Effects of Receive Buffer Size and Timer Granularity on TCP Performance over Erroneous Links in a LEO Satellite Network

Effects of Receive Buffer Size and Timer Granularity on TCP Performance over Erroneous Links in a LEO Satellite Network

Diplomarbeit

Berlin, December 1st, 2003

Marc Emmelmann

Department of Computer Science and Electrical Engineering

Institute of Open Communikation Systems (OKS)

Technical University of Berlin

Franklinstraße 28–29

D–10587 Berlin

Betreuer: Prof. Dr. Dr. h.c. Radu Popescu-Zeletin

Assistierende Betreuer: Hartmut Brand & Stehpan Steglich

Matrikel Nummer: 160499

Hiermit erkläert der Autor an Eidesstatt, daß er die Arbeit ohne unerlaubte Hilfsmittel und unter ausschließlicher Verwendung der genannten Quellen angefertigt hat.

Berlin, December 1st, 2003

Marc Emmelmann

Effects of Receive Buffer Size and Timer Granularity on TCP Performance over Erroneous Links in a LEO Satellite Network

Diplomarbeit

Berlin, December 1st, 2003

Marc Emmelmann

Department of Computer Science and Electrical Engineering
Technical University of Berlin

Berlin

Abstract

This master thesis presents a preliminary investigation into the impact of TCP's advertised receive buffer size and timer granularity on TCP performance over an erroneous link in a low earth orbit (LEO) satellite environment. Conducted simulations include over 200 different combinations of TCP flavor, advertised receive buffer size, timer granularity, and bit error rates. Results show that TCP can only approximate the variable propagation delay of the LEO satellite link for unsaturated channels and that the minimum timer granularity which prevents a premature expiration of the retransmission timeout (RTO) depends on the advertised receive buffer. Low bit error rates (BERs) do not influence TCP's capability to track the variable propagation delay in contrast to high BERs. Final results indicate that the relative performance degradation of TCP over erroneous LEO satellite links does not depend on the ability to estimate the variable propagation delay but rather on the employed TCP flavor. [35]

The presented work includes basic principles of TCP with respect to its congestion avoidance and retransmission mechanisms. Additionally, an introduction to satellite constellations is given and further detailed to highlight the characteristics of the analyzed satellite network. Based upon this analysis, this thesis will mathematically show that only the up- and downlink delay dominate within the end-to-end delay's variability. Thus, the developed simulation model might neglect any propagation delay associated with inter-satellite links (ISLs).

In order to simplify the simulation of erroneous satellite links, some aspect of up-to-day data encoding and forward error correction (FEC) schemes are presented. This work shows that on a transport layer level, the satellite link reveals either a uniformly distributed BER or is not available at all. [38, 39]

As the entire work was conducted within a BMBF-founded research project at Fraunhofer FOKUS, references to project constrains and system parameters are included whenever necessary. [6]

Abstrakt

Diese Diplomarbeit betrachtet einleitend die Auswirkungen der Größe des annoncierten Empfangspuffers (advertised receive buffer) und der Granularität der internen Zeitmessung (timer granularity) von TCP auf die Leistungsfähigkeit von TCP über eine mit Fehlern behafteten LEO Satellitenverbindung. Die durchgeführten Simulationen beinhalten über 200 verschiedene Kombinationen aus verschieden TCP Varianten (flavors), Empfangspuffergrößen, Granularitäten der internen Zeitmessung und Bitfehlerraten der Satellitenverbindung. Erste Ergebnisse Zeigen, daß TCP die zeitvariante Verzögerung aufgrund des Satellitenkanals nur für ungesättigte Übertragungskanäle wiederspiegeln kann. Ebenso zeigt sich, daß die Granularität mit der die TCP-internen Zeitauflösung mit der eine verfrühte Neuübertragung von möglicherweise verlorengegangenen Paketen vermieden wird, stark von der annoncierten Empfangspuffergröße abhängt. Niedrige Bitfehlerraten haben keine Auswirkung auf die Fähigkeit von TCP, die Zeitvariante Übertragungsverzögerung wiederzuspiegeln. Hingegen verhindern sehr hohe Bitfehlerraten diese Eigenschaft von TCP. Erste Ergebnisse zeigen, daß sich vor allem die eingesetzten TCP Varianten auf die Leistungsfähigkeit über einen fehlerbehafteten Satellitenkanal auswirken, wohingegen die Fähigkeit, die zeitvariante Übertragungsverzögerung wiederzuspiegeln, nur marginalen Einfluß hat. [35]

Um den Leser die Möglichkeit zu geben, sich mit für diese Diplomarbeit notwendigen Grundlagen im Bereich Satellitenkonstellationen und TCP Algorithmen zur Vermeidung von Netzüberlastsituationen vertraut zu machen, werden beide Themengebiete kurz anhand der Systemspezifikation des zugrundeliegenden Satellitennetzes erörtert. Auf dieser Grundlage wird der Leser auf mathematischem Weg von der LEO Satellitenkonstellation zu einem für die Simulation geeignetem Model der zeitvarianten Verzögerunszeiten geführt.

Um die Simulationen insbesondere in der Behandlung von auftretenden Übertragungsfehlern zu vereinfachen, werden dem Leser zusätzlich Aspekte der aktuellen Datenkodierung und Fehlererkorrektur (forward error correction schemes) vorgestellt. Auf dieser Basis kann gezeigt werden, daß aus der Sicht der Transportebene (transport layer), eine uniform verteilte Bitfehlerrate angenommen werden kann. Andernfalls steht der Satellitenkanal aufgrund zu hoher Fehlerraten in keinster Weise zur Datenübertragung bereit. [38, 39]

Die vorgestellte Arbeit ist eingebettet in ein Projekt, welches durch das Bundesministerium für Bildung und Forschung, BMBF, gefördert wurde. Die Durchführung erfolgte am Fraunhoferinstitut für Offenen Kommunikationssysteme FOKUS. Referenzen auf projektspezifische Randbedingungen und Systemparameter sind in der Arbeit an für sie entscheidenden Stellen enthalten. [6]

Acknowledgments

Scientific research results are most of the time not the fruits of an individual but of various people contributing their experience to the researcher's work. Thus, I would like to take the chance to thank them for their support to this thesis:

Hartmut Brandt: I know you had to pilfer time from project work to supervise a master thesis. Thanks that you made it never to look like a burden. The numerous discussions on TCP implementations and internal timer structures were a big help.

Bernd Bochow: It's not only your suggestions on how to structure a master thesis but also the numerous times I had the chance to listen to your ideas and explanations on signal theory and protocol design that made working with you a pleasure for the last five years. Thanks for introducing a "greenhorn" like myself into scientific working methods.

William Ivancic: The discussion on Opnet and other simulation tools that we had during SIW'02 was most fruitful. I still remember our conversation on useful TCP enhancement for satellite networks that your colleague Mark Allmann and we had during my visit at NASA Glenn in 2003.

Jürgen Winkler: Thanks for fighting with the math. Your hints on what should be further explained in order to allow a reader not familiar with TCP and satellite constellations to follow the presented work, was invaluable.

Prof. Dr. Dr. h.c. Radu Popescu-Zeletin, *Stephan Steglich*, and *Thomas Luckenbach*: You made this master thesis possible by offering me the chance to work at Fraunhofer FOKUS within the ATM-Sat project and by supervising this work at the Technical University of Berlin.

Vera Fahrig: The emotional background that you've provided me during the last years is priceless. Thanks for tolerating me staying several nights at the office to conduct simulations.

M. E.

Contents

List of Figures

List of Tables

Acronyms

ABLP	Availability Burst Length Product
ACK	Acknowledge (segment)
AER	Azimuth Elevation Range
ASCII	American Standard Code for Information Interexchange
BER	Bit Error Rate
BDP	Bandwidth Delay Product
COTS	Commercial Off-The-Shelf
CS	Control Station
DLC	Data Link Control
E_b	Energy per Bit
E_s	Energy per Signal
EIRP	Equivalent Isotropic Radiated Power
FEC	Forward Error Correction
FSM	Finite State Machine
FTP	File Transmission Protocol
GEO	Geostationary Orbit
GUI	Graphical User Interface
HEO	Highly Elliptical Orbit
ICO	Inter Circular Orbit
ITU	International Telecommunication Union
ID	Identifier
IP	Internet Protocol
ISS	International Space Station
ISL	Inter-Satellite Link
LAN	Local Area Network

LEO	Low Earth Orbit
LMS	Land Mobile Satellite (communication)
MAC	Medium Access Control
MEO	Medium Earth Orbit
MSS	Maximum Segment Size
N_0	Noise Power
OBP	On-Board Processing
PAWS	Protection Against Wrapped Sequence numbers
PSTN	Public Switched Telephone Network
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase-Shift Keying
RS	Reed Solomon (Code)
RTO	Retransmission Time-Out
RTT	Round-Trip Time
SCE	Satellite Channel Emulator
SDLC	Satellite DLC
SNMP	Simple Network Management Protocol
SRTT	Segment RTT
STK	Satellite Took Kit
STP	Satellite Transport Protocol
TDLC	Terminal DLC
TDMA	Time Devision Multiple Access
TCP	Transmission Control Protocol
TDA	Transmission Data Attribute
UDP	User Datagram Protocol

1 Introduction

1.1 MOTIVATION

TCP/IP, the protocol suite the internet is based on, is now very widely deployed. Originally designed for — in today's terms — rather slow connections providing a good link quality, packet loss is interpreted as an indication of network congestion and appropriate back-off mechanisms are being taken in order to prevent further overload. The fact that packet loss due to transmission errors, disclosing to be possibly bursty, might end up in congestion control back-off strategies, reveals the need for TCP enhancements when used over satellite links. [18, 73]

In a satellite environment, the radio signal — whose strength falls in proportion to the square of the distance travelled — has to overcome a distance ranging from one thousand up to tens of thousands kilometers. Besides the energy loss, multi-path distortion and shadowing might weaken the signal before it reaches the receiver. These effects result — depending on the employed FEC scheme — in rather high and possibly bursty bit error rates which yield to packet loss. As TCP does always interprets the latter to be caused by congestion (and not be corruption), there is a strong need to analyze the effects of satellite networks on TCP in order to highlight ways to possible improve TCP's performance especially as terrestrial bound networks will most likely exist in conjunction with satellite based networks or might even be connected with each others using satellites.

The following paragraphs will show the need for satellite based communication networks and the differences in between satellite-based and terrestrial communication. Further, some aspects of the current research related to TCP over satellite networks are highlighted.

1.1.1 The Need for Satellite Based Communication Networks

Satellite communications now form an "integral part of our new 'wired' world" [5]. The rapid evolution of mobile and wireless communication has been fueled by the growing demand for mobile information services and today's users expect their mobile handset to work "everywhere". Additionally, state-of-the-art communication devices shall support an increasingly wide range of bandwidth-demanding information services. Thus, the future growth and development in cellular and land mobile satellite communication is likely to be driven by the demand for advanced information services over information networks, such as multimedia via the Internet.

In order to provide these services, future communication networks all over the world, which are most likely to be geographically separated, will be connected via satellite networks. Besides a satellite's native capability to provide broadcast services, it might be the only feasible and cost efficient alternative to deploy broad-band multimedia services, especially in rural areas and developing countries.[1] Apart from the traditional "bend-pipe" relay architecture, intelligent satellite techniques, i.e. on-board processing (OBP) and satellite switching, will improve overall system efficiency and will help enable future land mobile satellite (LMS) communication services. These concepts may realize a promising view of a satellite network to consist of "switches in the sky" which in turn will most likely push satellite networks to act as an independent communication network providing multimedia services. [40]

1.1.2 Differences to Wired Communication Networks

Satellite based communication links are characterized by a rather long and possibly varying propagation delays, higher bit error rates as compared to a terrestrial network, and more stringent bandwidth limitations.

A long propagation delay and thus a large round trip time (RTT) have an influence on the buffer dimensioning at the receiver and sender side in order to allow each communicating partner to utilize the available bandwidth as efficiently as possible. To achieve this, local buffers should be dimensioned in the same magnitude as the bandwidth-delay-product.[2] As long as the RTT is (somehow) constant, long propagation delays do not significantly part satellite based networks from terrestrial networks offering a high bandwidth to their users. This aspect is, at least in the "wired community" well known and discussed under the term "long-fat pipelines in communication networks". [51] This leaves the focus of further research on the variability of the RTT and its effects on upper layer protocols.[3]

In addition to long and possibly variable propagation delays, satellite channels reveal a higher bit error rate than terrestrial communication paths. BERs encountered may rise up to 10^{-3} and desclose to be rather bursty for carrier frequencies prone to rain attenuation. Nevertheless, currently available commercial-of-the-shelf transponders in conjunction with appropriate forward error correction schemes may reduce BERs up to 10^{-8} with an almost uniform distribution. [20,38,39] With this perspective, the ITU defines "performance objectives for satellites" to connect back-bone ATM networks in terms of a cell loss ratio of $7.5 \cdot 10^{-8}$. [5]

Finally, the available link capacity might divide a satellite based communication system from wired networks. Besides that the available bandwidth is rather low[4], a possible asymmetry might occur if a satellite systems is only providing a broadband down-link while the up-link is realized via a low bandwidth dial-up line.

1.1.3 Related Research

Research related to TCP over satellite networks can roughly categorized into those experiments which propose a change of the TCP protocol in order to distinguish in between congestion and corruption based loss, solutions which split the end-to-end semantic of TCP in order to "insert" performance enhancing proxies in the middle of the connection, and those merely analyzing TCP performance over satellite links by tuning parameters available in standard-conform TCP implementations. Even though this thesis focuses on the evaluation of RFC conform, widely spread TCP implementations, all three research categories are briefly referenced.

Samaraweera and Fairhurst [69] propose a fundamental change of TCP in which they include mechanisms to explicitly notify the sender of transmission losses. For that purpose, they propose a new algorithm to gain a more accurate RTO. In contrast, other research rather focus on a modified TCP flavor which is backward compatible with RFC conform implementations. These proposals include, e.g., TCP-Peach which uses low-priority dummy segments to discover the available bandwidth. Afterwards, the

[1]Although the growth of the satellite part on intercontinental telephone connections has been slowed by the installation of new, optical fiber submarine cables, "satellites continue to carry most of the PSTN traffic between the developed and the developing world". [5]

[2]The ITU proposes buffer sizes of $0.5 * BDP$ to achieve a link utilization by TCP of approximately 98%. [5]

[3]The reader is asked to note that extremely long propagation delays which occur in inter-stellar communication links, are not included in the "long fat pipe" discussion and might have effects on upper layer protocols which are worth to be studied. As these effects are rather specific to a space mission, they are not considered within this work.

[4]Even though modern satellite systems provide links offering several Mbit/s, the available bandwidth of a terrestrial network is expected to be always at least one magnitude larger.

congestion window (CWND) may be increased or decreased by an addition to the Fast Retransmission and Fast Recovery Algorithm, namely Sudden Start and Rapid Recovery. [9,10]

From a satellite constellation point of view, literature featuring performance enhancing proxies (or so called "protocol boosters") put their main emphasis on GEO satellite networks. TCP connections are split transparently to the user. Data is afterwards forwarded over the satellite link with either (a modified) TCP or other protocols tuned for long delay, erroneous links. [58] These "boosting" protocols in the middle include the satellite transport protocol (STP) [57,70] or UDP-based approaches [67]. Ishac and Allman complement performance enhancing proxy based experiments by including congestion conditions in their experiments. [48]

The final group contains the evaluation of TCP over satellite channels using standard mechanisms. [13] As satellite links can somehow be compared to wired networks with a large bandwidth delay product, work related to TCP extensions over long delay networks (RFC 1072 [51]) belongs as well to the group of relevant research. Historically, experiments focuses on GEO satellite networks with a rather constant propagation delay. E.g., the TCP window scale option and the selective acknowledge option (SACK) are evaluated within this context.[5] [29] Further work considered LEO networks with variable propagation delays. They show that TCP is well able to reflect a variable propagation delay in its RTO [14] and that TCP based file transfers perform better in a LEO environment than in a GEO based satellite networks. [30] These experiments utilized only standard TCP settings with respect to the timer granularity G and advertised window size. Thus, the latter aspects are included in the work presented in this thesis. [34,35]

1.2 OBJECTIVE AND SCOPE

The presented work is embedded in a project carried out be German Aerospace Agency (DLR), Fraunhofer Institute for Open Communication Systems (FOKUS), and Tesat Spacecom. This project focuses on the design of a LEO, ATM-based satellite network supporting multimedia traffic. [6,18,19] The evaluation of widely spread, standard conform TCP implementations is included within this project. These constrains limited the analysis of TCP to two flavors: Vanilla TCP and TCP Reno. Parameters that are tunable at the end-system include the buffer size (i.e. the advertised receive buffer) and the timer granularity of TCP.

In particular, this work focuses on the evaluation of TCP over the specified LEO satellite network with respect to the following questions:

1. Does reducing the timer granularity G significantly improve TCP performance?
2. Is TCP's timer capable to reflect a variable link delay?
3. How does the advertised receive buffer size effect TCP's RTT approximation and performance?
4. How does the target system specific BER effect these aspects?

1.3 OUTLINE OF THE THESIS

The following outline is followed in order to answer the above specified questions:

Chapter 2 introduces the reader to satellite constellations and satellite specific data encoding. Starting from the definition of Keplerian Orbits, the satellite's velocity and the orbit period are derived. Additionally, the nomenclature associated with the orientation of an orbit plane is presented. The introduction to satellite constellations concludes with a detailed look at circular orbits and one special circular satellite constellation, namely Walker Constellation. Finally, it presents the forward error correction scheme used within the ATM-Sat project.

Chapter 3 provides an overview on simulation tools for satellite and commercial networks. The reader has the opportunity to follow the reasoning on why Opnet Modeler was used throughout this study. To achieve this, other commonly used simulation tools are compared to Opnet. Finally, Opnet's modeling

[5] It should be noted that SACK did not result is significant performance enhancements for the experiments presented.

concepts are roughly presented to allow the reader to gain a deeper knowledge of the newly developed simulation models presented in Chapter 5.

