO algorithm paired with CPUP protocol, with respect to the improvement of delivery delay estimation and routing decisions, by simulating multi-hop, multi-path data transmissions that may be employed in typical Mars and Lunar missions. In the third scenario, we validate the implementation of CGR-ETO in ION, and complement the evaluation of the introduced routing algorithm with the emulation of a satellite data transmission scenario, in which we examine how the proposed algorithm impacts routing decisions and improves the delivery delay of routed data units. We highlight that this scenario has promoted the efficacy of CGR-ETO and resulted in its adoption within the standard CGR algorithm of ION implementation. Next, we simulate a data transmission scenario, where different number of nodes create and transmit various amounts of data that are being delivered to a destination through a single queue, resulting in significant fluctuations of the queueing delay. Here, we assess the improvement that proactive estimations of queueing delay bring to the prediction accuracy of the total delivery delay, and we compare it with the reactive queueing delay calculation approach of CGR-ETO. Finally, we deploy DTPC-dRTO framework in an emulated, complex, deep-space scenario with challenging network conditions (i.e., varying error rates and sporadic cross-traffic through the duration of experiments). In this scenario, we examine how the introduced scheme improves RTT predictions, enhances the overall retransmission framework of DTPC protocol, by promoting timely retransmission of lost data, and provides more efficient storage management.

1.5 Thesis Results

Through the extensive evaluation process described in the previous subsection, using both simulations and experimentation, we demonstrate that all proposed methods achieve significant improvement in the prediction of end-to-end delivery latency. In particular:

- We show that the BDTE application can effectively exploit the obtained network statistics and provide an exhaustive, probabilistic delivery latency profile, with adequate **accuracy in delivery time estimations**, which can be exploited in different administrative services.
- We illustrate that the challenging calculation of queueing delay can be efficiently handled in different ways, i.e., reactively, through CPUP network updates, and proactively, by storing and disseminating network rate statistics and applying time series forecasting methods. Both approaches achieve significant improvements in queueing delay estimation, and, consequently, provide **more accurate predictions of the overall latency**.
- We prove that the CGR-ETO routing algorithm paired with the CPUP dissemination scheme achieves **faster delivery** for a significant percentage of the transmitted data units, primarily in highly congested network conditions. Therefore, it can provide an **efficient routing** scheme for space internets with challenged environment and network conditions.
- Using the proactive approach of queueing delay estimation, we show that an efficient management and forecasting process that exploits disseminated information about transmission rates can be even more **precise regarding the prediction of queueing delays**, in comparison to the approach that involves network updates on queue lengths.
- We showcase that the introduced DTPC-dRTO framework provides a better, **more accurate RTT estimator** than the originally proposed, static retransmission scheme of DTPC protocol. Consequently, erroneous or lost data are retransmitted faster, and, hence, we observe a **great reduction in the overall data transmission time**, while keeping the **overhead -due to duplicate transmissions- minimum**.
- Finally, we illustrate that, by achieving faster retransmission of lost data, DTPC-dRTO framework provides **great reduction of the storage occupancy** and utilization, primarily at destination node, when the in-order delivery feature of DTPC protocol is applied.

1.6 Impact

The methodology introduced in this thesis, the developed protocols and mechanisms, and the observed experimental results have multifold significance: they can influence the research and engineering community that works on space networking and the advancement of DTN architecture and protocols; they can support future space flight operations, and provide a robust framework for accurate mission planning; and, in the long term, they have the potential to impact the future Internet and assist its evolution towards a more inclusive networking paradigm that involves off-Earth networks, as well as disconnected areas and people on Earth.

From a research point-of-view, DTN is an architecture that emerged more than ten years ago and many of its components and protocols are currently under standardization within two communities: the Consultative Committee for Space Data Systems (CCSDS) [13], which develops standards in communications and data systems for future space missions, and Internet Engineering Task Force's (IETF) newly founded DTN Working Group (DTNWG) [14], in parallel with Internet Research Task Force's (IRTF) DTN Research Group (DTNRG) [15], works towards the production of standards within the wide area of data communications in the presence of long delays and/or intermittent connectivity. In this context, the novel research performed in this thesis can contribute to the standardization processes of the aforementioned groups in different ways. In particular, CCSDS ongoing processes towards standardizing CGR and DTPC protocol may include the CGR-ETO algorithm enhancement and DTPC-dRTO retransmission framework, respectively. Furthermore, CPUP protocol may be included as a potential standard in IRTF, IETF, and/or CCSDS standardization groups in future considerations of network management services and protocols, since it provides a viable and efficient solution for the exchange of contact plan information and dynamic network updates.

As space protocols and standards evolve towards the IPN era, the introduced research can be of great importance for the communication infrastructure of the space Internet, since it accelerates the transmission of important data during challenging environment conditions, and enhances the network's inherent ability to predict data transmission delays. Since typically data are routed based on the earliest-arrival-time routing objective, inaccurate information on the delivery interval leads to suboptimal routing decisions with extra impact on the space operations: insufficient exploitation of available data transmission windows, negligence of possibly better alternate routes, lack of load balancing, etc. In this context, the use of the introduced, enhanced routing algorithms can overcome those deficiencies and greatly improve space data communications, while the CPUP protocol may tackle unanticipated network changes more efficiently and dynamically. Moreover, since routing decisions are more efficient and overall data delivered during the same time intervals increases, acquisition of scientific

and/or Earth observation data is expedited. The updated capabilities introduced in this thesis may be of benefit both to automated networking processes (e.g., automatic space-data transmissions such as telemetry, network management procedures), as well as user-oriented network operations, by providing mission operators, Principal Investigators (PIs) and payload end-users with more information on the data communications section and further insight into the duration of data deliveries and operations in general. Mission design can thus obtain significant gains from the additional information on delivery timespans, since time scheduling is important for agencies and mission operators. By and large, the proposed research can provide an important step towards a robust, unified space Internet architecture, which will provide efficient routing, accurate mission planning and more effectively coordinated data communications.

Furthermore, the various improvements that this thesis brings to the IPN and DTN communications have the potential to impact the worldwide research towards the Internet of the future, which is being designed with different, novel networking paradigms in mind. A lot of research and engineering efforts in the context of the future Internet focus on unifying even more, disconnected areas and people [16], and its design includes the core functionality principles of DTN (i.e., delay, disruption, and disconnection tolerance), as well as the networking capabilities of space and satellite communications, which are increasingly being exploited to extend the Internet services to disconnected areas. Hence, the protocols and mechanisms introduced in this thesis can influence the design of future Internet and, in the long term, provide multiple gains in different aspects, such as social (e.g., with a more inclusive Internet), scientific (e.g., increased amounts of obtained scientific data), environmental (e.g., with the enhanced dissemination of Earth observation data), disaster-resilient (timely notification about emergency events through novel Internet architectures), etc.

1.7 Thesis Structure

In Chapter 2, we introduce the research background within the context of the IPN, as well as DTN architecture. We discuss the end-to-end delivery delay and its components in the context of space internet, and we provide the related work on delivery delay estimation methods. We continue with the discussion of previous research on data routing, in space internets, and on end-to-end retransmission timer setup and configuration, in different networking environments.

In Chapter 3, we introduce the administrative framework that produces probabilistic profiles on the arrival times of data items. We present an analytical study about the retransmission

procedure of lost data and the algorithms that are used to predict future error rates based on network statistics. Furthermore, we describe the deployment of the proposed analytical and algorithmic methods within the Bundle Delivery Time Estimation tool, as an administrative application that estimates bundle delivery times.

In Chapter 4, we present two different approaches in estimating queueing delays for space networks: the reactive and the proactive method. For the former, we introduce the incorporation of queue length information into the ETO parameter of the contact plan. We present the enhanced CGR-ETO algorithm, which incorporates the available ETO information to achieve better delivery delay estimates and improved routing decisions, accordingly. We describe the format and functionality of the CPUP, designed to diffuse contact plan modifications, including information on dynamic network features (such as queue backlogs), and we detail the applied dissemination mechanism. We continue with the description of the latter approach for proactive prediction of queueing delays, based on network statistics and forecasting methods. We present a generic scenario and propose a sampling procedure that extract measurements of queueing rates and queue lengths for network queues. We describe the dissemination process of the extracted queueing information, and detail the algorithmic method used to predict future queueing rates and delays, based on time series forecasting.

In Chapter 5, we present the dynamic retransmission framework of DTPC that exploits the methods for improved delivery delay accuracy presented in the previous chapters. We begin by analyzing the main concepts of operation that we have exploited, in a protocol-independent way, and continue with the algorithmic implementation of the proposed retransmission framework within the space-oriented protocol stack.

In Chapter 6, we present the Evaluation Methodology that we followed in this thesis. We list the objectives of our evaluation process, and describe the scenarios designed to emulate data transmissions in space networks. We present the evaluation metrics, as well as the software tools used to facilitate the development, validation, optimization, and evaluation of all introduced methods and protocols.

In Chapter 7, we present our experimental results. We validate the operation of the proposed mechanisms, and evaluate their efficiency with respect to the targets of this thesis. We analyze the obtained evaluation results and draw meaningful conclusions, accordingly.

Finally, in Chapter 8, we conclude the present thesis. We discuss the most significant outcomes of the presented research and highlight our contributions to the problems that this thesis targets.

Chapter 2 Background and Related Work

In this Chapter we present the background and related work that constitutes the basis of our research. Initially (Section 2.1), we describe the space internetworking background and discuss the concept of the Interplanetary Internet. In Section 2.2, we present the DTN architecture, which is the enabling paradigm for realizing space internets, and illustrate its core functionality, as well as the main DTN protocols. Within the context of the IPN, we identify the need to study the challenging issue of estimating delivery delays, and discuss the related work that has been presented towards this direction, i.e., the calculation of delivery delays in space networks (Section 2.3). We continue with the presentation of the relative aspects of the delivery delay calculation that we study in this thesis. In Section 2.4, we provide an overview of the related work on routing in space networks and specifically in space DTNs, discussing in detail the most distinctive routing algorithm, CGR. Finally, in Section 2.5, we present the studies that have been performed on end-to-end retransmission timers.

2.1 Interplanetary Internet

One of the crucial elements of every space mission is the communications system, which is responsible to carry commands and other information from Earth to a spacecraft or to a remote planet, and downlink scientific data to Earth, as well as telemetry or other data important to the welfare of the spacecraft, the scientific equipment, and potentially of people boarded on the spacecraft. Therefore, communications systems are central to the success of space missions. In this context, large amounts of data need to be transmitted on a daily basis from different spacecraft that reside in near-Earth or deep-space environments, or even from landers or rovers on planetary surfaces. Moreover, as science data requirements for future missions increase with the employment of more sophisticated instruments that generate more data, the demand for data transmissions is expected to grow even more in the future [17]. Hence, there is an increasing need for availability of high network transfer rates.

Furthermore, space communications systems have to maintain their functionality within difficult network conditions that pose serious communication challenges: long signal propagation delays and RTTs; intermittent connectivity due to the disruption of LOS; low and asymmetric data rates; likelihood of data loss due to errors on the communication link; possible channel disruptions; and coverage issues at high latitudes and in challenging terrain. At the same time, a space communications system must be reliable, since, typically, it is the only way

to interact with a spacecraft, and to diagnose and repair potential problems that may arise, as well as be enduring over time due to the long duration of space missions.

During the last decades, engineers and space agencies have tried to alleviate those issues by providing new protocols and services with cross-mission and cross-agency support, as well as improved functionality and reliability features. CCSDS [13], a standardization group that comprises the majority of space agencies worldwide, has supported many efforts towards this direction, in different layers of the protocol stack, which can be seen in Figure 2-1. Among the proposed standards, Space Packet Protocol (SPP) [18] was designed to interconnect different subnetworks under a common network layer, providing an initial step towards space internetworking, with functionality, however, limited to a simple, unreliable data transfer. In the transport layer, CCSDS proposed Space Communications Protocol Standards-Transport Protocol (SCPS-TP) [19] that extended the functionality of TCP and UDP protocols, to cover space environments. The CCSDS File Delivery Protocol (CFDP) in the application or transport layer [20] includes store-and-forward operations that form the basis for reliable multi-hop transfer, but without providing services other than file transmission (e.g., streaming, messaging, etc.).

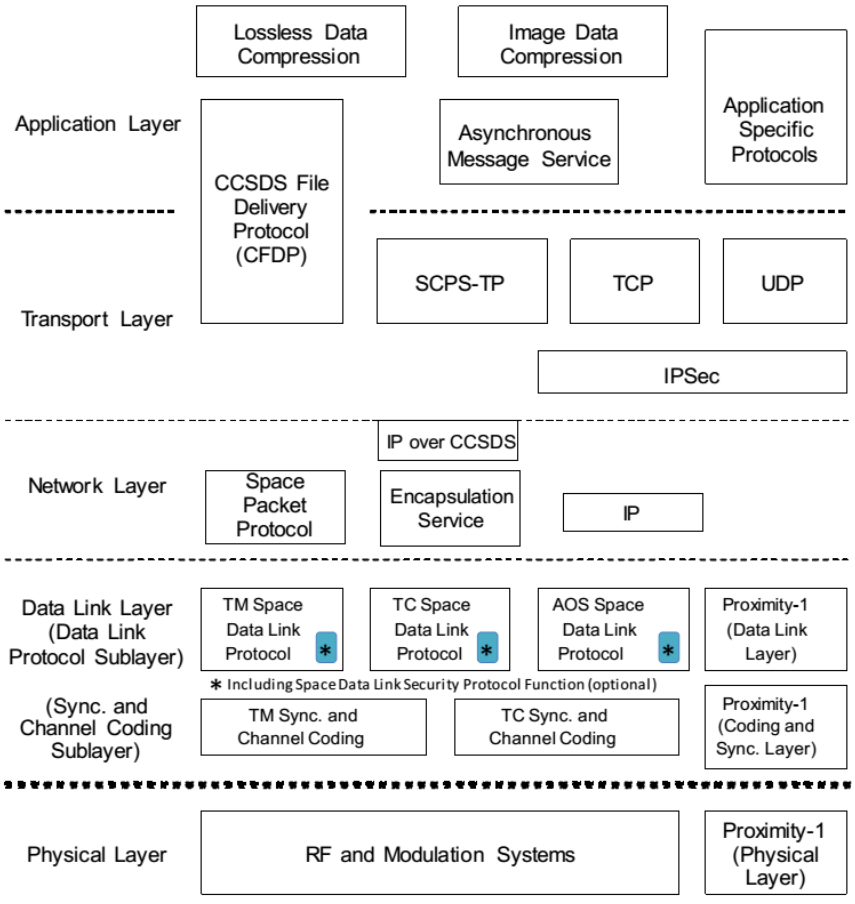

Figure 2-1 Space Communications Protocols Reference Model [21]

The increasing demand for higher data transmission rates, the requirement for more robust and reliable space communications systems that provide multiple services, and the increased number of space assets, created the need for a different, unified networking architecture: the Interplanetary Internet [2] [22] [23]. The core concept of the IPN is to forward the space communications concept from simple, scheduled operations over single point-to-point links, to include complex scenarios and topologies with more nodes and data transmissions over multiple hops. Future space operations in the upcoming IPN era are scheduled to be more dynamic and flexible. They will involve interoperability among various missions and space assets that belong to different, collaborating agencies, and will enhance the ability to share and better exploit the available network resources, which are rare and expensive in space networks. The improved resource exploitation, along with the constantly improving communications equipment will boost the data transmission capabilities and will better support space-data dissemination [24] [25] in the upcoming missions. Many of the procedures, which are now human-operated, will become automated, interoperable and collaborative.

In this context, space communications will benefit from the Internet architecture, assisting at the same time the internetworking provisioning in disconnected areas and communities and thus, providing extra capabilities in the future Internet [26]. Hence, space communications will evolve similarly to the TCP/IP architecture in five major aspects [1]:

i) Internet-related protocols will be utilized or adjusted to support low-latency (e.g., near-Earth or planetary) internetworks.

ii) A deep-space backbone network with long-haul links will interconnect various heterogeneous subnets.

iii) A common network layer will unify the "network of internets" [1] (e.g., deep-space, satellite, planetary networks, the Internet) and function over various subnet-specific protocols, in a way similar to the TCP/IP protocol suite over various Internet regions.

iv) The routing function will transcend the data forwarding processes from predetermined manual procedures to autonomous, dynamic operations, and will thus provide the ability to exploit possible alternative paths for data delivery.

v) Reliability of data transfers will be enhanced with the employment of retransmission mechanisms.

Since the standard Internet architecture was not adequate to solve the space internetworking issues [27], the efforts for realization of the IPN concept resulted in the introduction of a novel networking architecture: Delay- and Disruption-Tolerant Networking [5] [22] [28] [29]. DTN architecture was designed to operate as an overlay above different interconnected networks, and to provide key services such as in-network data storage and retransmission, interoperable naming, authenticated forwarding and coarse-grained classes of service [28]. Although initially proposed as an alternative for interplanetary communications [29], DTN was used to solve the

challenging networking conditions of other environments as well, including sensor networks [30], military ad-hoc networks, vehicular networks, underwater networks, etc. [31].

The advanced functionality of the DTN architecture resulted in its adoption by CCSDS into its standardization procedures [32]. Furthermore, due to the increased agency interest in internetworked space communications architectures, the Interagency Operations Advisory Group (IOAG) chartered a Space Internetworking Strategy Group in 2007, "to reach international consensus on a recommended approach for transitioning the participating agencies towards a future network-centric era of space mission operations" [33]. This effort resulted in documentation of the operations concept for a SSI [4], which specifies DTN protocol suite as the core networking architecture to realize the SSI and cover the required operational services. The interoperability features of DTN have been successfully proven within the DTNBone network [34] and the DTN Engineering Network (*DEN*) [35], and various worldwide projects have been working towards the improvement of networking capabilities in many aspects, with the use of DTN [10] [24] [36] [37] [38] [39] [40] [41] [42] [43] [44] [45].

In recent years, DTN was effectively tested in real space experiments. In 2008, NASA's JPL conducted Deep Impact Network Experiment (DINET), a 27-day experiment, to test essential elements of DTN technology on the Deep Impact spacecraft (i.e., bundle origination, transmission, acquisition, dynamic route computation, congestion control, prioritization, custody transfer, and automatic retransmission procedures), both on the spacecraft and on the ground. The DINET experiment demonstrated DTN readiness for operational use in space missions [46]. In 2009, the efficiency of DTN for large data transfers was proven in a two-day experiment, where the proactive and reactive fragmentation capabilities were demonstrated using the United Kingdom Disaster Monitoring Constellation (UK-DMC) satellite and two independent ground stations [47]. DTN was recently deployed in the International Space Station (ISS) to be tested within different activities and experiments [48] [49], including the Multi-purpose, End-To-End Robotic Operations Network (METERON) Project [50], an experimental architecture for the validation of human-robotic operations from space using the ISS. Within the METERON project, the applicability of DTN in space operations was validated in a successful experiment in 2014, where astronauts from ISS managed to operate a robot on Earth, exploiting the end-to-end networking capabilities of DTN [51].

By and large, DTN constitutes the most mature architecture that can effectively realize the Interplanetary Internet concept, providing efficient solutions to the necessary internetworking services for future space mission operations. In the next subsection we provide the core functionality and elements of the DTN architecture and present the basic protocols of the DTN protocol stack.

2.2 Delay/Disruption Tolerant Networking Architecture

DTN is a communications architecture that was introduced [5] [28] to internetwork challenging environments where traditional networking protocols typically fail. DTN can be successfully applied in different networks that may be characterized by:

- Transient network partitioning (e.g., due to moving out-of-range or disruption in the LOS) that leads to absence of end-to-end connectivity between a data source and its peer(s);
- Long signal propagation delays between network nodes, which may reach a few seconds for cislunar communications [52], 4-24 minutes for cis-martian communications [53] and even more for other missions such as the Voyager Interstellar Mission [54];
- Relatively low data rates with large asymmetry (e.g., in the order of 1000:1 or higher for deep-space communications [27]); and/or
- Low signal to noise ratio that leads to large packet drop probabilities.

To overcome those challenges, DTN incorporates at its core **Bundle Protocol** [6] as an overlay network protocol, with the potential to unify different internets under a global layer, in the same way IP connects Internet regions across the globe. The position of BP in the DTN protocol stack and its functionality as an overlay protocol is illustrated in Figure 2-2.

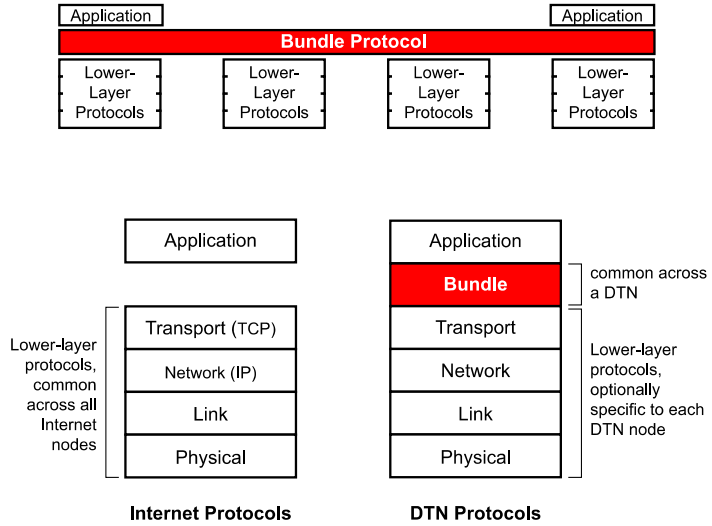

Figure 2-2 Bundle Protocol overlay, position within the DTN protocol stack and comparison with the Internet protocol stack [55].

BP implements the store-and-forward policy of DTN, which emphasizes on keeping in persistent storage of transmitted or relayed data, rather than temporary buffering, and enhances the reliability of data delivery. BP incorporates the transmitted data in variable-length PDUs,

which are called *bundles*. Bundles comprise a set of blocks that may contain meta-data or application data, "bundled" together to reduce the number of information exchanges and transactions (and corresponding RTTs), which is important in networks with long delays [56]. Key elements of the BP functionality include:

- *Custody transfer* [22] [57], which utilizes hop-by-hop transfer of reliable delivery responsibility through the end-to-end path of a bundle. The intermediate node that takes custody (i.e., responsibility) of a bundle is called *custodian node*.
- *Time-to-live* (*TTL*), i.e., the lifetime of a bundle.
- Three different *Classes of Service*, namely *Bulk*, *Normal*, and *Expedited*, for differentiation of the bundle delivery priority. The Class of Service and the TTL are determined by the transmitting application.
- Proactive and reactive bundle fragmentation; the former to tackle intermittent periodic connectivity when the amount of data that can be transferred is known a priori, the latter, which works a posteriori, when disruptions interrupt an ongoing bundle transfer.
- A routing algorithm, which is responsible to decide on the optimal path to forward the bundles, based on some routing objective. Detailed information on the routing functions of the BP are provided in Section 2.4.
- *Late binding*, where the name-to-address mapping is not required to be performed prior to the start of a transmission, but also at an intermediate node towards the destination region. For example, when a bundle destination endpoint's identifier includes a Domain Name System (DNS) name, only the penultimate DTN node might have to resolve that DNS name to an IP address, while data routing for earlier hops can be purely name-based.
- Flexible naming that can comprise different schemes, based on the Uniform Resource Identifier (URI) syntax [58]. Every DTN node is part of one or more *Endpoint Identifiers* (*EIDs*), which are text strings that are used to determine the bundle source node, destination node, custodial, etc.

In order to interconnect different subnetworks, BP may interface with different, underlying, transport-protocol-specific CLs, which are mainly used to add reliability for bundle transmissions between a pair of BP nodes [28]. Multiple CLs may connect different pairs of nodes within the same network. A number of different CL protocols have been introduced to cover a variety of network conditions. Several of those are based on typical Internet protocols, such as the TCP Convergence Layer (TCPCL) [59], UDP and Datagram Congestion Control Protocol (DCCP) Datagram CLs [60], the NetInf Bluetooth Convergence Layer [61], etc. CLs have also been developed for protocols specific to DTNs, such as the Uni-DTN CL protocol for Unidirectional Transport [62], the Underwater Convergence Layer (UCL) [63], the LTP CL

protocol [64] for long-delay and frequently-disrupted links, the Saratoga CL protocol [65] for long-distance space links and for terrestrial mobile ad-hoc networks, the CL for SpaceWire [66] etc.

Among the aforementioned CL protocols, LTP [67] [68] [69] [64] constitutes the most robust solution for reliable data transfers over point-to-point, long-haul links in a space network, and was included in CCSDS standardization processes [70]. LTP operates directly over the link layer, e.g., CCSDS Telemetry (TM) [71], Telecommand (TC) [72], Advanced Orbiting Systems (AOS) [73], Proximity-1 [74], but can also operate over datagrams [60]. It can successfully handle long disruptions without data loss, and employs a selective negative acknowledgement (selective NAK) Automatic Repeat reQuest (ARQ) procedure to ensure data delivery. LTP supports partial reliability, i.e., for a percentage of the data transmitted, which is called *red part*, while the unreliable data constitute the *green part*. It operates in a session-based mode: Each LTP session has a sender and a receiver node and assembles bundles ready for transmission into *LTP blocks*. Then, LTP fragments blocks into *LTP segments*, which are encapsulated in lower layer frames and forwarded to the LTP reception node. The last segment within a red-part set of segments, called a *checkpoint* (*CP*), triggers a NAK segment by the receiver, called a *report segment* (*RS*), which in turn triggers a *report-acknowledgement* (*RA*) *segment* from the sender. Any potential missing parts of the block are transmitted in response to the RS and the process repeats until the block is successfully received. The reliable delivery of CP, RS, and RA segments is established with retransmission timers, which are typically set equal to the signal propagation time from source to destination and back, plus some margin time [75]. During anticipated connectivity disruptions, LTP timers are suspended, and continue when connection is re-established, hence providing a robust solution against the typical intermittency of the space links. The operation of LTP during the transmission of a single block that consists solely of red-part data is illustrated in Figure 2-3. Since LTP is one of the most prominent CL protocols for long-haul, interplanetary links, part of the research performed in this thesis pertains to the analytical study of the total delay required for the successful delivery of an LTP block, in data transmissions over space links.

A variety of applications or services can operate on top of the DTN architecture, either directly above BP, or using an end-to-end, transport-layer protocol, to employ necessary end-to-end features. Among those features, the end-to-end reliability of data delivery constitutes a crucial element of space communications. That is, although BP provides a set of reliability features (e.g., custody-based retransmissions), there are some cases that it fails to assure the end-to-end delivery of data [76]. In this context, the transport layer complements the reliability of BP and potentially of the CL protocol by providing a "safety net" [8] for the end-to-end data delivery. Other end-to-end features include aggregation of data, in-order delivery, duplicate suppression, etc. [8]. Some of the **transport-layer protocols** introduced to operate on top of

the DTN architecture include CFDP, Erasure-Coding Transport Protocol (ECTP), DTPC, which are briefly described below.

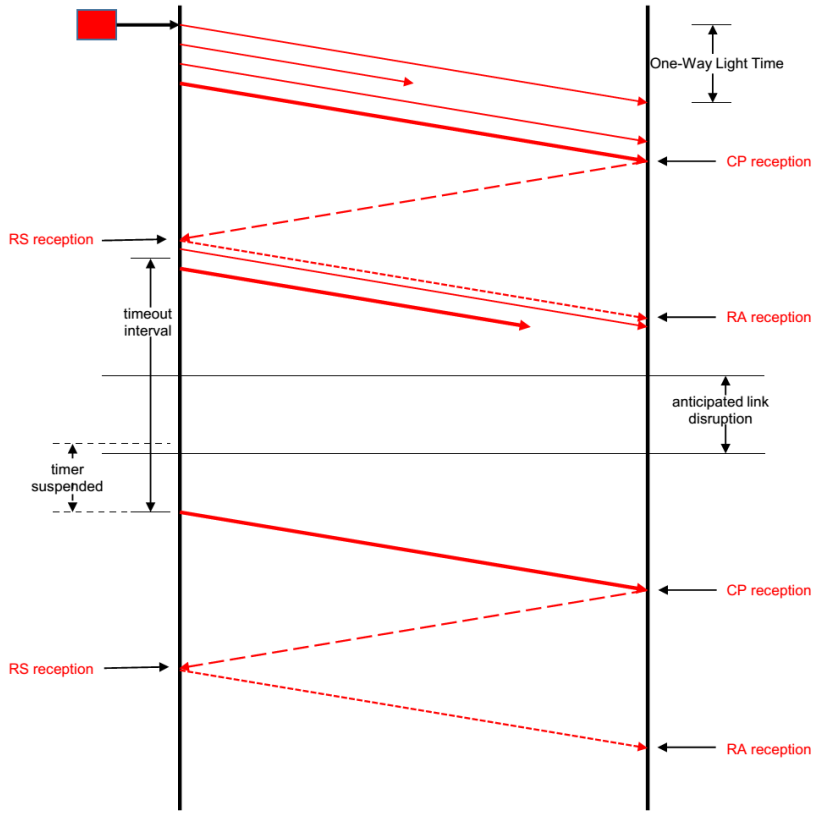

Figure 2-3 Operation of LTP during a block transmission

Although **CCSDS File Delivery Protocol** [20] [77] [78] was originally proposed as an application layer protocol, it also provides transport-layer functionalities, such as detection and retransmission of lost or corrupted data, and can be applied on top of BP [79]. CFDP provides reliable (acknowledged mode) or unreliable (unacknowledged mode) file transmissions and includes four modes for sending NAKs, i.e., Deferred, Immediate, Prompted and Asynchronous, and it can also send Positive Acknowledgements (ACKs) for critical PDUs.

Erasure-Coding Transport Protocol [80] was introduced as a generic end-to-end transport mechanism that operates on top of the DTN architecture. ECTP achieves reliable data transmissions through a hybrid framework, which incorporates low-density parity-check (LDPC) erasure codes applied at the end nodes of the data transmissions, along with an ARQ retransmission scheme.

Delay-Tolerant Payload Conditioning (DTPC) protocol [8] is an application-independent protocol offering transparent application data conditioning services in an end-to-end fashion. DTPC protocol is an expandable, connectionless, reliable, sequenced transport protocol that

enables a set of end-to-end services: (a) application data aggregation, (b) application-level reliability, (c) in-order delivery, and (d) duplicate suppression. An example of the DTPC protocol operation is displayed in Figure 2-4. Since DTPC is the most distinctive example of the end-to-end transport layer of the DTN architecture, in this thesis, we study the retransmission mechanism of DTPC and its dependence on the RTT, and propose a new retransmission framework that is based on advanced estimation methods for calculating the maximum RTT.

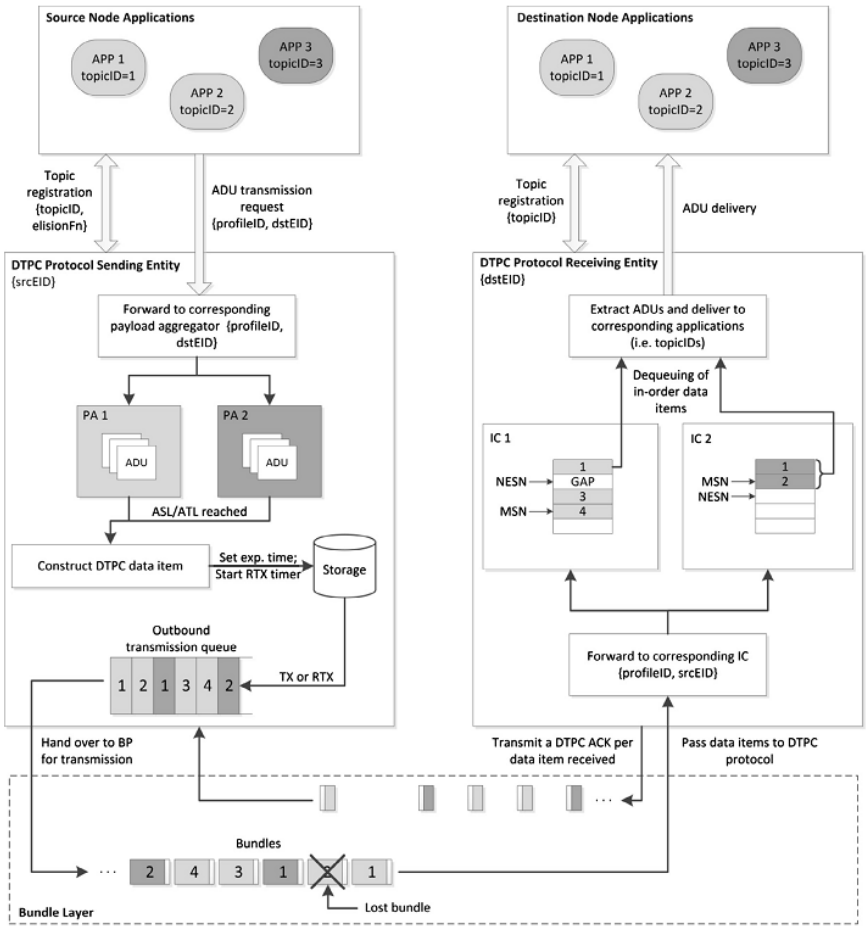

Figure 2-4 Example of DTPC protocol operation [8]

A typical example of a communications scenario for space includes the transmission of data (e.g., telemetry, captured images) from a Mars lander towards the Mission Operations Center (MOC) of the operating space agency on Earth, via a Mars satellite and a Deep Space Network (DSN) [81] receiving station. The DTN architecture can naturally support this scenario using different protocol stacks per each hop of the data transmission. As observed in Figure 2-5, the data transmission application initiates the communication, and transmits application data using DTPC as the end-to-end transport protocol (we note that the transport layer protocol is optional; application could interface directly with BP). Data are delivered in a hop-by-hop fashion and

stored in BP's persistent storage, while different CL, link layer and physical protocols are employed in each hop of the transmission. At the CL, in particular, LTP CL is used to transfer data over the space links (i.e., lander-satellite, satellite-DSN antenna), whereas for the data delivery from the DSN receiving station towards the MOC, BP interfaces with TCP CL, employing a typical TCP/IP data transfer over a dedicated terrestrial link or the Internet. Note that DTN hops may comprise more than one link-layer transmissions; for example, at the last segment of the described scenario, a single-hop DTN connection may be established using a multi-hop Internet connection.

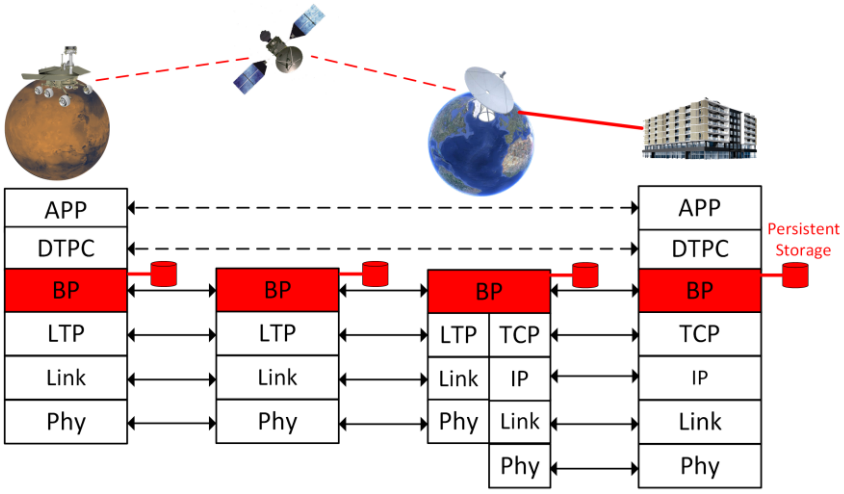

Figure 2-5 DTN protocol stacks example for a Mars-to-Earth data transmission scenario

2.3 End-to-End Delivery Delay

In the IPN era, communications systems of space missions are expected to function in an internetworked and automated fashion. Hence, the introduced networking functionality significantly reduces the necessary effort to manually setup and configure communication scheduling, data transmission procedures, and history tracking and management processes, vital to the health of space assets. However, the automated procedures necessitate a set of network functions that support the communications planning, manage in an efficient way the communications sessions, and provide advanced feedback to the mission operators upon request.

One fundamental characteristic involved in all network functions of a space internetwork is end-to-end delivery delay, which characterizes every data transmission. The accurate calculation of end-to-end delivery delays becomes more important in internetworked space communications, where data transmissions are performed automatically, rather than with preconfigured and manual scheduling processes. Moreover, since communications systems are

often the only way to interact with a spacecraft, the automated communications procedures need to be accurate and robust, with adequate information on the timespans and expected durations of data transmissions. The delivery time estimation affects a variety of mission operations aspects, which, respectively, require proper solutions for accurate estimates. Examples of such aspects include administrative services and applications, network management and planning, critical data transmissions, data routing, etc.

Similarly to the Internet [82], end-to-end delivery delay in a space internetwork is defined as the time required to deliver a bundle (or packet) from source to destination node. Each bundle generated by a source node is routed towards its destination node through a sequence of intermediate nodes, which correspond to networking elements that reside on space or terrestrial assets. The end-to-end delay is the sum of the delays experienced in all intermediate transmissions from source to destination. The individual delay for each hop transmission, in turn, comprises a set of different components:

Propagation delay is the time required for a single bit to be propagated over the communication link. It depends on the travel speed of the electromagnetic wave through the physical medium between the two communicating nodes, and the distance between them. Hence, in space point-to-point communications it is also called *One-Way-Light-Time* (OWLT). Due to the large distances, typical values of propagation delay for space links are significantly higher than terrestrial Internet links, where propagation times are limited to few milliseconds up to a few hundred milliseconds for inter-continental links. Typical values of propagation delays are 20-25 ms for Low Earth Orbit (LEO) satellites, 110-130 ms for Medium Earth Orbit (MEO) satellites, 250-280 ms for Geostationary Earth Orbit (GEO) satellites [83], 1-5 s for cislunar communications [52], 4-24 minutes for cis-martian communications [53] and may reach the order of hours or even days for missions in the deeper space, such as the Voyager Interstellar Mission [54].

Transmission delay is the time needed to transmit the entire bit sequence of a bundle over the communication link. Therefore, it depends on the length of the PDU, as well as the link capacity or bandwidth, measured in bits/s. The capacity of space communication links varies and may span from a few bits/s to the order of Gigabits/s, with the use of K_a band [84]. Space communication links are highly asymmetric, with a ratio in the order of 1000:1 or more, providing low data rates at the uplink channel and significantly higher at the downlink channel.

Queueing delay represents the time that the bundle waits in the persistent storage or buffer until all other bundles ahead are forwarded. Its actual value depends on the amount of interference with other flows of bundles through the path to destination. Queueing delays can become a considerable part of the total delay in low-latency (e.g., planetary) networks where propagation delay is not prohibitive. Furthermore, in networks with common disruptions and

rare transmission opportunities, such as space internets, even small queueing delays may significantly affect the overall delays, e.g., when they lead to loss of transmission opportunities.

Processing delay is the time required to process a bundle and prepare it for transmission. Similarly to the Internet [85], processing delay in a space network node depends on the complexity of the protocol stack and the computational power of the networking equipment. Since the equipment involved in space missions is not typically updated through the mission timespan, the employed technology may be old and the processing power low, resulting in relatively higher processing times than e.g., state-of-the-art Internet routers. However, processing delays rarely exceed the order of milliseconds, and hence, constitute only a minor percentage of the total end-to-end delay, in the presence of long-haul links and low data rates.

Besides the typical, aforementioned delay components that are present in all networking environments, space data transmissions include some other components as well, of stochastic nature, that are closely connected to the special nature of space internetworking.

One of the most significant delay components that differentiate space internetworks with the Internet is the **waiting delay** that the disconnected nature of space communications with common link disruptions incurs. Waiting delay represents the contact interruption time, that is, the time that a bundle needs to wait until the communication link is re-established. The waiting delays depend on the positioning and movement of communicating nodes, as well the orbits of celestial bodies or the presence of other bodies that may block the LOS and obstruct the communication signals. Furthermore, in some occasions, the available transmission windows can be shorter than the actual communication capability intervals, due to agency and resources limitations, e.g., when a ground station serves more than one missions, and it is configured to receive (and/or transmit) data during certain timeslots per mission.

Retransmission delay is another delay component that is also generated by the challenged nature of space environments, and represents the time required for a lost or corrupted bundle (or part of it) to be recovered and transmitted successfully at the destination. The retransmission delay depends on the reliability and recovery mechanism employed in the network, and, based on that mechanism, may actually comprise different types of delays. For an ARQ-based data recovery mechanism at the CL protocol, similar to the one employed in LTP [7], retransmission delay includes a series of consecutive retransmission rounds, where ACKs or NAKs (RSs in LTP) are transmitted towards the sending node and lost or corrupted data (reported missing) are retransmitted towards the destination node, until the complete bundle reception. Thus, the total retransmission delay is equal to the sum of the delays, for all retransmission rounds, including propagation, transmission, queueing, processing, and potentially waiting delays, for both ACK or NAK and retransmissions.

Although the space internetworking concept is a relatively new paradigm for space communications, the research community has identified the importance of delay estimation for data transmissions in space networks, and presented different methods to approach the subject. The majority of the published work that studies file transmission times in space environments examines CFDP or LTP, and focuses mainly on single-hop transmissions.

Lee and Baek in [86] [87] studied different CFDP schemes and delivery time expectations in deep-space scenarios, evaluating CFDP deferred NAK mode, with functionality equivalent to LTP [86], and CFDP immediate NAK mode [87]. In both papers, the authors considered single-hop file transmissions in a Mars-to-Earth scenario and defined rules for computing RTO intervals that minimize expected file-delivery time, with the constraint that throughput efficiency is not compromised. They evaluated variation in expected file-delivery time with varying BER, PDU length, and file size, and provided both analytic and simulation results. In [88], Sung and Gao studied K_a-band channels and their weather dependencies, modeled the effect of weather on K_a transmission as a Gilbert-Elliot channel with two weather states ("good" and "bad"), and analytically calculated file delivery latency with probabilities that depend on the channel weather on each transmission round. Propagation delay was the only component considered in that paper, and one of the measured metrics was the average number of transmission rounds (named spurts) required to complete the file delivery. One of the main features that pertain to the delivery delay calculation, in those three analytical studies [86] [87] [88], was the calculation of the expected number of transmission rounds, a single metric that corresponds to a precise arrival time at destination, rather than a detailed profile with different, plausible arrival times.

In [89], Gao and SeGui studied the performance of CFDP deferred NAK mode for Mars-to-Earth communications over K_a-band channels and evaluated file transfers over a deep-space link in terms of latency and storage requirements. The authors analyzed the probability distribution for varying numbers of transmission rounds that a file delivery may last and provided both analytical and simulation results for file transmissions of various sizes, with different error rates and "data completeness requirement" percentages. The analytical types used to extract transmission round probabilities were similar to those used in our methods. However, transmission delay was ignored as insignificant, and thus, the results were not sensitive to data rates. As in the other studies noted above, only single-hop transmission paths were examined in this paper.

Various studies have also been presented regarding the performance of transport protocols or protocol stack architectures, with respect to delivery delay, in different space environments including satellite [90] [91], cislunar [92] [93] [94] [95] [96] [97] [98], and deep-space [80] [99] [100] communications. The majority of them used file or PDU delivery time as a metric to evaluate the performance of the proposed configurations.

A multi-hop, heterogeneous satellite scenario that includes LEO and GEO satellites was studied in [90], where data file transfers were evaluated under different protocol stack approaches. In [91], file transmission times, transmission patterns, and throughput were studied as a means to compare the performance of different window-based and rate-based control mechanisms in satellite communications.

Cislunar communications and the varying conditions that characterize them were studied in [92] [93] [94] [95] [96] [97] [98]. The authors studied different protocols, such as the unreliable CFDP [92], TCP CL [93] [94], and LTP CL [95] [96], in terms of transmission effectiveness and goodput, and provided evaluation comparisons [97] [98] of different DTN convergence layers based on their goodput performance.

Within the same context, performance studies of protocols have also been presented for deep-space communications, through emulation measurements on the delivery latency [80] [99] [100]. In [99], the impact of LTP segment size, LTP block size, and bundle size in bundle delivery delay were studied in a single-hop, Mars-to-Earth file transmission scenario. In [80], ECTP was evaluated in terms of file deliveries in space; the authors studied the tradeoff between the gain in delivery latency and the loss in redundancy introduced by erasure coding, while the metrics used were normalized to the bandwidth-delay product. Finally, in [100], the authors assessed the impact of LTP aggregation in space communications in goodput as well as the file transmission times.

Although a variety of aforementioned studies attempted to measure or predict delivery delays in space internets, and besides the existence of different DTN protocols and services that provide -to some extent- functionality to this end, some important elements are still missing. In particular, there is a lack in administrative tools that assist network or mission operators in mission planning, or that provide estimations for future data transmissions. Moreover, existing automated delay estimators (e.g., for routing algorithms) provide rough predictions for data delivery time, since they ignore important components of the total delay (e.g., delays for recovery of lost data, queueing delays), and do not consider the entire contact plan and possible alternate routes for multi-hop bundle deliveries. Therefore, since miscalculation of delivery delay may result in sub-optimal functioning of several network functions, there is a need for more advanced protocols and methods that can accurately estimate the complete end-to-end delivery delay.

Furthermore, there is no work, to the best of our knowledge, to focus on the queueing delay component, notwithstanding the potential impact it might have on the total delay for space data transmissions. The significance of estimating packet queueing delays is not limited to the space internetworking context; it has been identified and studied since the early stages of the Internet, and numerous research papers have been published on the topic. In as early as 1985, Takagi and Kleinrock [101] studied a Carrier Sense Multiple Access with Collision Detection (CSMA-

CD) system and analytically calculated the average queueing delay of packets. In 1989, Demers et al. [102] suggested the use of the average queueing delay as a metric to control traffic in datagram networks, as opposed to flow control algorithms. Bolot [82] analyzed end-to-end packet delay using a probing process and discussed, among other factors, the queueing delay distribution. In the same context, Karam and Tobagi [103] studied voice traffic over the Internet and emphasized on the queueing component of the delay, as the only source of jitter, while Garetto and Towsley [104] studied TCP traffic generated by file transmissions and its significant impact on queueing delays in the Internet.

In the DTN paradigm, queueing delay modelling and analysis significantly differ from Internet-based internetworks; the main motivation for scientists to study queue lengths and the corresponding delays in DTNs has been their impact in routing efficiency, and various studies have been presented on the subject. In [105], the authors presented and compared different source routing algorithms based on the amount of knowledge that is available at the transmission initiation node. In particular, they exalted the knowledge of queueing occupancies in network nodes and stated that, amongst all "oracles" that provide different types of information (e.g., contact plan, buffer/queueing occupancies, traffic demand), the "queueing oracle" is the most difficult to realize, in order to achieve a complete knowledge of the queue occupancies in network nodes. In [106], a DTN-based link-state algorithm was applied on wireless networks in developing regions. The used link information included queue occupancy, among other data, and routing decisions considered the queueing delay that was calculated based on the most recent cached copy of the link information. Queueing delay has also been used as part of a performance metric in [107]; Seligman et al. proposed a DTN routing scheme with push and pull functions and measured its congestion control effectiveness with a time-weighted network storage metric. This metric was the product of the storage used by all queued messages and the amount of time they remain queued.

2.4 Routing in Interplanetary Internets

An essential networking function that impacts data transmissions in all network types, including the IPN, is routing. Routing of a bundle (or packet in general) is the procedure of defining the optimal path for conveying data from source to destination node in a network, based on some routing objective. Thus the routing algorithm on a space network decides on the optimal path based on the minimization of some cost or the maximization of some metric; the most commonly cited routing objective in space DTNs is the earliest delivery time for a bundle to reach its destination.

Therefore, the routing function is closely connected to the estimation of the delivery time; better delay estimators enhance the data routing procedure, while miscalculations or less accurate predictions may lead to suboptimal routing decisions. In complex space networks with multiple alternative routing paths from source to destination, the enhanced delay estimation may result in the selection of a better path, with faster data delivery. Another potential benefit is the improved distribution of limited resources that are present in space networks, since optimal routing and expedited data deliveries result in faster release of the available resources. Hence, improved routing decisions can also provide some indirect gains in the network, such as proactive congestion control, load balancing, better storage efficiency, etc. All of the aforementioned benefits justify the significance of accurate delivery delay calculations in routing, and motivated the research detailed in this thesis.

Given the challenges of space communications, described in subsection 2.1, the methods used for computing routes in a space network differ from those used in Internet routing; however in both environments, the procedure is not trivial, as networks may be complex and include many nodes. In recognition of the routing complexity, a network host plans a route for a data item before issuing it. The network state information on which this planning is based includes the network's "topology" and a list of all known connections between nodes. In a DTN-based network, this list may include additional information such as the speed of each connection and potentially the storage capacity of each node. However, network state information may change over time while traffic is traversing the network, and therefore the most efficient route may change while data are en route. For this reason, routing path may be updated at every branch point to take advantage of the most recently available information; consequently, the output of routing decisions is actually a neighboring branch point to transmit the data to, with the expectation that this branch point is on the optimal path for conveying the data to destination.

As far as Internet is concerned, routing decisions can be taken with high confidence because information regarding changes in network state information can be propagated quickly, and, therefore, each node's current understanding of the state of the network is almost always correct. However, that understanding may be incomplete, because routing in the network may be compartmentalized: the network state information exposed to any node may be limited to nodes in the local "domain" (including nodes that are on the border between the local domain and adjacent domains that serve as "gateways" between domains). Nonetheless, routing decisions can be made confidently in the expectation that the distribution of network state information within other domains is as rapid and comprehensive as within the local domain.

On the other hand, in a space network, or in other challenged networks where DTN architecture applies, network nodes may lack accurate network state information for other nodes: since connectivity is intermittent and/or signal propagation times are long, changes in

the network state may occur more rapidly than information about those changes can be propagated. Routing is still a matter of choosing a neighboring node to transmit the data directly to, but determination of the best path is constrained by lack of knowledge of the current state of the network; it may even not be possible to transmit immediately to the neighboring node that is the nearest branch point on the best path.

2.4.1 Routing in DTNs

Strategies for dealing with the aforementioned obstacles have been the focus of DTN research for longer than a decade. A key discriminator among these strategies is the assumed timeliness and accuracy of the network state information available to every node in the network. Approaches that assume minimal accurate network state information have historically been considered "opportunistic" while those that assume complete network state information are regarded as "deterministic."

Significant algorithms that belong to the category of opportunistic routing include single-hop multi-cast forwarding, such as the Spray-and-Wait algorithm [108], in-network exchange of link information, such as DTLSR [106], probabilistic analysis of predicted node contact such as PRoPHET algorithm [109], utility-based schemes [110], erasure-coding-based methods [111] [112], and a great variety of routing algorithms and protocols. All of these rely on the exchange of infrastructure and/or in-network measurements in a timely manner to support on-demand calculations of routes and forwarding hops. Opportunistic approaches may use a single copy of the message [113], or may apply a replication-based strategy [114]; using this strategy, messages are typically duplicated a fixed number of times, a variable number of times based on contact probability, or even in epidemic fashion [115], i.e., upon every encounter between nodes. In networks with high node mobility and nearly random contact establishment, the delivery success rate of multiple-copy class of approaches is higher than approaches that rely on the accuracy of current network state information.

On the other hand, in networks where contacts are predictable, such as the IPN, more deterministic algorithms can achieve higher rates of delivery success with less waste of bandwidth and buffer space. Typical algorithms that belong to this category include Space-Time Routing Framework [116], MARVIN [117], and CGR [9]. The first approach exploits the predictability of node motion to identify paths over space and time, and construct routing tables, accordingly. In the last two approaches, accurate contact predictions are distributed to the nodes in the network, enabling network graphs to be built, which are then used to make routing decisions on a hop-by-hop basis. MARVIN encodes information about the operational

environment (planetary ephemeris data) and infers contact opportunities from this knowledge. The CGR algorithm is a formulation of the perfect knowledge approach, and is currently being extended to work in less-perfect knowledge systems [118]. CGR constitutes the most suitable routing choice for the SSI, and is also acknowledged by the IOAG in the definition of operations concept for SSI [4]. Therefore, it comprises a central part in our research; we evaluate its use with respect to the delivery delay estimations, and we attempt to improve its prediction capabilities, providing a modified version of CGR. A detailed description of CGR is provided in the following subsection.

2.4.2 Contact Graph Routing

Contact Graph Routing [9] [119] [118] is a dynamic algorithm that computes routes through a time-varying topology of scheduled communication contacts in a DTN network. It can be successfully applied in different SSI environments, including deep-space communications [120], lunar communications [121], and LEO satellite communications [122], and in general it covers the majority of cases where link availability is known a priori, outperforming routing algorithms designed for terrestrial DTNs [123]. The applicability of CGR in space networks in general has been also proven with results from real experimental experiences [118], including the DINET experiment (employing the EPOXI space cruise) [46], the Japan Aerospace Exploration Agency (JAXA)-NASA joint experiments with JAXA's GEO relay satellite called Data Relay Test Satellite (DRTS) [124], the Space Data Routers European Project [24] [125], and the pilot operation of a DTN implementation on the ISS [48] [49].

The perfect connectivity knowledge that CGR assumes does not reduce the complexity of the route computations, as links are intermittent and the network connectivity varies through time. The basic strategy of CGR is to take advantage of the fact that, since space flight communication operations are planned in detail by mission operators, the communication routes between any pair of "bundle agents" in a population of nodes, all of which have been informed of one another's plans, can be inferred from those plans rather than discovered via dialogue.

The foundation of CGR is the "contact plan", a time-ordered list of scheduled, anticipated changes in the topology of the network. The entries in this list are of two types, "contacts" and "ranges" [9]. Each contact has the following information:

- The starting time of the communication interval;
- The stop time of this interval;
- The transmitting node number;

- The receiving node number; and
- The planned rate of transmission from transmitting to receiving node over this interval.

Note that the contact has a unidirectional concept; if communication between nodes *A* and *B* during a time interval [*T1 – T2*] is bidirectional, there should be two contacts in the contact plan, with the following elements:

[{*T1, T2, A, B, tx_rate_A_B*} and {*T1, T2, B, A, tx_rate_B_A*}]

The range includes the following information:

- The starting time of the communication interval;
- The stop time of this interval;
- The transmitting node number;
- The receiving node number; and
- The anticipated propagation delay between transmitting and receiving nodes, in light-seconds (i.e., OWLT).

Ranges and contacts should overlap, that is, data routing and transmission between two nodes during a timespan can happen only if both a range and a contact are active during this time span. We note that the contact information also defines the *volume* (or *capacity*) of the contact, which is the maximum amount of data that can be transferred during the contact, given by the product of contact interval (*T2 - T1*) and contact's nominal transmission rate.

Each node uses the contact plan information to build a "routing table" data structure. A routing table is a list of "route lists," one route list for every possible destination node in the network. Each route in the route list for node D identifies a path from the local node to destination node D that begins with transmission to one of the local node's neighbors in the network, i.e., the initial receiving node for the route, termed the route's "entry node." The route list entry for each neighbor contains the best route that begins with transmission to that neighbor. The route also includes information on (i) all other contacts that constitute the remaining segments of the route's end-to-end path, (ii) its estimated "cost" (e.g., the end-to-end delivery latency), and (iii) the "forfeit time," i.e., the latest time by which the bundle must have been forwarded to the route's entry node in order to have any chance of traversing this route.

The CGR algorithm is actually a family of algorithms with similar behavior; from those algorithms, we indicatively describe the standard algorithm that is included in the ION implementation [11], which is (at the time of writing) the only implementation of the CGR algorithm, and contains the core functionality described in [126] and [9]. As observed in Figure 2-6, whenever a new bundle with destination node D is passed to CGR for routing decision, a route list has to be determined for node D. If the contact plan has changed since the previous

routes' computation (or it is a first transmission to a new destination node), routes have to be recomputed. This procedure comprises two distinct steps:

i) Construction of an abstract contact graph, that is, a directed acyclic graph whose root is a notional contact from the local node to itself and whose other vertices are all other contacts that can contribute to some end-to-end path to node D, with no loops, and the terminal vertex being a contact from node D to itself.

ii) Execution of several series of Dijkstra searches [127] within the graph, one series for each payload class. Each search outputs the best route (based on the routing objective) from root to terminal vertex, adds it to the generated list of routes, and removes its root contact from the contact graph; the process continues iteratively by computing all available routes, finishes when no other routes can be found, and returns the list of obtained routes. Note that the routing objective used here may vary; however, the typical objective, introduced in [126] and widely used since, is earliest estimated delivery time.

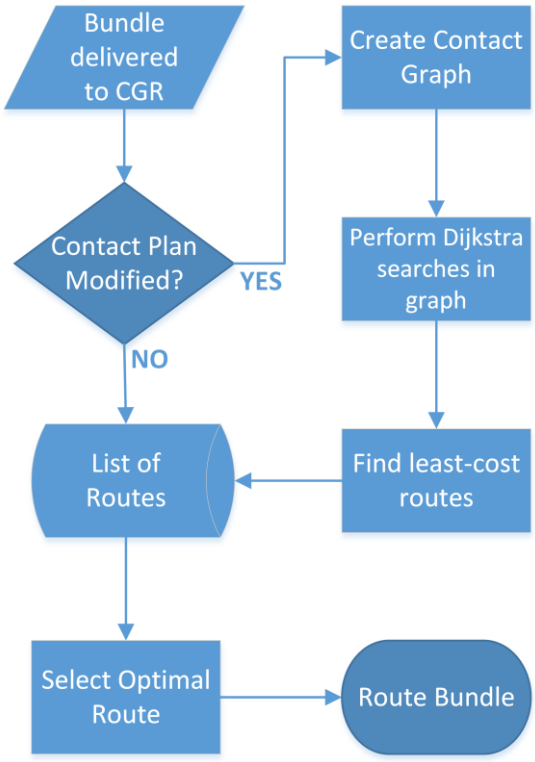

Figure 2-6 Contact Graph Routing Procedure

On the other hand, if there is no contact plan change when a new bundle is delivered to CGR, the decision is taken from the existing routes. This check is performed to avoid re-

calculation of routes, and re-execution of processing-demanding Dijkstra searches in the contact graph, and thus reduce the computational overhead of the CGR algorithm.

In both cases (i.e., with and without route recalculation), CGR uses the retrieved route list to decide on the optimal path for the input bundle. To this end, CGR searches the route list for available routes destined at D. Some of the routes in the list may be unusable. For example, a route may be temporarily unavailable (e.g., when transmission to the entry node is "blocked" due to a detected or asserted loss of connectivity), the expected delivery time on a route may be greater than the bundle's time-to-live, or the "residual capacity" (i.e., the capacity that has not been allocated yet) of the initial contact on the route may not be enough to contain the bundle. Out of the usable routes, CGR chooses the one with the lowest cost, based on the employed routing objective, and queues the bundle for transmission to that route's entry node. If the list of bundles queued for transmission on some route is non-empty at the time that the route's forfeit time is reached, new routes must be computed for all of these bundles.

An important element of the CGR algorithm that makes it ideal for the IPN is that the routes in the route list need not be continuous. Once a bundle has reached an intermediate node through the path to destination, it may reside in storage for some period of time, awaiting the start time of the first contact to the next node; this process continues until the final segment of the path, which concludes at the destination node.

Another key advantage of CGR is that, like Internet routing, it can be done with high confidence, as it is based on accurate information about the network's topology. With CGR, the topology on which routing is based is not the current topology but rather an anticipated time-varying topology. Nevertheless, since changes in the network topology are scheduled in the course of mission planning, information about these changes can be propagated long before they occur. In this way, each node's understanding of the topology of the network at any moment is almost always correct: while propagation of information about network topology changes is slow, it is still faster than the rate at which the changes themselves occur. Therefore, although the neighboring node that is the nearest branch point on the best path may be unreachable at the routing decision moment, calculation of optimal path is possible because topology knowledge is generally accurate.

Ever since CGR first appeared [119], the research community has worked on improving its functionality and usage. In [126], the authors proposed Enhanced Contact Graph Routing (ECGR), where they included Dijkstra algorithm [127] in routing path selection of CGR, and replaced the earliest-forfeit-time (initially proposed to minimize underutilization of large-capacity links and waste of expensive communication opportunities [119]), with earliest-arrival-time as the routing objective. Since this was the standard version accepted by the community and included in the ION DTN implementation [128], it is the state-of-the-art CGR algorithm, described previously in this subsection, and used for comparison purposes in our

evaluation. Various other works were also proposed to further enhance the usage and performance of CGR. In [120], the authors proposed the use of source routing, and suggested that path information, extracted at the source node, be encoded and transmitted in a Bundle Extension Block [6], effectively reducing the complexity of the algorithm. For the same purpose of reducing CGR processing requirements, the authors in [129] proposed the use of Cache-CGR, a computationally efficient CGR version that caches information about next neighbors and respective contact residual capacities, to avoid frequent route re-computations. The introduction of iCGR [130] extended the applicability of CGR in sensor-based internetworks, as an overlay routing across various homogeneous domains. Finally, CGR was exploited and adjusted accordingly to cover different networking environments that feature intermittent and scheduled connectivity, such as the Connectivity Plan Routing Protocol (CARPOOL) protocol [131], which exploits the connectivity plan of public transportation, achieving high delivery ratio with minimum overhead [132].

The CGR family of algorithms, however, is missing some important aspects. Although the forwarding decisions are made dynamically, end-to-end delays are estimated based on information that may not reflect current network dynamics: important information such as queueing delays is omitted. Therefore, since the routing objective, i.e., earliest delivery time, is not accurately estimated, routing decisions may be performed in a sub-optimal way.

In this context, part of the research conducted in the framework of the present thesis is stimulated by the importance of queueing delays in routing [105]. We acknowledge the absence of IPN routing schemes that take into account queueing delays in delivery delay estimations and routing decisions, accordingly, and we attempt to enhance CGR algorithm with improved delay estimators that incorporate the available information on queueing delay, as well.

The major assumption for the CGR functionality is the perfect connectivity knowledge. The different, aforementioned versions of CGR worked on the improvement of the algorithm *per se*, without examining the contact plan design procedure or the dissemination of the obtained contact plan through the network. The former issue of contact plan design has been recently studied and several works have been published on the topic [133] [134] [135] [136]. In the CGR algorithms studied and introduced in this thesis, we also assume that the contact plan is accurately designed, and consider the specifics of this design procedure out of scope.

The contact plan dissemination procedure has also been of interest during the recent years, particularly in standardization discussions about DTN management protocols [137] that CCSDS and DTNWG are involved in. However, no specific dissemination protocols have been presented to this end; therefore, in our research, we also design the CPUP that fills this gap and efficiently manages the dissemination of contact plan updates through the network.

2.5 End-to-End Retransmission Timeout

The importance of accurate retransmission timers is a subject that has concerned researchers since the introduction of the end-to-end reliability concept, at the early stages of the Internet, and it is bound together with the performance of reliable end-to-end transport protocols like TCP. Proper estimation of TCP retransmission timers was intended to find a balance between timely detections of packet losses or delays, and avoidance of unnecessary retransmissions, and was strongly correlated to the RTT, i.e., the time interval between the transmission of a packet and the reception of its acknowledgement. The initial TCP algorithm [138] included a smoothed RTT value estimation (SRTT) using a low-pass filter on the RTT measurements, and calculated RTO as a product of the SRTT: $RTO = \beta * SRTT$, with a proposed constant value of $\beta = 2$. Since then, the research community has attempted to improve the efficiency of RTO calculation with various ways, e.g., by overcoming the ambiguities of RTT measurements for retransmitted packets [139] [140], by incorporating the measured variance of RTT in the RTO calculation and providing a more balanced estimator [141], by facing the problem of spurious retransmissions [140], or eliminating "RTO outliers" [142], etc.

As challenged network environments emerged, the standardized TCP RTO calculation [143] was clearly unable to cope with the challenges of intermittent connectivity in WSN or ad-hoc networks [144] [145] [146] [147]. The proposed solutions attempted to provide better RTT estimators [144] for the challenged environments, or to differentiate link disconnections from network congestion using connection-oriented RTT estimations [145] or control messages [146] [147].

The importance of properly configured retransmission timers is an even more crucial task in space internets [8] [148] [149], due to the limited resources and the necessity of keeping the retransmission overhead as low as possible. However, high RTT variances make the configuration of retransmission timers a challenging task [148]. In [149], Akan et al. have applied a TCP-like retransmission timer and attempted to overcome the disruptive interplanetary communications by introducing the "blackout state;" during this state, although new packets are not transmitted and congestion control mechanism is suspended, packets with expired timers are retransmitted normally.

Among the most significant end-to-end protocols that reside on top of the DTN architecture, DTPC [8], CFDP (reliable mode) [20], and ECTP [80] employ similar retransmission mechanisms, triggered by RTO expirations, to achieve reliable data transfers. The retransmission schemes of these protocols, however, are missing an automated way to dynamically configure RTOs based on some input, e.g., the contact plan or network state, and provide accurate timers. In the original DTPC implementation [8], the timeout interval was

calculated equal to the data item lifetime divided by the maximum number of retransmission rounds plus one. The rationale for this configuration was to exploit every possible retransmission opportunity for each data item. However, its static value does not leverage the available network information, and, depending on the DTPC parameters and the topology, it might significantly delay the recovery of the lost data items, or lead to a great amount of redundant retransmissions. In the present thesis, we propose an updated retransmission framework that constitutes the first attempt for a dynamic calculation of retransmission timeouts in DTNs, exploiting accurate end-to-end delay predictions based on cross-layer provided information.

Chapter 3 Bundle Delivery Time Estimation

In this Chapter, we describe Bundle Delivery Time Estimation tool, which was designed to provide, for a given bundle that will be transmitted at a specific future time, analytical results on the plausible delivery times at destination, along with the corresponding probabilities. BDTE exploits contact plan information, and an instrumentation database (DB) that gets management data from each node to obtain statistics on the error rates. Using these elements as input, we develop an analytical method that forecasts future error rates, and estimates and sums up the different delay components that the bundle is expected to follow en route to destination, while the expected routes are obtained with the use of CGR algorithm. Furthermore, we develop an application that exploits this method and implements the BDTE functionality.

The Chapter continues with the description of the BDTE operation concept (Section 3.1). In Section 3.2, we present the main BDTE functionality including the core delivery time estimation algorithm. In Section 3.3, we describe the statistics DB, and in particular the useful information that BDTE extracts from the DB, and exploits to analytically calculate error rates for past time intervals, with a method described in Section 3.4. In Section 3.5, we present the statistical forecasting method that predicts future error rates based on past values. Finally, in Section 3.6, we discuss the assumptions that we have applied in our method.

3.1 Description

BDTE [150] is, in essence, an administrative network simulation tool that applies the CGR algorithm on every network node throughout the route of the bundle. The BDTE algorithm performs hop-by-hop calculations, provides possible arrival times for each hop of the path to destination, and continues iteratively through the entire predicted bundle route, ultimately resulting in the arrival time at the bundle destination.

The calculated latency for each hop is based on both deterministic and stochastic latency components. The former comprises propagation delay (i.e., OWLT), and transmission delay for bundle delivery (including overhead) via the link channel, i.e., the interval that will be required to transmit the bundle given the transmission rate on the link. The stochastic component is introduced by uncorrected channel errors, which compel packet retransmissions; it accounts for the propagation and transmission delay for retransmitted packets.

In our analysis, we make some simplifying assumptions. We assume that processing delays are insignificant in contrast to the long propagation delays. We also assume that the bundle is

transmitted at the highest possible priority and, consequently, omit consideration of queueing delays. Calculation of the queueing delay component is by itself an important and challenging task that is studied in detail in the next Chapter of this thesis.

BDTE's computation is based on the fact that the deterministic components of the bundle's latency can be accurately calculated, whereas the stochastic latency can only be statistically predicted using each link's history observations. The result of this analysis is a link error rate forecast that provides several estimates of the number of transmission rounds that may be required for successful bundle delivery to the next node; each estimate is assigned a probability. For each possible number of transmission rounds, a different delivery time to the next node is calculated. Each of the calculated delivery times is then used as the transmission initiation time for the next hop of the route, and a new simulation is conducted, accordingly. This method continues consecutively to the final destination and ultimately results in a set of distinct bundle arrival times, with different probabilities that theoretically sum up to 100%.

This network simulation and all analyses are performed in an administrative node that may or may not be a part of the space internetwork. This node is assumed to have current knowledge of the overall network contact plan, as well as access to a central DB that contains network instrumentation statistics through time. Past measurements from this statistics DB are used to predict channel error rates for future bundle transmissions. The accuracy of BDTE will always be limited to the accuracy of these information resources.

We note that BER observations are performed in the convergence layer and thus incorporate the uncorrected errors that have eluded channel Forward Error Correction (FEC).

The notation used in our analysis and algorithms is displayed in Table 3-1.

Table 3-1 Bundle Delivery Time Estimation Analysis: Notation

Symbol	Quantity
P{p.r. >< = k}	Probability that the number of packet transmission rounds is bigger than / smaller than / equal to *k*
P{t.r. >< = k}	Probability that the number of bundle transmission rounds is bigger than / smaller than / equal to *k*
BDT	Bundle Delivery Time
TPList	Time-Probability List: a list of BDTs at destination and the corresponding probabilities
MBS	Mean Block Size
MPL	Mean Packet Length
ANP	Average Number of Packets (per bundle)
ATR	Average Transmission Rounds

3.2 Main BDTE functionality

In Table 3-2, we present the core BDTE algorithm that performs the simulations and leads to possible bundle delivery times at destination. This is a recursive algorithm that uses equations developed in the following sections, and concludes when *next_node* is the *destination_node*, i.e., when last hop probabilities have been calculated. Its output is a list of pairs, termed a "time probability list" (*TPList*). Each of these pairs consists of a bundle delivery time at the receiving node and the corresponding probability.

Table 3-2 BDTE Algorithm

Input: *sending_node*, *destination_node*, *bundle_creation_time*, *bundle_lifetime*, *bundle_size*, *CL_packet_size*
Output: *TPList*

// Initialization
current_node = *sending_node*;
initial_probability = 1;
start_time = *bundle_creation_time*;
bundle_expiration_time = *start_time* + *bundle_lifetime*;

CalculateNextHopDeliveryTimes (*current_node*, *destination_node*, *initial_probability*, *start_time*)
[*next_contact*, *start_xmit_time*] = simulateCGR(*current_node*, *destination_node*, *start_time*);
span = [*current_node*, *next_contact.to_node*];
BER_time_series = StatisticDB.ExtractBER(*span*); // Procedure analyzed in 3.3-3.4
future_BER = time_series_forecasting(*BER_time_series*, *start_xmit_time*); // Section 3.5
future_PER = calc_PER_from_BER(*future_BER*); // Using Eq. (5)
cumulative_probability = 0;
for (*k* =1; *k* <= *MAX_TRANSMISSION_ROUNDS*; *k*++)
cumulative_probability += $P\{t.r. = k\}$; // Calculated from Eq. (13)
if (*cumulative_probability* >= *PROBABILITY_THRESHOLD*)
break;
end if
end for // *k* has now been set as the max number of transmission rounds to be examined
for (*j*=1; *j* <= *k*; *j*++)
$total\_propagation\_delay = next\_contact.OWLT \cdot (2j-1)$; // we omit last round's ACK
$total\_transmission\_delay = bundle\ size \cdot \left(1 + \sum_{i=1}^{j-1} PER^i\right) \Big/ next\_contact.tx\_rate$;
$BDT_j = start\_time + total\_propagation\_delay + total\_transmission\_delay$;
$P_j = initial\_probability \cdot P\{t.r. = j\}$; // Calculated from Eq. (13)
if (*next_contact.to_node* == *destination_node*)
TPList.add([BDT_j, P_j]);
continue; // with next transmission round *j*
end if
current_node = *next_contact.to_node*;
start_time = BDT_j;
initial_probability = P_j;
CalculateNextHopDeliveryTimes(*current_node*, *destination_node*,
initial_probability, *start_time*);
end for
end // CalculateNextHopDeliveryTimes
return *TPList*;

Note that the two configuration parameters *MAX_TRANSMISSION_ROUNDS* and *PROBABILITY_THRESHOLD* are globally configured a priori and not given as application input. The reason for using these parameters is to control the number of iterations and reduce the computational cost. Since the number of transmission rounds could theoretically be infinite with probability that tends to zero, we can either set a maximum number of transmission rounds or a probability threshold below which the calculation is negligible. In our testing configuration we have used both of these control parameters and have set *MAX_TRANSMISSION_ROUNDS* = 4 and *PROBABILITY_THRESHOLD* = 0.001.

3.3 Statistics Database and Obtained Information

Each DTN node in the IPN is expected to keep records of several types of events in both bundle and convergence layer and measure incoming, as well as outgoing bytes, bundles, and CL packets. For the purposes of BDTE, we assume that these measurements are transmitted to a central administrative node and stored in an instrumentation DB for detecting network defects and failures and for further processing through Delay/Disruption Tolerant Network Management Protocol (DTNMP) [137]. Although the network management procedures have not been standardized yet, and hence, DB structures are not in a final form, ION implementation includes an initial instrumentation DB that collects various information for the DTN network management procedures.

Some of the measurements stored in this central DB can be used to evaluate the quality of a given link that is a part of the space internetwork and predict its behavior in the future. In order to quantify the link quality, we consider a metric called *Average Transmission Rounds* (*ATR*) that indicates the anticipated number of retransmission rounds for the convergence layer block. We note here that this applies on the LTP CL, on which he have focused, since it is the most commonly accepted CL protocol for long-haul links, and was adopted by the CCSDS. We expect that the recovery or retransmission delay for different CL protocols could be calculated in a similar way, based on the recovery strategy used by these protocols (e.g., ARQ, FEC), in a straightforward way.

In Eq. (1), we consider the specific LTP export span in order to calculate the average number of retransmission rounds required to deliver each LTP block successfully to the next hop, and thus calculate *ATR*. The same results could be extracted, if the corresponding LTP import span was used.

$$ATR = 1 + Average\ LTP\ retransmission\ rounds$$
$$= 1 + \frac{NAK\ reports\ rcvd + checkpoints\ rexmitted}{xmit\ sessions\ completed + xmit\ session\ cancelled} \quad (1)$$

$$MBS = \frac{Data\ segments\ dequeued}{xmit\ sessions\ completed + xmit\ sessions\ cancelled} \quad (2)$$

$$MPL = \frac{Data\ bytes\ dequeued}{Data\ segments\ dequeued} \quad (3)$$

$$ANP = round\left(\frac{Mean\ block\ size}{Mean\ packet\ length}\right) \quad (4)$$

With Eq. (1) we can calculate *ATR* for time intervals stored in the DB. Using *ATR* along with the useful metrics *MBS*, *MPL*, and *ANP* extracted from Eq. (2), (3), and (4), we can estimate the average observed BER with a technique that is introduced in the next subsection. We note that we have used a one-to-one correspondence between bundle and LTP block. Even when some bundle is encapsulated in a larger LTP block together with other bundles, it will be delivered to the receiver BP agent when the entire LTP block arrives successfully at the receiver side. Therefore the calculated metrics apply both to the bundle and the corresponding LTP block.

3.4 Error Rate Approximation Method

Our algorithm calculates the distinct probabilities for each number of transmission rounds that bundle delivery may require. The computed *ATR*, however, is the average number of transmission rounds and includes no information about the probabilities of the distinct number of rounds, which is deemed necessary, in order to obtain a detailed bundle delivery profile. For this reason, we need to obtain the BER using *ATR*. However, there can be no such straightforward calculation; therefore, we present here an inverse method that uses the explicit calculation of *ATR* from BER and applies a binary search algorithm to estimate BER with adequate precision.

In our analysis, we assume that bit errors are independent. If *s* is the packet length in bits, and assuming that all packets are of equal length, the loss probability of each packet is termed *Packet Error Rate* (*PER*) and can be calculated as follows:

$$PER = 1 - (1 - BER)^s, \text{ where } s = 8 \cdot MPL \quad (5)$$

The probability that a packet reaches the next node on the end-to-end path at *exactly* one transmission round (or else in less than two rounds) is:

$$P\{\text{p.r.} = 1\} = P\{\text{p.r.} < 2\} = 1 - PER \tag{6}$$

Bundle sizes ordinarily exceed the CL packet size. So, when BP delivers a bundle to the convergence layer beneath, the bundle is normally truncated into multiple segments to be delivered to the link layer. Since we have already assumed independent bit errors, the probability that a given packet is successfully transferred is independent from the transfer success probability of all other packets. Therefore, if a bundle consists of *n* packets, the probability $P\{t.r. = 1\}$ that its transmission lasts *exactly* one round corresponds to the probability that all *n* packets are successfully transmitted during the first transmission round, which equals to the product of the probabilities $P\{p.r. = 1\}$, for all *n* bundle packets:

$$P\{\text{t.r.} = 1\} = P\{\text{t.r.} < 2\} = \left(1 - PER\right)^n \tag{7}$$

In our algorithm, *n* is equal to *ANP*, which is calculated using Eq. (4). In order to calculate the probabilities for transmission rounds greater than 1, we have to consider the convergence protocol functionality. Here we consider the use of LTP that is, as explained previously, a CL protocol-of-choice in CCSDS standardization procedures, to be used in high-latency space links, and which includes ARQ-based retransmissions. Requests for retransmissions are initiated upon the delivery of the last packet of the block, namely the LTP segment flagged as *End of Block* (*EOB*). In case of an unsuccessful EOB delivery, no positive or negative acknowledgment report is sent from the destination node. Hence, the retransmission timer at the sender expires and triggers EOB retransmission, thus delaying the block delivery by one round. The exact amount of delay produced is equal to the timeout time length plus the OWLT required to retransmit the EOB. In ION implementation [11], timeout is computed as twice the OWLT plus the imputed inbound and outbound queueing delays in both communicating nodes, for the enqueuing and dequeuing of EOB and the RS. As noted above, however, we assume that these queueing delays are insignificant compared to the long space propagation delays. Consequently, timeout interval equals to 2·OWLT (i.e., one RTT). Hence, the time granularity used in our analysis is in terms of transmission rounds.

An LTP block, which we here assume corresponds to a single bundle, is truncated into *n* segments, *n*-1 regular red-block segments and a last segment, denoted as EOB. Therefore, the probability that a packet is transmitted in less than *k* rounds equals to 1 minus the probability that the packet is not successfully transferred in the *k*-1 first rounds:

$$P\{\text{p.r.} < k\} = 1 - PER^{k-1} \tag{8}$$

The corresponding probability for n-1 packets to be transferred in less than k rounds is:

$$P\{\text{t.r.} < k\} = \left(1 - PER^{k-1}\right)^{n-1} \tag{9}$$

Using Eq. (9) we calculate the probability for the first n-1 bundle packets to be successfully delivered in exactly k transmission rounds, $k \geq 1$:

$$\begin{aligned} &P\{\text{t.r.} = k, \text{ for } n-1 \text{ red-block segments}\} = \\ &= P\{\text{t.r.} < k+1\} - P\{\text{t.r.} < k\} = \\ &= \left(1 - PER^{k}\right)^{n-1} - \left(1 - PER^{k-1}\right)^{n-1}, \end{aligned} \tag{10}$$

which is always greater or equal to zero, since

$$\begin{aligned} &PER \leq 1 \Rightarrow \\ &PER^k \leq PER^{k-1} \Rightarrow \\ &1 - PER^{k-1} \leq 1 - PER^k \Rightarrow \\ &\left(1 - PER^{k-1}\right)^{n-1} \leq \left(1 - PER^k\right)^{n-1} \Rightarrow \\ &\left(1 - PER^k\right)^{n-1} - \left(1 - PER^{k-1}\right)^{n-1} \geq 0 \end{aligned}$$

We have thus calculated the probability for the transmission rounds of the n-1 first LTP segments. In order to estimate the total transmission time of the LTP block, we have to consider its checkpoint-based ARQ functionality, assuming that LTP configuration incorporates one checkpoint per block, the EOB segment. In case EOB transfer fails, the destination node transmits no RS, resulting in timer expiry and EOB retransmission. This event triggers extra transmission rounds until the successful delivery of EOB.

We denote the probability of m number of lost EOBs during a bundle transmission as $P\{m \text{ lost EOBs}\}$. For a bundle transmission that lasts k rounds, the probability that m EOB packets are lost in the first k-1 rounds is given by the probability mass function of the binomial distribution:

$$P\{m \text{ lost EOBs in } k\text{-}1 \text{ rounds}\} = \binom{k-1}{m} \cdot PER^m \cdot \left(1 - PER\right)^{k-1-m}$$

The last EOB in the *k*-th round arrives successfully with probability 1-PER, to complete bundle reception after exactly *k* rounds. Thus, the total probability that exactly *m* EOBs are erroneously transferred in the first *k*-1 transmission rounds and the *k*-th EOB is successfully transferred is:

$$P\{m \text{ lost EOBs}\} = \binom{k-1}{m} \cdot PER^m \cdot (1-PER)^{k-m} \tag{11}$$

In case the EOB is retransmitted in *m* rounds ($m \leq k$), in a bundle transmission that lasts *k* rounds, the other red-data segments of the block are not transmitted during these rounds. Therefore, the successful delivery of the *n*-1 segments has to be achieved within the remaining *k-m* rounds, with a probability computed with Eq. (10). Hence, there are *k* distinct cases for a bundle transmission that lasts exactly *k* rounds: the loss of 0, 1, 2,..., *k*-1 EOBs. The total probability of a successful bundle delivery in exactly *k* transmission rounds is the sum:

$$\begin{aligned} &P\{\text{t.r.} = k \mid \text{error-free backward channel}\} = \\ &\sum_{m=0}^{k-1} P\{m \text{ lost EOBs}\} \cdot P\{\text{t.r.} = k-m, \text{ for } n-1 \text{ red-block segments}\} = \\ &\sum_{m=0}^{k-1} \binom{k-1}{m} \cdot PER^m \cdot (1-PER)^{k-m} \cdot \left[\left(1-PER^{k-m}\right)^{n-1} - \left(1-PER^{k-m-1}\right)^{n-1} \right], \end{aligned} \tag{12}$$

for $k > 1$ and $m < k$. If $k = 1$, the probability is calculated from Eq. (7). In the degenerate case where a bundle is incorporated into a single LTP segment, which is also the EOB segment, the probability of *k* transmission rounds equals to:

$$P\{k-1 \text{ lost EOBs}\} = PER^{k-1}$$

We have so far assumed that the return (acknowledgment) channel is error-free. The plausibility of this assumption may be increased by the use of small Report Segments (RSs), or by the use of strong encoding schemes at the underlying link and/or physical layers, that greatly reduce the statistical significance of a RS loss. Nevertheless, the DB field *checkpoints retransmitted*, in Eq. (1), includes the lost RSs as well. So, for a more accurate result, RS error rate can be incorporated in Eq. (12) by adding the loss probability of *j* RSs, with $0 \leq j < k$ and $j + m < k$. Useful RSs (*i.e.*, not retransmissions of already received RSs, e.g., when the *RA* is lost) are not transmitted when an EOB has not successfully arrived. Thus, RSs are not transmitted in the *m* rounds in which EOB is lost, but only in *k-m* rounds. The successful or unsuccessful

transmission of the *k*-th RS (i.e., the one transmitted after the *k*-th round) is not considered, since the transmission has been completed at *k* rounds. Therefore, the considered values for *j* losses are [0, 1, …, *k*-*m*-1], the number of different combinations for *j* is $\binom{k-m-1}{j}$ and the corresponding probability for each *j* is:

$$P\{j \text{ lost RSs in } k-m-1 \text{ rounds}\} = \binom{k-m-1}{j} \cdot PER_{RS}{}^{j} \cdot \left(1-PER_{RS}\right)^{k-m-1-j},$$

where PER_{RS} is the loss rate for the RSs, calculated from Eq. (5), if we assume the same BER in the return channel. The final probability becomes:

$$P\{\text{t.r.} = k\} = \sum_{j=0}^{k-m-1}\sum_{m=0}^{k-1}\left\{\begin{matrix} P\{j \text{ lost RSs in } k-m-1 \text{ rounds}\} \cdot P\{m \text{ lost EOBs}\} \cdot \\ P\{\text{t.r.} = k-m-j, \text{ for } n-1 \text{ red segments}\} \end{matrix}\right\} \Rightarrow$$

$$P\{\text{t.r.} = k\} = \sum_{j=0}^{k-m-1}\sum_{m=0}^{k-1}\left\{\begin{matrix} \binom{k-m-1}{j} \cdot PER_{RS}{}^{j} \cdot \left(1-PER_{RS}\right)^{k-m-1-j} \binom{k-1}{m} \cdot PER^{m} \cdot \\ \left(1-PER\right)^{k-m} \cdot \left[\left(1-PER^{k-m-j}\right)^{n-1} - \left(1-PER^{k-m-j-1}\right)^{n-1}\right] \end{matrix}\right\} \quad (13)$$

Theoretically, the number *k* of transmission rounds could be infinite, with probability that tends to zero. However, as we have already mentioned, we apply the *MAX_TRANSMISSION_ROUNDS* and *PROBABILITY_THRESHOLD* filters, in order to reduce the calculation cost for insignificant probabilities. Since we have so far calculated a finite number of probabilities for the distinct number *k* of transmission rounds, we can use them to evaluate *ATR*, as shown below:

$$ATR = \sum_{k}\left(k \cdot P\{\text{t.r.} = k\}\right) \quad (14)$$

In Eq. (5)-(14), we have presented a method that uses *BER* to compute the probabilities for a bundle to be successfully transmitted in 1, 2, ..., *k* rounds, as well as the aggregated *ATR* number. This method has a twofold significance: If BER is a known (or estimated) quantity, we can extract useful information about the bundle transmission in terms of delivery times at destination. On the other hand, if *ATR* is known, as with past DB measurements from Eq. (1), this method can be used in an inverse binary search algorithm (see

Table 3-3), which attempts to approximate PER from the known *ATR*. Channel BER can then be inferred, using the inverse of Eq. (5). This inverse calculation method has a unique solution because *ATR* is a genuinely ascending function of PER.