Chapter 4 focuses on TCP over satellite based communication networks. Starting with fundamental knowledge of how the TCP header is structured and which options can be negotiated during connection setup, the reader is pointed to TCP's data flow control mechanisms. A deeper understanding of these mechanisms, especially the sliding window and retransmission concepts including TCP's internal timer structure is required to further evaluate the presented simulation results. Due to this requirement, special emphasis is put on the congestion control and retransmission algorithms, namely slow start, congestion avoidance, fast retransmit, and fast recovery. The chapter ends with a rather general overview on the effects of satellite based networks on TCP.

Chapter 5 presents the actual evaluation of TCP over the LEO satellite link specified within the ATM-Sat project. After placing the work into the context of the ATM-Sat project, an overview on the simulation workflow is provided. Afterwards, the simulation models are developed, parameter for the simulation run specified, and the simulation results depicted.

The development of the simulation models is twofold: It starts out with a STK based analysis of the LEO satellite network in order to gain a propagation delay model which can further be used in Opnet. Afterwards, the Opnet specific models are explained, i.e. the satellite channel model is developed and the final simulation model used for TCP evaluations is depicted. The simulation settings specified can be divided into two major groups: the first simulation group focuses on the verification of the developed simulation model itself while the second group actually analyzes TCP over the ATM-Sat specific network. The chapter ends with the presentation and interpretation of the gained simulation results.

Finally, Chapter 6 summarizes this work and reveals some aspects which might be worthwhile to study in further simulations.

Chapters 2,3, and 4 are independent of each other. Thus, a consecutive reading of each chapter is not mandatory but recommended in order to follow the main aspect of this work as presented in Chapter 5. If already familiar with TCP's congestion control and retransmission algorithms, the reader might skip Chapter 4. Chapter 2 can be omitted if the reader does not wish to follow the derivation of the satellite channel model in all its mathematical details.

Additionally, the choice is left to the reader to acquaint himself with mathematical proofs of the applied formulas to analytically describe Keplerian orbits and the velocity of orbiting satellites (Chapter 2) or with the interpolation theorem which allows a better reconstruction of the propagation delay function based on the measured delay samples provided by STK (Chapter 5). These proofs and auxiliary information are started with the word *Lemma* written in italics and concluded with a bullet (●).

Mathematical proofs requiring more than one page are placed in the appendix. Additional measurement results which do merely support the results gained in this work are placed there. With respect to the source code of the implemented simulation models, only code manually written by the author is added to this thesis. C-Code which can easily generated via the graphical interfaces of either used simulation tool is therefore omitted as it would require several hundreds of pages.

Please note the glossary and index register provided in the back of this work whereas the acronym list is given at the beginning of this thesis.

2 Introduction to Satellite Constellations and Satellite Specific Data Encoding

This chapter shortly introduces basic satellite constellations and data encoding and forward error corrections schemes for satellite systems.

Starting from the Keplerian Laws, the satellite velocity and orbit periods are derived. Later on, the reader is introduced to basic terminology of the orbit plane's orientation. Based on this general introduction, circular orbits are highlighted. The Walker orbit, as a special circular orbit, is presented in all its details as this constellation is used to derive the satellite channel's delay characteristics later on.

The second part of this chapter provides a detailed explanation of data encoding and forward error mechanisms using the example of the FEC scheme applied within theATM-Sat project. Based on this FEC scheme, the error model which is later on applied to the TCP simulations is presented.

2.1 BASICS OF SATELLITE CONSTELLATIONS

2.1.1 Overview on Satellite Orbits

After the problem of determining the position and path of a satellite in space as a function of time has occupied scientist for thousands of years, Johannes Kepler discovered some important properties of planetary motion in the early 17th century. Kepler's Laws state that:

1. The planets move in a plane; the orbits around the sun are ellipses with the sun at one focal point (1602).
2. The line between the sun and a planet sweeps out equal areas in equal intervals of time (1605):
$$r^2\frac{d\theta}{dt} = \text{const.} \tag{2.1}$$
3. The ratio between the square of the orbit period T and the cube of the semi-major axis a of the orbit ellipse is the same for all planets (1618):
$$\frac{T^2}{a^3} = const. \tag{2.2}$$

It remained for Isaac Newton's discovery of the Universal Law of Gravitation (1667) to identify the forces associated with Kepler's Laws. Specifically, every mass M_1 attracts another mass M_2 with a force directed along the line joining the two masses and having a magnitute F_g of

$$F_g = G\frac{M_1 M_2}{r^2} \tag{2.3}$$

where G is the Universal Gravitational Constant[1] and r the distance in between M_1 and M_2. [5,55]

2.1.1.1 Keplerian Orbits The "Keplerian assumption" which underlies Keplerian orbits is a first order approximation of the forces that apply to any satellite orbiting around a central body. Under this assumption, the interaction between the attracting bodies is restricted to a two-body problem. [5] Therefore, these laws can be applied to describe the motion of a satellite around the Earth. [55] The influence of other moon's or sun's attraction is neglected and the only acting force is the Newtonian force (or central force) μ/r^2. As the mass m of the satellite can be neglected with respect to the mass M of the Earth (central body), μ is given by [5]:

$$\mu = GM \tag{2.4}$$

which is approximately $398600\, km^3/s^2$ for the Earth.

Figure 2.1 illustrates the geometry of such an orbit. The ellipse is described by the semi-major and semi-minor axes a and b with the Earth placed at one focal point. The radius r denotes the distance of the satellite from the Earth's center; the point of the orbit where r is smallest is called *perigee* with $r = r_p$ and the point with largest r is denoted *apogee* with $r = r_a$. The angle between the perigee and the satellite as seen from the Earth's center is commonly referred to as *true anomaly*. In contrast, the *eccentric anomaly* E traces a point on a circle, with the radius equal to a, that circumscribes the elliptical orbit.[2] The shape of the ellipses is defined by the eccentricity ($0 \leq e \leq 1$). It determines, the distance c of the Earth (i.e. one focal point) from the ellipse center and is given by

$$c = ea \tag{2.5}$$

$$e = \sqrt{1 - \frac{b^2}{a^2}} \tag{2.6}$$

The distance between the Earth's center and the perigee (apogee) is accordingly

$$r_p = a(1-e) \tag{2.7}$$

$$r_a = a(1+e) \tag{2.8}$$

which can directly be derived from the geometry illustrated in Fig. 2.1. [5,25,26,55]

Lemma: As for an eccentric anomaly of $90°$ the radius r is equal to the major-axis a. This relation is given by the definition of the ellipses itself:

> An ellipse is the locus of points the sum of whose distance from two fixed points is constant. The two fixed points are called the foci of the ellipse. [53]

The constance is known to be $2a$ and thus, we can write

$$\begin{aligned} E = 90° \quad &\Rightarrow \quad r = a \\ &\Rightarrow \quad a^2 = c^2 + b^2 \end{aligned}$$

which, in conjunction with Eq. (2.5) can further be simplified to Eq. (2.6). ●

2.1.1.2 Satellite Velocity and Orbit Period

Velocity For a satellite orbiting around the earth, the total energy E_b consisting of the potential energy E_U and the kinetic energy E_K is constant and commonly referred to as *binding energy*. [55,68,71]

$$E_K + E_U = \text{const.} = E_b \tag{2.9}$$

The kinetic energy is given by

$$E_K = \frac{1}{2}mv^2_{satellite} + \underbrace{\frac{1}{2}mv^2_{Earth}}_{=0} \tag{2.10}$$

[1]Gravitational Constant $G = 6.6730010^{-11} m^3 kg^{-1} s^{-2}$

[2]Lutz et al. [55] use a different definition of the eccentric anomaly. They measure the angle in between perigee and the satellite on its orbit (and not on the circumscribing circle) as seen from the ellipses center. The definition provided in this work is, nevertheless, more commonly used [5,25,26].

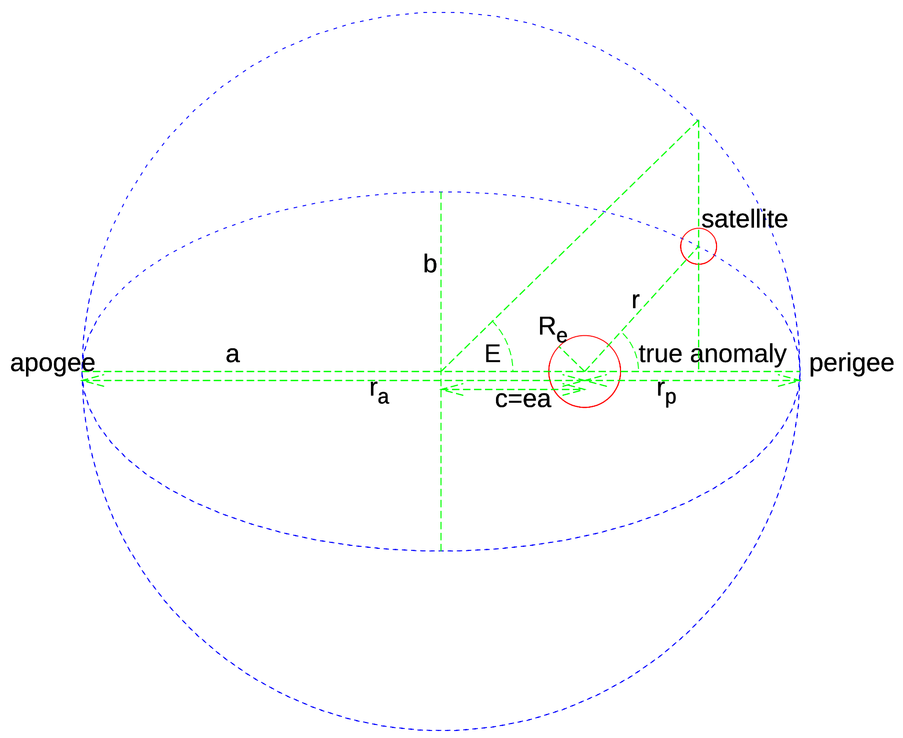


Fig. 2.1 Parameters of Elliptical Orbits, 2-D-view [5,25,26]

The satellite's orbit is observed from the Earth; thus $v_{Earth} = 0$ and we can write for $v \equiv v_{satellite}$:

$$E_K = \frac{1}{2}mv^2 \tag{2.11}$$

The potential energy can be derived from Eq. (2.3):

$$\begin{aligned} E_U &= -\int_{r_0}^{r} F_g\,dr + E_U(r_0) && (2.12)\\ &= G\frac{mM}{r} \quad \text{for} \quad E_U(r_0) \overset{\text{def}}{=} 0 && (2.13)\end{aligned}$$

and the binding energy E_b is known to be inversely proportional of the major axes[3] [71]

$$E_b \sim \frac{1}{a} \tag{2.14}$$

The proportional factor $-\mu m/2$ in combination with Eq. (2.4) allows us to rewrite Eq. (2.9)

$$\underbrace{\frac{mv^2}{2}}_{E_K} \underbrace{-\frac{\mu m}{r}}_{E_U} = \underbrace{-\frac{\mu m}{2a}}_{E_b} \tag{2.15}$$

which finally yields to the satellite's velocity[4]

$$v = \sqrt{\mu\left(\frac{2}{r} - \frac{1}{a}\right)} \tag{2.16}$$

[3]For a detailed discussion of the binding energy including energy diagrams and the analysis of the resulting hyperbolic, parabolic, elliptical, and circular orbits, reference is given to R. G. Brown's lecture notes. [27]
[4]The interested reader is referred to Bob Jenkins website which provides several interactive illustrations of orbits and the corresponding plant's velocity. [52]

The velocity depends on the the (current) radius r which is a function of the true anomale θ. The velocity is smallest for a large radius r which is the case for the satellite passing the apogee, and largest in the perigee.

Lemma: The proof of Eq. (2.16) is straight forward for a circular orbit ($r \equiv a$). [68] A planet moving in a circular orbit around the Earth is accelerating toward the latter with the centripetal acceleration v^2/r. As this acceleration is caused by the gravitational force F_g which is given in Eq. (2.3), we can determine the satellite's velocity in conjunction with Eq. (2.4)

$$\begin{aligned} \text{acceleration} &= \frac{v^2}{r} = \frac{\mu}{r^2} = \frac{F_g}{m} \\ v &= \sqrt{\frac{\mu}{r}} \end{aligned}$$

which is identical to Eq. (2.16) for ($r \equiv a$).

Further, we can define a radius-independent mean motion $\bar{v}$ of the satellite in order to find a relation between the satellite's velocity and the eccentric anomaly:

$$\bar{v} \overset{def}{=} \frac{v}{r} = \sqrt{\frac{\mu}{r^3}} \tag{2.17}$$

and rewrite Eq. (2.17) as

$$v = r\bar{v} = rE\frac{d}{dt} \tag{2.18}$$

using both, the satellite's mean motion and the angular velocity of the eccentric anomaly $E\,d/dt$ as, for a circular orbit, the eccentric anomaly is identical to the true anomaly. ●

Orbit Period The orbit period can be calculated on the basis of Kepler's observations. [26] His famous equation:

$$t = \frac{E - e\sin E}{\bar{v}} \tag{2.19}$$

calculates the time elapsed since the satellite has passed the perigee as a function of the eccentric anomaly E, the satellite's mean motion $\bar{v}$, and the eccentricity of the orbit e. If the spacecraft were traveling on the circumscribing circle shown in Fig. 2.1 rather than the elliptical orbit, it would have an angular velocity equal to the mean motion. Thus we can write according to Eq. (2.18)

$$\bar{v} = \sqrt{\frac{\mu}{a^3}} \tag{2.20}$$

and dereive from Eq. (2.19) and an eccentric anomaly of $E = 2\pi$ the orbit period T

$$T = \frac{2\pi}{\bar{v}} \tag{2.21}$$

$$= 2\pi\sqrt{\frac{a^3}{\mu}} \tag{2.22}$$

2.1.1.3 Orientation in the Orbit Plane

The following parameters uniquely describe the orbit plane's orientation in an inertial reference frame (Fig. 2.2):

1. inclination,
2. right ascension of the ascending node (RAAN), and
3. argument of perigee.

The *inclination* defines the angle between the orbit plane and the equatorial plane. It is counted positively with respect to the ascending satellite orbit track.

The *ascending node* is passed when the satellite enters the northern hemisphere. The *right ascension of the ascending node* (RAAN) determines the angle in the equatorial plane between a reference direction,

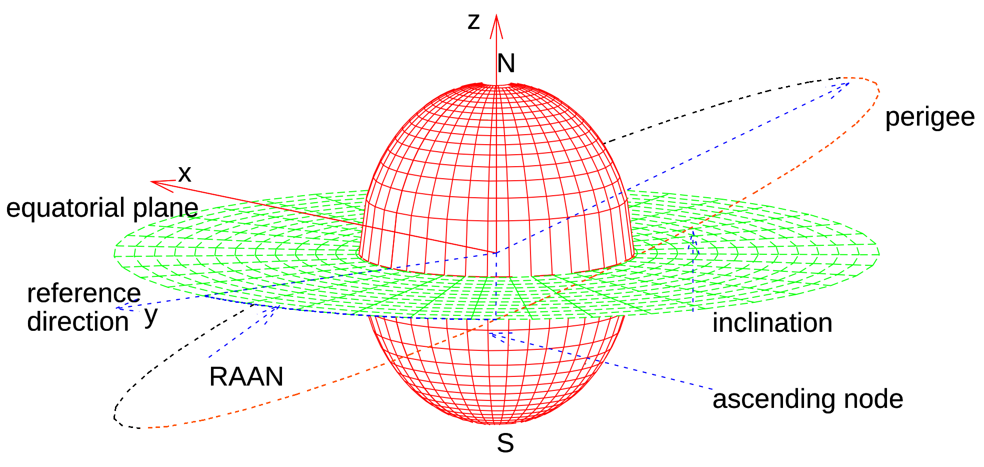


Fig. 2.2 Orientation of the Orbit Plane [55]

which is usually the direction from the Earth's center to the sun at vernal equinox, and the ascending node.

The *argument of perigee* is the angle in the orbit plane between the ascending node and semi-major axis of the ellipses (location of the perigee). [55]

Thus, the orientation and shape of a satellite orbit is defined by the following five parameters [5]:

1. Semi-major axis of the ellipse,
2. eccentricity of the ellipse,
3. inclination of the orbit plane
4. right ascension of the ascending node, and
5. argument of perigee.

If another parameter, i.e. the true anomaly, is included, the position of a satellite is described. This set of six parameters is then referred to as *Kepler elements*. [55]

2.1.1.4 Typical Circular Orbits Besides the shape of the orbit, i.e. either elliptical or circular, the orbit altitude h and its inclination i are the most important orbit characteristics. [55] Besides characterizing two special kinds of circular orbits, i.e. Low Earth Orbits (LEOs) and Geostationary Orbits (GEOs), this section mathematically derives the orbit altitude and the associated propagation delay.

For circular orbits, the semi-major axis a is equivalent to the orbit radius r as seen from the Earth's center ($a \equiv r$) as illustrated in Fig. 2.3. The slant range d can directly be derived according to the Cosine Law [24]

$$d = \sqrt{R_e^2 + r^2 - 2R_e r \cos E} \tag{2.23}$$

where E denotes the eccentric anomaly which is identical to the true anomaly for circular orbits. The radius r can be retrieved by simple rewriting Eq. (2.22)

$$r = \sqrt[3]{\mu \left(\frac{T}{2\pi}\right)^2} \tag{2.24}$$

which finally yields to an orbit altitude h as seen from the Earth's surface of

$$h = \sqrt[3]{\mu \left(\frac{T}{2\pi}\right)^2} - R_e \tag{2.25}$$

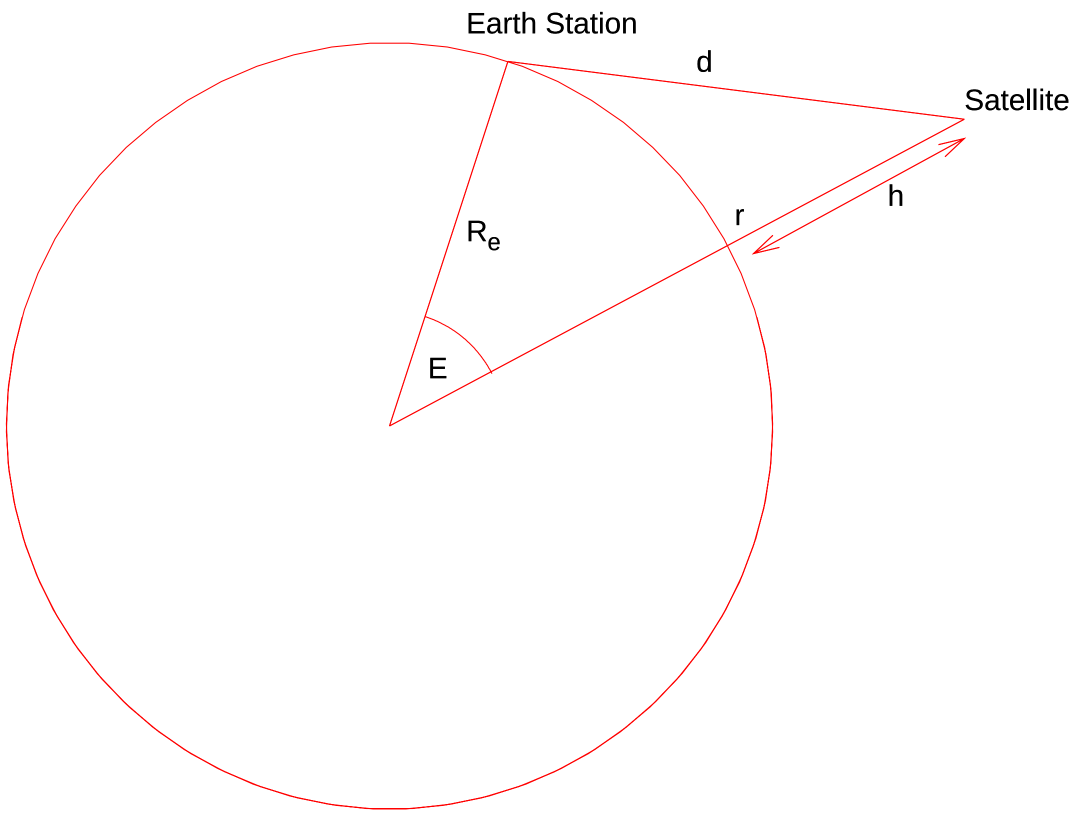


Fig. 2.3 Geometry of a Circular Orbit [55]

Table 2.1 Orbit Comparison for Satellite Communication Applications [5]

Orbits	LEO	MEO	GEO
Orbital Period	1.5 – 2 h	5–10 h	24 h
Altitude Range	500 – 1500 km	8000 – 18000 km	40000 km (i=0)
Visibility Duration	15 – 20 min./pass	2 – 8 h /pass	Permanent
Elevation	Rapid variations; high and low angles	Slow variations; high angles	No variation; low angles at high latitudes
Propagation Delay	Several milliseconds	Tens of milliseconds	$>$ 250 milliseconds
Link budget	Favourable; compatible with small satellites and handheld user terminals	Less favourable	Not favourable for handheld or small terminals
Instantaneous ground coverage (diameter at 10° elevation)	$\approx$ 6000 km	$\approx$ 12000 – 15000 km	16000 km

where, for simplicity, R_e denotes to the mean equatorial radius of the Earth ($R_e = 6378.14\ km$).

If the orbit altitude h is determined by an orbit period T which is an integer divisor of a day (T [houres] $\in \{1, 2, 4, 6, 12, 24\}$), the satellite positions reiterate periodically each day. [55] Without regard to the orbit period, certain regions of operation are useful as they do not fall into the Van Allen belts:[5]

- Geostationary Orbit (GEO)
- Medium Earth Orbit (MEO)[6]
- Low Earth Orbit (LEO)

Table 2.1 compares the orbits with respect to satellite communication applications.

Geostationary Orbit A satellite launched in the equatorial plane (inclination $i = 0$) whose circular orbit has an orbit period equal to one sidereal day[7] has a rotation exactly to the rotation of the Earth's surface directly below. Thus, the satellite appears to be stationary as observed from Earth. Its orbit altitude is accordingly $h = 35786\ km$ [25]; the useful coverage is limited by an elevation angle of 5 deg. As the satellite appears to be stationary to an observer on Earth, its clear advantage is that antennas can be pointed to the satellite without tracking capabilities. [5]

Non-Geostationary Orbit Non-GEO satellite constellations are mostly used to perform special functions and services, i.e. weather observations, remote earth exploration, radio-navigation, communications, surveillance, etc. One of the unique features of these satellites is the ability to view the entire Earth's surface periodically from a single satellite. If simultaneous coverage is required, a number of satellites can be employed. [5]

Especially *Low Earth Orbit* satellites can help to provide global access to the telecommunication infrastructure. The propagation delay associated with their orbit altitude of 500 - 5000 km makes them feasible to provide delay sensitive communication services. For this purpose, more than one satellite is required to guarantee continuous coverage. This requirement is met by, e.g., a Walker constellation.

[5]The Van Allen belts are regions of the ionosphere with high ion concentration which dramatically reduces the satellite's lifetime. [5,25,26,55]
[6]A synonym for MEO is ICO: Intermediate Circular Orbit [55]
[7]The time required by the Earth to rotate once with respect to the stars equals one sidereal day ($= 23\ h\ 56\ min\ 4.09\ s$)

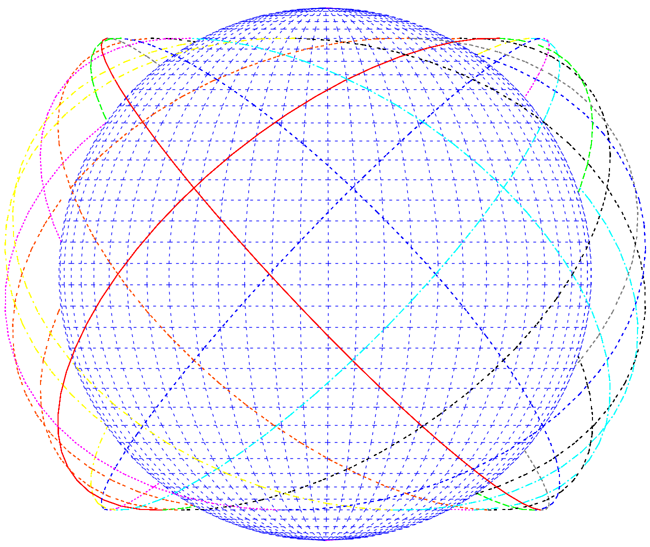


Fig. 2.4 Orbit Paths associated with Walker Constellation of ATM-Sat Project

2.1.2 The Walker Satellite Constellation

J. G. Walker developed satellite constellations for global coverage. Deriving from his work, a satellite constellation of

- regularly spaced,
- circular orbits with
- equal orbit altitudes and
- equal inclinations

is often called Walker constellation. [25] Thus, a Walker constellation can be described by the total number of satellites N, the number of orbit planes P, and a phasing factor F.[8] This triplet is called the *Walker Notation: N/P/F*. Figure 2.4 illustrates the orbit paths of a Walker 72/12/1 constellation as used within the ATM-Sat project. The orbits have an inclination of $25°$. [18]

Assuming, a Walker constellation implies inter-satellite links (ISLs) and a shortest-path routing algorithm, the following two relations hold:

1. To come as close as possible to the point which is the farest away from the current satellite, i.e. on the other side of the Earth, only by ISLs in the same orbit plane, the number of intra-ISLs p is limited by

$$p \leq \left\lceil \frac{N}{2P} \right\rceil \tag{2.26}$$

2. If only ISLs which change the orbit plane (inter-orbital ISLs) are used, the number of inter-orbital ISL q is limited by

$$q \leq \left\lceil \frac{P}{2} \right\rceil \tag{2.27}$$

[8]The phasing factor F, $F \in [0, P-1]_{\mathcal{N}_0}$, determines the angular offset between the satellites in adjacent orbit planes. [55] It is not further discussed within the contents of this work (and sometimes even omitted), even though the interested reader is referred to C. Brown [25] who provides further details on the relation between the phasing factor and a global coverage of a Walker constellation.

Even though this characteristic is obvious due to the symmetry of a Walker constellation, it will later on allow to tremendously reduce the complexity of used simulation models.

Additional characteristics with respect to the distance in between two satellites (and the associated propagation delay) can be found:

The geometry of Walker orbits directly yields to the fact that the distance in between two satellites in the same orbit plane is constant. So is the associated propagation delay.

Due to the regular geometric characteristic of the employed Walker constellation, further characteristics can be noted with respect to the distance in between two satellites on neighboring planes:

All satellites have the same orbit period. Thus, their position with respect to each other is also periodic which directly yields to a periodic function describing the distance / associated propagation delay in between two satellites.

In the contents of the ATM-Sat project, further simplifications can be made. ATM-Sat considers only inter-orbital ISLs to the next-most satellite. Thus, the associated propagation delays of all inter-orbital ISLs resemble each other. For a fixed time frame, the associated propagation delays are equal except for a phase shift and we can write:

Let $\mathcal{L}$ be the set of all possible inter-orbital ISLs, then the delay associated with each member of $\mathcal{L}$ resembles each other except for a phase shift φ:

$$\bigwedge_{l_1,l_2\in\mathcal{L}} \bigvee_{\varphi} (d_{interISL}(l_1,t) = d_{interISL}(l_2,t+\varphi)) \tag{2.28}$$

As satellites are evenly spaced within an orbit, the phase shift φ can be expressed as

$$\bigvee_{n\in[0,p-1]} \left(\varphi = n\frac{T_{orbit}}{p}\right) \tag{2.29}$$

where, for the ATM-Sat project, according to Eq. (2.22), the orbit period $T_{orbit} \approx 112$ minutes and the number of satellites per plane is $p = 6$.

2.2 ENCODING OF DATA AND FORWARD ERROR CORRECTION

The fundamental of coding is to decrease the bit error rate of a communication channel by increasing the symbol rate or bandwidth. A common bit error rate (BER) for uncoded information in a satellite environment is approximately 10^{-2} for an energy-per-bit over noise-power ratio (E_b/N_0) of 4 dB. Biorthogonal block coding or Reed-Solomon coding can decrease the BER by a factor of 10 and convolutional coding is even able to reduce the BER to 10^5. [26]

The following sections will introduce the adaptive FEC scheme used within the ATM-Sat project and derive from that scheme the effective error model which is later on used for TCP analysis.

2.2.1 FEC Method Applied within the ATM-Sat Project

The ATM-Sat target system uses an adaptive MAC layer protocol with reduced contention probability in the uplink and an improved bandwidth utilization due to its adaptive FEC scheme. [38, 39] The variable structure of the MAC frames[9] guarantees a fixed useful data rate by adapting the MAC packet length to the employed FEC scheme. The MAC framing structure does not effect the BER. Therefore, the following excerpt, taken from Emmelmann and Bischl [38], does only relate to the employed FEC scheme:

> The designed satellite system utilizes the Ka-band which mainly suffers from high rain attenuation and — in case of moving vehiculars — also from signal shadowing due to obstacles like trees or buildings. As for this project, vehicles are equipped with directional antennas, multipath fading is slight [45]. Signal shadowing due to obstacles is too high to be compensated by forward error correction (FEC) thus leaving its focus on the attenuation caused by rain.

[9]The variable structure of the MAC frames is not further detailed as it does not effect the BER. Please refer to [38] for a detailed discussion of the framing structure.

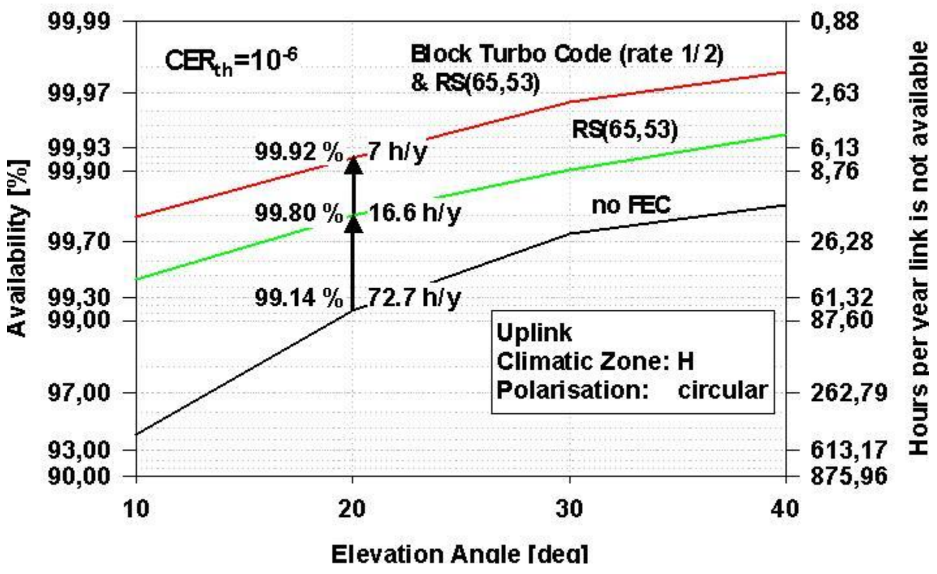


Fig. 2.5 FEC-Depending Link Availability ($CER_{th} = 10^{-6}$) [38]

As rain attenuation appears only from time to time and its maximum fade slope value for 0.01% of an average year is about 0.6 dB/s [28], adaptive forward error correction and also adaptive modulation techniques are applied to efficiently use the available bandwidth. This technique is a major improvement in contrast to traditional systems with non-adapting FEC schemes as the latter requires a rather high link margin which unnecessarily delimits up- and down-link capacity. The employed FEC scheme guarantees a fixed useful data rate by adapting the MAC packet lengths according to the used coding and modulation scheme.[10] The system design includes three major FEC schemes:

- No FEC is applied. The ATM cells are merely guarded with a 32-bit CRC with a block length of 57 bytes. The CRC is used for error detection only.
- Reed Solomon Code, RS(65,53) over GF 2^8, which can correct up to 6 byte errors.
- Rate 1/2 block turbo code concatenated with a RS(65,53) code.

Figure 2.5 illustrates that even at a minimum elevation angle of 20°, the system can guarantee a cell error rate threshold (CER_{th}) of 10^{-6}. If we define link availability as the percentage of time the given CER_{th} is guaranteed, a simple CRC error detection scheme without any FEC correction methods results in a link availability of 99.14% (i.e. the link is not available for 72.7 hours per year). The RS(65,53) code increases the availability to 99.80% and the concatenation of RS and Turbo codes improves link availability to 99.92%.[11] As all used codes are systematic, the uncoded block error rate of the information part can be used together with the measured received power to decide when to switch between the codes.

We use the availability burst-length product (ABLP) to compare the efficiency of this adaptive FEC scheme with a permanent RS(65,53) coding at QPSK modulation. The ABLP of the permanent scheme is 1.22 ($= 99.80\% * 65/53$). The adaptive FEC scheme has a higher availability of 99.92% and has nevertheless a lower ABLP of only 1.08 ($= 99.14\% * 57/53 + 0.66\% * 65/53 + 0.12\% * 130/53$).

In addition to adapting the FEC to the experienced rain attenuation and elevation angle, the modulation scheme can be changed in rain-less periods and high elevation angles to 8 or 16 QAM. Figure 2.6 illustrates the expected E_s/N_0 for a satellite overpass (i.e. for different elevation angles) at different rain intensities. The shaded background depicts the areas where the given CER_{th} of 10^{-6} can be guaranteed by using a specific FEC and modulation scheme. E.g., for a rain-less period, the system may switch to QAM, 8 QAM with a mere CRC checksum (at $t = -5.5min$ in Fig. 2.6 which corresponds to an elevation angle of approx. 30°) or even to QAM, 16 QAM (at $t = -2.5min$, elevation angle of

[10] The design supports also a fixed MAC packet length by adapting the useful data rate even though this option is not implemented in the demonstrator.

[11] Reference constellation: 72 satellites in 12 Walker orbits at 1350 km, 47° inclined. Link parameter: 30 GHz QPSK with 2447 ksymbols/s, EIRP 42.7 dBW, G/T 4.5 dB/°K, 2.5 dB Implementation losses.