The PER estimation algorithm parameters that need to be configured are the starting *MIN_BER*, *MAX_BER*, and the *MAX_ ITERATIONS*, which are configured as global parameters and are not used as application input. Similarly, an error threshold could also be applied to lead to a desired accuracy. However, this is a matter of application configuration and should be user-defined, based on the desired results and the computational resources. In our sample configuration, we have used *MIN_BER* = 10^{-8}, *MAX_BER* = 10^{-5}, and *MAX_ ITERATIONS* = 12.

Table 3-3 PER Estimation Algorithm

```
Input: ATR
Output: PER

times = MAX_ ITERATIONS;
min = calc_PER_from_BER(MIN_BER); // Using Eq. (5)
max = calc_PER_from_BER(MAX_BER); // Using Eq. (5)

while ((times >= 0) && (min < max))
        times--;
        per = (min + max) / 2;
        ATRtemp = calcATR(per); // Using Eq. (14)
        if (ATRtemp < ATR)
                min = per;
        else
                max = per;
        end if
end while
return (min + max) / 2;
```

3.5 Forecasting Method

In the previous subsections, we have described a method to extract useful information from the instrumentation DB for past time intervals, which could be quantified as a BER value. This method can thus provide a time series that consists of BER values through time, for the links that form the predicted bundle route.

The composed time series can provide a means to predict the future link behavior, in terms of error rate. Our rationale for this assertion is that the observed error rate through time is not an entirely random variable; it is generated by time-dependent events such as space weather and solar activities and, therefore, it is an auto-correlated variable and thus can be estimated using observation history.

Many different forecasting techniques have been developed during the last decades, from simple Moving Average to Exponential Smoothing method [151], Autoregressive Moving Average (ARMA) and Autoregressive Integrated Moving Average (ARIMA) [152]. The majority of them is developed to fit explicit models and thus apply better in specific time series. As a rule, in order to apply the most suitable forecasting model in a time series, one needs to study its evolution through time, as well as analyze its trend and periodicity, if such exist. After this careful study, time series may need decomposition into components (e.g., periodicity, trend), which can be used for forecasting.

BDTE, however, is designed to quickly and automatically reply to user input, with no intermediate manual inspection of any time series. Furthermore, the use of a specific prediction technique requires a complete and thorough analysis of error rates in space in general, for all times and seasons and for different weather and environmental conditions. The DTN architecture, however, has not yet been widely employed in space missions, and therefore there is no access to sufficient space DB sampling and measurements. For these reasons we cannot base our model on the actual space-channel behavior and form a realistic time series model yet. Instead, we use an exponential smoothing method as an initial forecasting method for testing purposes, which can be described as a simple and robust generic technique of time series forecasting that may fit different time series models. The rationale behind our choice is mainly the fact that exponential smoothing can achieve accurate predictions with minimal effort in model identification [151]. The prediction method employed by BDTE may be optimized or reconsidered when access to real space measurements is available and error rate distributions are studied in depth.

The triple exponential smoothing technique used is also referred to as Holt-Winters method, [153] [154]. This is an extension of exponential smoothing model designed for time series with trends and seasonality and does not require a large amount of time series data. Although we have as yet had no access to real channel information, we can foresee that both trend and seasonality will need to be included in data analysis. The incorporation of trends can be justified by the fact that space channel behavior is greatly affected by space weather conditions, which may have a linear behavior in short time intervals. Seasonality analysis on the other hand is included due to the periodicity of planetary and space assets' movement. In a case of a BER time series where either trend or seasonal components (or both) are missing, the corresponding factor can be set to zero and thus excluded from the model construction.

We now briefly describe the Holt-Winters prediction technique we have used in our application to forecast BER values. The additive Holt-Winters prediction function for time series with period length N (notation from [153]) has linear trend and additive seasonality. The time series forecast $\overline{S}_t$ is obtained from:

$$\bar{S}_t = a(S_t + P_t) + (1-a)(\bar{S}_{t-1} + r_t),$$

where S_t is the observed time series value at time t, P_t is the periodic adjustment increment for the t-th period, r_t is the trend adjustment increment for the t-th period, and a is a constant that determines how fast the exponential weights decline over the past periods.

The periodic and trend adjustment increments are calculated correspondingly as follows:

$$P_t = b(\bar{S}_t - S_t) + (1-b)P_{t-N}$$

$$r_t = c(\bar{S}_t - \bar{S}_{t-1}) + (1-c)r_{t-1},$$

with b and c the exponential weight constants for the periodic and trend components correspondingly. Time series forecasts T periods in the future are estimated as:

$$\bar{S}_{t+T} = \bar{S}_t + r_t T - P_{t+T-N},\ T = 1, 2, ..., N$$

The optimal values of a, b, and c (if applicable, i.e., if seasonality and/or trend exist) are estimated by minimizing the squared one-step prediction error.

All statistical functions, including Holt-Winters model, the forecasting method, as well as the optimization technique, were integrated into ION from *R*, a free software environment for statistical computing [155].

In order to determine the periodicity of a specific time series, we follow an algorithm introduced in [156]. In this study, according to Peter Turchin, a way of determining the seasonal component of a time series is based on its Auto-Correlation Function (ACF). The statistical significance of ACF can be found with the use of a simple algorithm described in Table 3-4.

We note that the maximum period that may be available for this algorithm is half the sample size. The seasonal term is ignored, if the BER time series has no periodic component. According to [154], for seasonal models, *S*, *P* and *r* initial values are inferred by performing a simple decomposition in the trend and seasonal components using moving averages on a number of initial periods, while a simple linear regression on the trend component is used for starting level and trend. For trend models without any seasonal component, start values for *S* and *r* are *S*[2] and *S*[2] - *S*[1], respectively. For ordinary exponential smoothing, i.e., for BER time series with no seasonal and trend components, the start value for *S* is *S*[1].

Table 3-4 Algorithm for Statistical Significance of ACF

```
Input: BER_time_series // with values t = 1 .. n
Output: period

ACF = calculate_ACF(BER_time_series);
T = first_local_maximum(ACF);
if (ACF[T] > 2 / sqrt(n))
        // we have "strong evidence of statistical periodicity"
        period = T;
else if (ACF[T/2] < - 2 / sqrt(n))
        // we have "weak evidence of statistical periodicity"
        period = T;
else
        // no evidence of periodicity
        period = 1;
end if
return period;
```

3.6 Model Assumptions

In the proposed model for bundle delivery time estimations, we have followed some assumptions, which are discussed in the following paragraphs.

An important factor for accurate BER prediction is DB sampling, which we assume is ideally performed for all spans in equal time intervals and not in random times. We also assume that LTP span configuration information exists in the DB for all time intervals in the past. In practice, there could be several missing values, due to network unavailability or other reasons that may lead to DB sampling failure. In such cases, interpolation techniques should apply.

For Eq. (13) we assumed the same BER for both forward and backward channel, in order to compute PER_{RS}. The inverse binary search algorithm is otherwise unable to compute the two different BERs (forward and backward) and another, more sophisticated inverse search algorithm has to be applied, thus increasing the complexity of the application.

In our forecasting technique, we predict BER for the time interval containing the moment of bundle transmission initiation, rather than for the total bundle transmission interval. In other words, if bundle transmission time exceeds the time interval that has been predicted, the statistical error of BER prediction might be higher. We can state, however, that the statistical significance of this deviation decreases, since the majority of errors occur in the first transmission round of the bundle.

The equation that uses BER to calculate PER assumes independent bit errors (a Gaussian bit error distribution on the channel), which is not always the case; burst errors, for example, occur on space channels. On the other hand, the capture rate for the statistics DB is not expected to be high due to the disconnected nature of the IPN. This results in BER calculation over

relatively long time intervals, which may be expected to exhibit “average” channel behavior. Therefore, burst errors are in essence outliers that are intentionally excluded from our prediction calculations.

For simplicity reasons, in PER calculation (Eq. (5)) we round the number of packets up to the next integer and consider them all equal to the input packet length. This assumption could lead to a significant error for small bundles that are truncated into LTP segments with a small last segment. For example, if convergence layer packet length is 1400 bytes and the delivery time of a 1500-byte bundle is to be estimated, application will consider the transmission of a bundle that is truncated into two 1400-byte packets. However, the provided analysis can be easily extended to include the remaining bytes of the last segment, and thus eliminate the rounding errors.

Finally, metrics measured during the DB time interval may have a significant standard deviation and, therefore, average values could be inaccurate metrics for block size (Eq. (2)), packet length (Eq. (3)), and number of packets in block (Eq. (4)). The distribution of these network statistics depends on the mission design and protocol configuration parameters. Further examination and reconfiguration of the total framework can be conducted with the obtainment of real measurement data, from future space DTN missions.

Chapter 4 Queueing Delay Estimation for Space Networks

In Chapter 4, we deal with the challenging problem of estimating queueing delays in space networks. To this end, we propose two different approaches, a reactive and a proactive method. The former is based on a network notification system that we introduce in Section 4.1, namely the Contact Plan Update framework, which distributes information on queue lengths through the network, and estimates queueing delays, accordingly. The latter exploits network statistics collection and distribution, and a time series forecasting procedure to proactively predict future queueing delays based on historical values (Section 4.2).

4.1 Contact Plan Update framework

The reactive method for queueing delay estimations is based on a notification method, the Contact Plan Update framework, which encodes queue lengths into the contact plan, disseminates contact plan updates through the network, hence improving the network's awareness on queue lengths, and exploits the obtained queueing information by incorporating it into routing decisions. In particular, the Contact Plan Update framework comprises three complementary elements:

(i) An update to the contact plan that integrates the queue length information, with the introduction of the ETO parameter (Section 4.1.1).

(ii) An enhanced routing algorithm, namely CGR-ETO, member of the CGR family of algorithms, which takes into account the ETO parameter and, accordingly, includes information on queueing delays during route computations. In this way, CGR-ETO provides improved prediction on the arrival time at destination, and, since CGR employs earliest arrival time as the routing objective, improves the overall functionality of the routing algorithm (Section 4.1.2).

(iii) An update protocol, namely Contact Plan Update Protocol, which is part of the network management process, and is responsible to disseminate contact plan modifications, including any available information on significant queue length changes through the network (Section 4.1.3).

4.1.1 Earliest Transmission Opportunity Parameter

In order to measure and further incorporate queueing delays into the network contact plan, we define ETO as the earliest plausible time, during a contact, that a bundle with a certain priority can be forwarded [157]. In this way, ETO measures the occupancy in outbound queues. ETO values range from the contact start time, which is set as the default and minimum value, up to the contact end time, which is the maximum value. Different priority levels are reflected in corresponding ETO parameters for each transmission opportunity. Hence, the contact structure with the ETO parameters incorporations becomes:

{*T1*, *T2*, *A*, *B*, *tx_rate_A_B*, *ETO*[*PRIORITY_LEVELS*]},

where *T1* is the starting time of the communication interval, *T2* the stop time of this interval, *A* the transmitting node, *B* the receiving node, *tx_rate_A_B* the planned transmission rate from transmitting to receiving node over this interval, and *ETO*[*PRIORITY_LEVELS*] the table that contains the distinct ETO values for all priority levels. In BP, for example, the three priority levels defined are bulk, normal, and expedited [6]. In this case, the ETO table contains three elements: *ETO_bulk*, *ETO_standard*, *ETO_expedited*, which represent the ETO values for the three priority levels, accordingly. For different priority levels, e.g., in the ION implementation [11] with 255 ordinal extended class-of-service levels, the local node's contact plan stores additional ETO values per contact, for each distinct level of service: ETO for bulk priority, ETO for standard priority, and 255 values of ETO for expedited priority. Thus, the ETO value represents the estimated transmission time, for each bundle, including the expected queueing delay that pertains to the bundle priority.

With regard to queueing information, ETO is updated: **(i) for local contacts** (i.e., contacts with neighboring nodes), after routing decisions are made at the local node: when a bundle is routed and inserted in an outbound queue towards a neighboring node, the local node increases the corresponding ETO for the specific contact during which bundle transmission is expected to occur; **(ii) for next-hop contacts** (contacts between other nodes) that the bundle is expected to follow through the path to destination: when a local routing decision is made for a bundle, the routing algorithm outputs the total path (set of contacts) that the bundle is expected to follow, and local node increases the corresponding ETO for the contacts that comprise the routing path; and **(iii) upon reception of CPUP messages** from other nodes, containing ETO information about specific contacts, the local node updates its contact plan accordingly. In the former two cases, contact updates are only based on locally processed data, without the need

for CPUP transmissions, and the ETO parameter is updated for all priority levels equal to or lower than the priority level of the bundle to be transmitted.

Updates using local decisions for local contacts

Whenever a bundle is routed and queued for transmission towards a neighboring node and transmission is expected to occur during a specific contact with that node, ETO information of the contact is updated to include the specific bundle. In particular, the routing algorithm computes the *estimated capacity consumption* (*ECC*) [9] [11] of the bundle, which comprises the bundle payload length, the potential bundle extension blocks length, the BP header and an estimate of the underlying protocols overhead. The routing algorithm then converts ECC into transmission delay by dividing it with the transmission rate of the contact, and the calculated transmission interval is added to the previous value of ETO or the current time (i.e., the routing time), whichever is later. The same update occurs for all priority levels that are equal or lower to the routed bundle's priority level, as it does not affect higher priority bundles. Hence, the updated ETO value represents, as its name suggests, the new *earliest transmission opportunity* for new bundles that will be transmitted over the same contact.

Updates using local decisions for next-hop contacts

Taking into account queueing delay on multiple hops (i.e., not just for local contacts with neighboring nodes) is inherently much more complex for three main reasons: First, length information of outbound queues in other nodes is not immediately available to the local node where CGR is performed. Second, queue length updates cannot always be timely disseminated to other nodes in a DTN network, due to link intermittencies and long propagation delays. Third, for the same reasons, the present state of the queues in the nodes of the DTN route to destination do not necessarily reflect the queue length (and the corresponding queueing delay) that a bundle will actually experience *when* this bundle will eventually reach the node (in space networks it may happen hours later).

A primary solution for the queue length calculations on next hops is to take into account only data originated from the local node, without any information exchange between network nodes. Whenever a bundle is routed and queued for transmission, CGR outputs all contacts that comprise the route, i.e., the contacts that the bundle is expected to follow during the path to destination. Then, for each contact of the route, bundle's ECC is translated into transmission delay, using the transmission rate of the contact, and the calculated transmission delay is added in the contact's ETO (i.e., the most recent queueing delay information of that contact), or the expected arrival time at the contact's transmitting node, whichever is later. Since also in this

case, the contact includes different ETO values for all distinct bundle priority levels, ETO is updated for all priorities equal to or lower than the routed bundle's priority. Thus, the network node maintains, in its contact plan, queue length information for all contacts, based only on the locally routed bundles.

Being based on locally processed bundles only, these ETO values might be an underestimation of the actual queueing delays. Nevertheless, they approach the actual queueing delay values better than using the contact start times. This update can lead to improved delay calculations, in all contacts of the path, and a better estimate of the bundle delivery time, although it cannot clearly consider the impact of traffic that is not processed locally (e.g., cross traffic).

Updates triggered from remote messages

Updates in contact plan's ETO information can be also updated upon reception of corresponding notification messages from other nodes. For this purpose, we have designed the CPUP protocol, which is responsible to disseminate the contact plan update messages (including information on ETO updates) within the network. Thus, whenever a node receives a CPUP message containing non-obsolete (i.e., not expired; this is explained in detail in Section 4.1.3.2) ETO information about a set of contacts, the local node updates its contact plan, accordingly. Details on the structure, usage and dissemination mechanism of the CPUP protocol are presented in Section 4.1.3.2.

The three aforementioned types of updates can be also employed in a complementary fashion; that is, a node can update ETO information in its contact plan after local routing decisions, for both local and non-local contacts, as well as upon reception of CPUP messages that contain ETO update information. The resolution of potential conflicts depends first on where the information is generated (local node's information is considered more accurate than information transmitted from other nodes), and, then, on the information generation time (more recent information is preferable, while older information is considered obsolete).

The dissemination of ETO information involves transmission overhead, whereas the updates from locally processed information does not include any information exchange and is, therefore, transmission-overhead-free and more cost-effective. However, in both cases, updates in ETO values result in a contact plan modification, and the CGR algorithm recalculates routes using Dijkstra searches whenever the contact plan changes (as described in Figure 2-6). This procedure leads to additional computational overhead, whenever bundles are routed and ETO values are updated, accordingly. To alleviate this processing overhead, we have employed a *contact plan update threshold*, expressed as a percentage of the contact duration, which defines

the successive contact intervals that ETO has to exceed, for contact plan to be flagged as *modified*. For example, if contact duration is 1000 seconds and the *contact plan update threshold* is set to 10%, Dijkstra recalculation of optimal routes is not triggered until ETO has increased more than 100 seconds (i.e., 10% of the 1000 s contact duration) since the previous calculation. For this check, we use the lowest priority (bulk) ETO value, which is updated for every forwarded bundle.

4.1.2 Contact Graph Routing with Earliest Transmission Opportunity

The original CGR algorithm [9], with functionality analyzed in Section 2.4.2, exploits the perfect knowledge of the contact plan that every network node is assumed to have. The introduction of ECGR [126] updated the algorithm functionality to examine all available paths to destination and conclude on an optimal path based on the earliest arrival time. In route computation, however, ECGR algorithm assumes that bundles can be sent at the start of the contact, or, if the contact is already open, immediately. In other words, it does not consider queueing delay, i.e., the time necessary to transmit bundles already in the transmission queue.

CGR-ETO [157] [158] [159] exploits the information included in ETO parameter, and substitutes the contact start time with ETO to calculate the arrival time for each route during the route computation included in the CGR algorithm (see Figure 2-6). This way it incorporates the most recent available queueing information, for the given bundle priority. In brief, the CGR-ETO enhancement aims to exploit knowledge of the queues to provide a better estimate of the actual transmission time of the bundle, rather than using the assumption that bundle is transmitted at the contact start time. The differentiation between CGR-ETO and the original CGR algorithm is clarified in Table 4-1, which displays the calculation of the destination arrival time, for a single route, for both CGR and CGR-ETO. Note that when, for a contact, *transmissionEndTime* exceeds contact end time, the route is considered invalid, since it cannot guarantee the bundle transmission through its set of contacts. The consideration of queueing delays provides CGR-ETO with an improved route filtering functionality, as it controls and modifies the data paths upon contact exhaustion. This can be interpreted as a means for congestion control (congestion here represents the transmission opportunity exhaustion).

Table 4-1 Calculation of route arrival time: CGR and CGR-ETO

```
Input: route, bundle
Output: arrivalTime // at destination

arrivalTime = currentTime;
for each (contact in route)
        switch (CGR_TYPE)
        case (CGR):
                transmissionStartTime = min(contact.startTime, arrivalTime);
                break;
        case (CGR-ETO):
                transmissionStartTime = min(contact.ETO[bundle.priority], arrivalTime);
                break;
        end switch
        propagationDelay = findRange(contact).owlt;
        /* Calculate ECC from bundle information (payload, header, extension blocks),
         * and convergence layer overhead, if available (local node)                    */
        ECC = bundle.payloadLength + estimateBundleOverhead(bundle) +
                estimateCLOverhead(bundle, contact);
        transmissionDelay = ECC / contact.xmitRate;
        transmissionEndTime = transmissionStartTime + transmissionDelay;
        if (transmissionEndTime > contact.endTime)
                return -1;          // contact end time is exceeded, invalid route
        end if
        arrivalTime = transmissionEndTime + propagationDelay;
end for each

return arrivalTime;
```

The enhanced functionality of CGR-ETO and its improved routing performance led to its inclusion in ION implementation [11], since version 3.2.1, where it was adopted as the standard CGR algorithm. In the version of CGR-ETO incorporated in ION, the configuration of ETO updates was modified, without changing however the core functionality of CGR-ETO. In particular, the incorporated in ION CGR-ETO algorithm includes only updates on local outbound queues based on locally routed bundles, that is, without considering outbound queues of other nodes and without ETO updates based on notification messages (e.g., using CPUP). Additionally, ETO variable is not included as a contact field, but is extracted using the already existing local information on the outbound queue lengths. In this way, the contact plan is not considered as modified, whenever a new bundle is put into an outbound queue, and, therefore, CGR-ETO does not perform new Dijkstra searches to consider the queueing delays in bundle delivery time estimation. Thus, instead of calculating the route arrival time at the Dijkstra algorithm searches, this version of CGR-ETO calculates it at the final algorithm step, where it compares all available routes in the route list based on the earliest delivery time, and decides on the optimal one (see Figure 2-6). In this way, the additional processing overhead is eliminated, and the algorithm retains its full functionality. The limitation of this algorithm version is that it only considers local updates on local outbound queues; the same approach

cannot be extended to consider outbound queues for nodes other than the local node. This extension requires incorporation of ETO information in the contact plan, and, consequently, route recalculations through Dijkstra searches, upon contact plan updates. We note that the results obtained with the two versions of CGR-ETO algorithm (i.e., CGR-ETO incorporated in ION standard CGR, and CGR-ETO with Dijkstra recalculations based on ETO updates on local contacts only) are the same, and thus we do not differentiate these two versions in the evaluation section.

4.1.3 Contact Plan Update Protocol

CPUP [157] is an update protocol designed to transmit modifications in the contact plan, including information that pertains to dynamic network features, encoded in the contact parameters (e.g., ETO). Possible update message options include the creation of new contacts, as well as the deletion or modification of existing contacts. The protocol format is presented in 4.1.3.1, while the employed dissemination mechanism is described in 4.1.3.2.

4.1.3.1 Protocol Format

The CPUP PDU allows for the efficient integration of multiple update commands within a single payload, as depicted in Table 4-2. Each command is encoded into a Command Block with the format displayed in Table 4-3.

The CPUP header contains the protocol version followed by the "Number of Command Blocks" field. The latter is represented by a Self-Delimiting Numeric Value (SDNV) format [160] and, therefore, has a variable length. The PDU header is followed by the sequence of Command Blocks. For convenience in representation, "Number of Command Blocks" field is depicted as a three-byte SDNV and each Command Block is shown as a four-byte field.

Table 4-2 CPUP PDU Format

Byte 0	**Byte 1**	**Byte 2**	**Byte 3**
Version num.	Number of Command Blocks (SDNV)		
Byte 4	**Byte 5**	**Byte 6**	**Byte 7**
1st Command Block			
...			
Byte 4×*n*	**Byte 4×*n*+1**	**Byte 4×*n*+2**	**Byte 4×*n*+3**
*n*th Command Block			

Table 4-3 Command Block Format

Byte 0	**Byte 1**	**Byte 2**	**Byte 3**
Creation Timestamp (SDNV)		Command Expiry (SDNV)	
Byte 4	**Byte 5**	**Byte 6**	**Byte 7**
Command Originator (SDNV)			Command Type
Byte 8	**Byte 9**	**…**	**Byte *n***
Command Parameter 1 (SDNV)		…	Comm. Param. *k* (SDNV)

The Command Block contains all necessary information pertaining to the update command. The "Creation Timestamp" field is used to detect and discard obsolete information: commands with creation time older than the most recent update time for a specific contact are ignored. Additionally, the timestamp value "Command Expiry" is used to identify the time after which the information contained in this Command Block is invalid or useless. The node that generated the command is stored in the "Command Originator" field, while "Command Type" is a one-byte field with different codes for adding, deleting, and editing contact and range registrations of the contact plan. The block ends with a sequence of "Command Parameter" fields that carry all the necessary information for the command execution. The number of Command Parameters is specific for each Command Type; for example, "Edit Contact bulk priority ETO" contains "Start Time", "From Node", and "To Node", which are the three necessary fields that uniquely identify the contact, followed by the field "New Value of bulk priority ETO". Given that SDNVs are of variable length, they are represented with different lengths in Table 4-3.

Since the contact plan information and the CGR algorithm have not yet been standardized, and the useful network information might vary, CPUP can be easily customized to include any parameter field, allowing the use of different routing objectives (i.e., other than earliest delivery time). CPUP is designed to use the DTN protocol architecture and, hence, each PDU is inserted into a single bundle payload, utilizing BP transport, routing, and security mechanisms to forward the updated information. CPUP functionality may alternatively be deployed as part of a unified DTN network management infrastructure, which is under standardization [137].

4.1.3.2 CPUP Dissemination Mechanism

CPUP data units can be generated either in an administrative framework, e.g., to initially setup a list of connection schedules, or automatically, triggered by significant changes in queue occupancy. It is essential that the produced command messages are delivered to all network nodes timely, before their validity expires. For this purpose, a flooding-based mechanism is utilized to disseminate CPUP data units.

Commands are automatically created when the queue occupancy in some contact increases beyond a predefined threshold, which may or may not be the same with the *contact plan update threshold*. The per-contact queue occupancy corresponds to the outbound queue that consists of stored bundles destined to the corresponding neighbor and expected to be transmitted during some given contact.

Commands are generated or updated separately per contact. The most recent information for a specific contact is disseminated to all other neighbors, provided that this information is delivered timely (i.e., before the command expiration). CPUP encodes all commands applicable to a specific neighbor into blocks and aggregates them into a CPUP PDU. It subsequently transmits this CPUP PDU to that neighbor at the next communication opportunity.

When a node receives a CPUP data unit, it updates the local contact plan with all applicable commands. Obsolete information, i.e., commands with creation time older than the most recent contact modification, are discarded. The node, then, performs the same flooding procedure for every received command; it checks whether information can arrive before expiration and delivers it to the CPUP engine for aggregation. The initial node that originated the corresponding command, as well as the previous neighbor that propagated the CPUP data unit containing this command block, are excluded from the flooding process.

According to our design, the granularity of the generated update commands for ETO is determined by a threshold level which can be either an absolute time interval or a percentage of the contact duration. As mentioned above, this threshold may be the same with the *contact plan update threshold*. In our initial configuration, we have applied the percentage model and followed the approach of a single threshold value, for both contact plan updates and CPUP command triggers; for example, a 1% update threshold level within a 5000s transmission opportunity triggers the generation of update commands each time ETO gets 50s greater than the previous ETO. The "Expiry Time" of the produced information is the new ETO; it will be delivered to the CPUP engine and conditionally forwarded to all neighboring nodes, if the CPUP expected delivery time precedes the new ETO. In this way, useless transmissions are restrained.

Finally, an update threshold level of 100% is associated with no dissemination of queue occupancy information when dealing with equal priority bundles, since contact capacity cannot be exceeded. However, a potential "overbooking" [158] [161] may occur when low-priority bundles, queued for forwarding during an imminent contact, are followed by high-priority bundles; the BP will enforce priority and transmit first the high-priority bundles, resulting in a potential oversubscription of the total contact capacity. An overbooking management mechanism, presented in [158], solves this issue by identifying the oversubscription and reforwarding the overbooked bundles. If the overbooking management mechanism is not applied, however, a different CPUP dissemination retention mechanism is required, since the

100% threshold can still be triggered. The solution can be provided by using a *CPUP generation flag* that retains the generation of ETO update commands and the production of CPUP messages, accordingly. This flag can be raised also in nodes that are not responsible for relaying data.

4.2 Queueing Delay Prediction Method

Since space internets are typically characterized by long propagation delays, the performance of the reactive, dissemination-based Contact Plan Update framework will always be constrained by the inability of network nodes to access recent information, created at a distant network node. To this end, we study a different approach of obtaining queue length estimates and solving the challenging task of queueing delay estimation, accordingly: the proactive prediction of future queueing rates and queueing delays, through network statistics and time series forecasting [162]. In particular, we examine a generic scenario and study the outbound queue of a network node that receives unicast data simultaneously and/or successively from a number of nodes, and enqueues the data in that outbound queue, for transmission to the next node. Even though the production and delivery rates of data cannot be foreseen, past measurements include valuable information that can assist in estimation of the corresponding future rates via time-series forecasting. The rationale for this argument is that, in space environments, data transmission flows follow a time-dependent scheme, since: a) mission data availability follows a time-dependent (rather than random) pattern, b) periodicity is imposed by planet rotations, satellite orbital movements and occasional high or low data rate passes [163], and c) linear dependency is inflicted by spaceship movements, as well as linearly evolving space weather phenomena.

Based on the generic scenario that we examine, we introduce a simple method in which all nodes extract queueing rate measurements in a per-contact granularity. Extracted measurements are then disseminated to all neighbors, and stored in each node's contact plan, composing different time series between each pair of network nodes. The available time series information are then used to forecast future queueing rates and the predictions are combined with the contact plan schedules to estimate the queueing delay, as well as the total delay for the bundles to be transmitted.

4.2.1 Generic Scenario

In order to study the queue occupancy and queueing delays in an outbound queue of a DTN node, we consider a generic scenario with topology as depicted in Figure 4-1. In this topology, *N* sender nodes are transmitting data to destination node *D* via intermediate node *A*. Thus all data sent from nodes 1, 2, ..., *N* to *D* or beyond need to be stored in the relay node *A* and then forwarded to *D*. This store-and-forward procedure inevitably imposes extra waiting time for any bundle queued in the outbound queue from *A* to *D*, until all previously queued bundles are forwarded. We consider a simple case where all bundles are transmitted with equal priority and, thus, there is a single outbound queue for the *A*-*D* link. The corresponding generic contact plan is illustrated in Figure 4-2, where a single period of transmission opportunities is depicted. The period starts from the end of the previous *A*-*D* contact and ends at the next *A*-*D* contact. Note that nodes 1-*N* may have more than one communication opportunities with *A* during a cycle. Our primary interest is in the bundle queueing delay and, consequently, in the total end-to-end bundle delivery latency, from bundle creation time until arrival at destination *D*. Note that this generic scenario can apply into different networking scenarios in the IPN that include data transmissions from different nodes in the same direction (typically at the downlink). An indicative scenario, e.g., for cis-Martian communications, includes a set of rovers, landers, or other satellites (nodes 1 – *N*), transmitting data via a Mars relay satellite (node *A*), e.g., Mars Reconnaissance Orbiter (MRO), towards the ground station (node *D*). The generated cross-traffic data contribute to the same outbound queue at the Mars satellite towards the ground station.

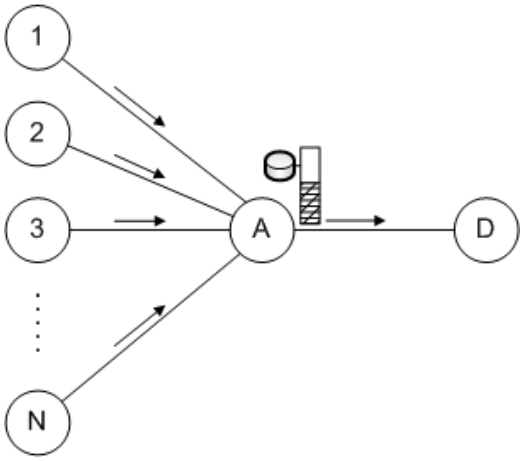

Figure 4-1 Generic Scenario Topology

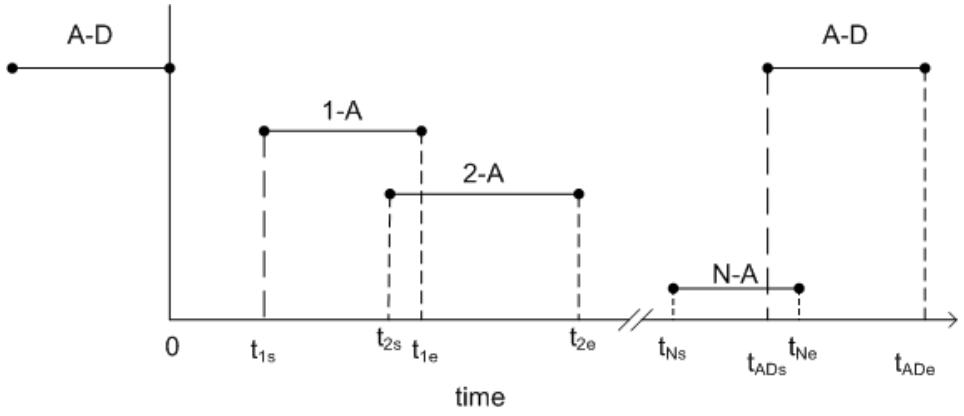

Figure 4-2 Generic Scenario Contact Plan

4.2.2 Queueing Rate Measurements

In order to study the queue length and all queueing rates through time, we apply a sampling process in a per-contact granularity. When a contact from node *k* to *A* ends, the number of bytes that arrived at *A* over this contact and were queued for delivery to *D* are counted. This amount is then divided by the contact duration to obtain the average queueing rate r_{kAD} that node k imposes into outbound queue *A-D*. Note that this queueing rate typically differs from the *k-A* nominal transmission rate, due to transmission and retransmission overhead and since some of the delivered bundles may not be forwarded to this link towards *D*. Then, upon the end of the *A-D* contact, node *A* calculates the remaining queue length $Qrem_{AD}$ at the specific link. Information about the extracted queueing rates and the remaining queue is transmitted to all neighboring nodes other than the link destination (i.e., *D* in this example) at the next available opportunity, via CPUP. The dissemination mechanism of CPUP is responsible to relay the information PDUs to all network nodes. In our two-hop scenario, PDUs are transmitted from *A* to nodes 1—*N* and no further transmissions occur.

Measurement granularity for the queueing rates could be improved, if sampling occurred in a number of time intervals during each contact. This would impose extra overhead, however, and would increase the complexity of historic rates management. Therefore, we choose the more conservative approach of per-contact measurement granularity.

4.2.3 Prediction of Future Queueing Rates

Following the dissemination of the measurements, all network nodes have received past values of queueing rates and remaining queue lengths. The past rate values comprise a time

series for each distinct pair of neighboring links. For example, for links $k \rightarrow A$ and $A \rightarrow D$, the time series contains past average values of r_{kAD}, i.e., data transmitted during contacts $k \rightarrow A$ and queued in the outbound link $A \rightarrow D$.

Due to the mainly deterministic and periodic nature of space communications, we argue that the past observations can be useful to predict future values such as queueing rates, with some accuracy. Time series may incorporate periodicity and/or a linear trend. In this context, a number of different forecasting techniques can apply into our model. The procedure described in Section 3.5, for example, utilizes a triple exponential smoothing method, which incorporates possible trends and/or periodicities in the BER time series under study. Here, however, we do not focus on the optimization of the time series forecasting method, but prove the applicability of our proposal. Therefore, we choose a simple exponential moving average (EMA) forecasting method for low-complexity and low-processing-overhead purposes. Further optimization is possible after obtaining a sufficient set of space data transmissions statistics, with the deployment of DTN in space missions.

For any contact j, the EMA S_j is calculated recursively, by computing $S_j = \alpha r_j + (1-\alpha) S_{j-1}$, where r_j is the measured rate for contact j, and α the constant smoothing parameter, $0 < \alpha \leq 1$. The forecast rate is set equal to the EMA of the previous time step (i.e., contact) $\hat{r}_{j+1} = S_j$. In the evaluation process, we examine different values for the smoothing parameter α, including $\alpha = 1$, which is equivalent to a random walk model. When the time series include missing values, due to delays in the arrival of information updates, the last computed EMA is reused.

Queue lengths at the end of contacts $A - D$, noted as $Qrem_{AD}$, also form a time series and a similar forecasting procedure applies.

4.2.4 Bundle Delivery Delay Calculation

The introduced method applies on the output of any contact-plan-based routing algorithm, that is, the path to destination, and calculates the total delay from bundle creation time to the arrival at destination. In our generic scenario, a bundle is created in node k and the routing algorithm selects path $k \rightarrow A \rightarrow D$. The transmission from k to A comprises the following components:

i) propagation delay $d_{pr.k\text{-}A}$,

ii) transmission delay $d_{tr.k\text{-}A} = (bundle\_size + overhead) / tx\_rate_{k\text{-}A}$,

iii) processing delay,

iv) queueing delay $d_{q.k\text{-}A}$, and

v) total waiting time $d_{w.k\text{-}A}$ until transmission opportunity is available.

Propagation and transmission delays are calculated based on the information contained in the contact plan. Queueing delay for the first transmission hop is calculated based on the queue information that is available for the local outbound queue and may exceed the duration of a single contact. Waiting time is also extracted from the contact plan and may also span across more than one time periods, if the data ahead have filled the capacity of the next contact(s). In contact plans where contacts are not often, the waiting time can be a major part of the total delay. Based on the aforementioned delay components and assuming trivial processing delays, expected arrival time at node *A* is calculated as follows, assuming t_{cr} the bundle creation time:

$$t_{arr.A} = t_{cr} + d_{w.k-A} + d_{q.k-A} + d_{pr.k-A} + d_{tr.k-A}$$

For the next transmission hop, the total delay has the same, aforementioned components, with starting time equal to $t_{arr.A}$. Calculation of the queueing delay $d_{q.A\text{-}D}$ exploits the contact plan information, the queueing rates $\hat{r}_{iAD}, i = 1..N, i \neq k$, as well as the remaining data in-queue $\hat{Q}rem_{AD}$ at the end time t_0 of the most recent *A*-*D* contact before $t_{arr.A}$. All these values can either be the actual measurements, if the corresponding information has already arrived at *k*, or the values predicted using the proposed forecasting procedure. Queueing delay for the bundle in outbound queue *A*-*D* is computed as follows:

$$d_{q.AD} = \frac{\hat{Q}rem_{AD} + \sum_i \hat{r}_{iAD} \times \tau_i}{tx\_rate_{A-D}}, \tag{15}$$

where *i* represents all contacts that may cause backlog (or, in other words, contacts that are active during the time interval from the end time t_0 of the last contact *A*-*D*, until the expected bundle arrival time at node *A*, $t_{arr.A}$), and τ_i the contact duration. If the expected bundle arrival time at node *A*, $t_{arr.A}$ exceeds the *i*-th contact's start time, the contact duration is calculated as the interval from contact start time to the bundle arrival time, $\tau_i = t_{arr.A} - t_{is}$. The waiting time $d_{w.A\text{-}D}$ for the bundle is the interval between the arrival time $t_{arr.A}$ and the next available contact *A*-*D*, plus all intervals between consecutive *A*-*D* contacts that the bundle waits in queue. Using these calculations and Eq. (15), bundle arrival time at destination node *D*, which is the output of our method, becomes:

$$t_{arr.D} = t_{arr.A} + d_{w.A-D} + d_{q.A-D} + d_{pr.A-D} + d_{tr.A-D}$$

Chapter 5 End-to-end Retransmission Framework for Space Networks

In this Chapter, we exploit the improved delivery delay estimation, obtained with the analytical methods and algorithms presented in the previous Chapters, to improve the functionality of end-to-end transport protocols that reside over the space DTN architecture, and in particular of DTPC.

Although the requirements of a reliable transport protocol for DTN networks may differ from that of typical reliable transport protocols for the Internet, the retransmission scheme, and primarily the RTOs still remain the crucial components that regulate the tradeoff between faster recovery of lost data and minimization of transmission overhead due to redundant transmissions. Controlling the dynamics of this tradeoff in networks with scarce connectivity, limited resources, and typically higher error rates, such as DTNs, is a challenging task and still requires enhanced mechanisms for dynamically calculating RTTs and setting RTOs based on network conditions.

To this end, we develop an efficient end-to-end retransmission framework [164] for the recently emerged transport layer of the DTN architecture [165]. The core idea is to estimate routing-aware end-to-end delays exploiting cross-layer interactions between the end-to-end transport layer and underlying protocol's routing function. That is, routing path decisions about transport data units are made available to the transport facility and RTO intervals are estimated using the maximum -within some boundaries- expected end-to-end delay, based on the worst-case network conditions that may be experienced on the selected routing paths. In this context, we attempt to estimate the major delay components that contribute to the end-to-end delays observed in the layered DTN architecture, and we combine these estimates to calculate efficient RTO intervals for the corresponding data units. Such delay components, as previously explained in Section 2.3, include not only typical sources of delay, i.e., propagation, queueing, and transmission delays, but also non-typical events associated with disruptive circumstances due to absence of connectivity (i.e., waiting delays), or delays that are inextricably linked to protocol operations (e.g., lower-layer protocols retransmission delays). The delay components pertaining to each layer of the protocol stack are computed independently and are combined hierarchically following a top-bottom approach. Since the delay estimation may produce reasonable processing overhead on its own, we reach a solution beyond a costly one-timer-per-data-unit approach and enable group-based calculations.

We begin by presenting the introduced framework in a protocol-agnostic manner, using generic, DTN-compatible concepts. In Section 5.1, we describe the design concepts that pertain

to the different components and layers of the proposed scheme, while in Section 5.2, we provide an overview of the overall operation of the proposed framework. Finally, we introduce the operation algorithms within the technical context of DTPC protocol and the specifics of space DTN architecture, in Section 5.3.

5.1 Main Concepts of Operation

In this Section, we describe the main design concepts of the proposed retransmission framework and provide the rationale behind our design choices.

5.1.1 RTO Considerations

The design of RTO intervals, in principle, affects and, in turn, regulates a critical tradeoff between fast retransmission of erroneous or lost data, and minimization of redundant retransmissions. In the RTO design we lean towards a more conservative approach, and favor the minimization of retransmission overhead over the fast recovery of lost data, since (i) constrained DTN environments typically have limited resources, and, thus, the importance of overhead reduction increases, and (ii) DTPC constitutes a safety net over the DTN architecture reliability, rather than the core reliability mechanism. Hence, RTO intervals are based on worst-case estimates of end-to-end delay, within some confidence level captured by the *delayTolerance* parameter. The *delayTolerance* parameter expresses the minimum percentage of confidence that the framework should consider when estimating the transfer time of a data unit over a single hop and qualitatively represents, as its name suggests, how tolerant an application is to the slower retransmission of lost data units. The latter can also be translated into how intolerant an application is to possible transmission overhead due to redundant transmissions. For example, a *delayTolerance* value equal to 0.99 means that the delivery time of a bundle at the next node will be estimated, so that the arrival probability of that bundle by that time is at least 99%, and thus the probability of a redundant retransmission is less than 1%. Smaller *delayTolerance* values result in shorter RTO intervals and faster retransmissions, while greater values are more tolerant to statistical deviations in the estimated delays for each hop, and, thus, less susceptible to redundant transmissions. We argue that overhead minimization is more important than timely retransmissions, and, therefore, stretch *delayTolerance* values close to maximum (100%).

5.1.2 Routing-aware Estimations

In our main concept, the enabling idea for estimating efficient RTO intervals in DTNs with scheduled connectivity is the cross-layer communication between the transport and routing facility. The motivation behind this design is the fact that routing paths in such networks are rather predictable and known beforehand, and all CGR-based routing algorithms [118] calculate complete routing paths for the transmitted bundles, along with their expected arrival time at destination. We expect future routing algorithms to fully exploit contact-schedules information and, thus, calculate complete routing paths, without being confined by the routing objectives. In this context, it is valid to assume that in DTNs with scheduled connectivity, routing path information will always be available at the sending nodes and thus can easily be made available to the transport facility, as well.

Along these lines, calculating efficient RTO intervals translates into setting timers according to the estimated arrivals computed by the CGR algorithm. There are, however, two drawbacks for this approach. First, such a design decision would make the proposed framework routing-dependent and limit its applicability; it cannot always be assumed that routing protocols base routing decisions on the expected delivery time. Second, estimated arrival times cannot be expected to have the level of accuracy required for end-to-end timer-based retransmissions, and typically provide minimum expected arrival times.

Given that our intention is to develop a retransmission framework that is independent of the routing algorithm deployed in the network and predicts worst-case, delay-tolerant RTOs, we choose to follow a different approach: RTO intervals are computed at the transport layer, combining the knowledge on routing decision (i.e., routing paths) that is available from the routing facility, with information about the worst-case conditions that the transmitted bundles may experience on the path, according to statistical observations, such as maximum outbound queue occupancy and maximum packet error rates.

Provided that accurate contact plan is distributed to all network nodes, as suggested in [9], we also introduce a different, interactive tool between the transport and the network layer that enables the transport protocol's ability to call the routing algorithm at will. This way, the sender transport entity, after calculating the worst-case arrival time at each intermediate node, simulates the routing decision that will be taken at that time, along with any potential modification to the actual end-to-end path due to additional delays (e.g., queueing delays, lower layer retransmissions, etc.). Thus, the initially predicted routing path may be revised according to the maximum-delay scenario, and RTO is configured according to the worst-case estimate of the end-to-end RTT.

5.1.3 Group-based Retransmissions

Since the worst-case RTO estimation involves a series of routing algorithm calls and might be a processing-demanding procedure, we follow a group-based design approach to keep processing overhead as low as possible. Towards this direction, in the proposed framework we consider group-based retransmissions only, where transmitted packets are grouped together based on their anticipated end-to-end path, and a single retransmission timer is set for each group. A group comprises all packets that are expected to follow the same end-to-end path, i.e., to be transmitted during the same set of contacts. The rationale for this grouping procedure is that, in disconnected environments, waiting times may significantly affect or even dominate end-to-end delivery delays, and, hence, it can be reasonably assumed that packets sharing the same set of contacts are expected to experience comparable delays. Assuming also that packets have the same priority and that a FIFO queueing policy is applied in every outbound link of the network, the worst-case end-to-end delay that may be experienced among all packets of each group is the worst-case delay of the most recent one. Based on the above logic, in the proposed framework, the computational-demanding calculations of RTO intervals are performed only once per group, for the most recent packet of each group. With this approach, processing overhead is kept low, at the cost of a slight delay in retransmitting the previous packets of each group, compared to a per-data-item RTO granularity. The described grouping concept may also be further optimized, e.g., by providing finer granularity, reducing the number of packets per group and possibly improving the retransmission initiation time for a percentage of the packets, at the cost of additional processing overhead. The study of this tradeoff and the optimization of the grouping mechanism is an interesting research subject, and constitutes a possible direction for further research on the topic.

5.1.4 Distributed Storage Occupancy Information

As described in Chapter 4, one of the major challenges for computing accurate estimations for end-to-end delays is how to measure the storage occupancy and thus the corresponding queueing delays along some network path. This is particularly important in DTNs, where contacts among nodes are rather sporadic, and increased queueing delays can lead to contact opportunity loss, with significant effect on the total delivery latency. Even when connectivity is scheduled and the available capacities are known beforehand, data production rates may not follow a predefined pattern. That is, even though a node may be aware of its local data production and forwarding rates, and can thus partially predict the congestion imposed at the

other nodes of the network, it is impossible to accurately forecast congestion, when data originated from other nodes share a common path (i.e., when cross-traffic exists). Furthermore, "traditional" approaches for measuring congestion using end-to-end close-loop schemes cannot have practical application to DTNs; the absence of continuous end-to-end connectivity and the long delays that characterize these networks cannot assure timely information delivery at the sender, e.g., through CPUP update messages. This is also in line with a recent study on congestion control schemes for DTNs, where the mechanisms that are considered as highly-deployable in DTNs are those that attempt to perform congestion control using local information only [166].

To address the above challenge, we follow an approach that does not highly depend on timely information delivery about the storage occupancy of the nodes of the network. In particular, we assume that each node maintains global information about the maximum backlog that may be experienced in every outbound BP transmission queue of the network. Such type of information can be easily maintained and updated periodically using a network update protocol like CPUP [157] and DTNMP [137]. Although the previously introduced approaches can be used to disseminate queue state information [157] or predict queueing delays using time-series forecasting [162], the use of a maximum backlog information has a twofold significance: first, it conforms to the worst-case delay approach that we follow, and, second, it makes estimation less sensitive to lack of timely information. We note that here we assume that maximum backlogs are fixed and known a priori based on statistical observations, and thus we do not study the details on how this information is disseminated through the network.

5.1.5 Distributed Convergence Layer Information

One of the primary means that assures reliable data transfer in DTNs is the transmission of bundles over sub-net specific reliable transport protocols (i.e., CL protocols), wherever this is feasible. Lost packets are usually recovered through typical ARQ methods, or hybrid ARQ-FEC (i.e., erasure coding) approaches for faster bundle delivery. Regardless of the error control method applied, hop-by-hop bundle delivery time over erroneous links highly depends on the PERs observed in the link. This is particularly true for deep-space communication links, where the additional RTTs that may be required for packet recovery can extend communication in the order of minutes or even hours. Thus, in order to model the performance of CL protocols and be able to estimate hop-by-hop transfer latency when scheduling end-to-end retransmission timers, CL-related information also becomes important. Here we assume that each node is aware of the CL protocol in every link of the network, along with other information such as the

respective packet size or the estimated overhead consumption per packet. Furthermore, similarly to the maximum backlog information described in the previous subsection, and in agreement with the worst-case approach, we assume that each node is aware of the maximum PER that may be observed in every outbound link of the network. Statistics about the observed values of the aforementioned CL protocol parameters can be maintained at each node and disseminated through the network using network update procedures, like CPUP [157] and DTNMP [137]. We note here that, since the maximum values of the observed network statistics (e.g., PER, RTT, maximum backlog) may be constant or oscillate moderately through large periods of time, frequent and timely updates are not critical for configuring RTO properly.

5.1.6 End-of-contact Policy

Finally, a factor that significantly affects bundle delivery delay is the policy on how the underlying network handles ongoing transmissions, when the period of current contact opportunity ends. In general terms, two basic policies are considered here. The first one relates to the CL protocol failure detection mechanism that can be employed at the BP layer. That is, if the end of a contact causes a CL protocol to fail, BP can detect the failure through some sort of inter-layer signaling, and re-forward all bundles whose acquisition by the receiving entity is presumed to have been affected by that failure. Practically, this approach has the benefit of exploiting alternative contact opportunities with other neighbors, rather than waiting for the transmission resumption at the same outbound link. An alternative approach could be to suspend transmission over the current outbound link until connectivity is restored and transmission continues. This approach requires "contact-aware" CL protocols and has the advantage that partially received data need not be retransmitted. This reduction on transmission bandwidth is achieved at the cost of losing alternative contact opportunities that could result in shorter data delivery.

5.2 Overall Operation

The overall operation of the proposed end-to-end retransmission framework is shown in Figure 5-1. It becomes apparent that the core functionality is implemented at the transport layer, while inter-layer communication (represented by the red vectors) between the transport and network layer is required in several stages of RTO interval calculation process. As shown in Figure 5-1, each time a transport packet is generated and passed down to the network layer for

further processing, a bundle is constructed and a routing decision is made, based on the deployed routing algorithm (e.g., CGR-ETO). That is, the routing algorithm compares the eligible routes to destination and chooses an optimal route based on the desired criteria, such as earliest delivery time. Once a route is decided, BP invokes a *callback* function that notifies the transport layer about the chosen routing path towards the destination, the local residual capacity for the selected outbound link, and the estimated bundle arrival time. In case the routing algorithm does not output the estimated arrival time, a rough estimation can be calculated at the transport layer based on the selected route, in a way similar to that of the CGR algorithm. The operation, then, proceeds with a new routing algorithm call, to estimate the route of the acknowledgment packet that will be transmitted from the destination to the data source node, at the expected arrival time. We note that we consider the processing delay between the arrival time of the transport packet and the creation of the acknowledgement a negligible percentage of the round-trip delay, and thus omit it from the overall calculation. The estimated routes of the transport packet and the corresponding acknowledgment are then concatenated into a joint round-trip path, i.e., a sequence of scheduled contacts from source to destination node and back.

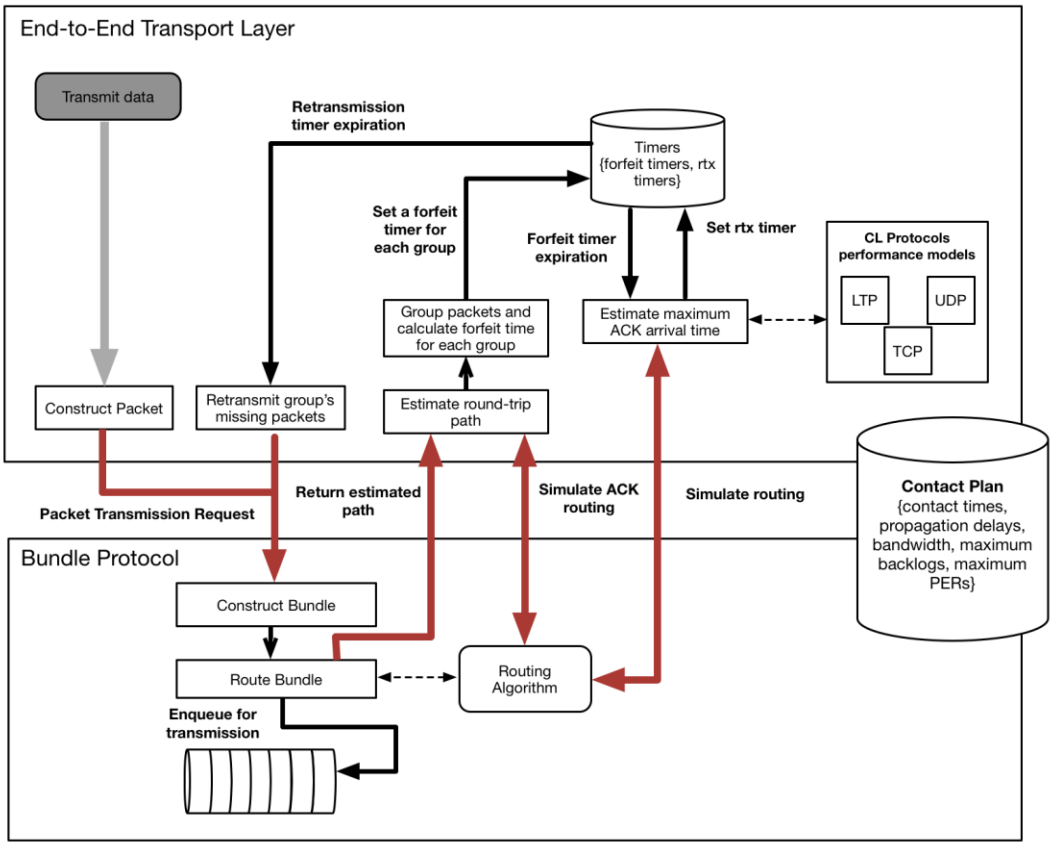

Figure 5-1 Operation diagram of the retransmission framework

As explained in Section 5.1.3, transmitted packets are grouped together based on the expected round-trip path and a single retransmission timer per group is set. A group is created as long as no other group associated with the same round-trip path exists and, for each group, a forfeit timer is set. Forfeit time is the earliest stop time of all contacts in the round-trip path and represents the latest time that a new transport packet can possibly be inserted into the group. Upon expiration of the forfeit timer, and since no new transport packets can be added to the group after that time, the transport protocol has enough information to calculate the worst-case acknowledgment arrival time, for the most recent packet of the group, and set the group's retransmission timer, accordingly. Given the aforementioned assumptions that packets have the same priority and lengt, and FIFO queueing discipline is applied, it is always assured that the worst-case acknowledgment arrival time for the most recent transport packet of the group is always greater or equal than that of the other packets of the same group. Thus, no spurious retransmissions are expected to occur, when RTO is calculated properly. Since network connectivity is also intermittent, the forfeit timer will expire before the expected RTT, and thus, no further delay will be imposed on the retransmission timer setup.

RTO interval for each group is calculated as follows: The worst-case arrival time of the group's last packet is calculated at each subsequent contact, based on (i) the transmission initiation time for this contact, (ii) the performance model of the deployed CL protocol, (iii) the worst-case conditions that are expected to be met (i.e., maximum backlog and maximum PER), and (iv) the particular network characteristics (i.e., contact times, bandwidth and propagation delays). If the estimated arrival time for a given contact exceeds the contact's stop time, i.e., if it is expected that the given bundle cannot be transmitted to the next-hop node successfully, within the *delayTolerance* confidence level, prior to the end of this contact, then a new route is calculated, according to the deployed end-of-contact policy. The transport protocol, then, uses the estimated, worst-case arrival time to simulate the routing algorithm call at the next node and calculate the –possibly updated– transmission initiation time for the next-hop contact. The process continues withal through the entire calculated path from source to destination and back, and completes when it estimates the worst-case acknowledgment arrival time at the sender node, configured as the corresponding group's RTO, and a retransmission timer is set.

Upon expiration of the retransmission timer, all transport packets that belong to the associated group and have not been acknowledged yet, are retransmitted. At the same time, relevant information associated with this group is discarded. Retransmitted data items are handled by the proposed retransmission framework as normal packet transmissions and therefore added in subsequent groups, based on their estimated round-trip path.

5.3 Implementation within Space DTN Architecture

In this Section, we detail the implementation of the proposed end-to-end retransmission framework within the confines of a typical space-oriented DTN architecture, which comprises of DTPC protocol as the end-to-end transport protocol, BP as the overlay protocol incorporating CGR routing, and LTP and UDP protocols as the available CL protocols below BP. In our implementation, it is always assumed that CL protocols operate at the edges of single point-to-point communication links and thus all hops in the network are also DTN hops. Our work can be easily extended to support multi-hop communication among DTN nodes, e.g., when BP operates over an IP-based network and TCP or UDP is used as a CL protocol.

In our description, we follow a bottom-up approach in order to describe the interaction between the different components of the proposed framework in a more comprehensive way. In Section 5.3.1, we briefly overview the shared contact plan information that must be available at any node employing the proposed retransmission framework. In Section 5.3.2, we present the performance models of the CL protocols considered in this work, while in Section 5.3.3, we describe the required modifications at the bundle layer and the associated CGR routing algorithm. Finally, in Section 5.3.4, we present the functional enhancements implemented in DTPC along with the corresponding RTO calculation algorithms, in order to establish the proposed retransmission framework.

5.3.1 Contact Plan Information

In the implementation of the end-to-end retransmission framework presented in this Section, we assume that all DTN nodes share a contact plan, in which each contact entry contains a minimum set of parameters pertaining to the different layers of the DTN protocol stack. A contact is identified by a *start time*, an *end time*, and the identities of the *transmitting* and *receiving nodes*. As far as the BP specific parameters are concerned, each contact entry should additionally contain the *maximum storage backlog* that may be experienced in the respective outbound transmission queue, and the employed *CL protocol*. The extra CL-protocol-specific parameters include the maximum utilized *packet size*, the imposed *overhead per packet*, the *maximum PER* that may be experienced in data transmissions over the given communication link, and the employed *end-of-contact policy*. The *end-of-contact policy* for LTP can be either *timer suspension* or *transmission failure*, while for other protocols, it should always be *transmission failure*. Finally, two link-specific information parameters are required: the anticipated link *propagation delay* (noted as OWLT in the range structure) and *transmission*

rate. Table 5-1 summarizes this minimum set of parameters, categorized according to the layer to which they pertain.

Table 5-1 Contact Plan Information

Category	Parameters
Identification	{Sending Node, Receiving Node}
	{Start Time, End Time}
BP specific	Maximum Storage Backlog (Bytes)
	CL protocol
CL protocol specific	Packet Size (Bytes)
	Maximum PER
	Overhead per packet (Bytes)
	End-of-Contact Policy
Link specific	OWLT (s)
	Data Rate (bits / s)

5.3.2 Delay Analysis Models of CL Protocols

In this Section, we describe how the performance of the various CL protocols considered in this work is modeled in order to accurately estimate bundle transfer time over each DTN hop. We note that two modes of operation are considered for the LTP protocol: (i) a fully reliable ("all-red") operation and (ii) a fully unreliable ("all-green") operation. We refer to the first mode as *LTP-Red* and to the second one as *LTP-Green*. Given the assumption that CL protocols operate over single point-to-point communication links, as far as bundle delivery time is concerned, the operation of LTP-Green matches UDP protocol operation: both protocols implement unreliable, rate-based data transmission. Thus, two different calculation algorithms are presented: one for LTP-Red protocol and one for LTP-Green and UDP protocols. Our concept can be easily applied in multi-hop CL connections, using the typical Internet transport protocols such as TCP or UDP, applying the appropriate performance analysis models (e.g., [19] for TCP). Given the bundle transmission start time and the contact parameters described above, the provided algorithms estimate the time needed for a bundle to be transferred over a single DTN hop:

(i) within a confidence level equal to the *delayTolerance* value, for the reliable LTP-Red, and

(ii) in a best-effort way, for the unreliable LTP-Green and UDP protocols.

5.3.2.1 LTP-Red algorithm

The algorithm presented in this Section refers to the operation of LTP protocol when a reliable ("all-red") bundle transmission is requested by BP. It leverages the work presented in Chapter 3 and uses part of the introduced analysis to estimate the maximum time interval required for a bundle to arrive at the destination node, within some confidence interval. Calculations have been simplified to restrain the processing overhead. In particular, we assume that both LTP EOB and report packets never get lost and, therefore, calculation of arrival probability is based on Eq. (9), instead of Eq. (13). Given that LTP protocol can support timer suspension when a contact ends, both possible end-of-contact policies are considered, i.e., timer suspension and transmission failure. The pseudo-code of the proposed algorithm is given in Table 5-2.

As shown in Table 5-2, the algorithm takes as input five parameters: *expectedTxTime*, *bundleSize*, *contact*, *prevContactProbability* and *delayTolerance*. Based on the values of the above parameters, bundle arrival time and the respective arrival probability are progressively estimated by considering a single (re)transmission round in each iteration. This basic process continues iteratively until either the *delayTolerance* confidence level is reached, or a retransmission cycle cannot be completed during the current contact. In the former case, calculations terminate and the algorithm returns the arrival time and the arrival probability values. In the latter case, though, different procedures are followed, depending on the applied end-of-contact policy, i.e., "transmission failure" or "timer suspension".

If the end-of-contact policy is set to "transmission failure", a transmission round can never span across more than one contact opportunities. That is, if the remaining contact duration is not sufficient for the current retransmission round to complete, the calculation procedure finishes, and the algorithm returns the arrival time and probability of the previous transmission round (i.e., the last complete round). If the achieved probability does not reach the desired confidence level, CL transmission is considered as *failed*, and bundle re-forwarding policy is applied at the BP layer. When the LTP-Red algorithm is called again to estimate the arrival time of a re-forwarded bundle, *prevContactProbability* is set equal to the previously returned probability value, which corresponds to the probability of having a successful bundle arrival during all previous contacts. As illustrated in Table 5-2, arrival probability calculations are always scaled by the term (1 – *prevContactProbability*), since the arrival probabilities for each contact are conditional to the probability of having an unsuccessful bundle arrival during previous contacts.

Table 5-2 LTP-RED_TX Algorithm

```
Input: expectedTxTime, bundleSize, contact, prevContactProbability, delayTolerance
Output: arrivalTime, arrivalProbability

txRound = 1; arrivalTime = 0; arrivalProbability = 0.0; startTime = expectedTxTime;
totalNumOfSegments = ceil(bundleSize / contact.packetSize);
bytesToTransmit = bundleSize + (totalNumOfSegments * contact.overhead);
while (arrivalProbability <= delayTolerance)
        if (contact.eocPolicy == "transmission failure")
                if ((startTime + bytesToTransmit / contact.txRate) > contact.endTime)
                        // Not enough contact duration
                        return [arrivalTime, arrivalProbability];
                end if
                txDelay = bytesToTransmit / contact.txRate;
                arrivalTime = startTime + txDelay + contact.propDelay;
                bytesToTransmit = bytesToTransmit * contact.PER;          // for next xmission round
                arrivalProbability = (1 - PER^txRound)^ totalNumOfSegments; // Eq. (9)
                arrivalProbability *= (1 – prevContactProbability);
                arrivalProbability += prevContactProbability;
                startTime = arrivalTime + contact.propDelay;
                // assuming txDelay = 0 for RS
                txRound++;
        else                                        // contact.eocPolicy == "timer suspension"
                remainingCapacity = max(0, (contact.endTime – startTime) * contact.txRate);
                while (remainingCapacity < bytesToTransmit)
                        // Not enough transmission capacity
                        contact <- nextContact;                 // Next contact with same rec. node
                        startTime = contact.startTime;          // Resumed timer
                        bytesToTransmit = bytesToTransmit – remainingCapacity;
                        remainingCapacity = max(0,  (contact.endTime – startTime)*contact.txRate);
                end while

                txDelay = bytesToTransmit / contact.txRate;
                arrivalTime = startTime + txDelay + contact.propDelay;
                if (arrivalTime > contact.endTime)      // LTP report transmission will be suspended
                        contact <- nextContact;         // and will arrive at the next contact
                        startTime = contact.startTime + contact.propDelay;
                        // assuming txDelay = 0 for RS
                else
                        startTime = arrivalTime + contact.propDelay;
                        // assuming txDelay = 0 for RS
                end if

                // Calculate new values for next transmission round
                arrivalProbability = (1 - PER^txRound)^ totalNumOfSegments;
                txRoundBytes = txRoundBytes * contact.PER;
                bytesToTransmit = txRoundBytes;
                txRounds++;
        end if
end while
return [arrivalTime, arrivalProbability];
```

The above process slightly differs in case the applied end-of-contact policy is set to "timer suspension". In this mode of operation, calculation of bundle arrival time can span across more than one contacts with the same receiving node. That is, if during a retransmission round the

remaining transmission capacity of the current contact is smaller than the total number of bytes to be transmitted (i.e., when *remainingCapacity* < *bytesToTransmit* in Table 5-2), LTP transmission is considered suspended, and resumes at the start of the next contact opportunity. This process can span across more than one contact windows, until all packets have been successfully transmitted and a transmission round is considered completed. Moreover, if during a transmission round the RS transmission time (i.e., the arrival time of the last RS byte at the receiver) is greater than the end time of the current contact window, the suspension of the report packet transmission is also considered. Report packet transmission is assumed to be resumed at the start of the next contact opportunity. Finally, in the special case where a transmission round is sent within the contact duration, but the last transmitted byte arrives at a time later than the contact end time[*], the new retransmission round always start at the beginning of the next contact. We note that, since in this mode of operation calculations are not limited within the boundaries of a given contact, it is guaranteed that the required confidence level will always be met, albeit possibly after several forthcoming contacts with the same receiving node.

5.3.2.2 LTP-Green / UDP Algorithm

Given our assumption that CL protocols operate over single point-to-point communication links, LTP-Green protocol operation closely resembles UDP. Bundles are segmented into packets that are directly transmitted to the remote peer over the deployed data link protocol. Packets are always transmitted unreliably at link rate and no intermediate relay nodes are used. Based on the above, only a single bundle arrival time calculation algorithm is presented here, which applies for both the LTP-Green and UDP protocol. As shown in Table 5-3, the algorithm accepts as input the same first three parameters of the LTP-Red algorithm, i.e., *expectedTxTime*, *bundleSize* and *contact*. Since no retransmissions occur, calculations are straightforward: bundle arrival time is simply the time that the last byte of the transmitted data is received at the receiving node. Also, since the protocols operate in unreliable mode, no confidence level can be guaranteed for bundle delivery. However, in order to maintain a standard output interface, bundle arrival probability is calculated and returned by this algorithm, as well. Its value, though, is not used at any point of the proposed framework implementation.

[*] The contact corresponds to the time window during which the sender can transmit data. The respective reception window spans until the last byte arrives to contact receiving node, at a maximum of (*End Time* + *OWLT*)

Table 5-3 LTP-GREEN_TX Algorithm / UDP_TX Algorithm

Input: *expectedTxTime, bundleSize, contact*
Output: *arrivalTime, arrivalProbability*

totalNumOfSegments = ceil(*bundleSize* / *contact.packetSize*);
bytesToTransmit = bundleSize + (*totalNumOfSegments * contact.overhead*);
arrivalTime = expectedTxTime + contact.propDelay + (*bytesToTransmit* / *contact.txRate*);
arrivalProbability = (1 - *PER*)^*totalNumberOfSegments*;

return [*arrivalTime, arrivalProbability*];

5.3.3 BP and CGR modifications

The proposed end-to-end retransmission framework is deployed as part of version 3.2.2 of ION implementation, which includes the RFC 5050 BP implementation [6] and the CGR-ETO variant of the CGR algorithm, presented in 4.1.2. In order to enable the required inter-layer communication between DTPC and BP/CGR, two modifications on the overall operation of BP/CGR are implemented.

The first modification includes the extension of the BP transmission request parameters, upon a DTPC data item transmission request. In particular, the set of parameters is extended to include the data item's transmission sequence number (*seqNo*), which uniquely identifies the data item within a DTPC payload aggregator, the *profile ID* of the transmission profile, and a pointer to the DTPC callback function that is responsible for the grouping of data items. Whenever a routing decision is made for a given bundle, BP invokes the respective callback function, notifying DTPC on the routing result of the data item that corresponds to the forwarded bundle. Routing information communicated -through the callback function- to the transport layer include: (a) the expected routing path, (b) the expected bundle arrival time at destination, and (c) the time that bundle is expected to depart from local node. We note that all this information is already available in CGR-ETO, the routing algorithm presented in 4.1.2 and implemented as standard CGR for ION (since v.3.2.1). More details about the input parameters and the operation of the callback function are given in Section 5.3.4.

The second modification refers to the implementation of a routing preview function, which simulates bundle routing, and can be directly called by DTPC:

[*route*, *expectedTxTime*] = SIMULATE_ROUTING(*routingTime*, *sourceEID*, *dstEID*, *itemSize*, *lifespan*, *priority*)

This function generates, for an item with given *itemSize*, *lifespan* and *priority*, routed from source node *sourceEID* to destination node *dstEID* at *routingTime*, a dummy bundle, and calls the routing algorithm, based on the input information and the global contact plan information. The routing algorithm, then, returns the expected path from source to destination, namely *route*,

as well as the estimated time that the bundle will be forwarded from source node, namely *expectedTxTime*. We note that CGR-ETO can exploit information about the current backlog in each outbound transmission queue of the source node, when calculating eligible routes, to account for the expected queueing delays. That is, the ETO information is restricted to local ETO for local outbound queues (see Section 4.1.1). Since, for routing simulations, the source node may not be the local node (i.e., the node that initiates data transmissions), it has no information about the outbound link queues' backlog. Therefore, we use the maximum backlog information contained in the contact plan, instead, to obtain the maximum possible queueing delays, and conform to the worst-case delay concept.

5.3.4 DTPC protocol modifications

The proposed enhancements of the DTPC retransmission framework comprise two distinct elements: i) the *Data Items Grouping Mechanism*, which exploits the BP routing decision in order to group data items based on the anticipated end-to-end path, and ii) the *Group RTO Calculation algorithm*, which calculates retransmission timers at group granularity, by estimating the worst-case acknowledgment arrival time of the last transmitted data item of each group, based on the desired confidence level (i.e., *delayTolerance*). These elements are described in the following two subsections, respectively.

5.3.4.1 Data Items Grouping Mechanism

The original version of DTPC protocol offers a simple retransmission mechanism that is based on one-to-one mappings between the applied retransmission timers and the outstanding data items. That is, DTPC sets one retransmission timer for each transmitted data item, based on a fixed retransmission interval, and the retransmission timer is canceled only upon the arrival of the ACK item that acknowledges the arrival of the respective data item at the receiver. In a DTPC sender, a data item is uniquely identified by its sequence number (*seqNo*) and the associated *payload aggregator*, where each payload aggregator is, in turn, uniquely identified by a destination EID (*destEID*) and a *profileID*. As a result, several data item flows can be produced by a DTPC sender, where the triplet {*srcEID, profileID, destEID*} uniquely characterizes a data item flow in the network and sequence numbers uniquely identify data items within each data item flow. Given that (a) different data item flows may call for different classes of service and (b) keeping the semantics of DTPC protocol intact is a reasonable design choice, in the proposed implementation of the end-to-end retransmission framework, we

consider a data items grouping mechanism that is applied on a per-payload aggregator basis. As described above, data items grouping is achieved through the incorporation of a DTPC callback function, which is called whenever a routing decision for a locally created bundle that carries a data item is made and notifies DTPC on the routing result. Table 5-4 shows the DTPC data aggregation callback function proposed in this work.

Table 5-4 DTPC Data Items Grouping Callback Function

```
Input: seqNo, profileID, destEID, expectedTxTime, forwardRoute, arrivalTime
Output: Create/Update RTO Group – Set RTO calculation timer

lifespan = dataItem(seqNo).expirationTime – arrivalTime;
[returnRoute, -] = SIMULATE_ROUTING(arrivalTime, destEID, srcEID, estimatedDTPCAckSize,
                    lifetime, profile(profileID).priority);
E2EPath = [forwardRoute, returnRoute];
payloadAggregator = findPayloadAggregator(profileID, destID);
group = findGroup(payloadAggregator, E2Epath);
if (group == NULL)
        group = createNewGroup(payloadAggregator, E2EPath);
        group.route = E2EPath;
        group.forfeitTime = min{E2EPath.contacts.stopTime};
        setForfeitTimer(group, group.forfeitTime);
end if

group.addItem(seqNo);
group.expectedTxTime = expectedTxTime;
return;
```

As displayed in Table 5-4, this function takes as input the parameters that uniquely identify the data item (i.e., the triplet *seqNo*, *profile ID*, *destEID*), as well as the routing algorithm output (*expectedTxTime*, *forwardRoute*, *arrivalTime*). Based on the estimated bundle arrival time, the route of the returned acknowledgment is simulated, using the SIMULATE_ROUTING algorithm described before, and the forward and return routes are concatenated into a single set of contacts, the *E2EPath*. *E2EPath* is used to classify data items into groups. In particular, all transmitted data items with the same expected *E2EPath* are considered to belong to the same group. Furthermore, whenever a new group is created, the group's forfeit time is calculated and a forfeit timer is set accordingly. We also note that, for each group, only the *expectedTxTime* of the most recent data item needs to be maintained and thus is updated whenever a new data item is added to the group. Upon expiration of a forfeit timer, i.e., when no further data item transmissions can be routed on the respective *E2EPath*, the Group RTO Calculation algorithm described in the next Section is executed, and a retransmission timer is set for the group, according to the maximum estimated acknowledgement arrival time. Upon expiration of a

retransmission timer, DTPC retransmits all data items of the respective group for which no acknowledgment has been received, and all group information is deleted. Retransmitted data items are treated as normal DTPC transmissions and, therefore, are regularly included into the RTO groups based on their expected *E2EPath*.

5.3.4.2 Group RTO Calculation algorithm

The Group RTO Calculation algorithm is the core algorithm of the proposed end-to-end retransmission framework; it coordinates its various components in order to calculate the desired worst-case RTO intervals. As illustrated in Table 5-5, the Group RTO Calculation algorithm takes as input the *sourceNode*, and the group data structure; the latter contains group-specific information (e.g., *E2EPath*, *lifespan* of the *lastDataItem*, etc.), as well as information on the corresponding DTPC aggregator (e.g., *destinationNode*) and its profile (e.g., *dataItemSize*, *estimatedBPHeaderSize*, *estimatedDTPCAckSize*, *delayTolerance*, *priority*, etc.). Since the size of the data items produced by a payload aggregator in DTPC protocol's aggregation mechanism can be variable in size, *dataItemSize* is always set equal to the aggregation size limit (ASL) of the corresponding transmission profile in order to calculate worst-case RTOs.

Starting from the first hop, Group RTO Calculation algorithm progressively calculates bundle arrival time at each hop; on the forward path, a bundle carrying the given DTPC data item is considered, while a bundle carrying the respective DTPC acknowledgment is considered on the return path. The algorithm completes with the calculation of the arrival time of the return bundle, at the *sourceNode*. For each hop, the corresponding CL protocol algorithm is used to estimate bundle arrival time. Whenever a hop transmission is completed, the SIMULATE_ROUTING algorithm is used to calculate the next hop on the path and the respective bundle transmission start time, based on the bundle arrival time from previous hop and the expected backlog storage. In the particular case where LTP-Red is the deployed CL protocol in a contact and the desired confidence level cannot be reached within the boundaries of this contact, the previous contact arrival probability is stored and bundle re-forwarding is simulated[*]. That is, SIMULATE_ROUTING algorithm is executed for the same source node and the value of *initTime* for the next iteration is set equal to the end time of the current contact.

[*] Bundles will be reforwarded only when *eocPolicy* = *transmissionFailure*; when *eocPolicy* = *timerSuspension*, the desired confidence level will always be met after (subsequent) timer suspensions, as mentioned in Section 5.3.2.1.

Table 5-5 Group RTO Calculation Algorithm

```
Input: group, sourceNode
Output: maxAckArrivalTime

previousContactProbability = 0.0; arrivalTime = 0; maxAckArrivalTime = 0;
contact = group.E2EPath[0];                // the first-hop contact of the path
profile = group.payloadAggregator.profile;
initTime = group.expectedTxTime;
lifespan = group.lastDataItem.lifespan;
// the lifespan value passed to BP when data item tx was requested
destinationNode = group.payloadAggregator.destinationNode;
bytesToSend = profile.dataItemSize + profile.estimatedBPHeaderSize;
while (contact != NULL)
        switch (contact.CLProtocol)
        case UDP:
                [arrivalTime, arrivalProbability] = UDP_TX(contact, bytesToSend, initTime);
                break;
        case LTP-Green:
                [arrivalTime, arrivalProbability] = LTP_GREEN_TX(contact, bytesToSend,
                        initTime);
                break;
        case LTP-Red:
                [arrivalTime, arrivalProbability] = LTP_RED_TX(contact, bytesToSend,
                        initTime, previousContactProbability, delayTolerance);
                break;
        end switch
        if ( (arrivalProbability < profile.delayTolerance) && (contact.CLProtocol == LTP-Red))
                previousContactProbability = arrivalProbability;
                [route, expectedTxTime] = SIMULATE_ROUTING(contact.endTime,
                        contact.fromNode, destinationNode, bytesToSend, lifespan,
                        profile.priority);
                contact = route[0];
                initTime = expectedTxTime;
                continue;
        end if
        // delayTolerance is met (LTP only), proceed with next hop
        if (contact.toNode == destinationNode)
                // data item has reached destinationNode, continue with ACK
                destinationNode = sourceNode;          // Update parameters for ACK calculations
                sourceNode = contact.fromNode;
                bytesToSend = profile.estimatedDTPCAckSize +
                        profile.estimatedBPHeaderSize;
                lifespan = group.lastDataItem.expirationTime – arrivalTime;
        end if
        if (contact.toNode == sourceNode)
                // ACK item has reached sourceNode, terminate algorithm
                maxAckArrivalTime = arrivalTime;
                break;
        end if
         [route, expectedTxTime] = SIMULATE_ROUTING(arrivalTime, contact.toNode,
                destinationNode, bytesToSend, lifespan, profile.priority);
        contact = route[0];
        initTime = expectedTxTime;
end while
return maxAckArrivalTime;
```

Chapter 6 Evaluation Methodology

In this Chapter, we present the methodology that we follow in order to evaluate the analytical tools, algorithms and protocols proposed in this thesis. In particular, we describe the goals of the described evaluation process (Section 6.1), the scenarios that we examine, including the topologies (Section 6.2), the metrics that we use to assess the performance of our mechanisms (Section 6.3), as well as the experimentation tools that we develop or exploit for evaluation purposes (Section 6.4).

6.1 Evaluation Goals

The main objective of any evaluation process is to examine how the proposed methods can be applied in the target environment or architecture, how they can solve the issues or inadequacies that motivated their introduction, as well as whether and to what extent they can achieve the goals set during the design process. In this context, the overall goal of the evaluation process that we follow in this thesis is to examine the applicability of the proposed methods in space internets, and in particular the space DTN architecture, and assess how they can improve the network's capability of estimating the delivery delay in different, space-oriented, data transmission scenarios. Furthermore, we define a set of evaluation goals that pertain to the individual methods that are part of the research performed in this thesis. These goals are summarized below:

- Evaluate the error prediction methods proposed in the BDTE analysis and validate its overall ability to provide detailed delivery profiles for bundle transmissions.
- Assess both the introduced queueing delay estimation methods, i.e., the reactive through the Contact Plan Update framework, and the proactive through management and forecasting of data rates, and investigate how they can improve the total delivery delay estimation, based on the consideration of queueing delays as well.
- Investigate the performance of CGR-ETO in bundle arrival time estimation and routing efficiency, and compare it to the previous CGR implementation, both with and without the employment of queue length update messages through CPUP transmissions.
- Compare the two distinct queueing delay estimation methods in terms of better queueing delay prediction.
- Examine the application of the proposed analytical methods in the DTPC dynamic RTO configuration, evaluate the introduced retransmission framework and compare it with the

original DTPC retransmission scheme, in terms of accurate RTT estimation, faster data delivery, and improved storage efficiency.