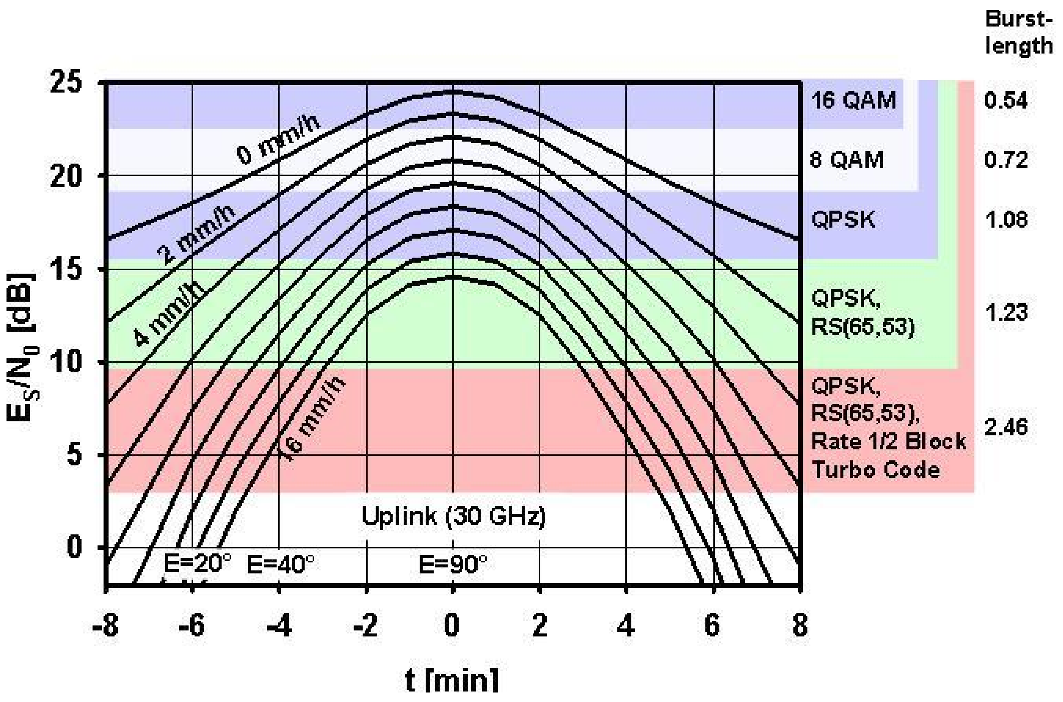


Fig. 2.6 E_s/N_0 for a satellite overpass at different rain intensities (in mm/h); required coding/modulation for an ATM CER_{th} of 10^{-6} and the resulting burst length

approx. $80°$). Only a E_s/N_0 below 10 dB requires QPSK with RS and Turbo code. Figure 2.6 also includes the burst lengths corresponding to each FEC-modulation pair.

The developed MAC protocol has been implemented on a FreeBSD based, modular prototyping environment [36] which allowed measuring the effective BER[12] of an end-to-end communication connection. These measurements

included the effects of rain at an intensity of 6, 10, and 16 mm/h at different FEC schemes. Fig. 2.7 depicts the experienced cell loss ratio for various coding schemes and rain intensities for an entire satellite overpass. For the reader's convinience, it also includes the current elevation angle for a given time. Even for severe rain falls of 16 mm/h, the required CER_{th} can be guaranteed with a simple CER FEC scheme for elevation angles higher than approx. $65°$. For lower elevation angles, the MAC has to switch to RS coding and finally to the concatenated FEC coding scheme (RS and Turbo). The measurements also illustrate that for slight rain falls of only 6 mm/h, the bandwidth efficient CRC FEC coding may be used for almost an entire satellite overpass. [13]

These measurements show that the adaptive FEC scheme results in a two-state link behavior: Either the link provides an effective BER better than 10^{-8} or it is not available at all as the dramatically increased BER prohibits any communication. [38,39]

2.2.2 Effective Error Model for TCP Simulations

The analysis of TCP over an erroneous LEO satellite link focus on the specific ATM-Sat system. The latter employs an adaptive MAC & FEC scheme which as been shown to result either in a rather good link state with an BER of 10^{-8} or a completely unavailable link. These "black outs" were not subject of the simulation, thus, the following three error rates were considered:

- An effective bit error rate of 10^{-8} which reflects the actual system behavior of the ATM-Sat project,

[12] Actually, Emmelmann et al [38,39] measure the cell loss ratio which can be put in correlation to an effective BER for an ATM cell stream.
[13] Source: Emmelmann and Bischl, 2003. [38]

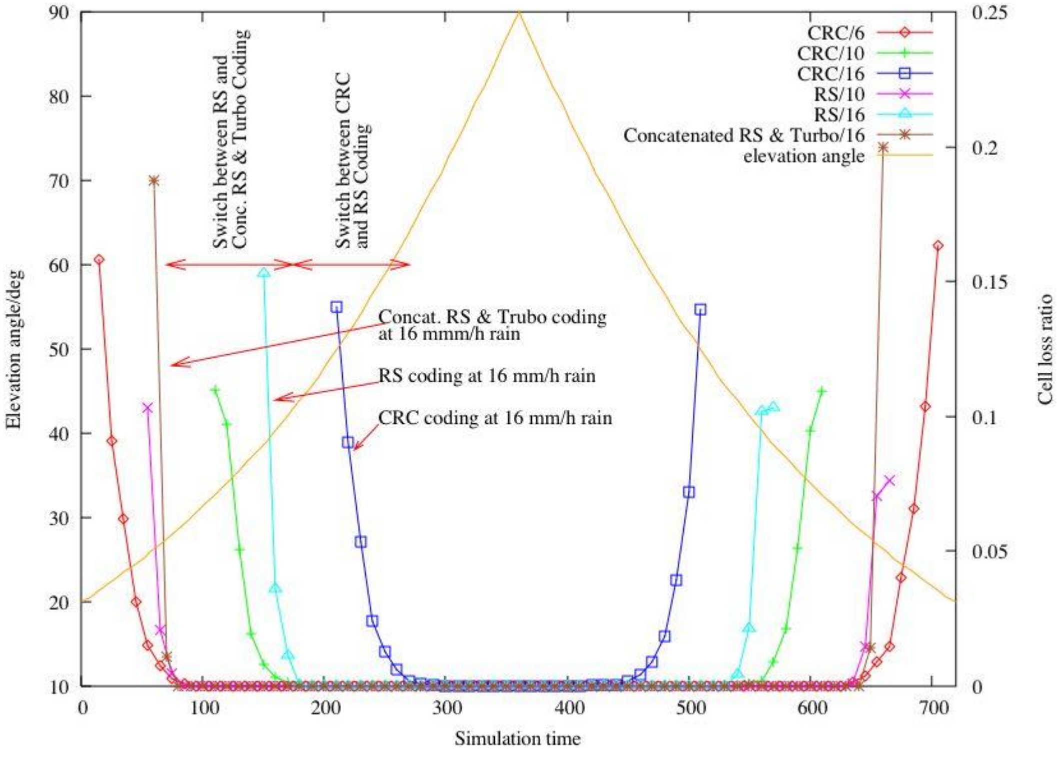


Fig. 2.7 Measured Error Rates for Various Codings and Rain Intensities

- an effective bit error rate 10^{-5} which is common for an uncoded satellite channel, [26] and
- a perfect, error free link which allows to isolate effects of a variable propagation delay channel apart from imposed bit errors.

3 Overview on Simulation Tools for Satellite and Communication Networks

This chapter will briefly introduce satellite constellation and channel simulators as well as network layer oriented simulators. Only those tools are discussed which had been evaluated in an early stage of the ATM-Sat project [6] or which were developed within the project itself.

Details on the modeling concept of Opnet Modeler are presented as the project partners agreed on using Opnet for both, MAC protocol development and TCP simulation.

3.1 SATELLITE CONSTELLATION AND CHANNEL SIMULATORS

3.1.1 Satellite ToolKit

The Satellite ToolKit (STK) is a commercial-off-the-shelf (COTS) analysis and visualization tool. The standard core of STK is available for free and includes various analytical capabilities (e.g. precise calculation of a satellite's position and altitude in time), allows orbit / trajectory generation, and features visibility analysis of any pair of STK objects. [15]

STK can perfectly be used to visualize any satellite orbit based on its orbit parameters and to retrieve so called AER (Azimuth Elevation Range) data with respect to any two STK objects. The latter functionality allows, in the contents of this work, to gain information on the distance (range) in between two satellites in neighboring orbit planes (ISLs) and in between a ground terminal and a satellite (up- and downlink).

3.1.2 The Ohio Network Emulator

ONE (the Ohio Network Emulator) is a tool to emulate a network between a pair of interfaces on a single Solaris-based workstation. Each packet arriving at one interface is forwarded via the other interface. Before leaving the other interface, a delay may be imposed on the packet and the packet can be corrupted with a given probability. In order to impose variable propagation delays, as encountered in a satellite network, ONE interfaces with the Satellite ToolKit (STK). [12, 63]

ONE might be used to run communication flows over an emulated satellite network while abstracting entirely from any underlying, satellite-specific MAC layer. As this functionality limits ONE to experiments employing existing TCP implementations, it is not used within this work. Additionally, conduction numerous experiments in real-time can take up to several weeks for a single run of a simulation set which is not reasonable in a first approach to evaluate TCP.

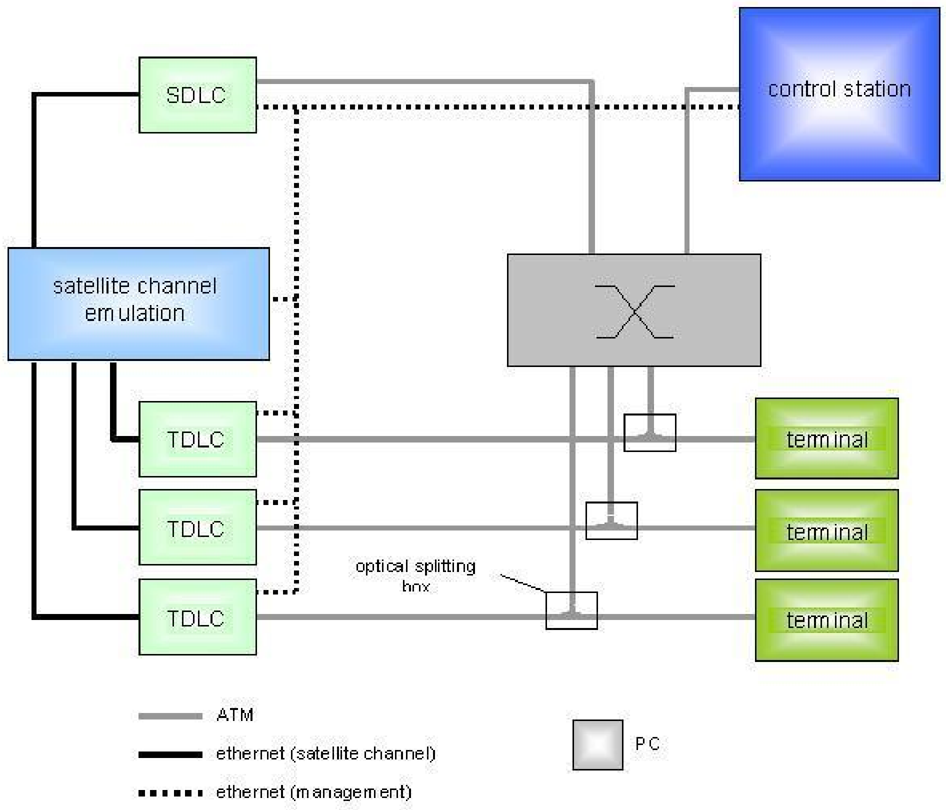


Fig. 3.1 ATM-Sat Satellite System Emulator

3.1.3 ATM-Sat Satellite System Emulator

The ATM-Sat Satellite System Emulator provides an environment to prototype, test, and evaluate MAC, tools to simulate and signaling protocols. [22,23] Its three major components, i.e. the *satellite channel emulator* (SCE), *satellite* and *ground terminal DLCs* (SDLC & TDLC), and the *control station* (CS), act independently (Fig. 3.1). Thus, the emulation of a satellite channel with respect to its associated variable propagation delay and channel error rate is possible without prior knowledge of the MAC protocol.

The SCE units is able to impose a predefined, time-variant propagation delay on packets being forwarded. Corrupting packets based on a given error rate is also possible. The control station propagates via SNMP to the SCE unit look-up tables which include the variable propagation delay and current error rate. These tables can be derived from a STK-based analysis of the considered satellite system.

A further advantage of the system is its capability to run end-to-end significant (user) applications over the emulated satellite channel as any possible set of terminals running user applications can be connected via the system with or without using a specific MAC layer protocol.[1] [38] Even though this allows to conduct experiments concerned with higher layer protocols, e.g. TCP, over a satellite channel as long as the an implementation of the probed protocol can be run on any kind of user terminal, the ATM-Sat Wireless Emulator was not used for the work presented in this paper as it was not available at the time the simulations took place. Additionally, running TCP analysis in real-time can take up to several weeks if a large number of TCP flavors and buffer size settings are considered.[2]

[1]If the SDLC and TDLC units are omitted, the system acts merely as a link / network emulator comparable to ONE.
[2]This network centric view has been extended into an *Integrated Prototyping and Simulation Architecture for Space Specific Protocol Developments and Verifications* which merges the Opnet-based simulation approach presented in Section 5.2 with the target system oriented MAC protocol prototyping. [36]

3.2 NETWORK SIMULATORS

3.2.1 NS

The Network Simulator (ns-2) is a discrete event simulator targeted at the networking research and educational community. It is focused on modeling network protocols, e.g. TCP, UDP, and other multi- and uni-cast protocols. Besides wired and wireless link models (including satellite channels), ns-2 includes application models (web, ftp, telnet) as well as routing models (ad hoc routing in sensor networks). Its open source code is freely distributed which allows a large community in the research and educational area to share simulation models. [1,43]

As ns-2 focuses on the network layer, it makes it harder to extend simulation models by including target system specific MAC layer protocols. In the latter case, Opnet is more favorable.

3.2.2 Opnet Modeler

3.2.2.1 Introduction Opnet Modeler is an object-oriented, discrete-event based network simulator. It comes with an interface to network management tools (e.g. HP openview) and with a large library of standard network components from network devices (e.g. switches, server, and firewalls) over protocol stacks (which include the entire TCP/UDP/IP protocol suite) up to link layer protocols. Besides network specific simulations, Opnet Modeler allows to set financial cost attributes for devices which allows an market oriented view of new and existing network topologies and their performance. Thus, besides in the educational and research community, Opnet Modeler is well accepted in commercial environments. [8,64]

Opnet provides all protocol and application models in C-based source code which easily allows modifications. Besides its of being well accepted in a commercial environment, Opnet's capability to incorporate physical layer characteristics and environmental effects in the simulations was a major argument within the ATM-Sat project in favor for Opnet vs. ns-2. [32]

The following section will provide a brief overview on Opnet's modeling concepts and nomenclature to allow the reader to easily follow later on the detailed description of the developed TCP-related simulation models.

3.2.2.2 Overview on Opnet's Modeling Methodology Opnet simulation models are based on a series of hierarchical views: the network view, the node view, and the process view. Each view represents parts of the structure of real networks, equipment, and protocols and has its own specific editor for modifications (Fig. 3.2).

Modeling Networks In order to model any kind of communication network, the topology of the latter has to be described. A network may consist of *network nodes* representing any kind of network device (e.g. router, switches, or workstations) and *subnetworks*. *Link objects* are used to connect network nodes with each other or with a sub-network; they can either provide a simplex or duplex point-to-point connectivity or have a bus characteristic.

Networks (and sub-networks) can be unlimitedly nested. In addition, each network can be put in a geographical context (dimensions of the view or the exact position of the network on the Earth) and can reflect physical characteristics i.e. mountains and rivers. [64]

In order to simulate "moving objects", i.e. cars or satellites, each network node can be attached to a trajectory description. With respect to simulating a satellite network, trajectory descriptions for each satellite of a constellation can be imported by Opnet from STK.

Description of Network Nodes As a network node can represent any kind of network device, the node view consequently captures the architecture of a network device or system. It illustrates the data in between functional elements within the network device. These functional units are called *modules*. Each module can generate, send, and receive packets from other modules to perform its function within the node. Modules typically represent applications, protocol layers, algorithms and physical resources, i.e. buffers, ports, and busses. Modules are assigned process models to achieve any required behavior. [64]

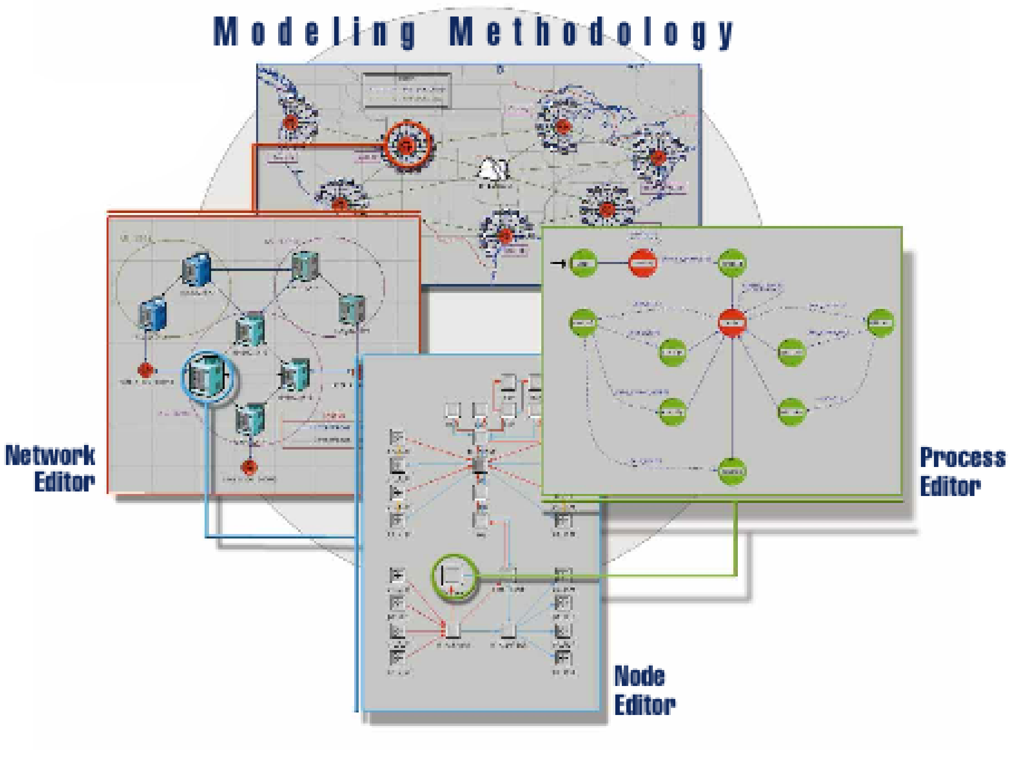


Fig. 3.2 Opnet Modelling Methology [64]

Two standard library network node components, a so called *application definition* and *profile definition*, allow easily to create predefined traffic on a network. Within the "application definition" network node, the user may describe any kind of standard application, i.e. IP-telephony or retrieving a large file via FTP. Parameters of the application definition include for the latter example, the size of the file to get, possible a preferred server (otherwise, a network node offering FTP Server functionality is randomly chosen) and a description, how often this kind of file is retrieved. Finally, the "profile definition" describes a user profile. E.g., a user might call twice a day a colleague via IP-telephony and start once a day an FTP file transfer. Consequently, its profile definition would specify the time of the day, the previously defined application "IP-telephony" would be started (of course twice a day) and when the FTP file transfer application should start. Finally, the "profile definition" can be attached to any network node representing a computer / workstation which executes the user profile.