6.2 Scenarios

In order to evaluate the introduced research methods and achieve the evaluation goals described in the previous subsection, we design a set of evaluation scenarios. All these scenarios cover different use cases of data transmissions, within the general context of space internetworking, and are specifically tailored to assess the different, introduced methods, protocols, and algorithms, with respect to the aforementioned evaluation goals.

6.2.1 Scenario 1: Validation of Bundle Delivery Time Estimation tool

The first scenario validates the BDTE analytical method and application, in a bundle transmission over a space network that consists of two communication hops. In particular, we examine how BDTE provides an analytical delivery delay profile for the transmission of a bundle with payload length 100,000 Bytes, expected to be generated at node 1 in a specific future time, and transmitted to node 3 via node 2. The topology of this scenario is depicted in Figure 6-1, and the parameters used are provided in Table 6-1. The connectivity between the nodes are continuous, i.e., with no intermittency. The convergence layer protocol applied in both 1-2 and 2-3 links is LTP, with segment size equal to 1400 Bytes. The link and data transmission parameters used in this scenario do not correspond to a particular space mission; their purpose is rather to represent indicative values that pertain to deep-space communications. To this end, we apply different error rates in the two space links, and we examine how the developed tool forecasts the future error rates that pertain to those links, based on the past rate values.

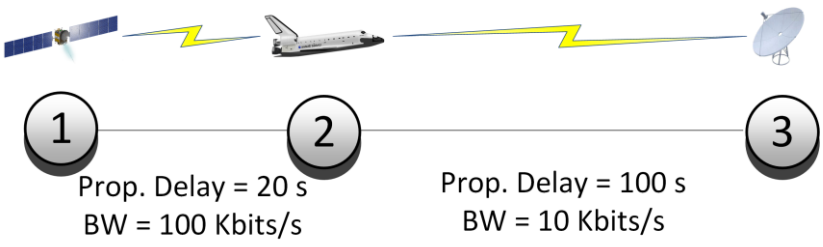

Figure 6-1 Scenario 1: Topology

In this scenario, BDTE is executed in an administrative node, which may or may not be one of the three participating nodes in the data transmission scenario. BDTE is assumed to have access to the contact plan information, and to have obtained the past network statistics through network management procedures. Based on this information, and on the application input, depicted in Table 6-2, BDTE predicts the future error rates that are expected at the two links during the transmission times of the bundle, calculates the possible transmission rounds that will be required to successfully transmit the bundle over each link, and, finally, outputs the complete profile of the bundle's plausible delivery times at destination, along with the corresponding probabilities.

Table 6-1 Scenario 1: Parameters

Parameter	Value
Packet size	1400 Bytes
Bundle size	100,000 Bytes
Propagation delay 1-2	20 s
Propagation delay 2-3	100 s
Bandwidth 1-2	100 Kbit/s
Bandwidth 2-3	10 Kbit/s
Transmission delay 1-2	1 s
Transmission delay 2-3	10 s

Table 6-2 BDTE Application Input

Parameter	Value
sending_node	1
destination_node	3
bundle_creation_time	11:00
bundle_lifetime	1000 s
bundle_size	100,000 Bytes
CL protocol packet size	1400 Bytes

6.2.2 Scenario 2: Evaluation of CGR-ETO and CPUP

In order to evaluate the Contact Plan Update framework, we design a multi-node, space data transmission scenario that may represent two reference space topologies, deep-space and near-Earth (both depicted in Figure 6-2), where a space asset (Node 1) extracts scientific data in-situ and transmits it to the Mission Operation Center (Node 6) via relay nodes 2 and 3, and Ground

Stations 4 and 5. The investigated topologies could refer to typical scenarios of Mars and Lunar missions. The network parameters used in this scenario are also depicted in Figure 6-2.

In this scenario, we evaluate the impact of CGR-ETO and CPUP on routing and delivery delay prediction. In particular, we study how far network knowledge about queueing delays can enhance the routing algorithm's ability to predict delivery times and improve routing performance, respectively. To this end, we examine the improvement of routing decisions, in terms of earliest delivery delay, and accuracy gain in delivery latency estimations. We perform simulations of one-week duration each, using the developed space DTN network simulator, namely *SpaceDTNSim*, which focuses on deterministic contact schedules; *SpaceDTNSim* is described in detail in Section 6.4.1. The contact plan is constructed using the Satellite Toolkit (STK) [167] for the space links, while the terrestrial links exhibit continuous contacts. Furthermore, to obtain results unbiased from the connectivity plan of specific days we also generate different contacts between the rover (node 1) and relay satellites 2 and 3, randomly generated during the simulation period. The amount of data generated per simulation is equal to the capacity that can be served by the network, which is the sum of the capacities of the first-hop contacts, since the first hop is the transmission bottleneck. The bundles have a size of 128kBytes, are of equal priority, and they are generated uniformly for the duration of the simulation period.

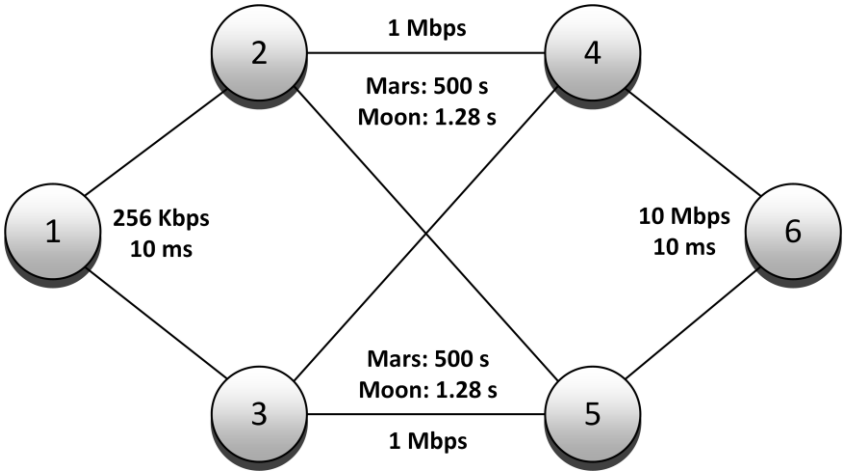

Figure 6-2 Scenario 2: Topology

6.2.3 Scenario 3: Evaluation of CGR-ETO in Satellite Communications

We continue the evaluation process with a Satellite Communications scenario; we deploy the scenario in an emulation environment, SPICE DTN Testbed, described in Section 6.4.3, and assess the performance of the implementation of CGR-ETO algorithm in ION DTN software.

For this scenario, we consider a simple four-node topology, as depicted in Figure 6-3. Node 1 represents a space asset, node 4 is the MOC, while nodes 2 and 3 are two terrestrial gateway stations, acting as DTN relays. In general, Node 1 may correspond to different space nodes that generate and transmit data to Earth, from deep-space or near-Earth environments. In this particular LEO satellite scenario under evaluation, the LEO satellite is the space asset, the first gateway is a terrestrial ground station of the LEO system, called LEO Control Center (LEO CC) in the figure, while the second gateway is the control center of a GEO constellation, acting as a space relay for the LEO satellite [168] [169]. Note that the GEO satellite, as non-DTN node, is transparent to CGR and thus not emulated in our deployment. The same topology could apply to other aspects of space communications as well. In Figure 6-3, dotted lines represent space intermittent links that use LTP as the convergence layer protocol, while the terrestrial continuous lines represent the continuous links that use TCP. In this scenario, we emulate both downlink and uplink data transmissions, with node 1 transmitting data to node 4, and vice versa, respectively.

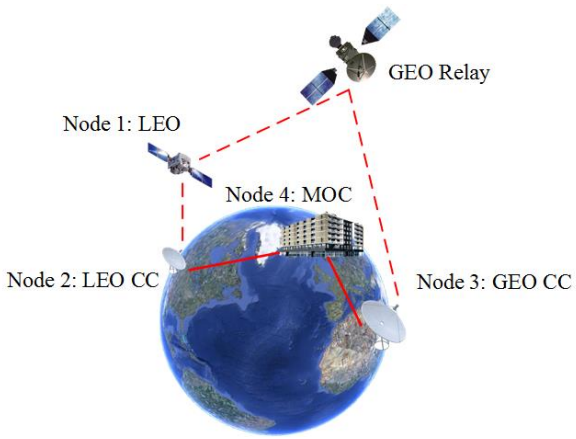

Figure 6-3 Scenario 3: Topology

In the contact plan employed in this scenario, space links (i.e., 1-2 and 1-3) are intermittent, whereas terrestrial links (i.e., 2-4 and 3-4) are continuous. We also examine a simplified routing case, where all contacts are continuous, in order to evaluate the ability of CGR-ETO to balance the transmission load between parallel equivalent routes. No propagation delays or data losses are artificially inserted in the emulated links, as they would make no difference to the present routing cost function. The contact characteristics are depicted in Table 6-3. In order to maintain a common ground for comparison, as well as for illustration purposes, we use symmetric links in our experiments, i.e., with equal transmission rates on the uplink and on the downlink. Although this symmetry is not typical in space communications, it has no particular impact in the present scenario, where we evaluate the routing function. Note that the sole contact between nodes 1 and 2 is nested in the first contact between nodes 1 and 3, and has a faster transmission

rate and a larger contact volume. As CGR lacks a specific syntax to denote continuous links, we have inserted a dummy contact for both terrestrial links (2-4 and 3-4), with end-time larger than the duration of the experiments. For convenience, in Table 6-3, the contact volume is also expressed in bundles, considering a bundle payload of 100kB, as in our experiments (with ECC = 107235B). The contact volume of continuous link is virtually infinite and is not displayed in the Table.

Table 6-3 Scenario 3: Contact plan

Link	*Contact#*	*Start-stop time (s)*	*Tx rate*	*Contact Volume*
1-2	1	65-90	512kbit/s	1.6 MB (14.9 bundles)
1-3	1	30-100	128kbit/s	1.12 MB (10.4 bundles)
1-3	2	125-195	128kbit/s	1.12 MB (10.4 bundles)
3-4 & 2-4	Dummy (cont.)	1-250	10Mbit/s	

In a first series of experiments, data are generated on board of the space asset and are transmitted to the MOC on Earth. Thus, the task of CGR in node 1 is to find the optimal route from 1 to 4 in the presence of intermittent links. Note that the best route may vary for successive bundles, because of this intermittency and limited contact volume. This is of primary interest in space communications, where the downlink of relatively large amounts of data (e.g., Earth observation images, results of scientific experiments on board of the asset, etc.) is often challenging due to the limited bandwidth of space links. The reverse direction is of interest as well, especially when the space asset is used as a DTN data relay, and is considered in a second series of experiments, where we examine data transmissions from the MOC towards the space asset (Node 4 transmits to Node 1).

To analyze the performance of CGR-ETO, we carry out a series of micro-analyses (i.e., bundle-by-bundle) using three different CGR versions: i) the "ECGR" implementation [126] present in old releases of ION (v.<3.2.1), lacking ETO functionality; ii) the official CGR implementation released in ION v.3.2.1, with CGR-ETO functionality limited to local information on first-hop queue lengths, called "CGR-ETO-first-hop", and iii) an experimental CGR version that includes local information for multiple hops, named "CGR-ETO-all-hops". All tests are carried out in four Linux machines in SPICE DTN Testbed, reproducing the layout of Figure 6-3.

As also described in Section 4.1.2, the CRG-ETO-first-hop version exhibits the same functionality and results with the version of CGR-ETO algorithm incorporated into ION standard CGR, since version 3.2.1, where consideration of ETO does not require Dijkstra recalculations, but ETO information is extracted based on the local outbound queue lengths at

the final step of route selection. Therefore, the latter version of the algorithm is omitted from the evaluation results.

6.2.4 Scenario 4: Evaluation of Proactive Queueing Delay Prediction Method

For the evaluation of the queueing delay prediction method, we consider the generic scenario of Section 4.2.1, and perform a simulation study on the *SpaceDTNSim* simulator, with different input parameters that fit the generic scenario, and compare the different delivery delay estimators. In particular, we provide comparisons of four different prediction methods:

i) the delivery time estimation implemented in "ECGR" [126],

ii) the delivery time estimation method that reactively exploits the queue data based on CPUP update messages, mentioned as "Contact Plan Update framework",

iii) the prediction method proposed in Section 4.2, mentioned as "Forecasting with Exponential Smoothing", and

iv) a prediction method similar to iii, where future values are not based on network statistics and time series forecasting methods, but are rather predicted with the assumption that all nodes transmit with nominal transmission rates. The latter is mentioned as "Forecasting with Nominal Rate".

We conduct a variety of simulations with different sets of parameters and periodic contact plans with period equal to half day and total duration equal to one week. Contacts are randomly put during this time period and follow a periodic pattern afterwards. For each set of parameters, we perform 100 repetitions to have a statistically adequate sample. The topology used is the one depicted in Figure 4-1, with different number of input nodes and varying parameters displayed in Table 6-4.

Table 6-4 Scenario 4: Parameters

Parameter	Value(s)
Number of Producing Nodes N	2, 5, 10, 20
Bundle Size	64 Kbytes
Capacities Ratio λ	0.1, 0.5, 0.9
Transmission Rate {1..N}-A	64 Kbits/s
Transmission Rate A-D	512 Kbits/s
Propagation Delay {1..N}-A	0.01 s
Propagation Delay A-D	1 s
Contact Duration {1..N}-A	600 s

We define λ as the ratio of the sum of all first-hop ({1..*N*}-*A*) contact volume capacities divided by the sum of all second-hop (*A*-*D*) contact volume capacities:

$$\lambda = \frac{N \times r_1 \times \tau_1}{r_2 \times \tau_2}, \tag{16}$$

where r_1 is the transmission rate of the first-hop links, r_2 is the transmission rate of the second-hop links, τ_1 is the duration of contacts {1..*N*}-*A* and τ_2 is the duration of contacts *A*-*D*. The value of λ is practically the ratio of the capacities of the two transmission hops. When $\lambda > 1$, the queueing system is unstable and can potentially lead to storage exhaustion and node failures. In our simulations, we use three different values of λ, 0.1, 0.5, and 0.9, and we set τ_1 = 600 s. The respective durations of the second-hop contacts are calculated using (16). We also examine different data production levels, with respect to the maximum amount of data that each of the first *N* nodes can transmit during the total simulation time. Bundle creation times are uniform for the total simulation period.

6.2.5 Scenario 5: Evaluation of Dynamic Retransmission Framework for DTPC

Finally, we evaluate the dynamic retransmission framework for DTPC protocol in a space emulation scenario. We implement the introduced changes of the proposed retransmission framework and integrate it into ION v3.2.2 [11]. We assess its performance in comparison to the original DTPC retransmission mechanism in a space scenario deployed in SPICE DTN Testbed [170]. In particular, we use a network topology (illustrated in Figure 6-4), with a Mars rover (NASA's Curiosity) capturing measurement data (e.g., images) and transmitting them to the MOC, via two relay satellites (MRO, and Odyssey) and three terrestrial ground stations (DSN in Canberra, Madrid, and Goldstone). The connectivity and topology details are extracted with the use of STK [167], based on the orbits and geography of the aforementioned space and terrestrial assets. In particular, Curiosity has an eight-minute link to each of the Mars satellites twice per day, while each satellite is connected to each of the three DSN Ground Stations (GSs) once per day for four hours. The contact plan has a period of one day and the daily connectivity pattern can be seen in Figure 6-5. The MRO – GS links are bidirectional, whereas the Odyssey – GS links are only used to downlink data. We use LTP green (i.e., unreliable) as CL protocol for the Odyssey-to-GSs downlink, as well as all uplinks, LTP red (i.e., reliable) for all other space downlinks of the network, and UDP for the terrestrial links. The BP-specific and link-specific parameters used in our evaluation experiments are illustrated in Table 6-5, the CL-

protocol-specific parameters used for all links are displayed in Table 6-6, and the DTPC-related parameters used in our experiments are displayed in Table 6-7.

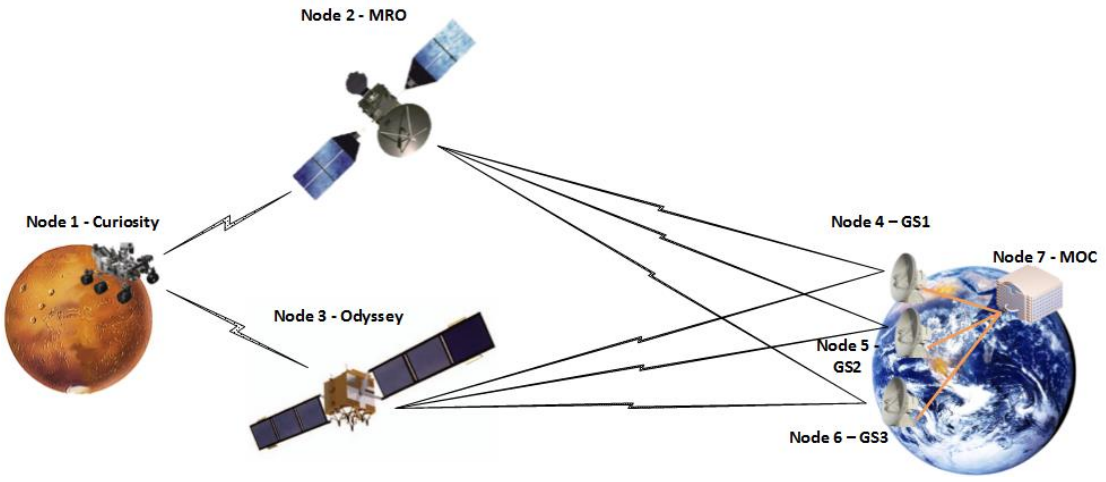

Figure 6-4 Scenario 5: Topology

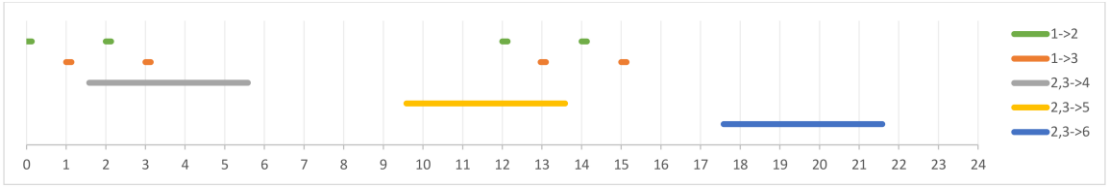

Figure 6-5 Scenario 5: Contact Plan

We name the updated DTPC, which incorporates the proposed retransmission framework, as DTPC with dynamic RTO (DTPC-dRTO), and the original DTPC retransmission mechanism as DTPC with static RTO (DTPC-sRTO). In all conducted experiments, Curiosity is generating application data units (ADUs) of size 9 Kbytes at a rate of 10 Kbits/sec, during the first day of the experiment (12000 ADUs in total), with MOC as the final destination. The Curiosity-Satellites links, as well as the terrestrial GS-MOC links are considered error-free, while for the deep-space links, we use PER = 1.1% (roughly corresponding to BER = 10^{-6} for the packet sizes used). During the time period of 9 – 14 hour (i.e., the time interval that corresponds to the second set of Satellite-GS contacts of the first day), the PER of the deep-space link increases to 10% (roughly corresponding to BER = 10^{-5}), to emulate bad weather conditions. Propagation delays and random artificial errors are injected using ION's one-way-light-time simulator [11].

Furthermore, we emulate the presence of sporadic cross-traffic generated at the two Mars satellites. In particular, MRO generates 150 Mbytes of data at time = 12 h, exactly before the third Curiosity-MRO contact of the first day, while Odyssey generates 30 Mbytes of data at time = 13 h, which is exactly before the third Curiosity-Odyssey contact of the first day. The

injected backlog inflicts extra queueing delay to a large set of data items that will arrive during the upcoming contacts from Curiosity and will be forwarded to the GSs.

We also evaluate the effect of the *delayTolerance* parameter, by setting the worst-case scenario to capture the 90% (*delayTolerance* = 0.9) and 99% (*delayTolerance* = 0.99) of the data items at the worst conditions (BER = 10^{-5}).

Table 6-5 Scenario 5: Topology Parameters

	BP Specific		Link Specific	
Links	***Maximum Storage Backlog* (KBytes)**	***CL Protocol***	***Prop. Delay* (s)**	***Data Rate* (kbits/s)**
1→2	-	LTP-Red	0.01	1,500
2→1	360	LTP-Green	0.01	256
1→3	-	LTP-Red	0.01	256
3→1	0	LTP-Green	0.01	32
2→{4,5,6}	200,000	LTP-Red	840	1,000
{4,5,6}→2	360	LTP-Green	840	128
3→{4,6}	200,000	LTP-Green	840	100
3→5	400,000	LTP-Green	840	100
{4,5,6}<->7	0	UDP	0.1	10,000

Table 6-6 Scenario 5: CL-Related Parameters for all links

Parameters	Values
Packet size	1400 Bytes
Maximum PER	10% (for *maximum BER* ~= 10^{-5})
Overhead per packet	50 Bytes
End-of-contact policy	"*transmission failure*"

Table 6-7 Scenario 5: DTPC-Related Profile Parameters

Parameters	Values
Lifetime	98 h
Maximum Number of Retransmissions (MNR)	3
DTPC-sRTO interval = Lifetime / (MNR + 1)	24.5 h
Aggregation Size Limit (ASL)	9000 Bytes
Aggregation Time Limit (ATL)	10 s
delayTolerance	0.9, 0.99
estimatedBPHeaderSize	26 Bytes
estmatedDTPCAckSize	4 Bytes

6.3 Metrics

Since the main focus of this thesis is the delivery delay, the majority of the metrics used for evaluation of the proposed methods and tools are also relative to the delivery delay of data. In order to assess the accuracy of the delay estimation methods, we also use prediction error and

prediction accuracy metrics, which, in essence, contain the same information but are expressed in a different way. In particular, we describe the specific metrics used for the evaluation of the different scenarios and the particular tools used in each scenario, in the following paragraphs:

In **Scenario 1**, we measure the BDTE application output, which contains the analytical profile of bundle delivery at destination. We obtain the *destination delivery time* and corresponding *probability*, as a detailed list. Furthermore, as an intermediate means for delivery delay prediction, we also predict the future *BER* for each link, through the time series forecasting procedure.

In **Scenario 2,** we evaluate the Contact Plan Update framework and compare it with the ECGR algorithm [126] using end-to-end *Bundle Delivery Delay* (*BDD*) measurements for two different cases, where data production is 50% and 100% of the maximum amount of data that can be forwarded to the network, respectively (i.e., from Node 1 to both relay nodes). We also compare ECGR against CGR-ETO (with ETO update thresholds equal to 1%, 5%, and 100%), in terms of both the CPUP overhead they employ, and the *Bundle Delivery Delay Prediction Accuracy* (*BDDPredAcc*), using the following metrics:

$$RelativeOverhead = \frac{TotalOverhead}{TotalDataPayload}$$

$$BDDPredAcc = \frac{BDD - Estimated\ BDD}{BDD}$$

Since delivery delays and accuracy percentages exhibit significant deviations, for the total number of bundles per simulation, we measure the *Cumulative Distribution Function* (*CDF*) of *BDD* and *BDDPredAcc*, additionally to their average values, in order to have a more indicative measure of the achieved results.

In **Scenario 3,** we perform emulation experiments to evaluate CGR-ETO and compare it with ECGR [126] in a realistic testbed environment, using a relatively small amount of bundles per experiment. Here, we also measure bundle delivery time at destination in a micro-analysis; that is, we do not provide any statistical metrics on the delivery times, as in the previous Scenario, but display them bundle-per-bundle, due to their small number.

In **Scenario 4**, in order to evaluate the accuracy of the queueing delay prediction method, we measure the *Bundle Delivery Delay Prediction Error*, both as an absolute time unit (*BDDPredErr*), and as a percentage (*NormalizedBDDPredErr*) of the *BDD*:

$$BDDPredErr = BDD - BundleDeliveryDelayEstimation$$

$$NormalizedBDDPredErr = \frac{BDD - BundleDeliveryDelayEstimation}{BDD}$$

Furthermore, in each simulation we calculate *Overhead*, which is the total number of bytes of the measurement information messages, as well as *RelativeOverhead*, which corresponds to the *Overhead* divided by the total number of data payload bytes, for the duration of the experiment:

$$RelativeOverhead = \frac{TotalOverhead}{TotalDataPayload}$$

Since simulations performed in this scenario involve a great number of bundles, with a significant variation in *BDDPredErr* and *NormalizedBDDPredErr*, we provide measurements of the CDF of both metrics, in addition to the average values, in the same way as in Scenario 2.

In **Scenario 5**, we focus on the DTPC transport protocol and its retransmission framework. Here, the important metric is the round-trip-time, which is defined as the interval between the transmission of a DTPC data item and the arrival time of the corresponding DTPC ACK item at the source node. In particular, we measure the *RTO Configuration Error*, which denotes the difference between the configured RTO and the actual RTT, based on the arrival time of the corresponding ACK item, as an increasing CDF function versus the data items percentile. We also calculate the destination reception times of the DTPC data items, in a micro-analysis (i.e., data item per data item), in order to illustrate the functionality of the proposed retransmission framework, as well as the distribution of *Payload Delivery* at destination, for the duration of the experiment. Finally, we measure the receiver node's *Storage Occupancy* through time, for the duration of the experiment, in order to showcase how the introduced retransmission framework improves the storage efficiency. To quantify the overall storage occupancy improvement through the whole experiment, we also calculate *totalStorageOccupancy* as the integral of the *storageOccupancy* for the duration of the data transmissions (measured in dataItems*days):

$$totalStorageOccupancy = \int storageOccupancy(t)\,dt,$$

as well as the *storageUtilization*, that corresponds to the *totalStorageOccupancy* normalized by the maximum amount of *maxTotalStorageOccupancy*, which is equal to 12000 dataItems * 3 *days* = 36000 dataItems*days, for all experiments.

$$storageUtilization = \frac{totalStorageOccupancy}{maxTotalStorageOccupancy}$$

6.4 Experimentation Tools

In order to evaluate the research methods proposed in this thesis, we conduct both simulation and emulation experiments, according to the scenario and evaluation objective. Simulation results are taken using *SpaceDTNSim*, a discrete-event simulator designed and implemented for the purposes of this thesis. Emulation experiments with real implementations of the developed elements are conducted using ION implementation, while the realistic, space-oriented network and link conditions are emulated in SPICE DTN Testbed. We use SpaceDTNSim to evaluate the Contact Plan Update framework and the proactive queueing delay prediction method, in particular in Scenarios 2 and 4, while ION DTN implementation and the SPICE DTN Testbed emulation environment are exploited in the validation of BDTE (Scenario 1), CGR-ETO implementation (Scenario 3), and the dynamic retransmission framework of DTPC (Scenario 5). In the following subsections, we summarize the functionality and usage of these experimentation tools.

6.4.1 SpaceDTNSim Simulator

In order to evaluate the queueing delay estimation methods introduced in Chapter 4, and in particular for Scenarios 2 and 4, we choose to perform a series of simulation studies, to obtain a variety of results, with sufficient statistical data and different parameter inputs, to better assess the performance of the introduced methods. Although a variety of widely used network simulators exist, such as NS2 [171], NS3 [172], Opportunistic Network Environment (ONE) [173], OMNET++ [174], etc., none of them is applicable to the space internetworking context, with deterministic and scheduled connectivity. Therefore, we chose to implement a new network simulator, namely SpaceDTNSim, dedicated to the functionality of the Interplanetary Internet. SpaceDTNSim is a Java-based, discrete-event simulator, and includes the core functionality of the space DTN architecture, including BP as the overlay network layer, while the connectivity of the nodes is based on deterministic contact plan schedules. It accepts as input a set of scenario parameters, configuration values, and the contact plan, and outputs detailed simulation results per bundle, as well as a set of values per experiment, e.g., total data delivery time, average delivery delays, average error in bundle delivery estimation etc.

SpaceDTNSim includes the CPUP protocol, as well as implementations of ECGR, CGR-ETO, and the queueing delay prediction method.

6.4.2 Interplanetary Overlay Network DTN Implementation

ION [11] is an implementation of the DTN architecture developed by JPL and released as open source software. It includes implementations of BP [6], Bundle Security Protocol (BSP) [175], DTNperf [176], CGR [9], a set of CLs such as LTP [7], TCP [59], Bundle Streaming Service (BSS) [177] [178], application-layer protocols like class-1 (unacknowledged) CFDP [20], Asynchronous Message Service (AMS) [179], DTPC protocol [8], etc.

ION is one of the most commonly used DTN implementations, together with DTN2 [180], which is the reference implementation of the DTN architecture, and IBR-DTN [181], an implementation of the Bundle Protocol designed for embedded systems. Since it has been specifically designed for delay-tolerant space communications, we chose it as the target implementation to incorporate and evaluate the developed tools, for the purposes of this thesis. ION was used in the DINET experiment [46], the JAXA-NASA joint experiments with JAXA's GEO DRTS [124], in the METERON project [50] and other ISS experiments [48] [49], as well as the Space Data Routers European Project [24].

6.4.3 SPICE DTN Testbed

SPICE DTN Testbed [170] was originally developed [182] [183] [184] as a prototype DTN testbed for space communications under a contract of ESA, within the project Extending Internet into Space [185], and received further funding from EC's FP7 Space Internetworking Center project [10], to be enhanced with more nodes and specialized components that accurately emulate the functionality of typical ground stations, space links and satellites. Its aim was to build an experimental research environment for developing and evaluating a variety of new architectures and protocols for space communications. In particular, SPICE DTN testbed presents the following key features:

i) Realistic emulation of space communications: Unlike the majority of existing DTN testbeds, which focus on terrestrial delay-tolerant communications, SPICE testbed provides a realistic experimental environment for satellite and space communications, including real and flight-ready components. Indeed, specialized hardware and software components have been incorporated into the testbed, enabling the testing, evaluation and validation of implemented

mechanisms and protocols. Furthermore, a link with a geostationary satellite, namely HellasSat 2, is utilized on demand, to provide real satellite link characteristics for experimental purposes.

ii) Compliance with typical equipment of major space agencies: SPICE DTN testbed incorporates typical components used by space agencies for the evaluation of protocols prior to mission launch. In particular, the Portable Satellite Simulator (PSS [186]) was built in compliance with ESA's requirements, while CORTEX CRT [187] is used by all major space agencies in their ground station facilities to support their missions. Finally, STK [167] is employed by mission designers as a tool to calculate not only exact satellite trajectories and contact durations, but also detailed communication characteristics, and perform link-budget analysis.

iii) Interface provision for multiple underlying protocols: SPICE DTN testbed not only supports a variety of convergence layers for underlying protocols that comply with CCSDS standards and major space agencies, but also facilitates the development of novel routing, transport, and management schemes. Taking advantage of this functionality, SPICE researchers are able to validate such schemes against standardized protocols, and perform interoperability testing.

iv) Scalability: SPICE DTN testbed includes numerous nodes for the evaluation of complex communication scenarios that involve several space assets and can be further enhanced with virtual nodes installed on a high-performance server. Therefore, complex scenarios involving constellations of satellites (e.g., cubesats) and several end-users can be realistically modeled. It should also be mentioned that this scalability comes without adding any complexity, since the testbed is easily configured and controlled through dedicated workstations.

Notionally, the testbed comprises two distinct parts, namely the data plane and the control plane, and its architecture is depicted in Figure 6-7. In the former, data are transferred between nodes to emulate communication among space and ground assets, while configuration scripts, control messages, and reports related to the emulation are managed through the latter.

The control plane is responsible for (a) configuring and controlling the testbed nodes in real time based on user input, (b) monitoring the correct node operation, (c) collecting any associated performance statistics, and (d) delivering the experimental results to the researchers. These operations are coordinated by a main controller accessible via the internal network or the Internet. Researchers configure the experiments to be conducted through a user interface (UI), available at the main controller. Link characteristics and emulation parameters are either imported directly by the users or provided by the STK workstation after conducting the relevant simulations. Upon the completion of an experiment, results are collected and stored in the main controller.

At the data plane, SPICE testbed supports the emulation of a wide variety of space and satellite communication scenarios, including present and future missions. These scenarios may

involve (a) a number of landed assets, such as landers and rovers, that generate scientific data and can possibly form a planetary network, (b) a set of space assets near Earth or in deep space (e.g., LEO/MEO/GEO satellites, spacecraft, planetary relay satellites etc.) that can produce and/or relay data, (c) terrestrial facilities such as typical ground stations (GS), mission operation centers (MOC) and end-users. Researchers are able to emulate all these types of space communications taking advantage of the diverse protocol stack configurations supported by SPICE DTN testbed (Figure 6-6).

SPICE DTN testbed includes three DTN implementations, ION [11], DTN2 [180], and IBR-DTN [181], ESA's implementation of CFDP [20], SIMSAT [188], which is a general-purpose real-time simulation infrastructure developed for ESA, STK [167], and Network Emulator (NetEm) [189]. For the evaluation purposes of this thesis, we particularly use the ION DTN implementation, specifically mentioned in the previous subsection. We also exploit the functionality of STK to obtain accurate experiment configurations from real missions, including information like bandwidth, error rates, propagation delay, disruption periods and connectivity schedules. Finally, we use NetEm to modify networking properties and emulate propagation delays, data losses, and transmission rates, according to the designed scenario.

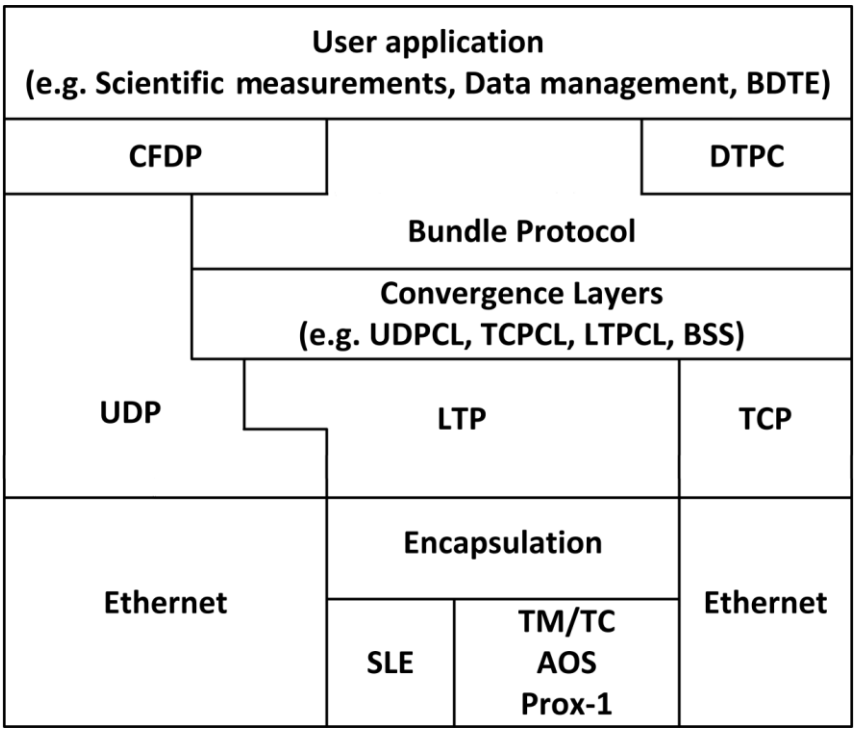

Figure 6-6 SPICE DTN Testbed protocol stack

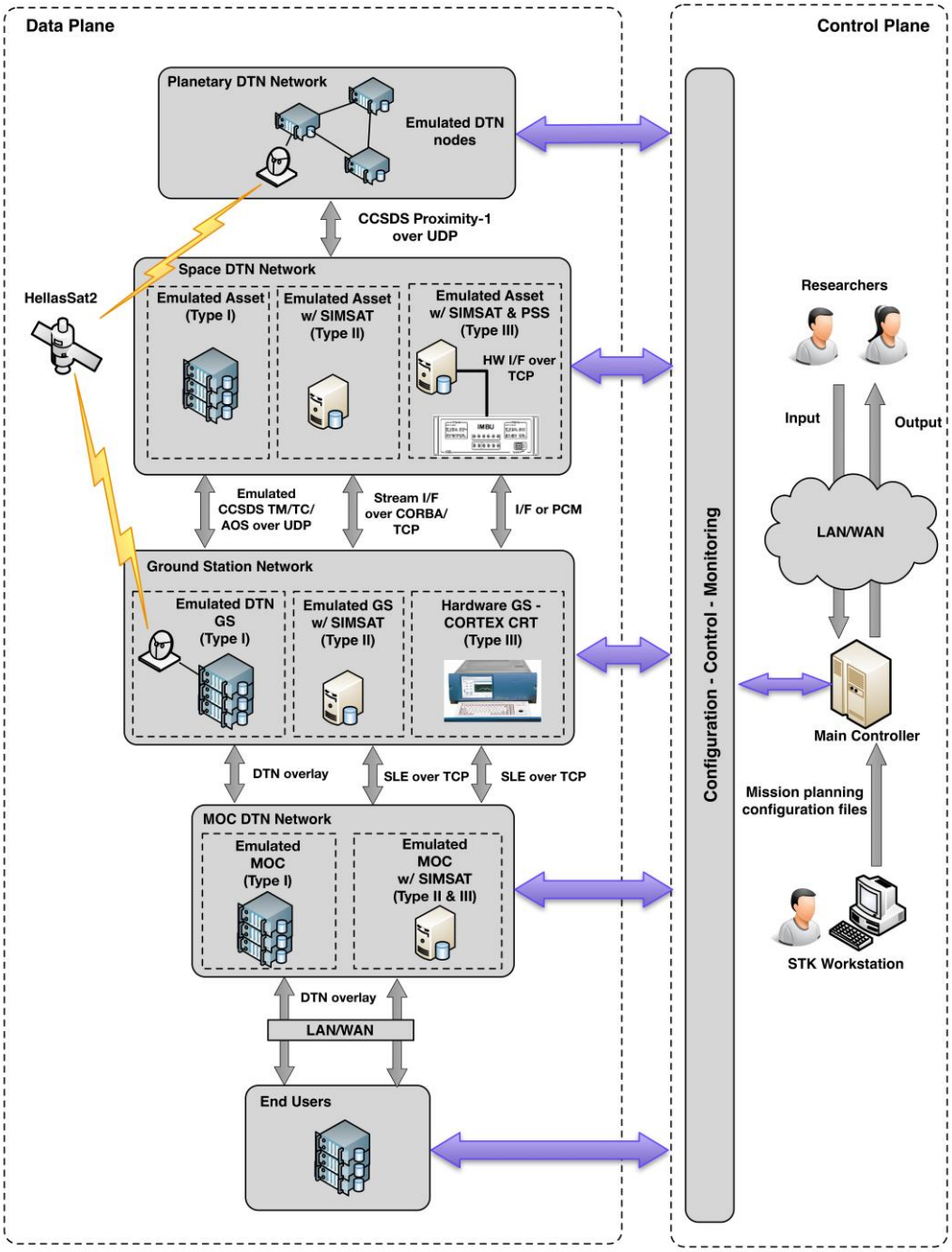

Figure 6-7 SPICE DTN Testbed Architecture

Chapter 7 Evaluation Results

7.1 Scenario 1

The goal of the first Scenario is to examine the applicability of the BDTE tool, and its corresponding analytical methods, in a space data transmission use-case. We validate the applied error rates forecasting method, and examine how the obtained forecasts assist in the estimation of the delivery times, for some bundle that will be transmitted in a specific future time. Finally, we examine how the different delivery times and the corresponding probabilities compose the analytical profile of the bundle delivery time.

The parameters used in this scenario (with topology depicted in Figure 6-1) are detailed in Table 6-1. The BDTE application, executed in an administrative node, gets the admin user input, illustrated in Table 6-2, and estimates the different delay components that pertain to the bundle transmission, based on the algorithmic and analytical methods detailed in Chapter 3.

We apply different error rates at the two space links. At the link between nodes 1 and 2, BER distribution includes seasonality with period = 9 time slots, a linear trend, and a random error with normal distribution. At the link between nodes 2 and 3 we apply a random error distribution with average value equal to $2 \cdot 10^{-7}$. BDTE forecasts the future error rate values, based on the corresponding past values that are stored in the DB. The BER distribution and the corresponding predicted values, for several periods of time, are depicted in Figure 7-1, for link 1-2, and in Figure 7-2, for link 2-3. The thick lines represent the forecast values, for a set of future time periods, while the grey areas represent the corresponding 95% confidence intervals. As observed in Figure 7-1, BDTE successfully identifies both the periodicity and the trend of the BER time series, and forecasts the future values based on the obtained configuration values. On the other hand, for the link between nodes 2 and 3, the random error distribution is interpreted as a Holt-Winters model without seasonal component and with an insignificant trend as illustrated in Figure 7-2, while the confidence intervals are significantly wide.

According to the application input, an administrative user wants to calculate the delivery profile for a bundle transmission that will be initiated at time 11:00:00, at node 1. Since the connectivity in this scenario is continuous, at that time, the bundle transmission initiates from node 1 to node 2. The predicted BER for this moment is $2.4177 \cdot 10^{-7}$ and the expected transmission rounds are calculated and displayed in Table 7-1.

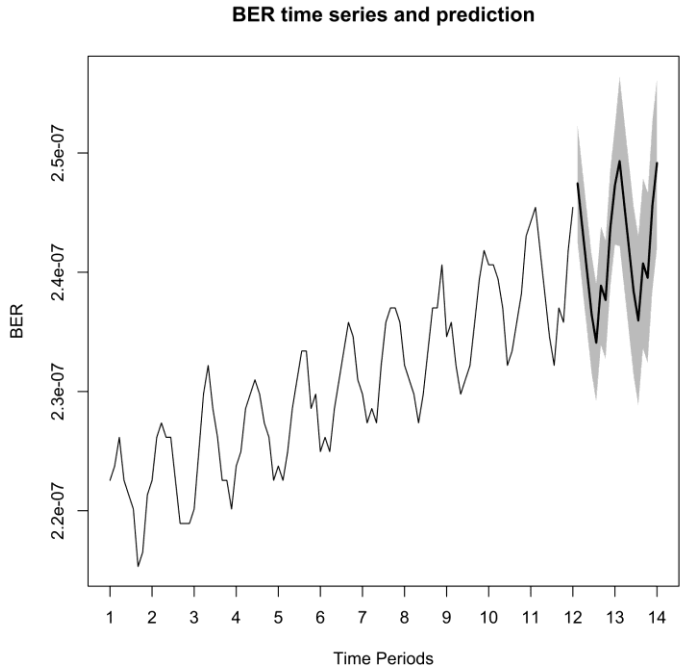

Figure 7-1 BER time series for link 1-2 with seasonality and trend

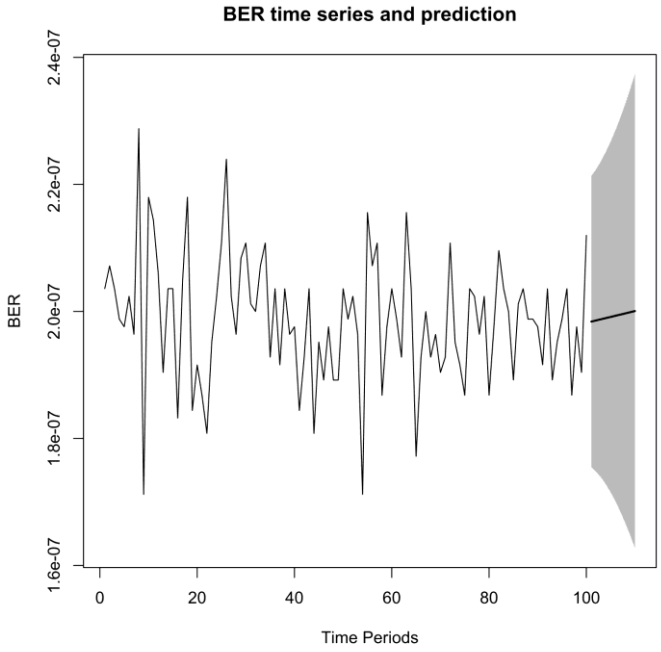

Figure 7-2 BER time series for link 2-3 with random values

After the first hop calculations, three distinct cases are extracted for the bundle to arrive at node 2: after 21, 61, or 101 seconds (1, 2, or 3 transmission rounds) with probabilities 0.822864, 0.176609 and 0.000525, respectively. Each one of them is then treated separately in new simulations for the next hop (2-3) with transmission initiation time equal to the bundle arrival

time at node 2, or the contact opening time between 2 and 3, whichever of the two times is later. In our scenario, the contact between 2 and 3 is always on (i.e., continuous connectivity), so the transmission initiation time from node 2 to 3 is 11:00:21, 11:01:01 and 11:01:41, correspondingly, for the 3 distinct cases. For each transmission initiation time, BDTE calculates BER for link 2-3, the possible transmission rounds, and the corresponding arrival times at node 3. For example, for transmission time 11:01:41, from node 2, BDTE estimates three distinct arrival times at node 3, at times 11:03:31, 11:06:51, and 11:10:11, with corresponding probabilities 0.84926, 0.15037, and 0.00037. The derived probabilities for second-hop transmissions are then multiplied with the previous ones (in this example with 0.00053) to calculate the final probability for each delivery time at final destination.

The times at destination are then sorted and the cumulative probabilities are calculated, accordingly. Table 7-2 shows, for the examined bundle transmission, the gathered cumulative probabilities, which represent the probabilities that a bundle will have been delivered at the final destination node before a specific future time.

Table 7-1 BDTE Calculations for Scenario 1

Link	Bundle Xmit Time	Predicted BER	Initial Probability	Probability	Time needed (s)	Arrival Time
1-2	11:00:00	$2.42 \cdot 10^{-7}$	1	0.82286	21	11:00:21
			1	0.17661	61	11:01:01
			1	0.00053	101	11:01:41
2-3	11:00:21	$2.02 \cdot 10^{-7}$	0.82286	0.84926	110	11:02:11
			0.82286	0.15037	310	11:05:31
			0.82286	0.00037	510	11:08:51
2-3	11:01:01	$2.02 \cdot 10^{-7}$	0.17661	0.84926	110	11:02:51
			0.17661	0.15037	310	11:06:11
			0.17661	0.00037	510	11:09:31
2-3	11:01:41	$2.03 \cdot 10^{-7}$	0.00053	0.84926	110	11:03:31
			0.00053	0.15037	310	11:06:51
			0.00053	0.00037	510	11:10:11

Table 7-2 Cumulative probabilities for bundle arrival time

Time	Probability
11:02:11	69.8829%
11:02:51	84.8817%
11:03:31	84.9262%
11:05:31	97.2994%
11:06:11	99.9550%
11:06:51	99.9629%
11:08:51	99.9932%
11:09:31	99.9998%
11:10:11	99.9998%

The reason that cumulative probability never reaches 100% percentage is the *MAX_TRANSMISSION_ROUNDS* and *PROBABILITY_THRESHOLD* filters that limit the number of considered rounds, as well as the bundle lifetime. If those limits were raised (i.e., more transmission rounds, smaller probability threshold), the result would consider cases of 4 or more transmission rounds, and the new percentages would be summed up to a percentage closer to 100%. The absolute 100% can be theoretically achieved if the *MAX_TRANSMISSION_ROUNDS* parameter is set to infinite and *PROBABILITY_THRESHOLD* is set to zero, with infinite connectivity and bundle lifetime.

Figure 7-3 shows the analytical profile of the bundle delivery time as a CDF, based on the results shown in Table 7-2. The probability distribution follows gradual increases with every plausible arrival time at destination, reaching the maximum value of 99.9998%. One of the useful results extracted from the application output is the earliest plausible arrival time, which corresponds to the best-case transmission scenario for the bundle, where no data are lost and no retransmissions are required. On the other hand, the latest plausible arrival time is theoretically infinite; in practice, however, it is limited by the end-time of the last contact in the contact plan between the communicating network nodes, as well as the bundle's lifetime. Since BDTE takes into account the lifetime, in routing procedures, it can also provide the theoretical maximum of a bundle transmission, when no *MAX_TRANSMISSION_ROUNDS* and *PROBABILITY_THRESHOLD* limits are used.

Based on the results of Figure 7-3, we can also use the application as a QoS-equivalent for space communications, in the sense of time delivery guarantee. That is, given a certain confidence *C* as user input, BDTE can estimate the time that its delivery is guaranteed with confidence *C*. For example, in our scenario, for a confidence input of 95%, we can guarantee that a bundle will have reached its destination with 95% confidence before 11:05:31. For this calculation we consider the cumulative probability that is greater or equal to C since, for the previous time (i.e., 11:03:31), we can't guarantee the delivery with confidence 95%.

Furthermore, BDTE can provide, for a given future time, the probability that the bundle will have reached the destination before that time. For example, in our scenario, if the administrative user wants to calculate the probability that the 100,000-Byte bundle will have been delivered before 11:04:00, BDTE will output the calculated probability, which is equal to ~84.9%.

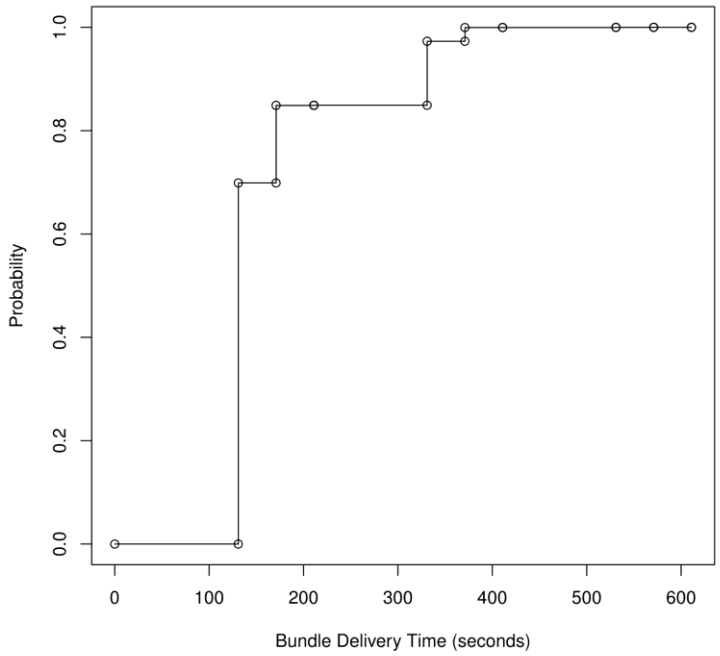

Figure 7-3 Cumulative distribution of bundle delivery times at destination

7.2 Scenario 2

In this scenario, we evaluate the efficacy of the Contact Plan Update framework in the queueing delay estimation problem. We investigate the applicability of the CGR-ETO algorithm and the CPUP protocol, as a collaborative framework that increases the efficiency of the network in estimating queueing delays, in a data transmissions use case (with topology illustrated in Figure 6-2), applicable to mars and lunar communications.

We conduct a set of simulations and compare CGR-ETO and ECGR using end-to-end *Bundle Delivery Delay* (*BDD*) measurements for two different cases, to simulate light-load and heavy-load network traffic, with data production levels equal to 50% and 100% of the maximum amount of data that can be forwarded to the network, respectively (i.e., from Node 1 to both relay nodes). We also compare ECGR with CGR-ETO, with the use of different *contact plan update thresholds* (1%, 5%, and 100%), in terms of both the transmission overhead they

inflict, with the disseminated CPUP data units, and the *BDD Prediction Accuracy* (*BDDPredAcc*).

Intuitively, we expect that CGR-ETO contribution will be more significant in scenarios with heavy traffic conditions, where queueing delay greatly affects network performance. In Figure 7-4, we illustrate the CDF of *BDD* in cases with different traffic load, versus the bundles' percentile. In this figure we observe that the contribution of CGR-ETO at the light-traffic scenario is relatively small, since queueing delay is a minor portion of the total delivery delay. Moreover, due to the intermittency of the connectivity between network nodes, the queueing delays in the first hops of the transmission may not affect the total delivery times, when they do not perturb the transmissions to next contacts with the same nodes. For example, a small queueing delay in a transmission between nodes 1-2 will detain the arrival of a bundle at node 2, but may cause no difference at the transmission start time for the next hop 2-4, if the corresponding contact has not started yet. At the heavy-load case, on the other hand, the significant amount of traffic imposes significant queueing delays, affecting also the final delivery time. Since CGR-ETO takes into account calculations on queueing delays, it achieves significant improvement compared to ECGR, in the delivery time for a percentage of the bundles. Indeed, queueing information assists node 1 in performing more prudent routing decisions, balancing the traffic load between the two relay nodes 2 and 3. Consequently, CGR-ETO efficiently mitigates the effects of intense network load.

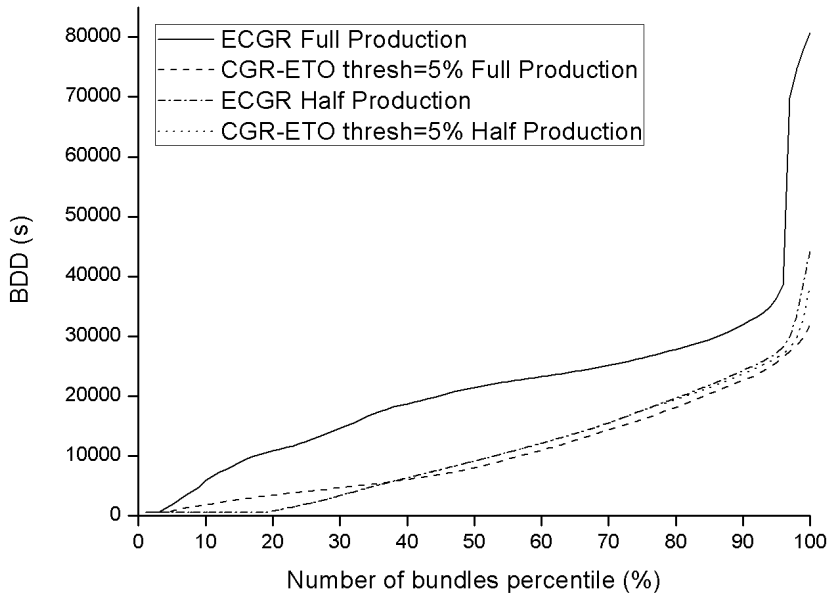

Figure 7-4 Bundle Delivery Delay (BDD) CDF

In Figure 7-5, we present the CDF of *BDDPredAcc* for ECGR and CGR-ETO with the three distinct *contact plan update threshold* levels, at the scenario with heavy data production rate. As mentioned before, in our evaluation process we have used a single threshold value, for both contact plan updates and CPUP command triggers. As observed in Figure 7-5, CGR-ETO performs 100% accurate predictions of the delivery delay for about 40% of the transmitted data, irrespective of the threshold value. The differentiation of threshold values affects the less accurate 30-40% of the bundles. In particular, CGR-ETO with 1% *contact plan update threshold* achieves at least 80% *BDDPredAcc* for *all* bundles, whereas the corresponding minimum accuracy is ~60%, for 5% threshold, and ~40%, for 100% threshold. Since, as mentioned before, bundles in this scenario have the same priority, a 100%-threshold configuration implicitly defines that no CPUP messages are disseminated through the network. As indicated in Figure 7-5, incorporating *ETO* in the *BDD* prediction has a substantial effect on the prediction error reduction even without disseminating the *ETO* information (*i.e.*, when no CPUP packets are transmitted).

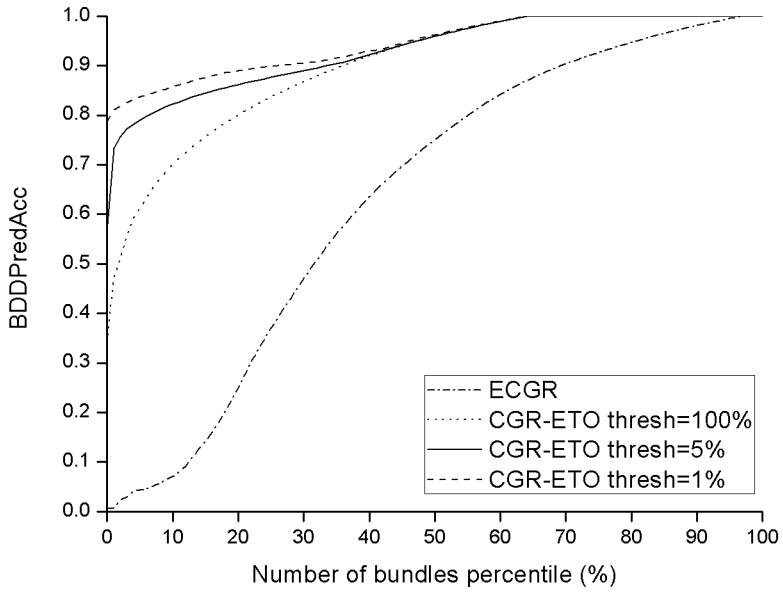

Figure 7-5 CDF of *Bundle Delivery Delay Prediction Accuracy* (*BDDPredAcc*)

In Figure 7-6, we illustrate the average values of *BDDPredAcc*, in conjunction with the overhead imposed by CPUP, for both low- and heavy- data production rates. Here, we observe that, in low-traffic network conditions, CGR-ETO improves the average delay prediction accuracy by ~10-20% in comparison to the ECGR algorithm. In heavy-traffic conditions, this improvement becomes more significant; the ~64% average *BDDPredAcc* of ECGR becomes 90-95%, with the use of CGR-ETO. We also observe that the use of finer granularity in ETO

updates and CPUP disseminations (with 5% and 1% *contact plan update thresholds*) has a slight improvement in the average delay prediction accuracy.

Finally, in Figure 7-6, we also depict the relative overhead that pertains to the queueing delay notifications, imposed by disseminations of CPUP messages. The use of ECGR, as well as CGR-ETO with 100% threshold, involves no CPUP transmissions, and, therefore, features zero overhead. We observe that the overhead caused by CPUP transmissions is in the order of 10^{-6} of the total size of data transmitted per simulation, with a maximum *relative overhead* equal to $3.3 \cdot 10^{-6}$, for 1% threshold and the heavy-traffic case. We also see that the improvement that smaller *contact plan update thresholds* have on the *BDDPredAcc*, come at a minor increase in the CPUP overhead.

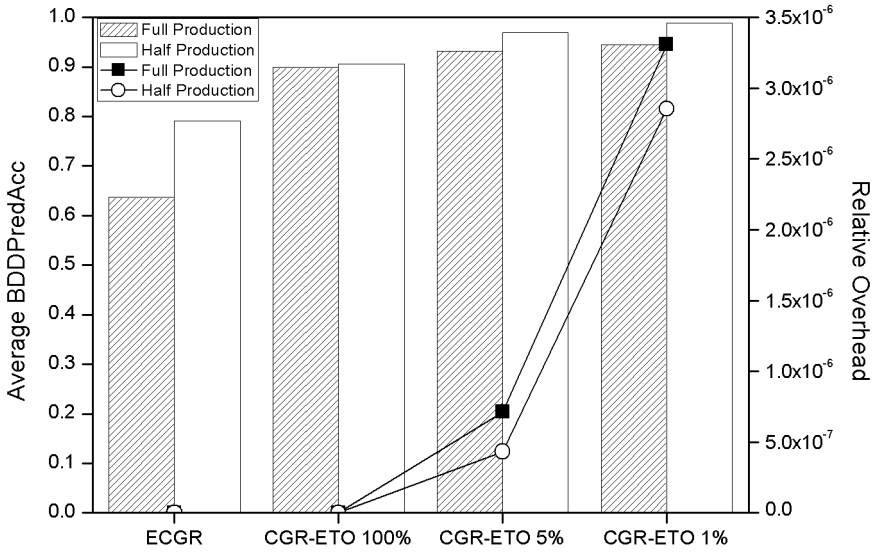

Figure 7-6 Average *BDDPredAcc* and *Relative Overhead*

7.3 Scenario 3

We continue with the evaluation of the CGR-ETO algorithm in a satellite communications scenario. An indicative topology of the scenario is depicted in Figure 6-3, which can also represent different scenarios with similar connection plans, e.g., for deep-space communications. In this scenario, our purpose is to assess the performance of the CGR-ETO algorithm in routing decisions and also evaluate its implementation in a realistic environment with nodes running the full DTN protocol stack.

To this end, we consider three different data transmission cases: At the first case, all links exhibit continuous connectivity and node 1 transmits data towards node 4. At the second case,

we modify the contacts based on the scheduled plan of Table 6-3, and evaluate downlink data transmissions (from node 1 to node 4 again) in the presence of intermittency. Finally, at the third case, node 4 transmits data at the uplink to node 1. Although the topology seems symmetrical, a major difference between downlink and uplink data transmissions is that, in the former, links are intermittent at the first hop and continuous at the second hop. Therefore, as short contacts are more susceptible to exhaustion, ETO updates are more meaningful for local outbound queues, at the intermittent links 1-4 and 2-4. On the other hand, for uplink data transmissions, the intermittent links and the corresponding short contacts are at the second hop to destination, and, hence, ETO updates are also necessary for non-local outbound queues.

In this context, we evaluate the routing performance of two different versions of CGR-ETO implementation, based on the different types of ETO updates, described in 4.1.1: at the first version, ETO is updated only for local contacts, based on local routing decisions, whereas at the second version, ETO is updated for all contacts that a locally-routed bundle is expected to follow through the path to destination. The two versions will be referred to from now on as *CGR-ETO-first-hop* and *CGR-ETO-all-hops*, respectively. Furthermore, we compare the two algorithms with ECGR [126], where no queueing delays are taken into consideration in routing decisions. To assess the routing performance of these algorithms, in this emulation scenario we carry out a micro-analysis and exhibit the delivery times, bundle-by-bundle.

7.3.1 Downlink data transmissions with parallel equivalent routes

We start by considering the extreme case of two equivalent parallel routes via 2 and 3. Here we deploy a different contact plan, where we use two equal contacts with duration = 110s, tx rate = 128 kbit/s, and contact volume = 1.76 MB, equivalent to 16.5 bundles. The contact to 2 starts just 1s before the contact to 3, at 29 s. This case, although clearly unrealistic, clarifies the improvements introduced by ETO, in terms of improved delivery time estimation and load balancing. Node 1 generates 16 bundles of 100kBytes each, all of the same priority. In Figure 7-7 we depict the routing decisions of ECGR, and we also illustrate the intervals of first-hop contacts (1-2 and 1-3), at the lower part of the diagram, for convenience.

The routing algorithm is executed for each bundle as soon as it is generated and, when the routing decision has been taken, puts the bundle to the corresponding outbound queue to the chosen neighbor. As ECGR does not consider the queueing delay caused by the previously routed bundles, and since the contact to 2 assures for ECGR a delivery time one second shorter than its competitor (i.e., the contact to 3), all bundles are routed via 2. When the contact starts, bundles are delivered one-by-one to 2, which relays them to 4 (“Delivered” series in Figure 7-7

shows the arrival time at 4). The last bundle is delivered at the end of the contact (contacts are shown at the bottom of the chart), in accordance with the estimated contact volume of 16.5 bundles. This behavior clarifies the miscalculation of data delivery times in the ECGR algorithm, without the use of ETO. ECGR estimates that the transmission for all bundles will start at the beginning of the contact, although it has already forwarded other bundles through the same route.

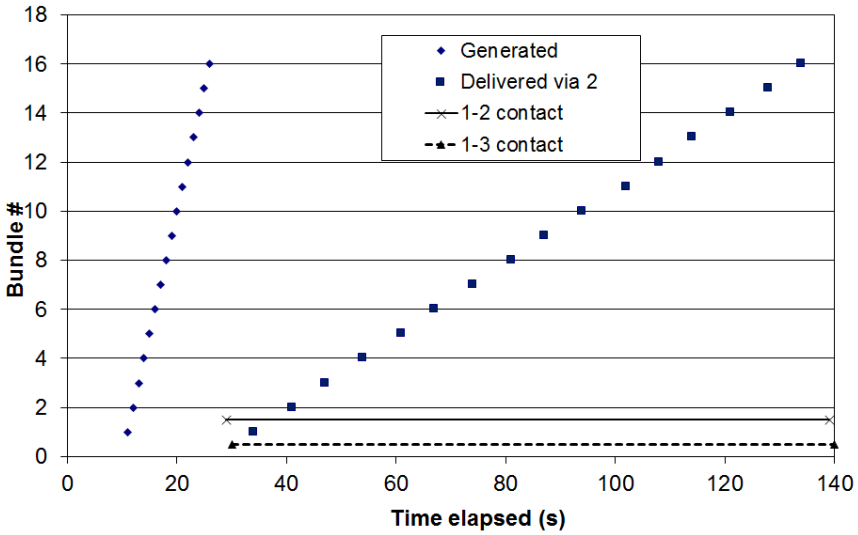

Figure 7-7 ECGR at the downlink, with parallel, continuous routes

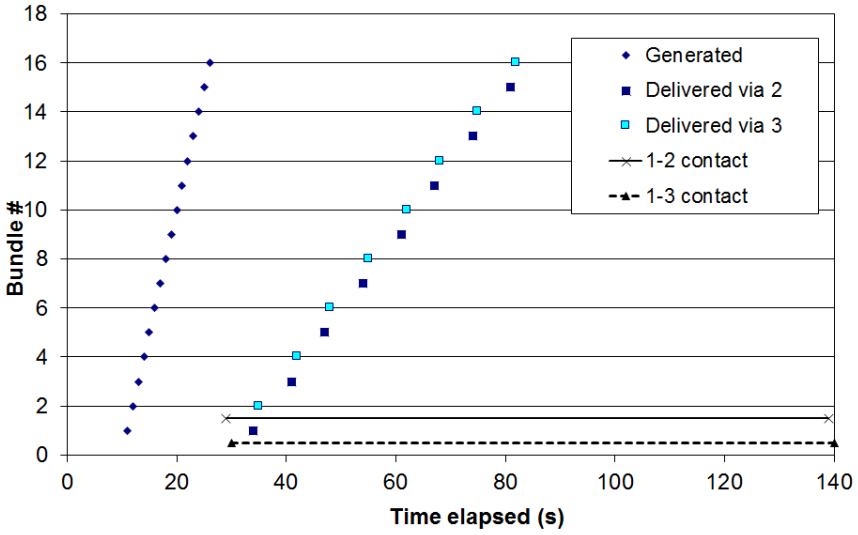

Figure 7-8 CGR-ETO-first-hop, with parallel, continuous routes

The same experiment is also conducted with the use of CGR-ETO, using a low *contact plan update threshold*, in such a way the routes are always re-calculated after each bundle is forwarded. The first bundle is forwarded to 2, as before; then, thanks to ETO's consideration

of queueing delay, the two contacts are used alternately. The results observed in Figure 7-8 highlight two advantages: first, there is a 50% reduction in total data delivery delay; second, CGR-ETO exhibits perfect load balancing, which is also an important element, since it leaves some capacity on both contacts, for subsequent traffic. CGR-ETO-all-hops is not displayed in this case, as its routing decisions are exactly the same as with the CGR-ETO-first-hop.

7.3.2 Downlink data transmissions with intermittent links

We continue the evaluation process of CGR-ETO with a more realistic use case, where connectivity for links 1-2 and 1-3 is intermittent, according to the contact plan detailed in Table 6-3. In particular, LEO satellite (node 1) communicates with LEO control center (node 2) through a short-duration (25 s) link with high transmission rate (512 kbit/s). It also uses an alternative downlink channel, through the GEO relay satellite and the GEO control center (node 3), with a pair of longer contacts (70 s each) with lower transmission rate (128 kbit/s). The terrestrial links between control stations and the MOC (node 4) are modelled as continuous, high transmission speed links for both alternative routes (with a rate of 10 Mbit/s each). Therefore, the transmission bottleneck is at the first hop, for both alternative routes, whereas the impact of the second hop to destination (from 2 or 3 to 4) on the delivery time is negligible and it is thus irrelevant in the choice of the best path.

We consider the transmission of a series of 20 bundles at the downlink from node 1 to node 4. CGR routes each bundle to either node 2 or 3, based on the shortest expected delivery time, and places it at the corresponding outbound queue with the selected neighbour. Initially we examine the routing performance of the ECGR algorithm. In Figure 7-9 we depict the bundle creation times and the arrival times at node 4, based on the selected route, as well as the intervals of the first-hop contacts (1-2 and 1-3), at the lower part of the diagram, for better understanding of the routing decisions. As observed in the figure, by neglecting the queueing delay caused by the previously forwarded bundles, all bundles are preferentially forwarded via 3 (“Delivered via 3” series in the diagram), as ECGR estimates shorter delivery times through this route. However, once the capacity of the first 1-3 contact is fully exploited, ECGR discards this route and forwards the remaining bundles via 2 (“Delivered via 2” series), which is the best of the residual choices, as the second contact to 3 begins much later. Although ECGR is able to make use of both parallel contacts (1-2 and the first 1-3) and to deliver all bundles during these contacts, we observe three sub-optimal effects: first, the delivery is significantly disordered: bundles 1 and 5 are delivered first, then 6-8 in parallel with 11-20, and 9-10 are delivered last. Although this is compliant with BP functionality [6], it is an undesirable behavior. Second,

since the 1-2 contact has not been fully exploited, bundle 10 could have been delivered earlier, if it was routed via 2, resulting in a lower total delivery time. Third, the first contact to 3 is no longer available for subsequent traffic; if a new bundle is generated at 90 s, node 1 will not be able to immediately forward it through the first contact to 3.

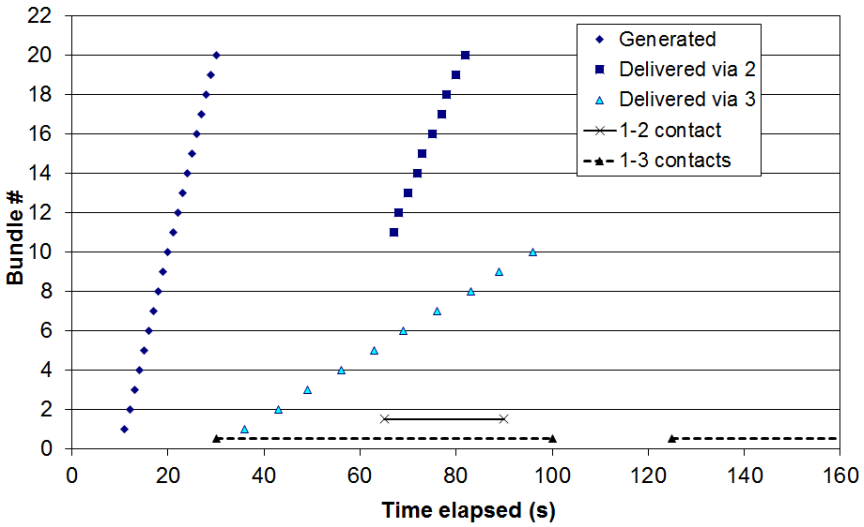

Figure 7-9 ECGR at the downlink, with intermittent connectivity

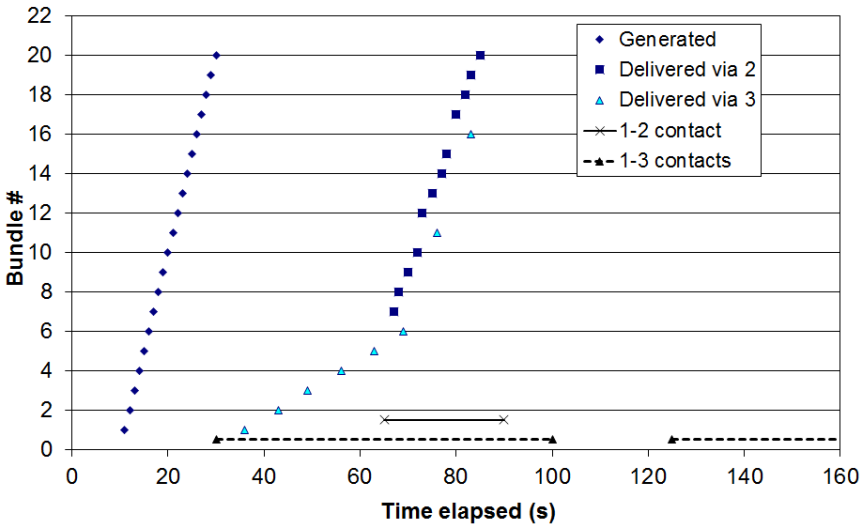

Figure 7-10 CGR-ETO-first-hop at the downlink, with intermittent connectivity

Next, we repeat the same experiment with the use of CGR-ETO-first-hop. We note that the use of CGR-ETO-all-hops, as well as the CGR-ETO version incorporated in ION standard CGR have the same exact functionality in this case, since the transmission bottleneck is the first hop and all versions of the algorithm feature the same functionality concerning local queueing

delays. Therefore, CGR-ETO-all-hops algorithm is not illustrated in a separate figure. Figure 7-10 shows the performance of CGR-ETO in terms of routing decisions. As observed in the figure, CGR-ETO increases the accuracy of the delivery time estimation, as the queueing delay on the first hop (the only relevant queueing delay here) is taken into account. The first bundles are sent via 3, as before, but as soon as the contact 1-2 opens, the next bundles start to be forwarded on both paths in parallel. The improved algorithm functionality provides three advantages, resolving the three aforementioned sub-optimal points of ECGR: first, the total delivery time is shorter by ~15 seconds; second, the large-scale disordered delivery has disappeared; third, when the two alternative routes are both open, bundles tends to use the fastest link in direct proportion to the transmission speeds (1-2 contact is 4 times faster than 1-3), achieving an almost perfect traffic balancing; this is evident in Figure 7-10, where we observe that four bundles are forwarded via 2, for each bundle forwarded via 3, when both contacts are open. As a result of that load balancing functionality, the first 1-3 contact is no more fully exploited and there is still some residual capacity available to other traffic. In this context, CGR-ETO alleviates the congestion, in terms of contact capacity occupation, caused by ECGR miscalculations, and preventing it, provides a basic form of proactive congestion control.

7.3.3 Uplink data transmissions with intermittent links

Here, we consider the opposite case of data transmissions at the uplink channel from Earth to space. Here, we consider the same transmission scenario as the previous subsection, i.e., with intermittent links, but instead, data are transmitted in the opposite direction, from node 4 to 1. In addition to being an interesting practical example for satellite and space communications, the uplink case enables the evaluation and comparison of the routing algorithms, when the transmission bottleneck appears at the second hop of the routing path. Here, similarly to the downlink case, the terrestrial hops have no effect in the choice of the best path, as their contribution to the delivery time is negligible. The difference with respect to the previous case is that now the terrestrial links are on the first hop. This means that challenges posed by intermittent space links, such as limited contact volumes and significant queueing delay can no more be tackled by the original CGR's residual capacity check, or by the CGR-ETO-first-hop, because the former recalculates routes only upon a contact exhaustion, based on the residual capacity check, and the latter takes into account queueing delays only at the first DTN hop to destination. So, for the initial route selection at node 4, local information about local queues (4 to 3 and 4 to 2) will not be enough for accurate delivery time estimations, since queueing delays

that affect delivery times are not observed at the local outbound queues of node 4, but at the second-hop outbound queues.

In this regard, we note that similar behavior is not exclusive at the uplink transmission direction, considered here, but can be observed in different multi-hop scenarios (both downlink and uplink), where the bottleneck is not on the first hop towards destination. This limitation could be justified by the fact that, in data transmissions from space, the most critical hop (i.e., with minimal contact capacity) is often the first. However, there are also some multi-hop data transmission scenarios where this is not the case. Furthermore, although the uplink in deep space communications is considered less demanding, since data are limited to small-sized commands, this does not hold true in DTN LEO satellite communications where the satellite is used as a DTN data relay, because in this case both link directions (to and from the satellite) have the same relevance. Therefore, the applicability and performance of routing algorithms is of interest for the reverse direction as well, and deserve to be fully investigated.

To this end, we generate on node 4 a series of 20 bundles, which are routed by CGR via 2 or 3 as soon as they are generated. We start by examining the case of CGR-ETO-first-hop. The achieved routing results are illustrated in Figure 7-11. We note that the obtained routing results in this experiment are similar to those obtained with ECGR, since the delivery time estimations calculated by both algorithms are the same. Due to the insignificant queueing delay on the first hop, and its virtually unlimited capacity, all bundles are forwarded via 3, since the first 1-3 contact opens first, and, thus, it seems to provide the shortest delivery time, with no calculation of next-hop queueing delays. However, once arriving at node 3, only the first 10 bundles can actually use the first 1-3 contact, whereas bundles 11-20 are delivered during the next 1-3 contact. We observe that, for the first time here, the volume of both contacts to node 3 are completely allocated, and therefore, there is no remaining capacity left for additional traffic. Note that, while CGR-ETO-first-hop on node 4 assumes that all bundles can use the first 1-3 contact, when the bundles arrive at node 3, they are routed again, and they are correctly scheduled, partially on the first contact to 1 (first 10 bundles) and partially on the next one (last 10). This is because the local decisions at node 3 are now made with perfect knowledge on the outbound queues, and CGR-ETO-first-hop has accurate queue length information. The same applies for ECGR, which performs residual capacity checks at the route selection, and, when the first contact is fully subscribed, routes bundles 11-20 via the next contact to 3. The overall results obtained in this experiment are worse than in the symmetrical case shown in Figure 7-9, as the contact 1-2 is not exploited, resulting in a significantly higher total delivery time.

The same experiment is conducted with the use of the CGR-ETO-all-hops algorithm version, which takes into account decisions on locally routed bundles to update ETO on all contacts through the path to destination. The obtained results, illustrated in Figure 7-12, show an improved utilization of contacts. By taking into account the queueing delay of the traffic

generated locally, not only for the first hop, as in CGR-ETO-first-hop, but also in subsequent hops, bundles are now optimally routed either via 3 or 2, with a perfect load balancing when both links are active. This behavior results in a significant reduction of the total delivery time, from 190 s to 85 s, as all bundles are delivered before the end time of the contact 1-2. Furthermore, the algorithm enhances the link utilization, since it exploits also the 1-2 higher-speed contact, while at the same time leaving the capacity of the second 1-3 contact completely available to subsequent traffic.

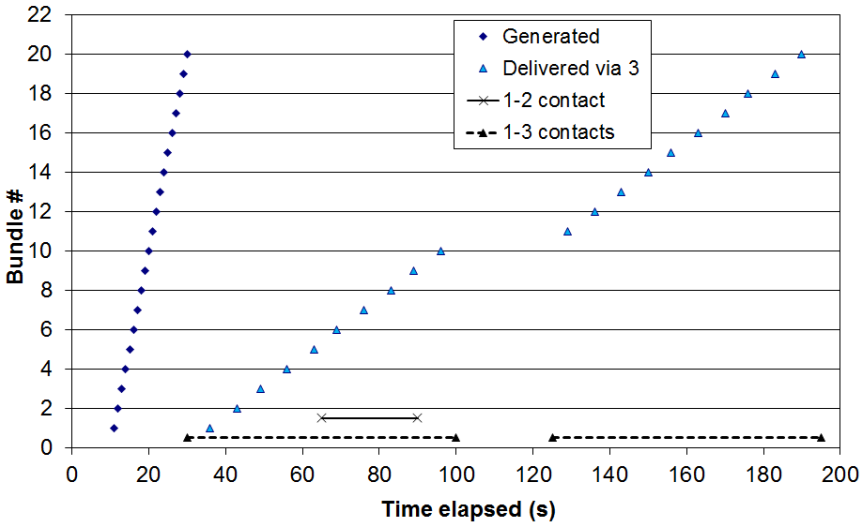

Figure 7-11 CGR-ETO-first-hop / ECGR at the uplink

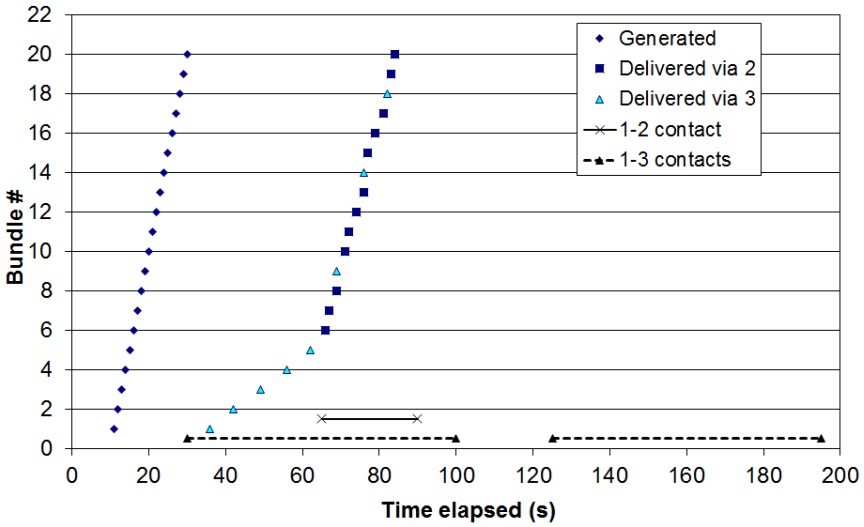

Figure 7-12 CGR-ETO-all-hops at the uplink, with *contact plan update threshold* = 1%

Thus, the improved calculation of delivery times, achieved by the enhanced algorithm version, results in a significantly improved routing performance, and provides perfect load

balancing capabilities, in-order delivery at the receiver, as well as basic congestion control enforcement, by alleviating the contact capacity congestion that was present with the other two CGR versions, CGR-ETO-first-hop and ECGR.

The aforementioned gains, however, come at the cost of additional Dijkstra computations, raising a tradeoff between improved routing decisions and increased processing overhead. This tradeoff involves different parameters, such as the amount of traffic and the processing power of the node where the routing computations are performed, and can be regulated using the *contact plan update threshold*, described in Section 4.1.1. On fixed terrestrial nodes, as is the case with nodes 2 and 3, in our scenario, processing power is less challenging than on space assets. Hence, the *contact plan update threshold* could be configured in a low value: in our experiment, we have used a value of 1%, which corresponds to new route recalculation for every routed bundle. This is the reason for the perfect load balancing functionality observed in Figure 7-12. In any case, it is within the ability of the mission planners or operators to decide on the optimal threshold use, depending on the mission objectives and criticality of the routing performance.

7.4 Scenario 4

In this scenario, we focus on the evaluation of the proactive framework that predicts queueing delays based on past values of data rates and the time series forecasting method presented in Section 4.2. We perform a set of simulations to assess the accuracy of the introduced mechanism in terms of delivery delay prediction, for different sets of input parameters. We note here that we paired the proposed method with the CGR-ETO-first-hop version of CGR. The same results would be obtained with the use of the ECGR version, since the congested hop in this scenario is the second hop, i.e., the contacts between A and D, and both algorithms consider no queueing delay on the second hop of the path to destination.

An initial observation that appears from the simulation results is that the occurrence of the contacts during the time period (set randomly as described in Section 6.2.4) has significant impact on the total bundle delivery delay. The reason for this is that the most significant portion of the total bundle delivery delay is the waiting time, in the order of tens of thousands of seconds, since contacts *A-D* occur twice per day. Thus, the queueing delays, although seemingly minor, may impose a lot of additional waiting delays. In other words, when a bundle arrives at intermediate node *A*, and finds a lot of backlog bundles ahead, it may be queued for a period of time longer than the *A-D* contact duration, and thus will have to wait for the next transmission opportunity, which is half a day later, in our simulations setup. We observed that,

depending on the contact occurrences, the simulation results were divided into two groups. In the first and most common one, all bundles were transmitted during the contacts initially predicted by CGR; or, in other words, there were no queueing delays large enough to cause any bundles to miss the transmission opportunity and wait for a total transmission cycle (half day). In these simulations, to which we will refer from now on as *Case 1* simulations, the *BDDPredErr* (i.e., the error in bundle delivery delay prediction) does not exceed the duration of a contact, and comprises a small percentage of the total delivery delay. In the second observed group of simulations (referred to from now on as *Case 2* simulations), on the other hand, queueing delays caused loss of transmission opportunities for a portion of the transmitted bundles, resulting in a significant *BDDPredErr*.