Description of Processes Opnet uses finite state machines (FSMs) to support the specification of protocols, resources, etc. Each state of a FSM can contain C/C++ code to fulfill the desired behavior of the state. [64] Even though using FSMs is a fast and easy approach to describe or develop new protocols, it is also possible to encapsulate the entire functionality of a module in external C/C++ code.

Modeling Communication Channels Communication channels are represented by so called "link models". Opnet distinguishes between point-to-point links, buses, and wireless transmission links. An open architecture is provided to allow developers to specify a customized behavior for each individual link on a per-transmission basis. Opnet denotes this architecture as the *transceiver pipeline* as it provides a conduit connecting a transmitter to one or more receivers.

The structure of the transceiver pipeline is similar for each link type. In order to transfer a packet from a transmitter to a receiver, a series of computation functions is applied to the packet. Each of this function models a particular aspect of the link behavior and is referred to as a *pipeline stage*. The interface of each pipeline stage and the order in which they are executed is standardized but the functionality of the pipeline stage itself can freely be implemented by the user outside Opnet's simulation kernel.

The principal objective of a transceiver pipeline is thus to determine whether or not a packet can be received at the link's destination including the time at which the packet is received. Depending on the kind of link, a transceiver pipeline consists of one[3] or several of the following pipeline stages:[4]

- Receiver Group (r)
- Transmission Delay (pbr)
- Link Closure (br)
- Channel Match (r)
- Transmitter Antenna Gain (r)
- Propagation Delay (pbr)
- Receiver Antenna Gain (r)
- Received Power (r)
- Background Noise (r)
- Interference Noise (r)
- Signal-to-Noise Ratio (r)
- Collision (b)
- Bit-Error-Rate (r)
- Error Allocation (pbr)
- Error Correction (pbr)

The order of pipeline stages in which they are executed is not considered here. [3]

[3] A transceiver pipeline of one (functional) stage simply implements "void-functions'" for its remaining stages.
[4] p — point-to-point transceiver pipeline; b — bus transceiver pipeline; r — radio / wireless transceiver pipeline

4

TCP/IP over Satellite Based Communication Network

This chapter provides a brief introduction to TCP fundamentals to allow readers not familial with TCP internals to easily follow the conducted simulations and results as described in Chapter 5.

Section 4.1 describes the general packet format and puts emphasis on TCP's flow control and retransmission algorithms. Second, widely spread TCP implementations are named and put into context of this work (Section 4.2). Finally, Section 4.3 outlines some major characteristics of satellite based networks and how they affect TCP performance.

4.1 FUNDAMENTALS OF TCP/IP

TCP provides a reliable flow of data between two hosts. Its internal functionalities include dividing the data passed to it from the application into appropriately sized pieces for the network below, acknowledging received data, and sets certain timeouts to assure packets are either retransmitted if they are lost or acknowledged by the TCP entity on the other end of the connection. [73] As an OSI-layer-3 protocol, it is not concerned with network-independent addressing of its interlocutor (which is realized by IP) but rather with addressing a certain application represented by port numbers. [72, 74] The TCP header is illustrated in Fig. 4.1. Besides any (optional) data and option field, it is 20-byte-long which results in a minimum IP datagram size of 40 Bytes. [73]

An introduction to all features and facets of TCP is out of the scope of this work which limits the following sections to the aspects of TCP which are most relevant for understanding the conducted analysis of TCP performance over a LEO satellite network with respect to the advertised receive buffer size and timer granularity. We follow the structure suggested by Stevens [73] which highlights features of the connection establishment phase, details TCP data flow aspects, and closes with today's most widely spread congestion control and retransmission schemes.

4.1.1 Connection Establishment

4.1.1.1 Maximum Segment Size In order to minimize the overhead associated with the TCP and IP header, TCP should determine the largest "chunk" of data it can transmit without fearing its fragmentation. Therefore, each TCP entity (might) announce the size of the largest segment that it can possibly transfer. This size is referred to as the maximum segment size (MSS). The two TCP interlocutors agree afterwards on the smaller value of the two announcements.

The announcement is not mandatory. If a MSS proposal is not received, an entity assumes a default of 536 bytes which results for a 20-byte-long TCP and 20-byte-long IP header in a 576-byte-long IP datagram. [73]

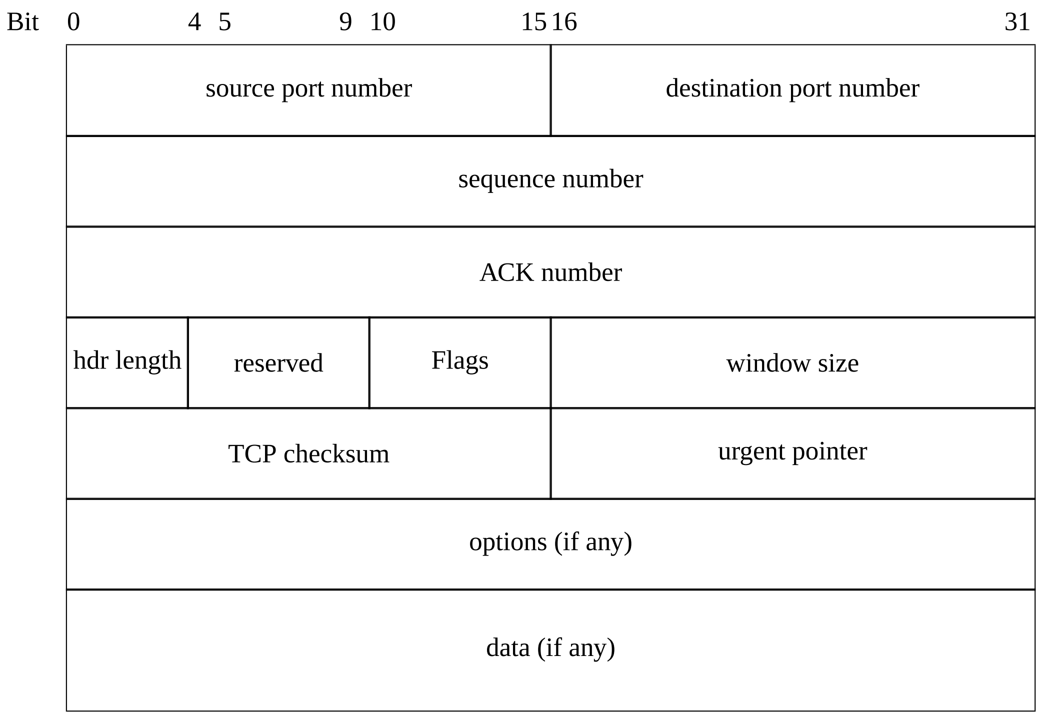


Fig. 4.1 TCP Header Format [73]

In an Ethernet-based environment, best performance is achieved for a MSS of 1460 bytes [59] whereas for a non-local IP-address of the communication partner, the MSS normally defaults to 536.

It should be noted that the MSS announcement does only consider the largest segment size each interlocutor can handle. Fragmentation may still occur if an intermediate router cannot forward these segments in one piece. To avoid this situation, MTU discovery had to be employed. [73]

4.1.1.2 Window Scaling During connection establishment, each TCP entity announces its local receive window size which reflects the amount of data that the receiving TCP entity may handle without having the upper layer application receiving the data (buffer size). This window size is continuously updated during the connection and is therefore a standard component of the TCP header (refer to Fig. 4.1). The window size is given in bytes which results to a maximum supported advertised receive window / buffer of $2^{16} - 1$. As this limit on the buffer might be to stringent, especially for high bandwidth connections or large round trip times associated with the link, TCP might employ a window scaling option. [49,51]

The window scaling option tells TCP by how many positions the value given in the header has to be shifted to the left. Valid shifts range in between 0 (no shift) and 14 which results in a maximum supported window size of $(2^{16} - 1) * 2^{shift}$ bytes.

The LEO satellite network considered here does not require the window scaling option. As its bandwidth delay product, which is the upper bound to fully saturate the link by a single TCP connection, is well below 65535 bytes, a 16-byte-long window size is efficient.

4.1.1.3 Other TCP Options

Selective Acknowledgment The Selective Acknowledgment TCP flavor / options was originally proposed in RFC 1072 [51] and allows the receiving entity to provide exact information about the packets correctly arrived which in turn allows the sender to maintain information on which packets were (presumingly) lost. During the fast retransmission phase, the sender first retransmits all suspectly lost packets before sending new ones. This allows the sender to recover from several lost segments within one RTT. [21,33,41,56]

At the time of conducting the experiments presented in this thesis, SACK was not a standard, widely spread TCP flavor and, thus, is not further analyzed within the contents of this work.

Timestamping The timestamp option lets the sender place a timestamp value in every segment. The receiver reflects this value in the ACK allowing the sender to calculate an RTT for each received ACK. This options is useful for extremely large window sizes as TCP can only estimate once the RTT per congestion window. [73]

Turning the timestamp option on might increase TCP performance but will most likely not degrade results gained in the work. Thus, the evaluation of the effects of placing a timestamp in every ACK is left for the future.

4.1.2 TCP Data Flow Control

4.1.2.1 Delayed Acknowledgments Normally, TCP does not send an ACK the instant it receives data. Instead, it delays the ACK and hopes to have data going in the same direction as the ACK, so the ACK can be sent along with the data. If this ACK piggybacking is not possible within a certain amount of time TCP sends a single ACK segment anyway. [73] This delayed ACK delay is usually set to 200 ms which is compliant to the Host Requirements RFC [47] stating that the delay must be less than 500 ms. This requirement is even more stringently met as most TCP implementations acknowledge every Kth segment[1] out of a group of segments arriving within the 200 ms interval which does avoid waiting for the delayed ACK timer to elapse even if there is no pending traffic on the reverse channel. [51]

4.1.2.2 Sliding Window Mechanism & Retransmissions TCP determines the number of segments (or bytes) that it can transmit before receiving an acknowledgment by a sliding window mechanism. IBM's System Network Architecture (SNA) refers to this kind of flow control also as *pacing group*. [42] The (effective) size of the sliding window may vary over the time and is set to the minimum of the receiver's advertised window size and the current value of the congestion window (CWND). Figure 4.2 numbers the bytes that have to be transferred from the sender to the receiver as 1 through 11. The first three bytes have been successfully transfered whereas bytes 4 through 6 were sent but their acknowledge is still pending. As the current effective window size allows 6 bytes to be transferred without receiving an acknowledgment, TCP may further transmit bytes number 7 through 9 right away. As soon as further segments are acknowledged, TCP may slide the window further to the right. [73]

The acknowledgment strategy of TCP has a direct effect on how the sliding window is allowed to move forward. As illustrated in Fig. 4.2, TCP numbers the bytes to be transferred sequentially. An ACK does always include the byte number of the next byte expected to be transmitted, i.e. an ACK(4) means that all bytes up to (and including) byte number 3 have successfully been received. Due to this mechanism, a TCP implementation compliant to RFC 793 [46] may not acknowledge byte number 9 until is has received both, bytes 4 and 5. On the sender side, TCP's local timer, the retransmission timeout RTO, expires as the ACK for byte number 9 does not arrive. As a result, TCP has to retransmit the entire current window starting after the last acknowledged byte.[2]

4.1.2.3 Advertised Receive Buffer Size The advertised receive buffer size, which is also denoted in literature as advertised window size, reflects the number of available bytes in the buffer of the receiving node. The number may drop down to zero if, e.g., the receiver's application is not able to retrieve its pending data from the TCP entity. [73] As TCP must not transmit data if the receiver announces a window size of zero, this behavior implements an implicit congestion control method.

The maximum advertised receive buffer size is controlled by the receiving process which can affect TCP's performance. Depending on the operation system, the maximum window size varies in between 2048 and 16384 bytes (refer to Table 4.1). [73] Mogul [59] as well as Papadopoulos and Parulkar [66] show that the most common default value for the advertised receive buffer of 4096 bytes is not optimal for Ethernet based communication links. This fact suggests to test TCP performance for different window

[1]K is usually set to 2.

[2]This retransmission behavior is the one implemented in RFC 793 which is also referred to as Vanilla TCP. TCP Reno, e.g., employs a slightly different mechanism as will be seen later on.

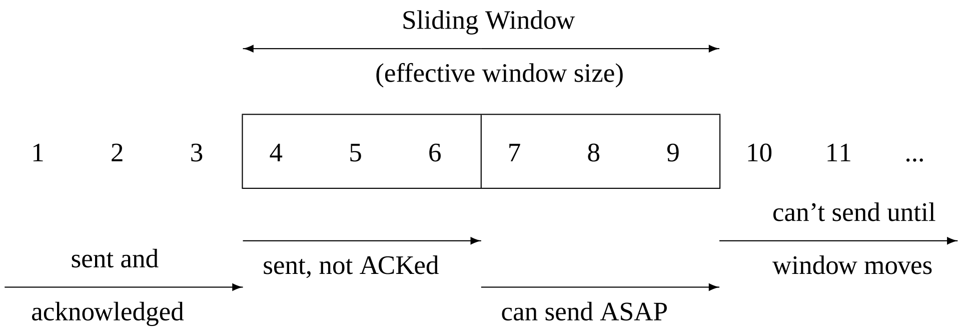


Fig. 4.2 Sliding Window Mechanism [73]

Table 4.1 Default Values of the Advertised Receive Buffer Size for various Operation Systems [73]

Operation System	Advertised Receive Buffer Size [a]
4.2 BSD	2048 bytes
4.3 BSD	4096 bytes
SunOS 4.1.3	4096 bytes
BSD/386	4096 bytes
SVR4	4096 bytes
Solaris 2.2	8192 or 16384 bytes
4.4 BSD	8192 or 16384 bytes
AIX 3.2	8192 or 16384 bytes

[a] The same amount of memory is allocated to the sending buffers.

sizes which should at least include experiments considering receive buffers equal to the bandwidth delay product.

4.1.3 TCP Timeout and Congestion Control

4.1.3.1 Round-Trip Time Measurement TCP recognizes a successful data transfer by receiving an acknowledgment. As RFC 793 [46] does not specify an algorithm to detect a segment loss or corruption other than waiting for an appropriate time for the ACK to arrive. This time is denoted retransmission timeout (RTO). If a segment is not acknowledged within this time interval, it is assumed to be lost and has to be retransmitted.

The RTO has to be determined according to the current round trip time (RTT) associated with the link. The earliest possible time at which a segment's ACK can arrive equals to one RTT. Nevertheless, the RTO should be larger than the RTT since there is not always a one-to-one relation between a transferred data segment and its corresponding ACK (one example is the delayed ACK mechanism described previously).

The original RFC [46] calculates the RTO based on a smoothed RTT estimator (called R) using a low-pass filter with a smoothing factor α and a variance factor β

$$R \leftarrow \alpha R + (1-\alpha)M \tag{4.1}$$
$$RTO \leftarrow \beta R \tag{4.2}$$

where M refers to the time in between sending a data segment and receiving the corresponding acknowledgment. Obviously, as the ACK might be delayed, this interim might be slightly larger the the actual RTT associated with the link. Recommended values for α and β are 0.9 and 2 correspondingly. [73] The smoothed RTT is calculated to reduce the influence of short peaks in the measured RTT as might occur due to queuing delays in the network but does not consider the variance of the measured RTT. Jacobson [50] shows that this lack of the original RTO calculation might result in an inappropriate RTO value which can cause false retransmissions. This behavior is tremendous to the network as it adds unnecessary network load and does not occur for the newly proposed RTO calculation:

$$\begin{aligned} Err &= M - A && (4.3)\\ A &\leftarrow A + gErr && (4.4)\\ D &\leftarrow D + h(|Err| - D) && (4.5)\\ RTO &= A + 4D && (4.6) \end{aligned}$$

where A is the smoothed RTT and D is the smoothed mean deviation. Err denotes to the difference between the current estimated (smoothed) RTT and the value that has most recently been measured. The gain g of the average and the gain h for the deviation are usually set to $1/8$ (0.125) and $1/4$ (0.25) correspondingly.[3] [73] Usually, the RTO is lower bounded in TCP implementations and has a minimum value of 500 ms.

Another aspect of measuring the RTT is TCP timer granularity G. TCP measures the round trip time in multiplies of G which is set to 500 ms for most analysis. As a result, TCP can only determine the RTT with an error of G which in turn might have effects on TCP's ability to correctly reflect the actual RTT associated with a link.[4] Smaller values of the timer granularity allow TCP to better measure the current segment round trip time but do also make this measurement more prone to sudden changes in the RTT. As a result, fine timer granularities might cause TCP's RTT measurement to reflect a RTT which is slightly smaller than the actual RTT which in turn might yield to a premature collapse of the RTO.

An optimal estimation of the current RTT and thus an optimal size of the RTO are fundamental for best possible TCP performance. If the RTO is too large, throughput will be low as a possible retransmission will start delayed. On the other hand, if the RTO falls below the current RTT, premature retransmissions will occur and network bandwidth is wasted.

4.1.3.2 Congestion Avoidance Algorithms

TCP assumes that packet loss is caused by network congestion (i.e., a router dropping packets due to over-full queues) rather than by corruption (high bit error rates). Based on this assumption, two retransmission algorithms, namely Slow Start / Congestion Avoidance and Fast Retransmit / Fast Recovery, try to avoid overloading the network. Both of them calculate a maximum number of (unacknowledged) segments that are allowed to be transmitted at the same time. This number is referred to as congestion window ($CWND$) which is used to determine the effective window size of the sliding window mechanism as described in Section 4.1.2.2.