The percentage of the *Case 2* simulations depends heavily on the number of network nodes and the randomly generated contact distribution in the contact plan. Table 7-3 shows how this percentage varies for different number of nodes, and, also, the corresponding average percentage of bundles (in *Case 2* simulations) that miss the contact opportunities due to heavy cross traffic and, thus, long queueing delays. In *Case 1* simulations, as mentioned above, no bundles have missed any transmission opportunity.

Table 7-3 *Case 2* simulations as a percentage of total simulations, and corresponding average percentage of bundles that missed contact opportunities

N	***Case 2* simulations (%)**	**Bundles that missed transmission opportunity (%)**
2	3.33	24.75
5	4	7.6
10	12	6.66
20	23.67	2.42

For example, when ten nodes generate and transmit bundles, an average of 11.33% of the conducted simulations are *Case 2* simulations, and an average of 6.66% of the bundles in each of these simulations are actually transmitted during a different contact than the one predicted by CGR. Even though this percentage of bundles seems minor, the *BDDPredErr* calculated by CGR for those bundles, approaches the time period, i.e., half day. This may have significant impact on the performance of the application or service layers residing on top of BP, such as unnecessary retransmissions due to timeout expirations, and delayed in-order delivery, when a transport-layer protocol such as DTPC [8] is used. In Figure 7-13, we present the average *BDDPredErr* for different values of capacity ratio λ, with $N = 10$ producing nodes and for *Case 1* simulations. We observe that the forecasting method results in a great reduction in the delivery delay prediction error, significantly lower than the reactive method with the use of CPUP update

messages. In *Case 2* simulations, as depicted in Figure 7-14, the bundles that have lost a transmission window are reflected in the significantly higher prediction error, when no forecasting is used. In our comparative simulations, we have observed that both Contact Plan Update framework and the proactive forecasting method are able to predict this deviation for all bundles (i.e., 100% of the bundles for all set of parameters), resulting in a major *BDDPredErr* decrease, and resolving the aforementioned misbehavior.

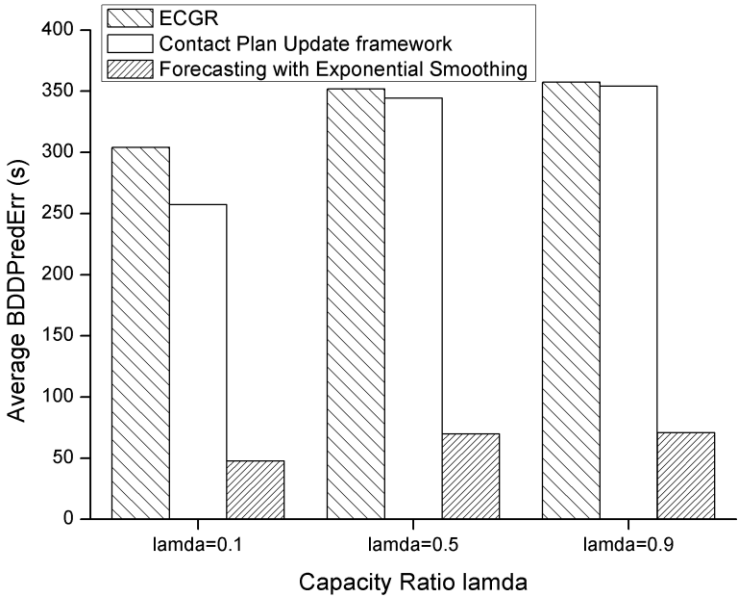

Figure 7-13 Average *BDDPredErr* versus the capacity ratio *λ*, with *N* = 10. *Case 1* simulations.

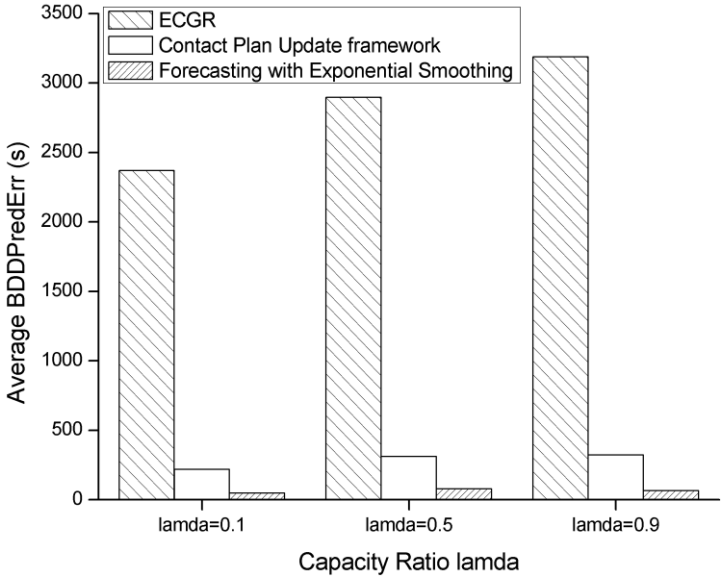

Figure 7-14 Average *BDDPredErr* versus the capacity ratio *λ*, with *N* = 10. *Case 2* simulations.

Due to the large fluctuation in the bundle delivery delay prediction, primarily for the bundle percentages that are depicted in Table 7-3, average values is not the most indicative statistical function. In order to capture the whole range of prediction errors we use the *NormalizedBDDPredErr* percentiles: all bundle delivery delay prediction errors are sorted in an ascending order and the *k*-th percentile corresponds to the *NormalizedBDDPredErr* that is greater than the *k* % of all bundle delivery delay prediction errors. In Figure 7-15 – Figure 7-18, we depict the *NormalizedBDDPredErr* percentiles for sample simulations of different parameter sets, for $N = 20$ & *Case 1* simulations, $N = 20$ & *Case 2* simulations, $N = 2$ & *Case 1* simulations, and $N = 2$ & *Case 2* simulations, respectively. In the first two figures (Figure 7-15 and Figure 7-16) we compare the proposed forecasting method with ECGR, and with a forecasting method that assumes nominal transmission rates, rather than predicting future rate values. As observed in Figure 7-15 and Figure 7-16, all algorithms achieve small prediction errors for the majority of bundles; there is, however, a ~2%-3% of the bundles that all algorithms err. The ECGR prediction error reaches 40% of the bundle delivery delay, for the *Case 1* simulation depicted in Figure 7-15, and 90% of the bundle delivery delay, for the *Case 2* simulation depicted in Figure 7-16. For the exponential smoothing method, the respective errors are less than 20%, whereas the forecasting with nominal rates provides an overall good prediction, leaving though a tail of overestimation for ~4% of the bundles at the lower percentage end.

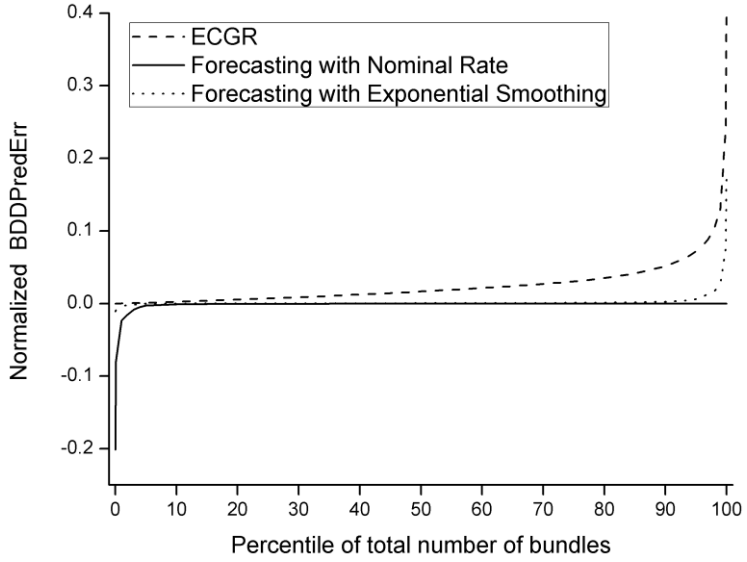

Figure 7-15 *Normalized BDDPredErr* versus the percentiles of total number of bundles for sample simulations with $N = 20$ and $\lambda = 0.9$. *Case 1* simulations.

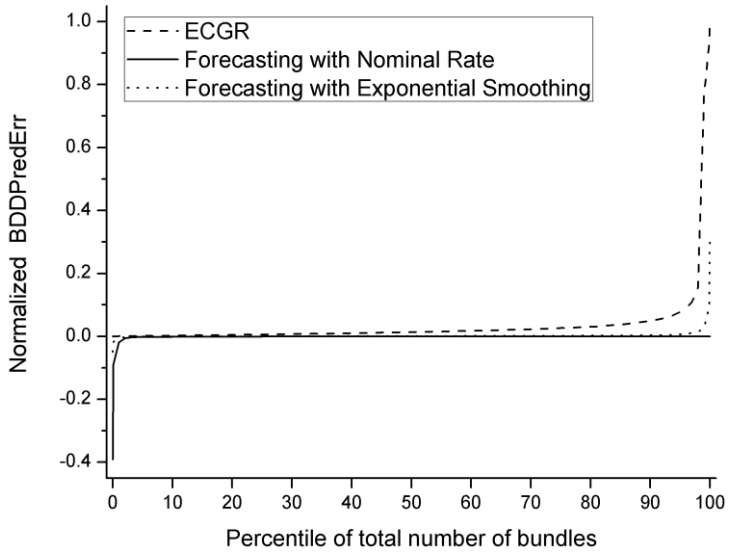

Figure 7-16 *Normalized BDDPredErr* versus the percentiles of total number of bundles for sample simulations with N = 20 and λ = 0.9. *Case 2* simulations.

In Figure 7-17 and Figure 7-18 we compare our exponential forecasting method with ECGR, and Contact Plan Update framework, for N = 2 producing nodes. As observed in the diagrams, *NormalizedBDDPredErr* is significantly improved for a larger percentage of all bundles, using both the reactive CPUP estimation and the proactive forecasting method. Figure 7-17 shows that, in a *Case 1* simulation, the prediction accuracy can be improved with the exponential smoothing forecasting method, for all bundles. However, since the queueing component is a tiny portion of the total end-to-end delivery delay, *NormalizedBDDPredErr* does not exceed the amount of 0.4%.

So far, we have used a uniform data production rate, equal to the maximum rate that the network can serve. The prediction method with the use of nominal transmission rates provides good accuracy, as depicted in Figure 7-15 and Figure 7-16. However, in cases where network nodes produce less data than the network can serve, its performance degrades. In Figure 7-19, we measure the Average *NormalizedBDDPredErr* for different production levels, presented as a percentage of the maximum amount of data that can be served. Although forecasting with nominal rates outperforms the other algorithms for large data rate productions, since it approaches the actual data rates, the results obtained for 10% of the maximum production rate become even worse than with ECGR. In our forecasting method, despite the fact that network nodes have no prior knowledge of the production rates of other nodes, they achieve a good estimation for all production rates, due to the past queueing values obtained through update messages, and the forecasting procedure. Note that in Figure 7-19, the average *BDDPredErr* represents the mean of absolute values, whereas in the percentiles figures we also provided the negative, overestimated values.

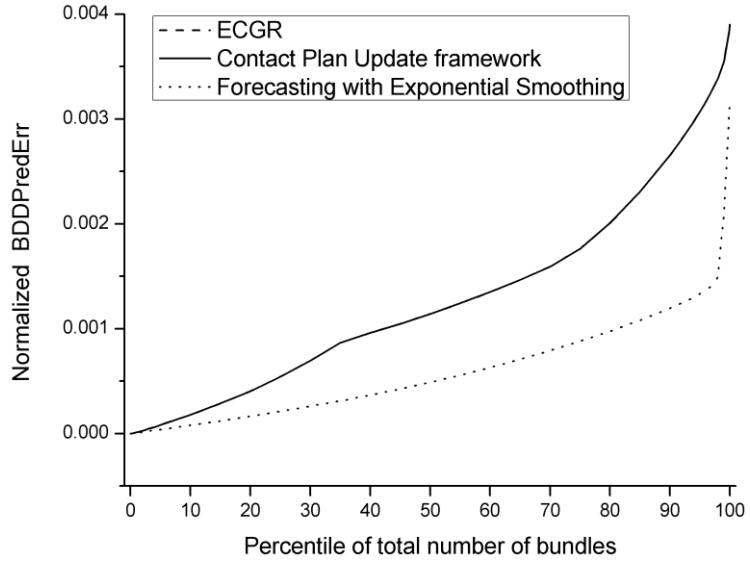

Figure 7-17 *NormalizedBDDPredErr* versus the percentiles of total number of bundles for sample simulations with N = 2 and λ = 0.9. *Case 1* simulations.

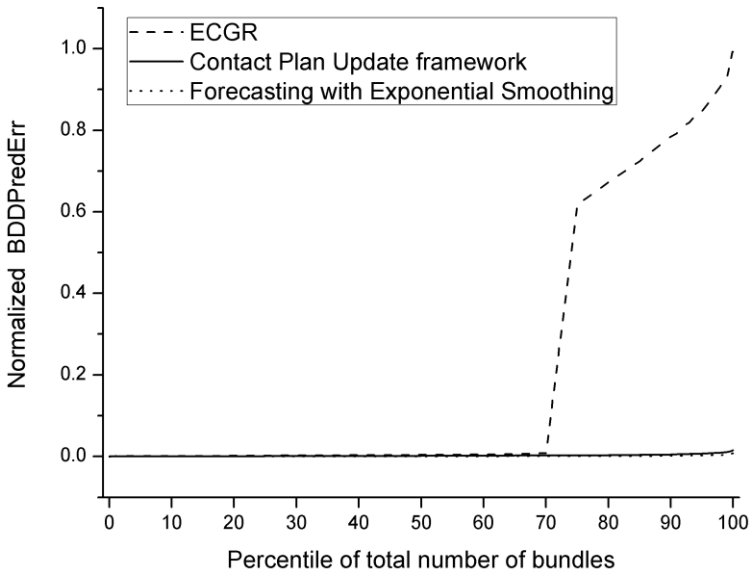

Figure 7-18 *NormalizedBDDPredErr* versus the percentiles of total number of bundles for sample simulations with N = 2 and λ = 0.9. *Case 2* simulations.

In Figure 7-20, we illustrate the overall overhead caused by the update messages in relevance to the transmitted amounts of data payloads. The amount of overhead bytes span from 11.7 Kbytes for simulations with data transmissions of 137 Mbytes (N = 2), to 818 Kbytes for simulations with data transmissions of 1.37 Gbytes (N = 20). We note here that the inflicted overhead depends heavily on the granularity of the rates extracting and the accordingly generated messages. In our scenario, as mentioned in Section 4.2.2, rates are extracted in a per-contact granularity, that is, whenever a contact ends. The use of a finer granularity (i.e., in

smaller intervals) would provide more accurate statistics for the distribution of rates through time, even during a contact, but would produce more transmission overhead. Using more coarse-grained measurements would on the other hand reduce the overhead at the cost of reducing the time series samples, with possible degradation of the forecasts precision. The study of this tradeoff is in itself an interesting research subject for future works.

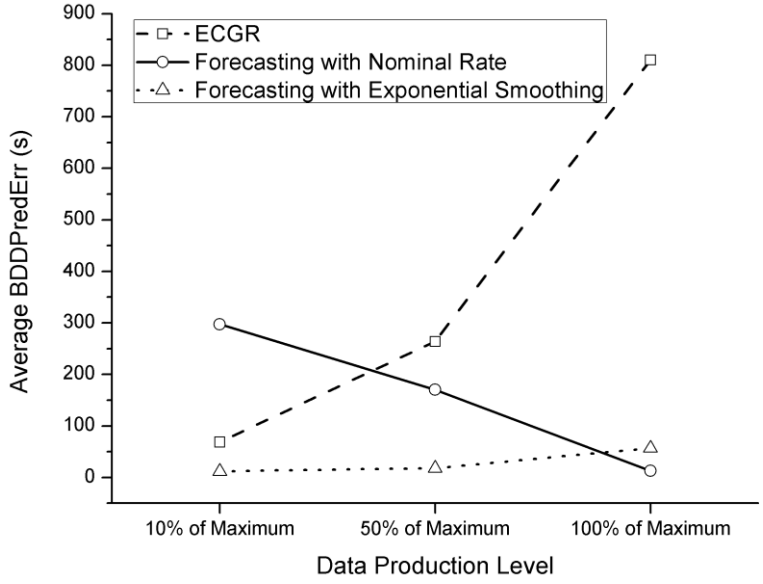

Figure 7-19 Average *BDDPredErr* versus the data production level

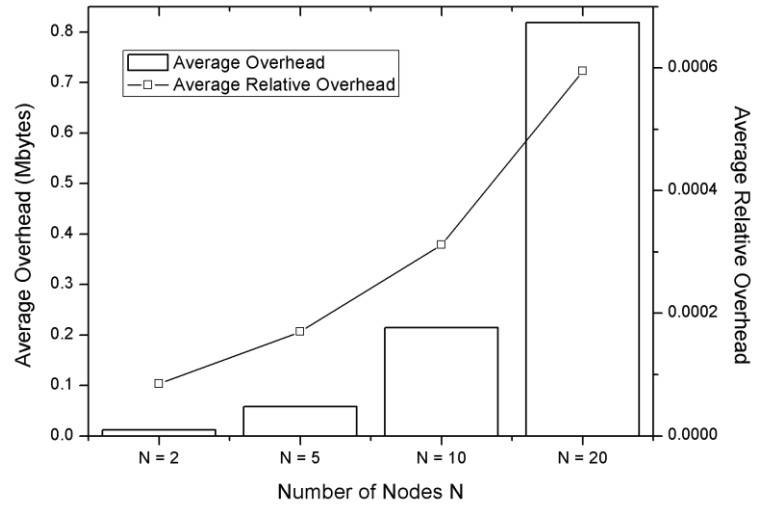

Figure 7-20 Total Overhead versus the number of nodes *N*

Finally, we study the impact of the exponential smoothing parameter in the *Average BDDPredErr* by using different values: a = 0.1, 0.5, 0.9, and 1. Figure 7-21 illustrates that the

predictions are more accurate for values of *a* near 1, (i.e., more sensitive to changes), which shows larger dependency on the recent values than on the history observations. This behavior is justified by the use of a uniform production rate in our simulations: the resulting transmission rates increase gradually from zero to the steady-state rate, stay there till the end of bundle productions and decrease gradually to zero again. Different production rates than the uniform we used in this work might require less sensitivity to fluctuations and increased weight on the history values.

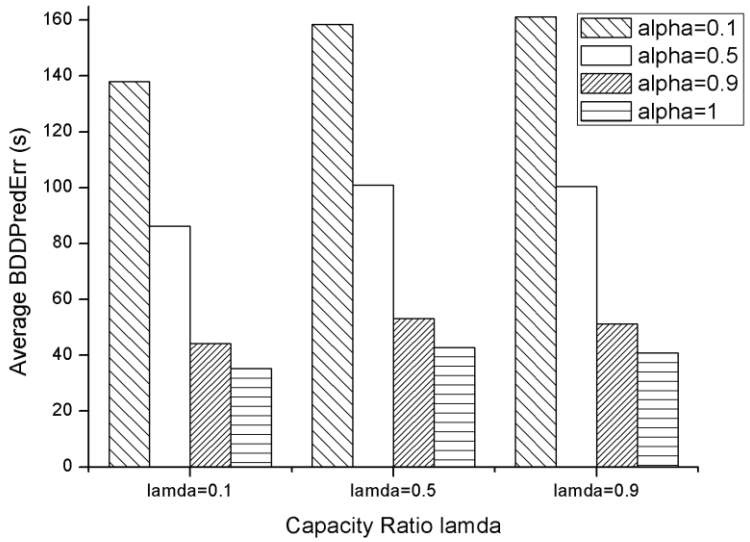

Figure 7-21 Average *BDDPredErr* for different values of the smoothing parameter *a*, with *N* = 5.

7.5 Scenario 5

In the last scenario, we evaluate the efficacy of the RTT estimation and RTO configuration methods, and the benefits that the end-to-end retransmission framework, introduced in Chapter 5, brings to the functionality of DTPC protocol. To this end, we emulate a complex, deep-space scenario specified in Section 6.2.5, and compare the updated DTPC, which incorporates the proposed retransmission framework, namely DTPC-dRTO, with the original DTPC retransmission mechanism, namely DTPC-sRTO, which configures retransmission timers in a static way.

We begin the evaluation process with an inspection of the accuracy of the configured RTO in relation to the DTPC ACK arrival time. Figure 7-22 shows the *RTO Configuration Error*, i.e., the difference between the configured RTO and the actual arrival time of the corresponding ACK item, as an increasing function, versus the data items percentile. For the proposed mechanism, the data item RTO is equal to the RTO of the corresponding block; however, the

RTO Configuration Error is displayed per data item in order to be comparable with DTPC-sRTO, where RTOs were set per data item. Figure 7-22 shows that the original mechanism results in a significant *RTO Configuration Error* for a great amount of data items, varying from 4 – 24 h. This observation was the factor that initially motivated the introduction of a modified RTO configuration scheme, based on the updated delivery estimation tools introduced in the previous Chapters of this thesis.

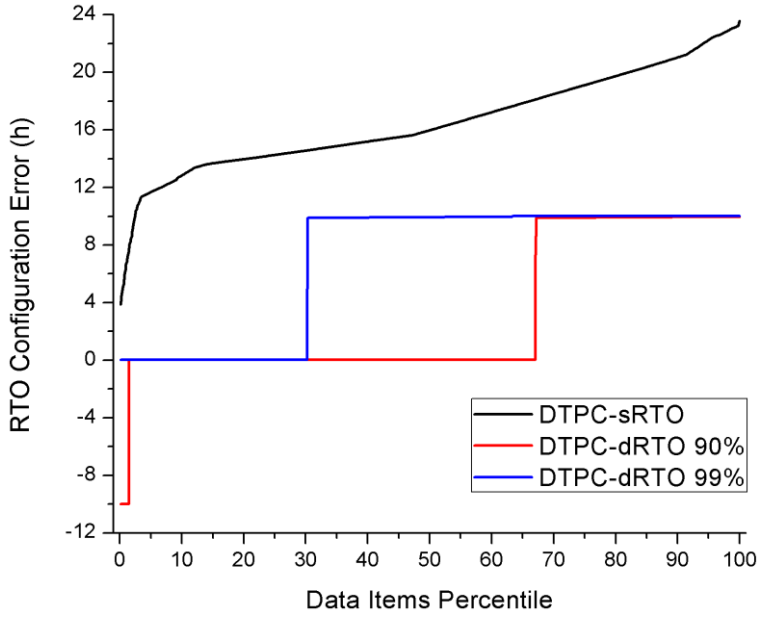

Figure 7-22 *RTO Configuration Error* vs data items percentile

The accuracy of the timer in the original DTPC depends on the maximum number of retransmissions and the data item lifetime, parameters configured in a static way. A proper configuration of the RTO (i.e., neither very large nor very small) requires careful observation of the network topology and the specifics of the data transmissions a priori, and thus is a manual, challenging task. The proposed, automatic RTO setup, on the other hand, responds to the network conditions and the contact plan very accurately, with a smaller *RTO Configuration Error* (max absolute value ~10h) for the majority of data items, as illustrated in the corresponding lines of Figure 7-22. The comparison between the experimental results with the use of two different *delayTolerance* values raises an interesting observation. The less delay-tolerant, more optimistic value of 90% provides better accuracy with ~zero *RTO Configuration Error* for a greater percentage of the data items (~65% of total data items in comparison to ~30% with 99% *delayTolerance*). This is achieved at the cost of some underestimation error, which results in spurious retransmissions for 172 data items (i.e., 1.43% of the total data items).

A detailed analysis of how our mechanism calculates the delay of the last data item of the block, at the end of the 2-5 contact can be observed in Figure 7-23. The total delay for the

delivery to node 5 comprises the max estimated queueing delay (calculated using *contact.maxBacklog*), and the transmission and propagation delays for multiple ARQ transmission rounds. The estimation difference between the two *delayTolerance* values is based on the ARQ delivery probability at the next hop: for a 9Kbyte bundle transmission, and the given packet size used, with *maxAcceptablePER* = 10%, a 90% *delayTolerance* is accomplished with two transmission rounds (original transmission and one retransmission round), whereas the achievement of 99% *delayTolerance* requires three transmission rounds (original transmission and two retransmission rounds). The difference of one transmission round corresponds to one deep-space link RTT (i.e., 28 minutes plus the corresponding retransmission delay), and thus results in a deviation of more than 28 minutes in the estimation of arrival time at next hop. Therefore, as illustrated in Figure 7-23, the "pessimistic" 99% *delayTolerance* value succeeds in capturing the extra delay caused by the contact loss, whereas the 90% *delayTolerance* value fails. This observation provides further insight on the distribution of the *RTO Configuration Error* depicted in Figure 7-22. DTPC-dRTO 90% underestimates the delivery time (and RTT, respectively) for the data items that were not successfully transmitted within two transmission rounds, providing a negative *RTO Configuration Error* of ~10h. On the other hand, for the data items that were transmitted at exactly two transmission rounds (percentile ~31-67%), DTPC-dRTO 90% succeeds in calculating the exact RTT, whereas DTPC-dRTO 99% provides an overestimation of 10 h, with the consideration of a third transmission round. We state here that the contact plan design and the choice of *delayTolerance* values were deliberately configured to provide an example that clarifies the significance of the *delayTolerance*, and shows in a more illustrative way the behavior of the proposed RTO estimation scheme.

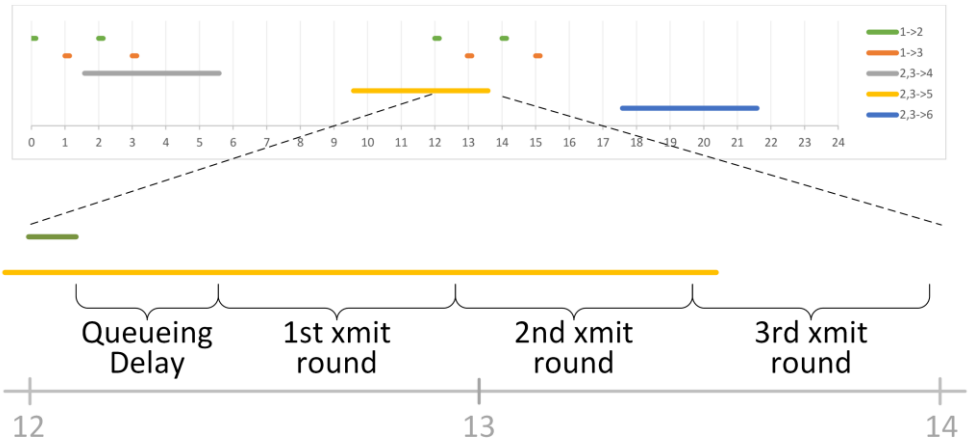

Figure 7-23 Contact Plan zoom at 12-14h of the experiment

A more detailed insight of the data transmissions can be obtained from the micro-analysis of the data item destination arrival times in Figure 7-24 and Figure 7-25. The 90% *delayTolerance* value leads to 172 spurious timeouts and, therefore, to 172 duplicate receptions at destination, for the data items that missed the 2-5 contact and were delivered at ~18h. On the other hand, the more delay-tolerant, less optimistic 99% *delayTolerance* value manages to capture the same set of data items that arrive at 18h, and minimizes the duplicate data items to a number equal to the lost ACKs.

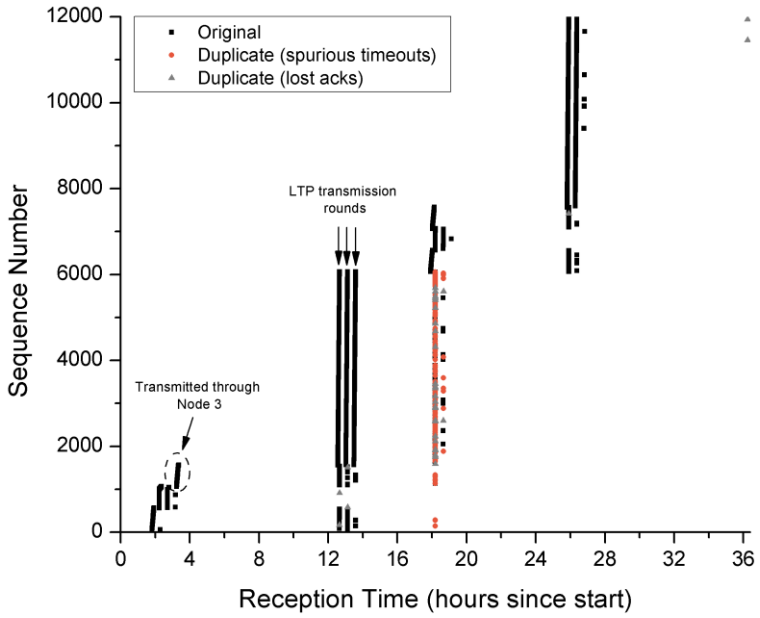

Figure 7-24 Data Item reception times at destination node (MOC), DTPC-dRTO with *delayTolerance* = 90%

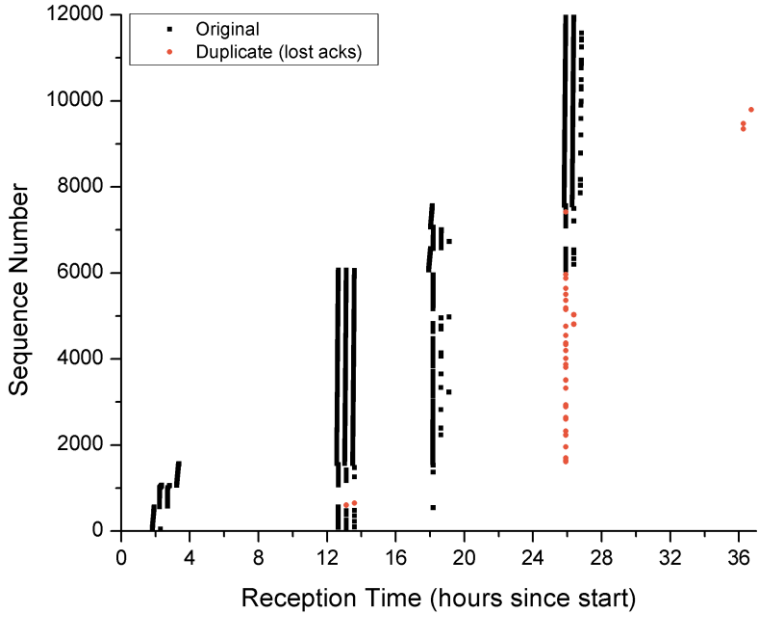

Figure 7-25 Data Item reception times at destination node (MOC), DTPC-dRTO with *delayTolerance* = 99%

The importance of timely retransmission of lost data items becomes more obvious when it comes to the data reception at the destination application. Figure 7-26 displays the payload delivery at the application of the destination node versus the experiment time. The total data delivery for this transmission scenario is reduced from 65.7 h, with DTPC-sRTO, to 26.8 h, with DTPC-dRTO. The relative improvement of 59.2% illustrates an important reason to adopt a faster, dynamic retransmission scheme rather than the originally proposed, static retransmission mechanism of DTPC-sRTO, and highlight the importance of accurate RTT estimations.

A lower *delayTolerance* value results in a faster retransmission of the data items that are actually lost due to e.g., uncorrected channel errors. This can be observed in Figure 7-26, where a slightly faster payload delivery occurs for a small percentage of the data items with 90% (red line) in contrast to 99% (blue line), at the cost of 172 redundant retransmissions for the aforementioned spurious timeouts. Hence, faster recovery with a smaller value of *delayTolerance* leads to redundant transmissions, increasing the overall transmission overhead by a percentage of 172/12000 = 1.43%. On the other hand, when *delayTolerance* is equal to 99%, the significant improvement in comparison to DTPC-sRTO comes at no redundancy cost, since RTO exceeds the actual ACK arrival time for all data items.

An implicit but also important improvement introduced by the proposed framework is the reduction of the data storage occupancy at the end nodes, during the duration of the experiment. The source node stores data items until they are acknowledged by the sender; when a data item or the corresponding acknowledgement is lost, the storage will be freed only after the first successful ACK reception of the retransmission. Thus, faster retransmission of the lost data items leads to a more timely release of the occupied storage. The improvement, however, is insignificant, as the lost and retransmitted data items are a small percentage of the total number. On the other hand, storage occupancy improvement becomes clear at the receiver node, when the end-to-end protocol supports in-order delivery, as is the case with DTPC. In Figure 7-27 we illustrate the receiver *storageOccupancy* (i.e., the data items that are stored at the protocol buffers) through time, for the duration of each experiment. The improvement is great; with both values of *delayTolerance*, maximum *storageOccupancy* is kept significantly lower, from 11405 data items in DTPC-sRTO to 6419 in DTPC-dRTO 99% and 5884 in DTPC-dRTO 90%. The explanation for this behavior is that, when in-order delivery is enabled, the out-of-order elements are stored and not freed until the reception order is restored. Hence, faster retransmission and recovery of lost items restores the packet order and releases the corresponding stored elements quicker.

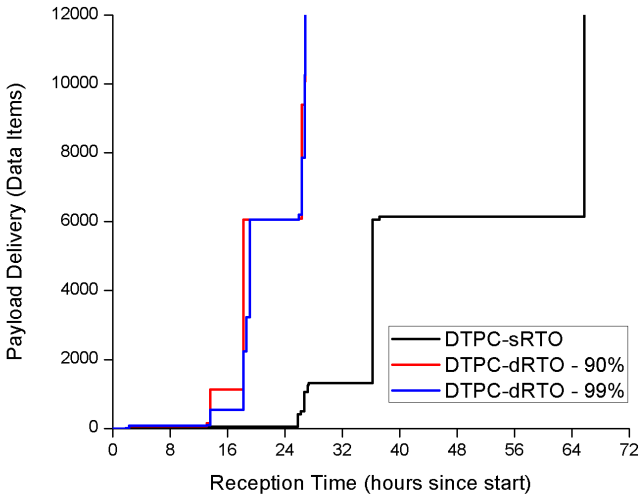

Figure 7-26 Payload Delivery at destination node vs time

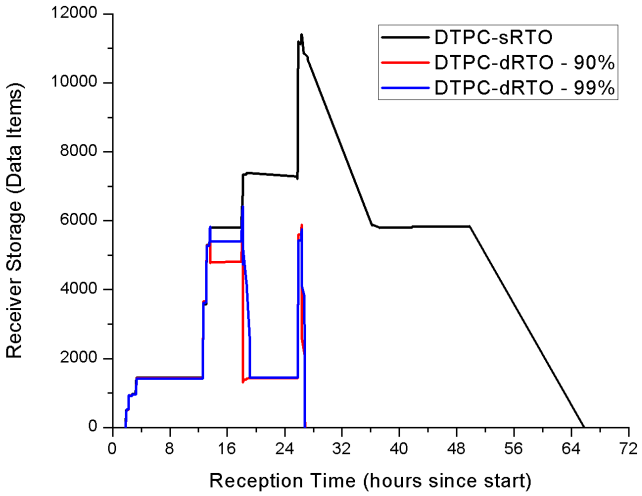

Figure 7-27 Receiver Storage Occupancy vs time

In Table 7-4, we show the *totalStorageOccupancy* and the *storageUtilization* both at the sender and receiver nodes. The improvement in occupancy and utilization is small at the sender node; the *storageUtilization* is reduced by less than 0.4% of the *maxTotalStorageOccupancy*. At the receiver side, however, the improvement is great; the *storageUtilization* is reduced from 44.6% (which corresponds to 16056 dataItems*days), with DTPC-sRTO, to 6.5% (2340 dataItems*days), and 7.2% (2592 dataItems*days) of the *maxTotalStorageOccupancy*, for DTPC-dRTO 90% and DTPC-dRTO 99%, respectively. The significant difference highlights an important feature of the timely retransmissions of lost data items. Storage is released faster,

and this can lead to possibly better usage of the available resources, which can be particularly beneficial to the storage-demanding nature of space and other DTN networks.

Table 7-4 *TotalStorageOccupancy* and *StorageUtilization* at Sender and Receiver Nodes

RTO Configuration	*totalStorage Occupancy* at Sender (dataItems*days)	*storageUtilization* at Sender	*totalStorage Occupancy* at Receiver (dataItems*days)	*Storage Utilization* at Receiver
DTPC-sRTO	4158	11.55%	16056	44.6%
DTPC-dRTO with 90% *delayTolerance*	4032	11.2%	2340	6.5%
DTPC-dRTO, with 99% *delayTolerance*	4046	11.24%	2592	7.2%

Chapter 8 Conclusions

8.1 General Conclusions

In the present thesis we have studied the problem of estimating end-to-end delivery delays in the context of space internetworks, and identified the need for more accurate estimations. To this end, we have exploited the DTN architecture that is the main candidate to realize the Interplanetary Internet concept. We have focused on the different components of the end-to-end delay that pertain to data transmissions in space networks, and in DTN architecture in specific, and developed a set of algorithms, protocols, and tools to tackle the challenging issue of estimating these components; In particular:

- We have provided an analytical method to calculate a detailed probabilistic profile of the plausible delivery times at destination, for a future bundle transmission, and developed Bundle Delivery Time Estimation, an administrative tool that exploits this analytical process and implements the designed functionality, in an algorithmic way.
- We have studied the issue of estimating queueing delays in the IPN context and, to this end, we have developed two different estimation methods. In the first method, namely Contact Plan Update framework, updates on significant queueing delay changes are reactively disseminated through the network, using CPUP protocol. The second method is based on regular exchange of information on network rates, and a time series forecasting procedure, to predict future queueing and forwarding rates, and estimate future queueing delays, accordingly.
- Finally, we have applied the developed analytical and algorithmic methods to improve the transport layer's capability of estimating RTT and configuring RTO, accordingly, based on cross-layer information, and implemented a dynamic retransmission framework for DTPC transport protocol that exploits the accurate RTT estimation functionality.

For evaluation purposes, we have developed *SpaceDTNSim* simulator for space networks with intermittent and scheduled connectivity, and deployed *SPICE DTN Testbed* that accurately emulates network conditions for space networks. We have performed simulation and emulation studies, accordingly, to assess the efficiency of the proposed methods in estimating and improving the end-to-end delivery delays. The evaluation results have provided us with several insights:

✓ The developed research tools and methods can significantly increase the delay prediction accuracy for data transmissions in space networks, in a variety of challenging network conditions.

- ✓ The network is injected with an inherent capability to estimate delivery delays in a more dynamic and flexible fashion that is aware of dynamic network features, such as the link error rates or the cross-traffic queue backlogs.
- ✓ The enhanced accuracy in delivery delay prediction, besides constituting by itself an important network element, has also the potential to improve different network functions that depend on delay estimations:
    - ➢ It grants advanced output information to administrative services, and allows for the extraction of useful metrics, for future, crucial data transmissions.
    - ➢ It provides the routing function with advanced delivery time estimators, and, since typically routing objectives are based on the minimization of earliest delivery time, enhances the routing algorithm performance.
    - ➢ It improves the estimation of actual transmission intervals and, therefore, enhances network awareness on the distribution of transmission load. Thus, it administers load balancing capabilities, in the presence of multiple paths to data destination.
    - ➢ It has the potential to alleviate and control contact or link congestion, and prevent contact capacity exhaustion, based on the improved routing and load balancing capabilities.
    - ➢ It enhances the round-trip-time estimators, and supports an advanced end-to-end retransmission scheme with better configuration of RTOs, thus boosting the functionality of end-to-end protocols that reside on top of the DTN architecture.

In the next subsections, we detail the conclusions drawn by the specific elements that were developed and presented in the present thesis.

8.2 Specific Conclusions

8.2.1 Bundle Delivery Time Estimation tool

In the first part of this thesis, we have provided an analytical method to calculate the different components of the end-to-end delivery delay required for a high-priority bundle transmitted in a space network, to reach its destination. Our technique is based on an instrumentation DB that stores management statistics for each network node. Past BER values are calculated through some metrics extracted from the DB, creating different time series for error rates per link, and future BER values are predicted via a Holt-Winters time series forecasting method. The forecast

error rates are then used to estimate the total number of transmission rounds that the bundle transmission will last, and this procedure continues successively for each hop that the bundle is expected to follow through the path to destination. The result of our algorithm is a list of plausible bundle delivery times at destination with the corresponding probabilities.

We developed the introduced analytical and algorithmic method in the BDTE application, which we integrated into the ION DTN implementation. BDTE exploits the CGR algorithm and the instrumentation DB that are included in ION, in order to provide the desired delivery time estimation functionality.

Validation experiments showed that the introduced method can be effectively used for administrative purposes, providing different outputs and useful metrics. It can administer analytical, probabilistic profiles for delivery time expectations of future, critical bundle transmissions. Based on the profile obtained for a given bundle, BDTE can determine the earliest plausible delivery time that it may reach its destination, and compute the probability that it is delivered prior to some given time in the future. Finally, it can output the time that ensures the delivery of a bundle within some desired confidence level.