Slow Start and Congestion Avoidance Besides the congestion window, slow start and congestion avoidance require an additional variable called slow start threshold $ssthresh$. If $CWND$ is less than or equal to $ssthresh$, the slow start algorithm is executed to calculate consecutive values of CWND, otherwise congestion avoidance is used. The two algorithms are defined as follows:

1. Slow Start

   Increase $CWND$ by one segment for each ACK that is received. As $CWND$ is implemented to limit the number of unacknowledged bytes, it is increased by the number of bytes that the ACK confirms to be received. This results in doubling $CWND$ every round trip time if the delayed ACK algorithm is not employed on the receiver side.

2. Congestion Avoidance

   Congestion avoidance dictates that $CWND$ is incremented by $1/CWND$ for each ACK received. This is an linear increase as compared to the exponential increase of slow start.

Final adjustments to $CWND$ and $ssthresh$ are made when congestion occurs which is indicated by either receiving duplicate ACKs or by the expired RTO timer. In both cases, one half of the effective

[3]Jacobsen's original work [50] included $2D$ in the calculation of RTO. A factor of $4D$ is currently most widely spread in TCP implementations.

[4]It should be noted that, e.g., FreeBSD calculates the RTT in multiplies of so called "timer tics" whose duration depend on the clock rate of the computer chip. As the tic's duration is far below 1 ms most of the time and since FreeBSD employ by standard the timestamp option, the granularity G has no effects for this real operation system. [7]

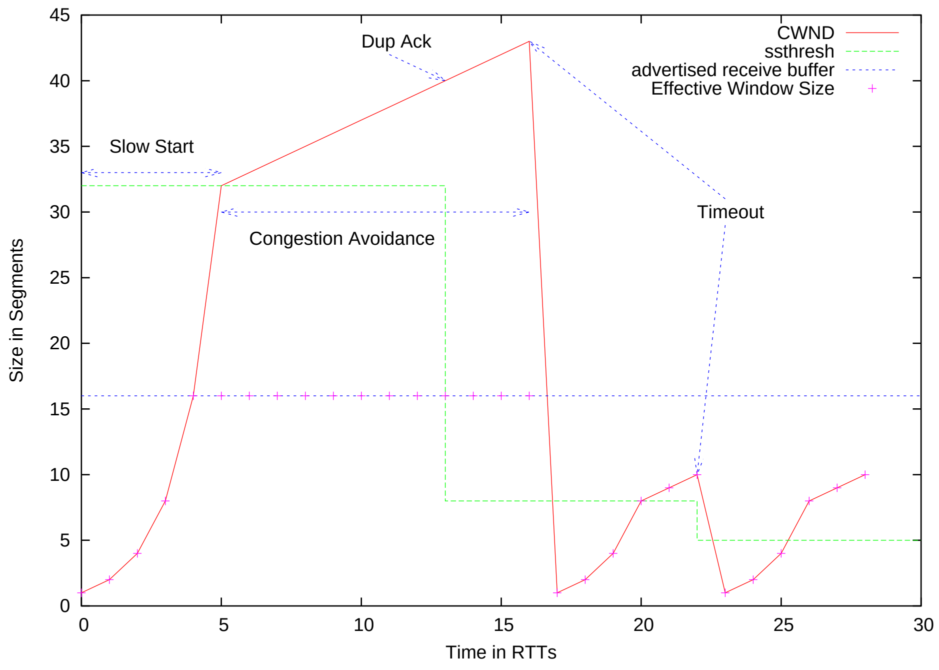


Fig. 4.3 Slow Start and Congestion Avoidance Algorithm

window size but at least two segments[5] is saved to $ssthresh$. Additionally, if the congestion is caused by a timeout of the RTO, the congestion window is decreased to one segment. [73]

Figure 4.3 illustrates these situations for an advertised receive buffer size of 16. $CWND$ is initialized to 1 and afterwards exponentially increased (slow start phase). After 5 RTTs, $CWND$ reaches the current value of $ssthresh$ and enters congestion avoidance in which the increase is only linear. It should already be noted that after 4 RTTs, the effective window size is limited by the advertised receive buffer size of 16.

After 13 RTTs, the sender detects an duplicate ACK and adapts the current value of $ssthresh$ to one half of the current effective window size (i.e. 16). The value of $CWND$ is left untouched. The first timeout occurs after 17 RTTs: $CWND$ is initialized to 1; the value of $ssthresh$ does not need to be changed as is equals the half of the current effective window size. In the further duration of the connection, $CWND$ is always lower than the advertised receive buffer size and thus, the effective window size equals the value of $CWND$. Due to these circumstances, the final timeout after 22 RTTs results in changing $ssthresh$ to 5.[6]

Fast Retransmit and Fast Recovery The retransmission scheme employed with slow start and congestion avoidance relies on the RTO time to expire before a lost segment is retransmitted. The fast retransmit algorithm avoids this problem and the fast recovery algorithm prevents the congestion window to entirely collapse after a single retransmission.

1. Fast Retransmit

   The reception of duplicate ACKs does most likely indicate the loss of the data segment which was transmitted right after the acknowledged data. Thus, if three or more duplicate ACKs arrive in a row, TCP performs a retransmission of the presumingly lost segment.

2. Fast Recovery

[5] According to Section 4.1.2.2, the effective window size is the minimum of the advertised receive buffer and the current congestion window.
[6] Existing implementations would rather change $ssthresh$ to either 4 or 8 as these values are a power of 2.

After starting the retransmission, $CWND$ is not entirely closed to one. Instead of slow start, TCP enters the congestion avoidance phase. $ssthres$ is set the one-half of the current effective window size (minimum of $CWND$ and the advertised receive buffer size). Afterwards, $CWND$ is set to $sshthresh$ plus 3 times the segment size. This is caused by the fact that we have received 3 duplicate ACKs which indicated that at least 3 segments have successfully been transferred thus allowing an increase of $CWND$.

Each time another duplicate ACK arrives, $CWND$ is increased by one segment; if the number of outstanding packets is less than the new effective window size, further packets may be transferred.

Finally, when the first ACK acknowledging new data arrives (which is the case when the retransmitted packet had arrived at the sender), $CWND$ is set to the current value of $ssthresh$.

This algorithm is able to recover from one lost segment within one RTT as a second lost segment can not be detected due the lack of corresponding duplicate ACKs. If a second segment is lost within one RTT, the RTO will expire and the congestion window will collapse to one segment. TCP restarts with slow start and congestion avoidance afterwards. [11, 72, 73]

4.2 TCP FLAVORS

Different TCP implementations do often feature different algorithms with respect to how they handle packet loss and how they avoid congestion (i.e. how the in- or decrease $CWND$). These different implementations are often referred to as "TCP flavors". The following section will shortly introduce the three most common flavors: Vanilla TCP, TCP Reno, and TCP SACK.

TCP Reno is the most widely deployed version of TCP when it comes to a stable, commercial environment. Even though TCP SACK is more and more deployed, it is not the standard TCP flavor and is thus not further considered within this work.

4.2.1 Vanilla TCP

The implementation of *slow start* and *congestion avoidance*, which is thus compliant to RFC 793 [46], is occasionally referred to as Vanilla TCP. The detection of congestion is only based upon the expiration of the retransmission timer after which SSTHRESH is set to the value of CWND. The slow start algorithm is applied to further increase CWND with a new initial value of one. Congestion avoidance takes over as soon as CWND reaches SSTHRESH. [16, 33]

4.2.2 TCP Reno

The TCP Reno implementation is based upon the *fast retransmit and recovery* algorithm. As an acknowledgment is sent for each, or in the case of delayed ACKs for every second packet received, a duplicate ACK containing the same number of bytes expected to be received, indicates a lost (or late) package. After three duplicate ACKs, the sending TCP retransmits the missing packet immediately. After this fast retransmit and the acknowledgment of the lost package, the fast recovery algorithm halves CWND and artificially increases it by one for each duplicate ACK. Afterwards, the congestion avoidance phase is entered. This behavior allows to recover from one lost segment within one RTT. [16, 33]

4.2.3 TCP New Reno

TCP New Reno is a set of bug fixes of the original. It avoids multiple window reductions in one window of data and constrains the burstiness of the sender upon leaving fast recovery. [44]

4.2.4 SACK TCP

Selective acknowledgments (SACKs) are used by the receiver to provide exact information about the packets correctly arrived. During the fast retransmission phase, the sender first retransmits all suspectedly lost packets before sending new ones. This allows to recover from several lost segments within one RTT. [16, 33]

4.3 THE EFFECTS OF SATELLITE BASED NETWORKS ON TCP

Satellite channels are dominated by the following fundamental characteristics which have an effect on TCP performance:

- noise and resulting transmission errors,
- bandwidth constrains,
- propagation delay characteristics different from a terrestrial environment, and
- asymmetric usage.

The influence of these characteristics can either be reduced by the satellite system which in turn almost eliminates the effect on TCP, or have to be coped with by TCP itself. [11]

4.3.1 Noise and Transmission Errors

The strength of a radio signal falls in proportion to the square of the distance traveled. In a satellite environment, the distance to overcome might range in between one thousand up to tens of thousand kilometers (LEO vs. GEO satellite environment). Thus, this signal of a satellite link becomes rather weak before it reaches its destination. Additionally, multi-path distortion and shadowing (e.g. blockage by buildings or trees) become more important for mobile user terminals. As a result, bit error rates are, compared to terrestrial wired networks, rather low. State of the art transmission systems are capable to provide a BER of 10^{-7} or less. [11]

TCP in its most widely spread flavors is not capable to distinguish in between packet loss caused by congestion or caused by transmission errors. It always assumes network overload to have caused intermediate switches to have dropped a packet and reduces its transmission rate accordingly. This approach is not feasible for a rather high error environment frequently corrupting packets as it does not fully utilize the available bandwidth.

Two approaches to overcome this problems can be considered: The first employs advanced error correction mechanisms, e.g. Reed Solomon or Turbo coding, which in turn results in BERs approaching todays fiber quality. [11, 22, 32] Second, one might modify TCP algorithms in order to distinguish in between packet loss caused by errors or network overload which results in optimized congestion control algorithms and hence in a better link utilization (refer to Section 1.1.3).

4.3.2 Bandwidht-Delay-Product

In order to possible reach a link utilization of 100%, TCP's effective window has to grow beyond the bandwidth-delay-product (BDP). The reasoning is quite simple: TCP has to inject its last allowed data packet just at the time it receives an acknowledgment for the first inflated packet (of its current effective transmission window). Depending on the satellite constellation, the delay might be higher than 500 ms (which is approximately the single-hop RTT for a GEO satellite) [5, 18, 26, 34] or the bandwidth may reach several Mbit/s for prospective satellite systems supporting multimedia traffic. [22] If the advertised receive buffer is not large enough, optimal link utilization is not possible. Additionally, the range of sequence numbers to acknowledge received data might not be large enough.

The problem coming along with large BDPs is not specific to a satellite environment but ubiquitous in all high performance networks. Jacobson et al discuss this problem in RFC 1072 and show how it can be solved, e.g., by TCP's window scaling mechanism or the PAWS (protections against wrapped sequence numbers) algorithm. [51]

4.3.3 Variable Delays

TCP has to estimate the current round trip in order to properly retransmit any possibly lost segments. If this estimation is not accurate, TCP might wait too long before it retransmits lost segments, or, in the worst case in which the RTO actually falls below the current RTT, conducts unecessary retransmissions. The propagation usually changes in a satellite environment apart from GEO constellations. Even though publications on how TCP's timer behaves in such a variable delay environment [14, 35] are limited, they show that TCP is, at least for a standard timer granularity of 500 ms, well capable to reflect satellite specific variable propagation delays.

4.3.4 Asymmetry

Literature usually distinguishes two kinds of asymmetry [4, 31, 44, 54, 62] which focus either in the link's capacity or its physical characteristics with respect to the associated delay:

1. bandwidth asymmetry and
2. asymmetric propagation delay of the forward and reverse channel.

The first is a frequent phenomenon as the satellite's downlink provides usually a high bandwidth whereas the uplink, especially for mobile terminal who are limited in their transmission power, lower bit rates are common. The second asymmetry might occur, e.g., if a user employs a satellite system to broadcast data to users using a low cost, dial-up channel to acknowledge the reception.[7]

[7]The discussion of both aspects is out of scope of this work; the interested reader is referred to [4, 31, 44, 54, 62] for further information on this matter.

5 Simulation of TCP over a LEO Satellite Link

This chapter focuses on the simulation of TCP over a erroneous satellite LEO satellite link as encountered within the ATM-Sat project. Aspects of the simulation include the effect of TCP's timer granularity G on both: TCP's ability to reflect the variable propagation delay of the satellite channel and the effect of a hand-tuned timer granularity on TCP's throughput performance in the presence of transmission errors. Therefore, the chapter is structured into three parts:

Section 5.1 presents the framework of this thesis, i.e. it presents the ATM-Sat project [6] and its aspects which influenced the simulation of TCP. Based on the project's constrains, the scope of the final simulations is defined. In Section 5.2, the reader is introduced to the applied simulation workflow, starting from the analysis of the satellite constellation over the definition of an appropriate simulation model of the link, up to the actual TCP evaluation. Finally, the simulation models are explained in all their details (Section 5.3) and the parameters used in conjunction with these models are outlined (Section 5.4). Finally, Section 5.5 summarizes the simulation results and evaluates them.

5.1 SCOPE AND CONTEXT OF CONDUCTED SIMULATIONS

The simulation of TCP over a LEO satellite link is embedded within the ATM-Sat project which "provides methods for a new ATM-based communication technology for Low Earth Orbit (LEO) satellite systems". [6] Thus, the scope of the simulation and constrains limiting the TCP flavors to be analyzed derive directly from the project requirements.

The ATM-Sat project studies and designs an entire LEO, ATM-based satellite network to support multimedia applications in a mobile environment. Besides the analysis of the IP-protocol suite, tasks include the definition of an appropriate satellite constellation including inter satellite links and the development of a MAC layer protocol supporting QOS guarantees. The final project stage envisioned a proof on concept prototype of the system.

An aspect of the project is the possible commercial deployment of the system which directly influences the constrains applied to the analysis of the IP-protocol stack. With respect to an assured data transfer, the following assumptions apply:

- Only commercial, off-the-shelf operating systems are run on the (end) user terminal. Thus, a satellite specific protocol stack is not available for end-user-applications.

- The end-to-end semantic of TCP has to be uphold, i.e. the user shall have the opportunity to encrypt its data right above the network layer without suffering performance decrease.

These constrains directly limit the scope of the simulation.

The simulation of Vanilla TCP and TCP Reno as the most widely spread TCP flavor is chosen. Even though TCP SACK is very frequently evaluated when it comes to TCP-related performance issues, it is not yet the state-of-the-art implementation (as of the time of the simulation runs) used by default on a commercial off-the-shelf operating system. This limitation obeys to the first constrain. The second restriction eliminates so called performance enhancing proxies which usually destroy the end-to-end semantic of TCP or cannot cope with data encryption right above the network layer.

Providing an assured data transfer by an application on top of UDP is not considered either. Even though UDP is not as prone to satellite specific network aspects as TCP (refer to Section 4.3) as it does not employ any kind of flow control or retransmission algorithms, the functionality to acknowledge received data had to be implemented on top of UDP which equals either an adapted protocol stack or an entire new user application.

These constrains result in a performance analysis of TCP Reno and Vanilla TCP with respect to parameters that can be tuned at the end-system without modifying TCP's algorithms (thus maintaining a standard-conform protocol stack). The parameters include TCP's timer granularity G and the available / advertised buffer size. In particular, the following questions are subject of the work:

1. Does reducing the timer granularity G significantly improve TCP performance?
2. Is TCP's timer capable to reflect a variable link delay?
3. How does the advertised receive buffer size effect TCP's RTT approximation and performance?
4. How does the target system specific BER effect these aspects?

As to the authors knowledge, question number one has so far not been discussed for a variable delay satellite environment. TCP's ability to reflect a variable link delay in a satellite environment has only been partially answered by Allmann [14] as only a standard timer granularity of $G = 500ms$ is evaluated. The effects of an decreased G in order to (possibly) increase TCP's performance has not been studied so far for satellite systems. These aspects are finally put into the context of different advertised buffer sizes as the latter have usually an influence on queuing delays and might therefore have an influence on results with regard to question one and two.

The analysis is carried out using a simulation based approach as neither an implementation of the MAC protocol (whose development was also part of the ATM-Sat project) nor a project specific demonstration / simulation environment was available. In addition, focusing on a simulation driven approach does not restrict analysis to happen in real-time. Instead of several weeks, the entire set of simulations was finished after a few days (of continuous simulation runs).

5.2 SIMULATION WORKFLOW

The performance evaluation of TCP over a LEO satellite channel breaks down into two main simulation steps: analyzing the target system's satellite constellation and simulating TCP over a specific satellite link. Accordingly, the simulation workflow incorporates two specialized simulation tools, namely STK and Opnet Modeler. Figure 5.1 illustrates the resulting simulation workflow which includes auxiliary tools. The latter allow an automatic exchange of satellite channel descriptions between STK and Opnet. [32, 34] The extension of this approach into an *Integrated Prototyping and Simulation Architecture for Space Specific Protocol Developments and Verifications* is well documented by Emmelmann [36] and not further discussed in this contents.

The simulation process starts with STK which is used to model the network in the sky focussing on the mere constellation of each satellite. Parameters of the target system[1], i.e. the orbit type and altitude (Walker 72/12, $h = 1350\,km$), are directly fed into the simulator in order to visualize the LEO satellite network. Based upon this model, STK analyzes the satellites' constellation with respect to the time-varying distance between two satellites or between a satellite and an earth station, the change of the elevation angle over the time, and the visibility of a particular satellite as seen from a ground station. The

[1]Please refer to appendix D for a complete list of the target system's parameters.