All in all, BDTE can provide a useful administrative tool to predict the performance of different space applications and adjust their functionality and usage. When DTN is deployed in space missions and access to data transmissions information is granted, the employed forecasting procedure can be further optimized, based on the observed network behaviors.

8.2.2 Queueing Delay Estimation Methods

In the research conducted for this thesis, we identified that the queueing delay component constitutes by itself an interesting research issue and a challenging factor to accurately compute. Therefore, we have presented two distinct approaches for estimating queue lengths and queueing delays in data transmissions that pertain to space communications: the reactive estimation through the Contact Plan Update framework, and the proactive prediction through network statistics and time series forecasting.

8.2.2.1 Contact Plan Update Framework

In the former approach, we have proposed the incorporation of queueing delay information into the contact plan, encoded in the Earliest Transmission Opportunity parameter. We have accordingly proposed an update in the CGR algorithm, namely CGR-ETO, to exploit the integrated queueing delay information in routing decisions. Furthermore, we have designed the

CPUP protocol to disseminate, through the network, changes in the contact plan, including information updates on significant queueing delay changes.

Our evaluation shows that the introduced framework has the potential to increase awareness on queueing delays and, consequently, improve routing decisions. Simulation results showed significant improvements on the delay prediction accuracy, irrespectively of the particular conditions of each scenario. The Contact Plan Update framework also administers significant delivery delay reduction, primarily in heavy-traffic scenarios.

CPUP can also provide a robust solution for notifying other network nodes about contact plan additions or modifications. Therefore, it can prove a valuable solution to the space DTN architecture, in terms of network management, for the dissemination of the predicted connectivity plans and dynamic network updates.

The efficiency of the CGR-ETO algorithm was also evaluated in realistic emulation scenarios in SPICE DTN Testbed. For this purpose, we incorporated CGR-ETO into ION DTN implementation as an alternative to the ECGR version, and conducted a variety of emulation experiments. Results showed that CGR-ETO can estimate the optimal path more accurately, based on its better insight in the earliest delivery time. Therefore, it offers a more efficient exploitation of future contacts, provides better load balancing when choosing between parallel alternative routes, and results in a shorter data delivery time, whenever possible. Furthermore, by improving not only delivery time, but also link utilization in the presence of multiple routes, CGR-ETO provides an additional benefit: it avoids contact exhaustion, and offers an initial form of proactive congestion control, which is a feature of particular interest in space communications, where contacts are intermittent and transmission rates often limited. In this context, and given the dominant role of DTN in the design of future space operations, Contact Plan Update framework constitutes an important step towards a robust, unified architecture, which will provide efficient routing and accurate mission planning.

An additional, significant outcome of this research work is that the obtained improvements of the CGR-ETO algorithm led to its adoption within the standard CGR algorithm of the ION DTN implementation, since version 3.2.1.

8.2.2.2 Proactive Prediction

In the latter approach, we introduced a novel method to predict queueing rates and queueing delays in contact-plan-based DTNs with application in space communications. Queue length and rate statistics are extracted in a per-contact granularity and disseminated to the network nodes via CPUP. These historical data are then used to predict future queueing rates via time

series forecasting and, ultimately, improve the estimation of bundle queueing delays en route to destination. Through extensive simulations we showed that the proposed prediction method provides significant accuracy in queueing delay estimation, and, consequently, improves the total delivery delay calculations. It outperforms both the calculation of end-to-end delays provided in the original CGR algorithm, as well as the estimations through the Contact Plan Update framework, without at the same time inflicting any significant transmission overhead.

The proposed method can assist the configuration of higher layer protocols and services, providing a more accurate end-to-end delivery delay estimate. It can also be used as an administrative tool to analyze queue length distributions and queueing delays in DTNs with deterministic contact schedules. As soon as real measurements from space DTNs are available, our mechanism can be further optimized with the analysis of different time series forecasting methods, such as triple exponential smoothing or ARMA/ARIMA, and the assessment of the tradeoff between the practicality of the prediction accuracy and the computational overhead that time series calculation will impose on the energy-sensitive space assets.

8.2.3 End-to-End Retransmission Framework

Finally, at the last part of this thesis, we introduced a novel, end-to-end retransmission framework for the newly emerged transport layer that resides over DTNs with intermittent and scheduled connectivity. We established the requirements and provided the design of the proposed framework in a cross-layer way, over a set of operation concepts, described in detail, which comply with the DTN architecture and the specificity of the connectivity intermittency. With the most commonly cited protocol stack for IPN as a reference, we developed algorithms that utilize these concepts, and incorporated them in ION DTN implementation. We evaluated the proposed retransmission mechanism in our space-oriented DTN Testbed, where we emulated a complex deep-space data transmission scenario with varying, challenging network conditions.

Evaluation results illustrate that the proposed framework provides a better, more accurate RTT estimator than the originally proposed, static retransmission scheme of DTPC protocol. Consequently, RTO configuration is improved, erroneous or lost data are retransmitted faster, and, hence, we observe a great reduction in the overall data transmission time, while keeping at the same time the overhead, due to duplicate transmissions, minimum. Finally, an implicit, albeit significant, benefit from fast retransmissions is the great reduction of the storage occupancy and utilization, primarily at destination node, when the in-order delivery feature of DTPC protocol applies. By and large, the proposed retransmission framework constitutes an

important enhancement for the DTN transport layer and specifically DTPC protocol, while its modular character makes it flexible for future modifications and enhancements in the transport protocol, the BP layer, as well as to cover different CL protocols.

References

[1] S. Burleigh et al., "The interplanetary internet: A communications infrastructure for Mars exploration," *Acta Astronautica*, vol. 53, no. 4-10, pp. 365-373, 2003.

[2] InterPlanetary Networking Special Interest Group (IPNSIG). [Online]. http://ipnsig.org/

[3] Interagency Operations Advisory Group (IOAG) Space Internetworking Strategy Group (SISG). [Online]. http://cwe.ccsds.org/ioag

[4] Interagency Operations Advisory Group, "Space Internetworking Strategy Group (SISG) Operations Concept for a Solar System Internetwork (SSI)," 2011.

[5] V. Cerf and et al, "Delay-tolerant networking architecture," RFC Informational RFC 4838, Apr 2007.

[6] K. Scott and S. Burleigh, "Bundle protocol specification," RFC Experimental RFC 5050, Nov. 2007.

[7] M. Ramadas, S. Burleigh, and S. Farrell, "Licklider transmission protocol specification," Experimental RFC 5326, Sep. 2008.

[8] Giorgos Papastergiou, Ioannis Alexiadis, Scott Burleigh, and Vassilis Tsaoussidis, "Delay Tolerant Payload Conditioning Protocol," *Computer Networks*, vol. 59, pp. 244-263, 2014.

[9] S. Burleigh, "Contact Graph Routing," Internet-Draft draft-burleigh-dtnrg-cgr-00, July 2010. [Online]. http://tools.ietf.org/html/draft-burleigh-dtnrg-cgr

[10] Space Internetworking Center (SPICE) FP7 project. [Online]. http://www.spice-center.org

[11] ION-DTN. [Online]. http://sourceforge.net/projects/ion-dtn/

[12] S. Burleigh, "Interplanetary Overlay Network: An Implementation of the DTN Bundle Protocol," in *4th IEEE Consumer Communications and Networking Conference (CCNC 2007)*, 2007, pp. 222-226.

[13] The Consultative Committee for Space Data Systems. [Online]. http://public.ccsds.org

[14] Delay-Tolerant Networking Working Group (DTNWG). [Online]. https://datatracker.ietf.org/wg/dtnwg

[15] Delay-Tolerant Networking Research Group (DTNRG). [Online]. https://irtf.org/dtnrg

[16] IRTF. Global Access to the Internet for All (GAIA). [Online]. https://trac.tools.ietf.org/group/irtf/trac/wiki/gaia

[17] Jet Propulsion Laboratory, California Institute of Technology. Exploration & Observational Systems. [Online]. http://scienceandtechnology.jpl.nasa.gov/research/ResearchTopics/topicdetails/?ID=67

[18] Consultative Committee for Space Data Systems, "Space Packet Protocol," Recommendation for Space Data Systems Standards, Blue Book CCSDS 133.0-B-1, 2003.

[19] Consultative Committee for Space Data Systems, "SPACE COMMUNICATIONS PROTOCOL SPECIFICATION (SCPS)— TRANSPORT PROTOCOL (SCPS-TP)," Recommendation for Space Data System Standards, Blue Book CCSDS 714.0-B-2, 2006.

[20] Consultative Committee for Space Data Systems, "CCSDS File Delivery Protocol (CFDP)," Blue Book CCSDS 727.0-B-4, 2007.

[21] Consultative Committee for Space Data Systems, "Overview of Space Communications Protocols," Green Book CCSDS 130.0-G-3, 2014.

[22] V. Cerf et al., "Delay-Tolerant Network Architecture: The Evolving Interplanetary Internet," Internet-Draft draft-irtf-ipnrg-arch-01.txt, August 2002. [Online]. https://tools.ietf.org/html/draft-irtf-ipnrg-arch-01

[23] Ian Akyildiz and et al, "InterPlaNetary internet: state-of-the-art and research challenges," *Computer Networks*, vol. 43, no. 2, pp. 75-112, October 2003.

[24] Space-Data Routers. [Online]. http://www.spacedatarouters.eu/

[25] D. Vardalis et al., "Decentralized Space-Data Dissemination for Low-Cost, Dense Satellite Networks," *IEEE Transactions on Aerospace and Electronic Systems, accepted for publication*, 2015. [Online]. http://utopia.duth.gr/~slenas/documents/DecentralizedDissemination.pdf

[26] S. Burleigh, V. Cerf, J. Crowcroft, and V. Tsaoussidis, "Space for Internet and Internet for space," *Elsevier Ad Hoc Networks*, vol. 23, pp. 80-86, Dec. 2014.

[27] R. C. Durst, P. D. Feighery, and K. L. Scott, "Why not use the Standard Internet Suite for the Interplanetary Internet?,". [Online]. http://www.ipnsig.org/reports/TCP_IP.pdf

[28] K. Fall, "A delay-tolerant network architecture for challenged internets," in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications (SIGCOMM '03)*, 2003, pp. 27-34.

[29] S. Burleigh et al., "Delay-tolerant networking: an approach to interplanetary Internet," *IEEE Communications Magazine*, vol. 41, no. 6, pp. 128-136, June 2003.

[30] M. Ho and K. Fall, "Poster: Delay Tolerant Networking for Sensor Networks," in *First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*, Santa Clara, CA, 2004.

[31] S. Farrell and V. Cahill, *Delay-and disruption-tolerant networking*.: Artech House, Inc., 2006.

[32] Consultative Committee for Space Data Systems, "Rationale, Scenarios and Requirements for DTN in Space," Green Book CCSDS 734.0-G-1, 2010.

[33] Interagency Operations Advisory Group, Space Internetworking Strategy Group. (2008, Nov.) Recommendations on a Strategy for Space Internetworking. [Online]. http://cwe.ccsds.org/ioag/Final%20Products/SISG%20Report%20v1.4%20FINAL.pdf

[34] DTNRG. DTN Bone. [Online]. https://sites.google.com/site/dtnresgroup/home/dtn-bone

[35] E. Birrane, K. Collins, and K. Scott, "The Delay-Tolerant Networking Experimental Network Constructing a Cross-agency Supported Internetworking Testbed," in *SpaceOps 2012*, 2012.

[36] N4C: Networking for Communications Challenged Communities: Architecture, Test Beds and Innovative Alliances. [Online]. http://www.n4c.eu/

[37] Haggle Project. [Online]. http://haggleproject.org/

[38] ResiliNets: Resilient and Survivable Networks Wiki. [Online]. https://wiki.ittc.ku.edu/resilinets/Main_Page

[39] S Guo et al., "Very low-cost internet access using KioskNet," *ACM SIGCOMM Computer Communication Review*, vol. 37, no. 5, Oct. 2007.

[40] CROWD: Consumer Generated Mobile Wireless Media. [Online]. http://anr-crowd.lip6.fr/

[41] SARAH: Delay-Tolerant Distributed Services for Mobile Ad Hoc Networks. [Online]. http://www-valoria.univ-ubs.fr/SARAH

[42] SCAMPI: Service Platform for Socially Aware Mobile and Pervasive Computing. [Online]. http://www.ict-scampi.eu/

[43] DTN-Bytewalla 5 Project. [Online]. https://archive.ssvl.kth.se/csd2011/csd.xen.ssvl.kth.se/csdlive/content/dtn-bytewalla.html

[44] UMOBILE: Universal, mobile-centric and opportunistic communications architecture. [Online]. http://www.umobile-project.eu/

[45] RIFE project. [Online]. http://rife-project.eu/

[46] J. Wyatt, S. Burleigh, R. Jones, L. Torgerson, and S. Wissler, "Disruption Tolerant Networking Flight Validation Experiment on NASA's EPOXI Mission," in *Advances in Satellite and Space Communications (SPACOMM) 2009*, 2009, pp. 187-196.

[47] W.D. Ivancic et al., "Large File Transfers from Space Using Multiple Ground Terminals and Delay-Tolerant Networking," in *2010 IEEE Global Telecommunications Conference (GLOBECOM 2010)*, 2010, pp. 1-6.

[48] B. Willman and S. Davidson. (2014, February) International Space Station (ISS) and Delay/Disruption Tolerant Networking. [Online]. http://ipnsig.org/wp-content/uploads/2014/02/ISS-DTN-Presentation-IPNSIG.pdf

[49] A. Jenkins, S. Kuzminsky, K.K. Gifford, R.L. Pitts, and K. Nichols, "Delay/disruption-tolerant networking: flight test results from the international space station," in *2010 IEEE Aerospace Conference*, 2010, pp. 1-8.

[50] Multi-Purpose End-To-End Robotic Operation Network (METERON) Project. [Online]. http://esa-telerobotics.net/meteron

[51] European Space Agency. (2014, August) METERON: RIDING HIGH ON SUCCESS. [Online]. http://blogs.esa.int/rocketscience/2014/08/08/meteron-riding-high-on-success/

[52] Consultative Committee for Space Data Systems, "Cislunar Space Internetworking—Architecture," DRAFT GREEN BOOK CCSDS 730.1-G-0, 2006.

[53] (2012, August) Mars Express blog. [Online]. http://blogs.esa.int/mex/2012/08/05/time-delay-between-mars-and-earth/

[54] Jet Propulsion Laboratory. Voyager: The Interstellar Mission. [Online]. http://voyager.jpl.nasa.gov/mission/

[55] F. Warthman. (2012, July) Delay- and Disruption-Tolerant Networks (DTNs) A Tutorial, Version 2.0. [Online]. ipnsig.org/wp-content/uploads/2012/07/DTN_Tutorial_v2.04.pdf

[56] K. Fall and S. Farrell, "DTN: An Architectural Retrospective," *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 5, pp. 828-836, June 2008.

[57] K. Fall, W. Hong, and S. Madden. (2003, July) Custody Transfer for Reliable Delivery in Delay Tolerant Networks. [Online]. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.59.1856&rep=rep1&type=pdf

[58] T. Berners-Lee, R. Fielding, and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax," RFC 3986, 2005.

[59] M. Demmer, J. Ott, and S. Perreault, "Delay-Tolerant Networking TCP Convergence-Layer Protocol," Experimental RFC 7242, 2014.

[60] Kruse, H., S. Jero, and S. Ostermann, "Datagram Convergence Layers for the Delay- and Disruption-Tolerant Networking (DTN) Bundle Protocol and Licklider Transmission Protocol (LTP)," Experimental RFC 7122, 2014.

[61] L. Sunde, "The NetInf Bluetooth Convergence Layer," Internet Draft draft-sunde-netinf-protocol-bluetooth-00, 2013.

[62] D. Kutscher, K. Loos, and J. Greifenberg, "Uni-DTN: A DTN Convergence Layer Protocol for Unidirectional Transport," Internet Draft draft-kutscher-dtnrg-uni-clayer-00.txt, 2007.

[63] D. Merani, A. Berni, J. Potter, and R. Martins, "An Underwater Convergence Layer for Disruption Tolerant Networking," in *2011 Baltic Congress on Future Internet Communications (BCFIC Riga)*, Riga, 2011, pp. 103-108.

[64] S. Burleigh, "Delay-Tolerant Networking LTP Convergence Layer (LTPCL) Adapter," Internet Draft draft-burleigh-dtnrg-ltpcl-05, 2013.

[65] L. Wood, W.M. Eddy, W. Ivancic, J. McKim, and C. Jackson, "Saratoga: a Delay-Tolerant Networking convergence layer with efficient link utilization," in *International Workshop on Satellite and Space Communications. IWSSC '07.*, Salzburg, 2007, pp. 168-172.

[66] M. Alfonzo, J.A. Fraire, E. Kocian, and N. Alvarez, "Development of a DTN bundle protocol convergence layer for SpaceWire," in *2014 IEEE Biennial Congress of Argentina (ARGENCON)*, Bariloche, 2014, pp. 770-775.

[67] S. Burleigh, M. Ramadas, and S. Farrell, "Licklider Transmission Protocol - Motivation," Informational RFC 5325, 2008.

[68] M. Ramadas, S. Burleigh, and S. Farrell, "Licklider Transmission Protocol - Specification," IETF RFC 2008.

[69] S. Farrell, M. Ramadas, and S. Burleigh, "Licklider Transmission Protocol - Security Extensions," Experimental RFC 5327, 2008.

[70] Consultative Committee for Space Data Systems (CCSDS), "Licklider Transmission Protocol (LTP) for CCSDS," Red Book CCSDS 734.1-R-3, May 2014.

[71] Consultative Committee for Space Data Systems, "TM Space Data Link Protocol," Blue Book CCSDS 132.0-B-1, 2003.

[72] Consultative Committee for Space Data Systems, "TC Space Data Link Protocol," Blue Book CCSDS 232.0-B-2, 2010.

[73] Consultative Committee for Space Data Systems, "AOS Space Data Link Protocol," Blue Book CCSDS 732.0-B-2, 2006.

[74] Consultative Committee for Space Data Systems, "Proximity-1 Space Link Protocol - Rationale, Architecture, and Scenarios," Green Book CCSDS 210.0-G-2, 2013.

[75] S. Burleigh. (2007, July) Licklider Transmission Protocol (LTP): An Overview. [Online]. https://www.ietf.org/proceedings/69/slides/tsvarea-0/sld1.htm

[76] E. Koutsogiannis, F. Tsapeli, and V. Tsaoussidis, "Bundle Layer End-to-end Retransmission Mechanism," in *2011 Baltic Congress on Future Internet Communications (BCFIC)*, Riga, 2011, pp. 109-115.

[77] Consultative Committee for Space Data Systems, "CCSDS File Delivery Protocol (CFDP)—Part 1: Introduction and Overview," Green Book CCSDS 720.1-G-3, April 2007.

[78] Consultative Committee for Space Data Systems, "CCSDS File Delivery Protocol (CFDP)—Part 2: Implementers Guide," Green Book CCSDS 720.2-G-3, April 2007.

[79] F. Flentge, "Study on CFDP and DTN Architectures for ESA Space Missions," in *The Third International Conference on Advances in Satellite and Space Communications (SPACOMM 2011)*, Budapest, 2011.

[80] G. Papastergiou, N. Bezirgiannidis, and V. Tsaoussidis, "On the Performance of Erasure Coding over Space DTNs," in *10th International Conference On Wired/Wireless Internet Communication (WWIC 2012)*, Santorini, 2012.

[81] Jet Propulsion Laboratory. Deep Space Network. [Online]. http://deepspace.jpl.nasa.gov/

[82] J.-C. Bolot, "End-to-end Packet Delay and Loss Behavior in the Internet," in *Conference Proceedings on Communications Architectures, Protocols and Applications (SIGCOMM '93)*, San Francisco, CA, 1993, pp. 289-298.

[83] I.F. Akyildiz and Seong-Ho Jeong, "Satellite ATM networks: a survey," *IEEE Communications Magazine*, vol. 35, no. 7, pp. 30-43, 1997.

[84] K. McCarthy, F. Stocklin, B. Geldzahler, D. Friedman, and P. Celeste, "NASA's Evolution to Ka-Band Space Communications for Near-Earth Spacecraft," in *Proceedings of the SpaceOps 2010 Conference.*, 2010.

[85] C. J. Bovy, H. T. Mertodimedjo, G. Hooghiemstra, H. Uijterwaal, and P. Van Mieghem, "Analysis of end-to-end delay measurements in Internet," in *Proceedings of ACM Conference on Passive and Active Leasurements (PAM)*, Fort Collins, 2002.

[86] D.C. Lee and W. Baek, "Expected file-delivery time of deferred NAK ARQ in CCSDS file-delivery protocol," *IEEE Transactions on Communications*, vol. 52, no. 8, pp. 1408-1416, 2004.

[87] W. Baek and D. C. Lee, "Analysis of CGSDS file delivery protocol: Immediate NAK mode," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 41, no. 2, pp. 503-524, 2005.

[88] I. U. Sung and J. L. Gao, "CFDP performance over weather-dependent Ka-band channel," in *Proceedings of the SpaceOps 2006 Conference*, Rome, 2006.

[89] J. L. Gao and J. S. SeGui, "Performance evaluation of the CCSDS file delivery protocol-latency and storage requirement," in *2005 IEEE Aerospace Conference*, Big Sky, MT, 2005, pp. 1300-1312.

[90] T. De Cola and M. Marchese, "Performance analysis of data transfer protocols over space communications," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 41, no. 4, pp. 1200-1223, 2005.

[91] R. Wang, B. I. Gutha, and P. V. Rapet, "Window-based and rate-based transmission control mechanisms over space-internet links," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 44, no. 1, pp. 157-170, 2008.

[92] R. Wang and et. al., "Unreliable CCSDS file delivery protocol (CFDP) over cislunar communication links," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 46, no. 1, pp. 147-169, 2010.

[93] R. Wang, X. Wu, T. Wang, X. Liu, and L. Zhou, "TCP Convergence Layer-Based Operation of DTN for Long-Delay Cislunar Communications," *IEEE Systems Journal*, vol. 4, no. 3, pp. 385-395, Sept 2010.

[94] R. Wang et al., "Experimental evaluation of TCP-based DTN for cislunar communications in presence of long link disruption," *EURASIP Journal on Wireless Communications and Networking - Special issue on opportunistic and delay tolerant networks*, pp. 8:1-8:11, Jan 2011.

[95] Ruhai Wang, S.C. Burleigh, P. Parikh, Che-Jen Lin, and Bo Sun, "Licklider Transmission Protocol (LTP)-Based DTN for Cislunar Communications," *Networking, IEEE/ACM Transactions on*, vol. 19, no. 2, pp. 359-368, 2011.

[96] Z. Yang et al., "Analytical characterization of licklider transmission protocol (LTP) in cislunar communications," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 50, no. 3, pp. 2019 - 2031, July 2014.

[97] W. Ruhai, X. Wu, T. Wang, and T. Taleb, "Experimental Evaluation of Delay Tolerant Networking (DTN) Protocols for Long-Delay Cislunar Communications," in *IEEE Global Telecommunications Conference, 2009 (GLOBECOM 2009)*, Honolulu, 2009, pp. 1-5.

[98] R. Wang et al., "Which DTN CLP is best for long-delay cislunar communications with channel-rate asymmetry?," *IEEE Wireless Communications*, vol. 18, no. 6, pp. 10-16, Dec. 2011.

[99] N. Bezirgiannidis and V. Tsaoussidis, "Packet size and DTN transport service: Evaluation on a DTN testbed," in *Proceedings of the International Congress on Ultra*

Modern Telecommunications and Control Systems and Workshops (ICUMT 2010), Moscow, Russia, 2010, pp. 1198-1205.

[100] R. Wang, Z. Wei, Q. Zhang, and J. Hou, "LTP Aggregation of DTN Bundles in Space Communications," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 49, no. 3, pp. 1677-1691, July 2013.

[101] H. Takagi and L. Kleinrock, "Mean packet queueing delay in a buffered two-user csma/cd system," *IEEE Transactions on Communications*, vol. 33, no. 10, pp. 1136-1139, 1985.

[102] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," *SIGCOMM Computer Communication Review*, vol. 19, no. 4, pp. 1-12, 1989.

[103] M. Karam and F. Tobagi, "Analysis of the delay and jitter of voice traffic over the internet," in *20th IEEE International Conference on Computer Communications (INFOCOM 2001)*, vol. 2, Anchorage, Alaska, USA, 2001, pp. 824–833.

[104] M. Garetto and D. Towsley, "Modeling, simulation and measurements of queuing delay under long-tail internet traffic," in *ACM SIGMETRICS 2003*, 2003, pp. 47-57.

[105] S. Jain, K. Fall, and R. Patra, "Routing in a delay tolerant network," *SIGCOMM Computer Communication Review*, vol. 34, no. 4, pp. 145–158, 2004.

[106] M. Demmer and K. Fall, "Dtlsr: delay tolerant routing for developing regions," in *NSDR 2007*, Kyoto, Japan, 2007, pp. 5:1-5:6.

[107] M. Seligman, K. Fall, and P. Mundur, "Storage routing for dtn congestion control," *Wireless Communications and Mobile Computing*, vol. 7, no. 10, pp. 1183-1196, 2007.

[108] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and Wait: An Efficient Routing Scheme for Intermittently Connected Mobile Networks," in *Proceedings of the 2005 ACM SIGCOMM Workshop on Delay-tolerant Networking*, pp. 252--259.

[109] A. Lindgren, A. Doria, E. Davies, and S. Grasic, "Probabilistic Routing Protocol for Intermittently Connected Networks," Experimental RFC 6693, 2012.

[110] Thrasyvoulos Spyropoulos, K. Psounis, and C.S. Raghavendra, "Spray and Focus: Efficient Mobility-Assisted Routing for Heterogeneous and Correlated Mobility," in *Fifth Annual IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops '07.*, 2007, pp. 79-85.

[111] Y. Wang, S. Jain, M. Martonosi, and K. Fall, "Erasure-coding Based Routing for Opportunistic Networks," in *Proceedings of the 2005 ACM SIGCOMM Workshop on Delay-tolerant Networking*, 2005, pp. 229-236.

[112] F. Tsapeli and V. Tsaoussidis, "Routing for Opportunistic Networks Based on Probabilistic Erasure Coding," in *10th International Conference on Wired/Wireless Internet Communications (WWIC 2012)*, Santorini, 2012.

[113] T. Spyropoulos, K. Psounis, and C.S. Raghavendra, "Efficient Routing in Intermittently Connected Mobile Networks: The Single-Copy Case," *IEEE/ACM Transactions on Networking*, vol. 16, no. 1, pp. 63-76, 2008.

[114] T. Spyropoulos, K. Psounis, and C.S. Raghavendra, "Efficient Routing in Intermittently Connected Mobile Networks: The Multiple-Copy Case," *IEEE/ACM Transactions on Networking*, vol. 16, no. 1, pp. 77-90, 2008.

[115] A. Vahdat and D. Becker, "Epidemic Routing for Partially-Connected Ad Hoc Networks," Duke Technical Report CS-2000-06, 2000.

[116] S. Merugu, M. H. Ammar, and E. W. Zegura, "Routing in Space and Time in Networks with Predictable Mobility," Georgia Institute of Technology, Technical Report GIT-CC-04-07, 2004.

[117] A. Sekhar, B.S. Manoj, and C.S.R. Murthy, "MARVIN: movement-aware routing over interplanetary networks," in *First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON 2004)*, 2004, pp. 245-254.

[118] G. Araniti et al., "Contact graph routing in DTN space networks: overview, enhancements and performance," *IEEE Communications Magazine*, vol. 53, no. 3, pp. 38-46, 2015.

[119] S. Burleigh, "Dynamic routing for delay-tolerant networking in space flight operations," in *AIAA SpaceOps 2008*, 2008.

[120] E. Birrane, S. Burleigh, and N. Kasch, "Analysis of the contact graph routing algorithm: Bounding interplanetary paths," *Acta Astronautica*, vol. 75, pp. 108-119, 2012.

[121] P. Apollonio, C. Caini, and V. Fiore, "From the far side of the Moon: Delay/disruption-tolerant networking communications via lunar satellites," *China Communications*, vol. 10, no. 10, pp. 12-25, 2013.

[122] C. Caini and R. Firrincieli, "Application of Contact Graph Routing to LEO satellite DTN communications," in *IEEE International Conference on Communications (ICC)*, 2012, pp. 3301-3305.

[123] I. Komnios, S. Diamantopoulos, and V. Tsaoussidis, "Evaluation of dynamic DTN routing protocols in space environment," in *International Workshop on Satellite and Space Communications (IWSSC 2009)*, Tuscany, Italy, 2009, pp. 191-195.

[124] K. Suzuki, S. Inagawa, J. Lippincott, and A.J. Cecil, "JAXA-NASA Interoperability Demonstration for Application of DTN under Simulated Rain Attenuation," in *SpaceOps 2014*, 2014.

[125] M. Goetzelmann et al., "Space Data Routers for the Exploitation of Space Data," in *SpaceOps 2012*, Stochholm, 2012.

[126] J. Segui, E. Jennings, and S. Burleigh, "Enhancing Contact Graph Routing for Delay Tolerant Space Networking," in *2011 IEEE Global Telecommunications Conference (GLOBECOM 2011)*, 2011, pp. 1-6.

[127] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269-271, 1959.

[128] J. Segui and S. Burleigh, "Contact Graph Routing Enhancements Developed in ION for DTN," NASA Tech Brief NPO-48186, 2013.

[129] J.A. Fraire, P. Madoery, and J.M. Finochietto, "Leveraging routing performance and congestion avoidance in predictable delay tolerant networks," in *2014 IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE)*, Noordwijk, 2014, pp. 1-7.

[130] E. Birrane, "Building routing overlays in disrupted networks: inferring contacts in challenged sensor internetworks," *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 11, no. 2-3, pp. 139-156, Nov. 2012.

[131] I. Komnios and V Tsaoussidis, "CARPOOL: extending free internet access over DTN in urban environment," in *2013 ACM MobiCom workshop on Lowest cost denominator networking for universal access (LCDNet '13)*, Miami, 2013, pp. 21-24.

[132] I. Komnios and V. Tsaoussidis, "CARPOOL: Connectivity Plan Routing Protocol," in *12th International Conference on Wired & Wireless Internet Communications (WWIC 2014)*, Paris, France, 2014.

[133] J.A. Fraire, P. Madoery, and J.M. Finochietto, "On the design of fair contact plans in predictable Delay-Tolerant Networks," in *2013 IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE)*, Baltimore, MD, 2013, pp. 1-7.

[134] J. A. Fraire, P. G. Madoery, and J. M. Finochietto, "On the Design and Analysis of Fair Contact Plans in Predictable Delay-Tolerant Networks," *IEEE Sensors Journal*, vol. 14, no. 11, pp. 3874-3882, Nov. 2014.

[135] J. Fraire and J.M. Finochietto, "Routing-aware fair contact plan design for predictable delay tolerant networks," *Elsevier Ad Hoc Networks*, vol. 25, pp. 303-313, Feb. 2015.

[136] J.A. Fraire and J.M. Finochietto, "Design challenges in contact plans for disruption-tolerant satellite networks," *IEEE Communications Magazine*, vol. 53, no. 5, pp. 163-169, May 2015.

[137] E. Birrane and V. Ramachandran, "Delay Tolerant Network Management Protocol," Internet Draft draft-irtf-dtnrg-dtnmp-01, December 2014. [Online]. http://tools.ietf.org/html/draft-irtf-dtnrg-dtnmp

[138] J. Postel, "Transmission Control Protocol Specification," IETF RFC 793 1981.

[139] P. Karn and C. Partridge, "Improving Round-Trip Time Estimates in Reliable Transport Protocols," in *ACM SIGCOMM, 1987*, 1987, pp. 2-7.

[140] R. Ludwig and R. H. Katz, "The Eifel algorithm: making TCP robust against spurious retransmissions," *SIGCOMM Computer Communication Review*, vol. 30, no. 1, pp. 30-36, 2000.

[141] V. Jacobson, "Congestion avoidance and control," in *ACM SIGCOMM '88*, 1988, pp. 314-329.

[142] H. Ekstrom and R. Ludwig, "The peak-hopper: a new end-to-end retransmission timer for reliable unicast transport," in *23rd IEEE International Conference on Computer Communications (INFOCOM 2004)*, vol. 4, 2004, pp. 2502-2513.

[143] Paxson V. and M. Allman, "Computing TCP's Retransmission Timer," IETF RFC 2988 2000.

[144] Y.G. Iyer, S. Gandham, and S. Venkatesan, "STCP: a generic transport layer protocol for wireless sensor networks," in *14th International Conference on Computer Communications and Networks (ICCCN 2005)*, San Diego, CA, 2005, pp. 449-454.

[145] G. Anastasi, E. Ancillotti, M. Conti, and A. Passarella, "TPA: a transport protocol for ad hoc networks," in *10th IEEE Symposium on Computers and Communications (ISCC 2005)*, La Manga del Mar Menor, Cartagena, Spain, 2005, pp. 51-56.

[146] D. Kim, C.K. Toh, and Y. Choi, "TCP-BuS: Improving TCP performance in wireless Ad Hoc networks," *Journal of Communications and Networks*, vol. 3, no. 2, pp. 1-12, 2001.

[147] K. Chandran, S. Raghunathan, S. Venkatesan, and R. Prakash, "A feedback-based scheme for improving TCP performance in ad hoc wireless networks," *IEEE Personal Communications*, vol. 8, no. 1, pp. 34-39, 2001.

[148] S. Burleigh, E. Jennings, and J. Schoolcraft, "Autonomous Congestion Control in Delay-Tolerant Networks," in *SpaceOps '06*, Rome, 2006.

[149] O.B. Akan, J. Fang, and I. F. Akyildiz, "TP-Planet: A Reliable Transport Protocol for Interplanetary Internet," *IEEE Journal on Selected Areas in Communications*, vol. 22, no. 2, pp. 348-361, Feb. 2004.

[150] N. Bezirgiannidis, S. Burleigh, and V. Tsaoussidis, "Delivery Time Estimation for Space Bundles," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 49, no. 3, pp. 1897-1910, 2013.

[151] E. S. Jr Gardner, "Exponential Smoothing: The State of the Art," *Journal of Forecasting*, vol. 4, pp. 1-28, 1985.

[152] G. E. P. Box and G. M. Jenkins, *Time series analysis: Forecasting and control*. San Francisco: Holden-Day, 1970.

[153] C. C. Holt, "Forecasting seasonals and trends by exponentially weighted moving averages," *International Journal of Forecasting*, vol. 20, no. 1, pp. 5-10, 2004.

[154] P. R. Winters, "Forecasting Sales by Exponentially Weighted Moving Averages," *Management Science*, vol. 6, no. 3, pp. 324-342, 1960.

[155] R Development Core Team. (2012) R: A Language and Environment for Statistical Computing. [Online]. http://www.R-project.org/

[156] P. Turchin, *Complex Population Dynamics: A Theoretical/Empirical Synthesis*.: Princeton Univ. Press, 2003.

[157] N. Bezirgiannidis, F. Tsapeli, S. Diamantopoulos, and V. Tsaoussidis, "Towards Flexibility and Accuracy in Space DTN Communications," in *8th ACM MobiCom Workshop on Challenged Networks, (CHANTS'13)*, Miami, FL, 2013.

[158] N. Bezirgiannidis, C. Caini, D.D. Padalino Montenero, M. Ruggieri, and V. Tsaoussidis, "Contact Graph Routing enhancements for delay tolerant space communications," in *7th Advanced Satellite Multimedia Systems Conference (ASMS 2014)*, Livorno, Italy, 8-10 Sept. 2014, pp. 17-23.

[159] N. Bezirgiannidis, C. Caini, and V. Tsaoussidis, "Analysis of Contact Graph Routing Enhancements for DTN Space Communications," *International Journal of Satellite Communications and Networking, Submitted for publication*, May 2015.

[160] W. Eddy and E. Davies, "Using Self-Delimiting Numeric Values in Protocols," May 2011.

[161] F. Apollonio, C. Caini, and M. Lülf, "DTN LEO Satellite Communications through Ground Stations and GEO Relays," in *5th International Conference on Personal Satellite Services (PSATS 2013)*, Toulouse, France, June 2013, pp. 1-13.

[162] N. Bezirgiannidis and V. Tsaoussidis, "Predicting Queueing Delays in Delay Tolerant Networks with Application in Space," in *Wired/Wireless Internet Communications 2014 (WWIC 2014)*, Paris, France, 2014, pp. 228-242.

[163] E. Jennings, J. Segui, J. Gao, L. Clare, and D. Abraham, "The impact of traffic prioritization on deep space network mission traffic," in *IEEE Aerospace Conference*, Big Sky, MT, 2011, pp. 1-6.

[164] N. Bezirgiannidis, G. Papastergiou, and V. Tsaoussidis, "Dynamic Calculation of End-to-End Retransmission Timeouts in DTNs with Scheduled Connectivity," *Submitted for publication*, April 2015.

[165] G. Papastergiou, C.V. Samaras, and V. Tsaoussidis, "Where does transport layer fit into space DTN architecture?," in *2010 5th Advanced satellite multimedia systems conference (ASMS) and the 11th signal processing for space communications workshop (SPSC)*, Cagliari, Sept. 2010.

[166] A. P. Silva, S. Burleigh, C. M. Hirata, and K. Obraczka, "A survey on congestion control for delay and disruption tolerant networks," *Ad Hoc Networks*, vol. 25, Part B, pp. 480-494, Feb. 2015.

[167] Satellite Toolkit. [Online]. http://www.agi.com/

[168] Z. Katona, "GEO Data Relay for Low Earth Orbit Satellites," in *6th Advanced Satellite Multimedia Systems Conference (ASMS 2012)*, Baiona, Spain, Sept. 2012, pp. 81-88.

[169] B. Johnston et al., "SB-SAT- Persistent Data Communication LEO Spacecraft via the Inmarsat-4 GEO Constellation," in *6th Advanced Satellite Multimedia Systems Conference (ASMS 2012)*, Baiona, Spain, Sept. 2012, pp. 21-28.

[170] I. Komnios et al., "SPICE Testbed: A DTN Testbed for Satellite and Space Communications," in *9th International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TRIDENTCOM 2014)*, Guangzhou, China, May 2014.

[171] The Network Simulator - ns-2. [Online]. www.isi.edu/nsnam/ns/

[172] NS3. [Online]. https://www.nsnam.org/

[173] The Opportunistic Network Environment (ONE) simulator. [Online]. http://www.netlab.tkk.fi/tutkimus/dtn/theone/

[174] OMNeT++ Discrete Event Simulator. [Online]. http://omnetpp.org/

[175] S. Symington, S. Farrell, H. Weiss, and P. Lovell, "Bundle Security Protocol Specification," Experimental RFC 6257, May 2011. [Online]. http://www.rfc-base.org/txt/rfc-6257.txt

[176] C. Caini, A. D'Amico, and M. Rodolfi, "DTNperf_3: A further enhanced tool for Delay-/Disruption- Tolerant Networking Performance evaluation," in *2013 IEEE Global Communications Conference (GLOBECOM 2013)*, Atlanta, GA, Dec. 2013, pp. 3009-3015.

[177] S.-A. Lenas, S. Burleigh, and V. Tsaoussidis, "Reliable Data Streaming over Delay Tolerant Networks ," in *10th International Conference on Wired/Wireless Internet Communication (WWIC 2012)*, Santorini, Greece, 2012, pp. 358-365.

[178] S.-A. Lenas, S. Burleigh, and V. Tsaoussidis, "Bundle streaming service: design, implementation and performance evaluation," *Transactions on Emerging Telecommunications Technologies*, vol. 26, no. 5, pp. 905-917, 2015.

[179] Consultative Committee for Space Data Systems, "Asynchronous Message Service," Blue Book CCSDS 735.1-B-1, September 2011.

[180] Sourceforge: Delay Tolerant Networking. [Online]. http://sourceforge.net/projects/dtn/files/DTN2/

[181] IBR-DTN. [Online]. https://www.ibr.cs.tu-bs.de/projects/ibr-dtn/

[182] E. Koutsogiannis, Diamantopoulos S., and V. Tsaoussidis, "A DTN Testbed Architecture," in *Workshop on the Emergence of Delay-/Disruption-Tolerant Networks (E-DTN 2009)*, St. Petersburg, Russia, 2009.

[183] C. V. Samaras et al., "Extending Internet Into Space - ESA DTN Testbed Implementation and Evaluation," in *1st International Conference on Mobile Lightweight Wireless Systems (MOBILIGHT 2009), Workshop on Research activities funded by ESA in Greece*, Athens, Greece, May 2009.

[184] E. Koutsogiannis et al., "Experiences from architecting a DTN testbed," *Journal of Internet Engineering*, vol. 3, no. 1, pp. 219-229, Dec. 2009.

[185] Internetworked Systems Lab Projects. [Online]. http://www.intersys-lab.org/pages/projects.php

[186] Portable Satellite Simulator (PSS). [Online]. http://www.spacelinkngt.com/PSSIMBU.html

[187] CORTEX CRT system. [Online]. http://www.zodiacaerospace.com/en/our-products/aircraft-systems/data-systems/space-applications/satellite-command-control/line

[188] SIMSAT Simulator. [Online]. http://www.terma.com/space/ground-segment/satellite-simulators/

[189] S. Hemminger, "Network emulation with NetEm," in *Linux conf.*, April 2005, pp. 18-23.

[190] C.D. Edwards, M. Denis, and L. Braatz, "Operations concept for a Solar System Internetwork," in *IEEE Aerospace Conference*, 2011, pp. 1-9.

[191] C. R. Durst, G. J. Miller, and E. J. Travis, "TCP extensions for space communications," *Wireless Networks*, vol. 3, pp. 389-403, 1997.