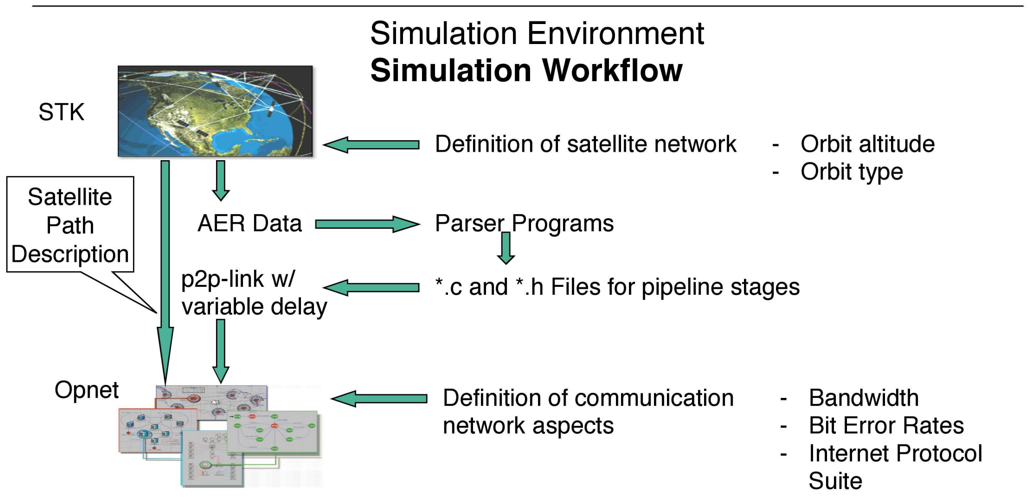

Simulation Environment
Simulation Workflow
STK
Definition of satellite network
- Orbit altitude
- Orbit type
Satellite Path Description
AER Data
Parser Programs
p2p-link w/ variable delay
*.c and *.h Files for pipeline stages
Opnet
Definition of communication network aspects
- Bandwidth
- Bit Error Rates
- Internet Protocol Suite

Fig. 5.1 Simulation Workflow and Interaction of used Toolsets [32,34])

simulation results can be retrieved from STK as an ASCII file containing the AER description (azimuth, elevation, and range) of a specific communication path.

Following the simulation workflow, a C-based parser program automatically converts STK's AER description of a distinct communication path into corresponding c- and h-files which can later on be included in a modified stage of Opnet's transceiver pipeline. The modified pipeline stage allows a point-to-point based (wired) communication link between two Opnet nodes to behave as if it were the satellite channel previously analyzed with STK.[2] The interested reader is referred to Appendix E for the source code and a short description of the developed conversion program.

The actual simulation of TCP over the LEO satellite channel is conducted with Opnet. Channel characteristics are encapsulated in a modified point-to-point link imposing a time-varying delay on data packets. Additional simulation parameter, e.g. the bandwidth and profiles of data to be transmitted via TCP, and the bit error characteristics of the transmission path, are directly fed into Opnet. Finally, this model of a point-to-point, TCP-based communication link is used to compare the performance of different TCP flavors over an erroneous LEO satellite channel.

5.3 SIMULATION MODELS

The following section will describe the simulation models developed within this work. The models are either composed out of available standard components shipped with the corresponding simulation tool or entirely developed from scratch.

Section 5.3.1 describes the model used along with the Satellite ToolKit STK to analyze the satellite constellation as proposed within the ATM-Sat project. Based on this model, information, especially with respect to the range in between two satellites and a satellite and an observer on earth, are gained. Further on, the accuracy of the delay model is evaluated which yields to a simplified delay model. Finally, the simplification is mathematically proven to be valid with respect to the scope of the analysis of TCP in a variable delay environment.

[2]It should be noted that STK provides a native interface to Opnet which is restricted to the exchange of orbit descriptions of all modeled satellites. Even though this approach allows Opnet to simulate a satellite network in its whole, it requires the usage of wireless communication links within the Opnet model. The wireless link pipeline stage in turn requires to write a new model for error allocation and correction, channel interference, and antenna characteristics. These pipeline stages can be omitted with an appropriate analysis of the satellite channel which is later on reflected in the error characteristics of the simulated communication path. These circumstances and the fact that the final simulation of TCP includes only a point-to-point communication plead for a modified point-to-point link.

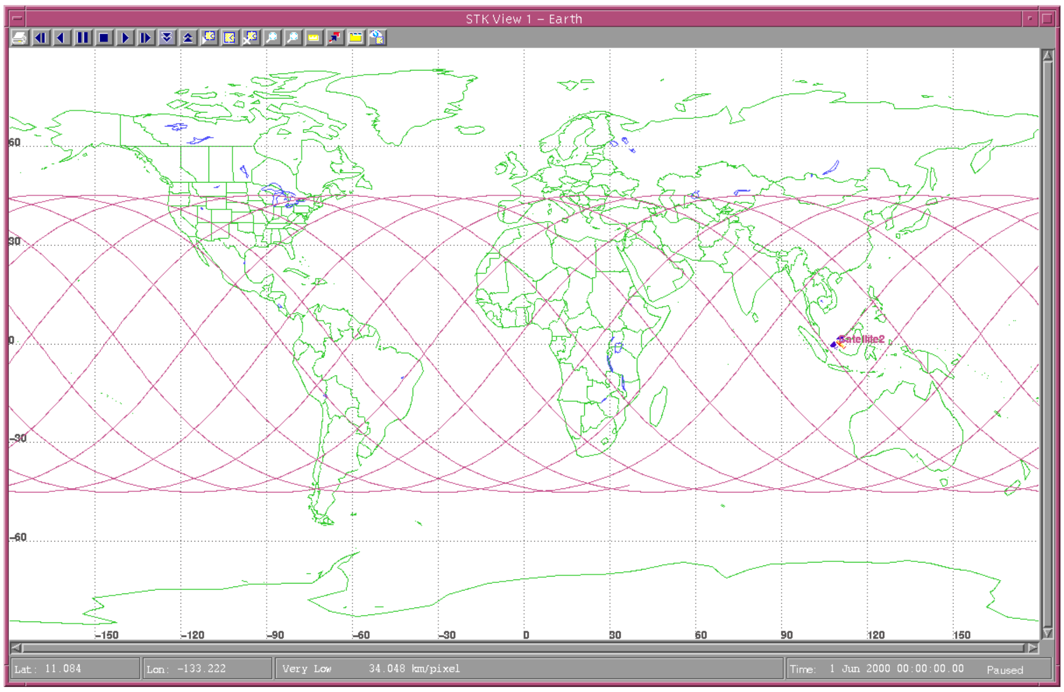


Fig. 5.2 Path Tracks of considered LEO Satellite System

Section 5.3.2 provides an analytical delay model based on the orbit geometry and compares this delay model with the one gained by STK based simulations.

Section 5.3.3 presents the simulation models used in conjunction with Opnet. The representation of the satellite channel and the models to actually analyze TCP over an erroneous, variable delay satellite channel are explained. Additionally, the reader is introduced to auxiliary simulation models which are used to verify the proper functionality of the overall simulation system in order to gain confidence in the simulation results.

5.3.1 STK Model

Model for AER Measurements The STK model describes the time-variant position of each satellite in its orbit. The model is based on the following target system parameters:

- The LEO satellite communication network consists of 72 satellites.
- The satellites are evenly distributed among 12 orbits.
- Each orbit is inclined by $47°$ and has an hight of 1350 km.

Figure 5.2 illustrates the path tracks of the resulting Walker 72/12 constellation. Based on this model, STK determines the intra- and inter-orbital distance to a satellite's next neighbor. In addition to the satellites, the model includes a ground facility (*Facility1*) which is used to analyze the time-varying distance between a satellite and a potential observer on earth.

Figure 5.3 illustrates all possible ISLs and highlights those deriving from a single satellite. Satellite *LEO063* (i.e. satellite number 3 in plane number 6) has two ISLs in the same orbit: to satellite *LEO062* and *LEO064*. The distance reported by STK is constant (7727.985 km) which is an obvious result due to the circular shape of each orbit. The range of the four ISLs to the satellites in the next adjacent planes, i.e. to satellite *LEO072*/*LEO073* and *LEO052*/*LEO053*, changes over the time from 3477.147 to 4482.404 km.

Whereas a satellite sees its next neighbors 24-hours a day, the communication path between a satellite and a potential observer on earth is characterized by its limited availability. The satellite overpasses the ground station three times a day and, in conjunction with a minimum elevation angle of $20°$ as

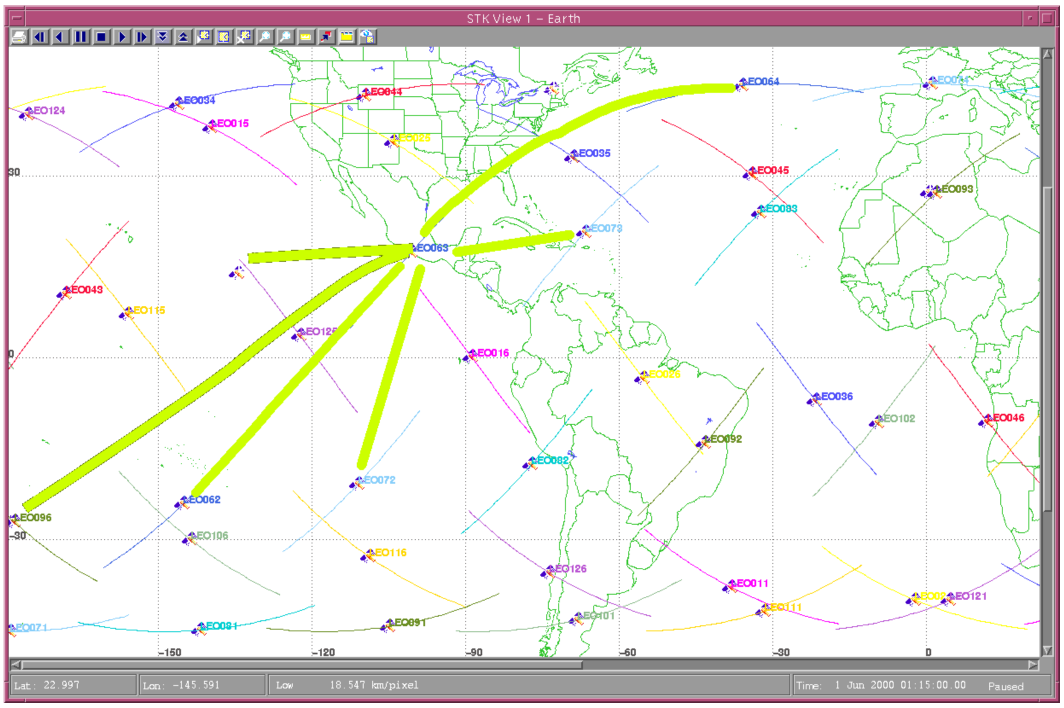


Fig. 5.3 ISLs to next neighbor in the same and left- and rightmost orbit plane

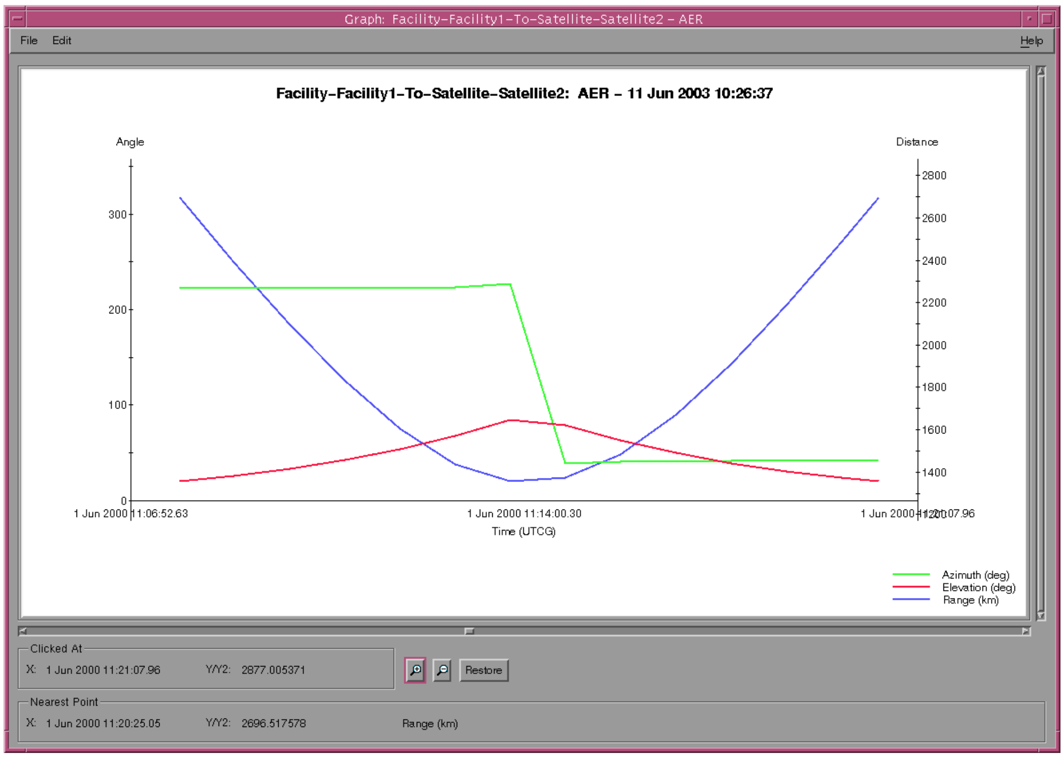
required by the target system specification, the access period is restricted to 12 minutes. After this time, the ground terminal has to invoke a handover to the next available satellite. In a worst case scenario, which is supposed to be a handover scheme resulting in the highest possible variation of the slant range, the ground terminal invokes the handover when the satellite disappears at the horizon and switches to the next upcoming satellite. As this kind of handover scheme results in a periodic repetition of the experienced slant range, further simulations abstract from the actual handover and merely consider the periodically repeated slant range characteristics of a single satellite's overpass. The measured AER (Access Elevation Range) data for a potential observer placed in the satellite's nadir is plotted in Fig. 5.4. The plot is restricted to a single access interval and shows how the slant ranges varies over the time in between 2111.884 and 2698.502 km.

Accuracy of the Model and the Resulting Propagation Delay Two aspects influence the accuracy of the measured AER data: the fact that STK reports AER data only every minute (of simulated real time) which in turn imposes an uncertainty about the exact value of the current AER information in between two measurements, and the improper manual placement of the ground facility in the satellite's nadir. The latter influences only the AER measurements of the up- and down-link delay and cannot be numerically characterized whereas the influence of the measurement's granularity of one minute imposes the following error:

Let r_t be the reported slant range at time t, $\triangle r_{t+\epsilon}$ be the maximum deviation of all $r_{t+\varepsilon}$ from r_t within $[t, t+1]$, and $\delta r_{t+\epsilon}$ be the maximum relative deviation (see Fig. 5.5). Then, according to the fact that r_t is strictly monotonic over $[t, t+1]$, the deviation and slant range is given by

$$\triangle r_{t+\epsilon} = r_t - r_{t+1} \tag{5.1}$$
$$r_{t+\epsilon} \in [r_t - \triangle r_{t+\epsilon}, r_t] \tag{5.2}$$
$$\delta r_{t+\epsilon} = \left| \frac{\triangle r_{t+\epsilon}}{r_t} \right| \tag{5.3}$$

According the reported AER data, the error of the up- and down-link delay has an upper bound of 308.258 km ($\delta r_{t+\epsilon} \leq 14.57\,\%$). The error of the ISL range to the next orbit is below 56.07 km ($\delta r_{t+\epsilon} \leq 1.44\,\%$); the slant range of the ISL within an orbit is constant and thus $\triangle r_{t+\epsilon} \to 0$.



Clicked At
X: 1 Jun 2000 11:21:07.96 Y/Y2: 2877.005371

Nearest Point
X: 1 Jun 2000 11:20:25.05 Y/Y2: 2696.517578 Range (km)

Fig. 5.4 AER Graph for Ground Facility to Satellite Connection

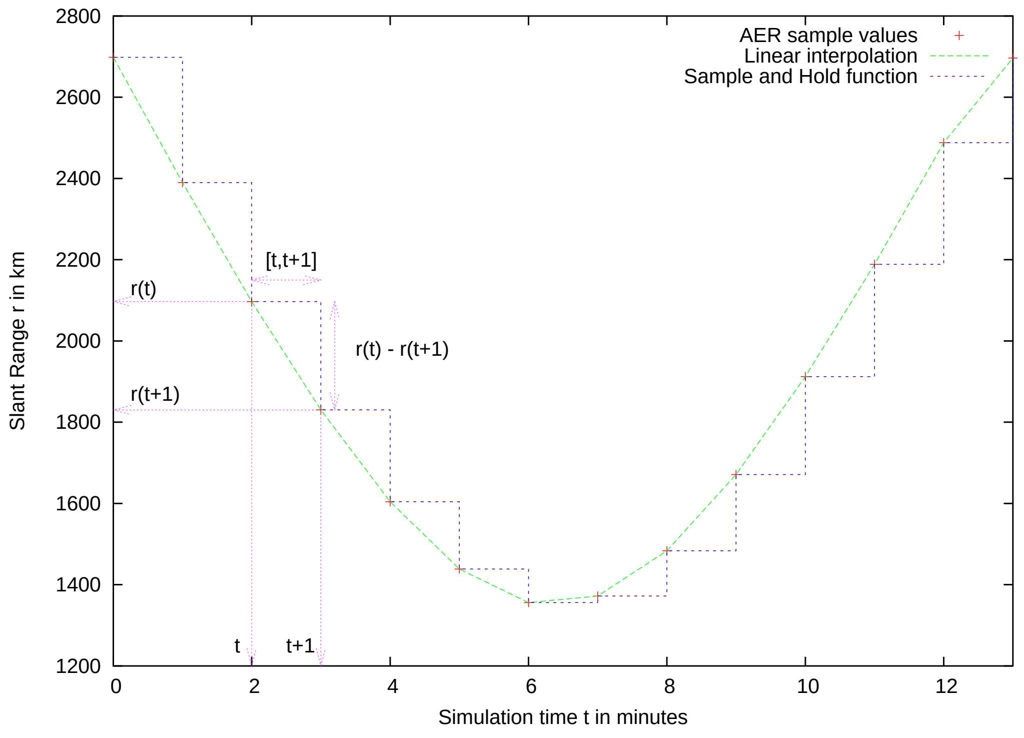


Fig. 5.5 Linear and "Sample and Hold" Interpolation of Slant Range Samples

Based upon the measured slant range, the propagation delay d_t and its relative error are given by

$$d_t = \frac{r_t}{c} \tag{5.4}$$

$$\delta d_t \leq \delta r_{t+\varepsilon} \tag{5.5}$$

where c is the speed of light. Figure 5.6 plots the propagation delay for intra- and inter-orbital ISLs as well as for the up- and down-link. The graph linearly interpolates values of d_t in between t and $t+1$.

End-To-End Propagation Delay of Satellite Link The overall end-to-end delay $d_{end2end}$ is given by the addition of numerous ISL delays d_{ISLs} to the up- and down-link delay d_{up} and d_{down}.

$$d_{end2end} = d_{ISLs} + d_{up} + d_{down} \tag{5.6}$$

Assuming that a signal's transmission starts at the time t and that the signal is routed from satellite A to satellite C via an intermediate satellite B, the transit time from A to B has to be taken into account when the delay for the ISL B–C is computed, i.e.:

$$d_{A\rightarrow B\rightarrow C}(t) = d_{A\rightarrow B}(t) + d_{B\rightarrow C}(t + d_{A\rightarrow B}(t)) \tag{5.7}$$

Measurements show that for any route the ISL delay d has only a neglectful influence on a possible consecutive ISL delay d^*, i.e.

$$\bigwedge_{t,\varepsilon} ((O\,(d_{ISL}(t)) = O(\varepsilon)) \Rightarrow (d^*_{ISL'}(t) \approx d^*_{ISL'}(t+\varepsilon))) \tag{5.8}$$

and hence, Eq. (5.6) can be approximated as

$$d_{end2end} \approx \underbrace{p\, d_{intraISL}}_{=\text{const.}} + \underbrace{q\, d_{interISL} + d_{up} + d_{down}}_{\text{variant over the time}} \tag{5.9}$$

The number of intra- and inter-orbital ISLs p and q depends on how the satellite network routes the connection. If we assume that the network applies shortest-path optimization to its routes, p and q have an upper bound for any satellite network with a Walker constellation. As the target system regularly distributes 72 satellite over 12 orbits, any route may not consist of more than 3 intra-orbital ISLs (reach the satellite in the same plane at the other side of the earth) or 6 inter-orbital ISLs (reach the orbit which is the farthest away).

Simplificaton of the Propagation Delay Model The end-to-end propagation delay given in Eq. (5.9) describes all possible routes containing any pair (p, q) of intra- and inter-orbital ISLs. The simulation of TCP over every possible combination of p and q is by far too complex. Thus, the route which suspectedly results in an end-to-end delay with the most significant influence on TCP (with regard to the delay's variability) is determined for further simulations.

Any ISL within the same orbit adds only a constant delay to $d_{end2end}$. As TCP's timer is well able to reflect constant propagation delays, further simulation models neglect the propagation delay caused by intra-orbital ISLs. The remaining addends are further analyzed with respect to their variability to exploit their possible affects on TCP.

The frequency of the inter-orbital ISL delay is by one magnitude smaller than the one of the up- and down-link ($f_{interISLdel} = 0.297619$ mHz as compared to $f_{upDel} = f_{downDel} = 1.388889$ mHz) and may at least be neglected if the satellite link only consisted of a single inter-orbital ISL. This simplification also holds with regard to the delay's derivation as Fig. 5.7 illustrates.

The further discussion will show that the inter-orbital ISL delay may be neglected with regard to the up- and down-link delay as any combination of inter-orbital ISL delays

- does not significantly increase the maximum derivation of the resulting delay and
- does not increase the frequency of the resulting delay compared to each addend.

Even though the inter-orbital delay is in the same order as the up- and down-link delay ($O(d_{interISL}) = O(d_{up})$) and might even outscore the latter for several consecutive inter-orbital ISL links, it can be ignored for an analysis of TCP with its focus on the influence of the delay's variability.

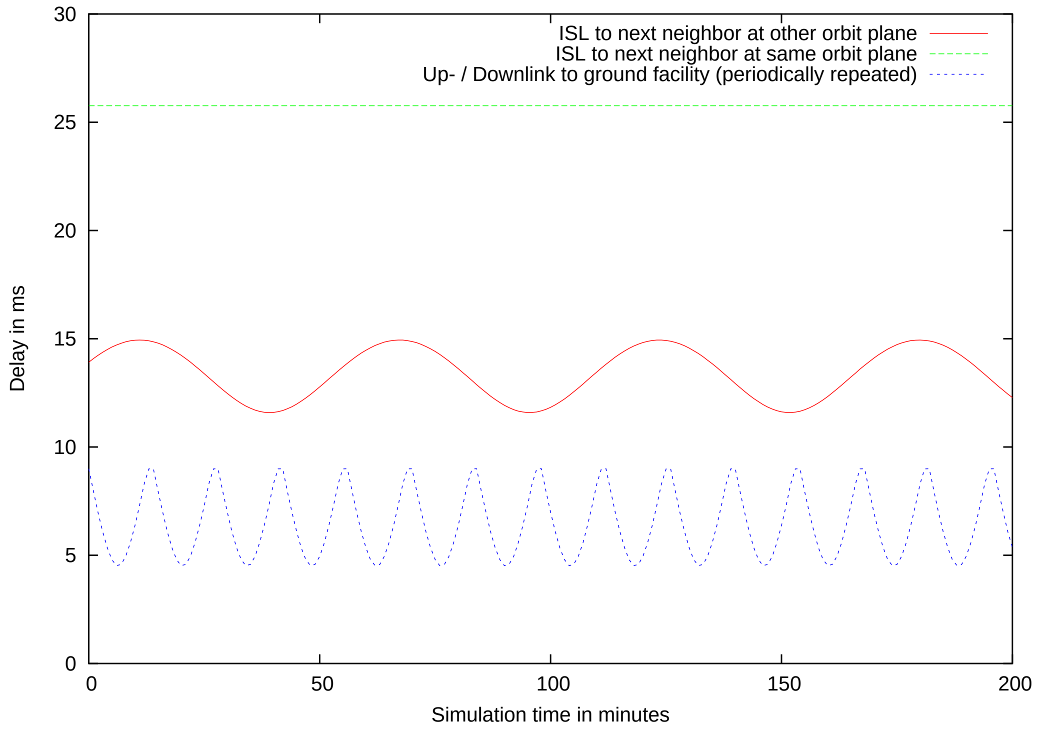


Fig. 5.6 Propagation Delays (top to down: intra-plane ISL, inter-plane ISL, Up- / Downlink)

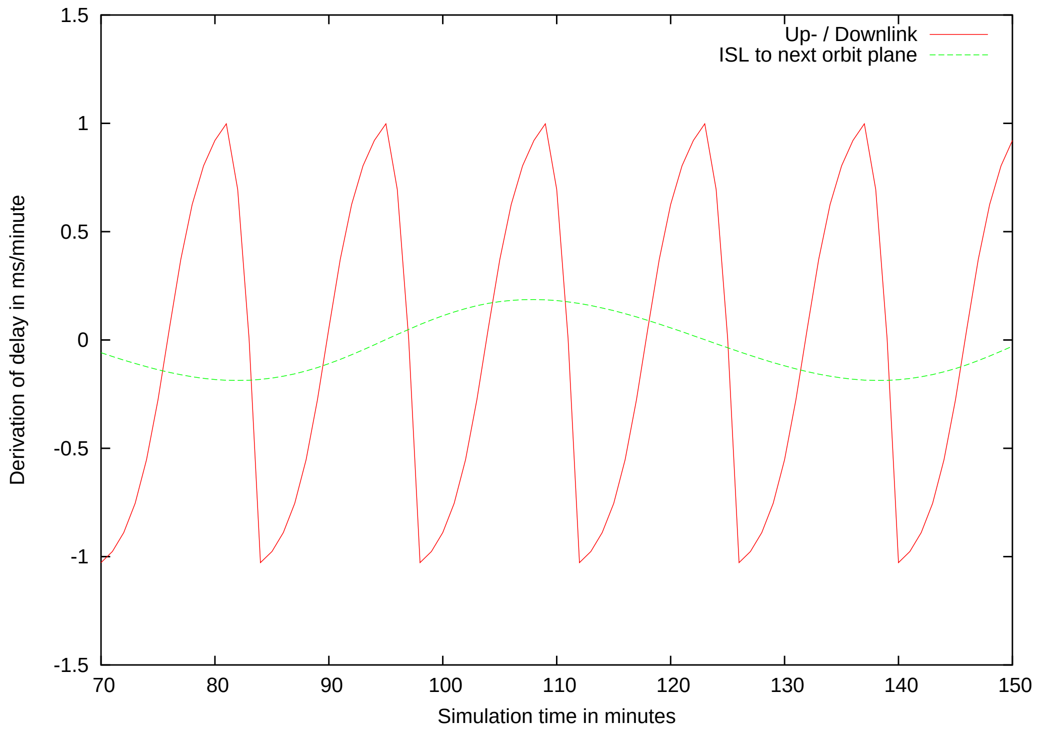


Fig. 5.7 Derivation of Propagation Delay for the Up- / Downlink and ISL to the next Orbit Plane

The derivation of the sum of two inter-orbital ISL delays[3] d and d^* is given by [4]

$$
\begin{aligned}
d &\equiv d_{interISL} \\
d_{sum} &\overset{def}{=} d + d^* && (5.10) \\
d'_{sum} &\overset{def}{=} d_{sum}\,\frac{d}{dt} = (d+d^*)' = d' + d'^* && (5.11)
\end{aligned}
$$

Hence, the highest possible maximal derivation $\max d'_{sum}$ occurs if the cross correlation function ($r_{dd^*}(\tau)$) reaches its maxima.

$$r_{dd^*}(\tau) \overset{\text{def}}{=} \int_{+\infty}^{-\infty} d(t)\, d^*(t+\tau)\, dt \qquad (5.12)$$

This is true if the phase shift φ is according to Eq. (2.29) a multiple of the delay's period, i.e. if d and d^* directly overlap each other:

$$\bigvee_{\tau} \bigvee_{n \in \mathcal{N}_0} \left((r_{dd^*}(\tau) = \max r_{dd^*}(\cdot)) \Rightarrow (\varphi := \tau = n\, T_{interISL} = n \frac{T_{orbit}}{2}) \right) \qquad (5.13)$$

The relation between $T_{interISL}$ and T_{orbit} is given by the nature of the inclined Walker orbit itself: Each orbit has two intersections with its next neighbor orbit (one for the ascending and one for the descending node). Thus the inter-orbital ISL delay's period is only half of the orbit period.

Eq. (5.13) also holds if instead of the delay d and d^* the delays' derivations d' and d'^* are considered. As the number of of inter-orbital ISLs is strictly limited by $q \leq 6$, the relative change of the resulting overall delay does not dominate over the up- and down-link delay variation:

$$(q \leq 6) \Rightarrow O\left(q * \frac{d'_{interISL}}{d_{interISL}} \right) \leq O\left(\frac{d'_{up}}{d_{up}} \right) \qquad (5.14)$$

Figure 5.8 illustrates this fact: In logarithmic scale, adding several derivations of delay functions which are in phase, results in an upward shift of the function itself.

The amplitude spectrum of the frequency of the overall inter-orbital ISL delay ($|\mathcal{F}\{d_{interISLsum}(t)\}|$) is independent of the addends' phase shift (φ_i) except for a factor i representing the weight of each spectral frequency. The spectral analysis [61] is given by:[5]

$$
\begin{aligned}
d_{interISLsum}(t) &= \sum_i d_i^*(t) = \sum_i d(t-\varphi_i) && (5.15) \\
\mathcal{F}\{d_{interISLsum}(t)\} &= \mathcal{F}\left\{ \sum_i d(t-\varphi_i) \right\} && (5.16) \\
&= \sum_i \mathcal{F}\{d(t-\varphi_i)\} \\
&= \sum_i \mathcal{F}\{d(t)\}\; e^{-j\omega\varphi_i} \\
&= \mathcal{F}\{d(t)\} \sum_i e^{-j\omega\varphi_i} \\
|\mathcal{F}\{d_{interISLsum}(t)\}| &= \left| \mathcal{F}\{d(t)\} \sum_i e^{-j\omega\varphi_i} \right| && (5.17) \\
&= |\mathcal{F}\{d_{interISLsum}(t)\}| \cdot \left| \sum_i e^{-j\omega\varphi_i} \right|
\end{aligned}
$$

[3]The discussion focuses on two inter-orbital addends d and d^* for simplicity. If the satellite link contains more than two inter-orbital ISL, we can perform an iteration starting with the proof for two addends d and d^* which results in a function $d_{sum} := d + d^*$. The result d_{sum} can than be compared to d^{**} etc.

[4]For simplicity, we abbreviate $d_{interISL}$ with d in some formulas.

[5]$\mathcal{F}\{d(t)\}$ denotes the Fourier transformation of $d(t)$ which is given by $\mathcal{F}\{d(t)\} := \int_{-\infty}^{+\infty} d(t) e^{-jwt}\, dt$

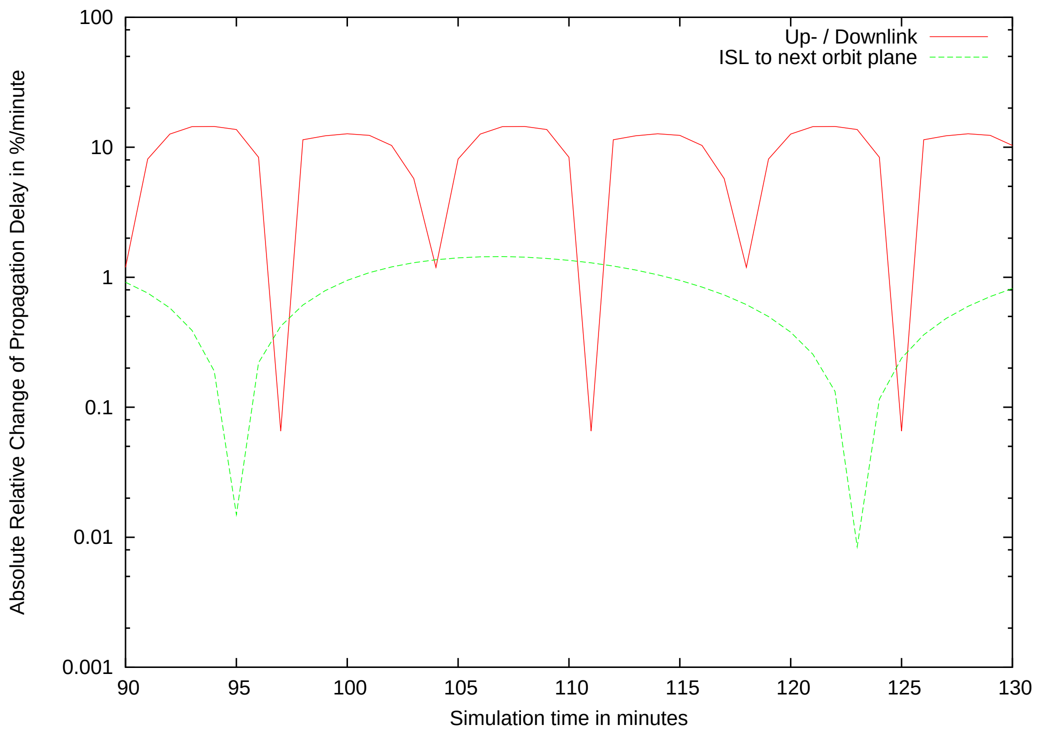


Fig. 5.8 Absolute Relative Variation of Propagation Delay $d'(t)/d(t)$

$$\begin{aligned}
&\leq \left|\mathcal{F}\left\{d_{interISLsum}(t)\right\}\right| \cdot \sum_i \left|e^{-j\omega\varphi_i}\right| \\
&= \left|\mathcal{F}\left\{d_{interISLsum}(t)\right\}\right| \cdot \sum_i \left|\cos(\omega\varphi_i) - j\sin(\omega\varphi_i)\right| \\
&= \left|\mathcal{F}\left\{d_{interISLsum}(t)\right\}\right| \cdot \sum_i \underbrace{\sqrt{\cos^2(\omega\varphi_i) + \sin^2(\omega\varphi_i)}}_{=1} \\
&= i\left|\mathcal{F}\left\{d_{interISLsum}(t)\right\}\right| \qquad (5.18)
\end{aligned}$$

The frequency of $d_{interISLsum}(t)$ cannot be larger than the largest frequency of all contributing harmonics. Thus, the frequency of the overall inter-orbital ISL is according to the measurements at least one magnitude smaller than the frequency of the up- and down-link delay.

$$\bigwedge_{\varphi_i} O(f_{interISLsum}) \leq O(f_{interISL}) < O(f_{up}) \qquad (5.19)$$

As a final result, Eq. (5.14) and (5.19) prove that the delay model can be reduced to the mere up- and down-link delay if the affects of the variability of the delay on TCP performance is the subject of further simulations.

5.3.2 Analytical Orbit Model

For validation purposes, the simplified STK model is compared to an analytical model of the up- and down-link delay. The analytical model directly results from the geometry of the given Walker constellation (ref. to Section 2.1.2) and returns the propagation delay associated with a link as a function of the simulation time. As the visibility of a satellite is limited to approximately 12 minutes, the delay is derived from the slant range as given in Eq. (2.23) and periodically repeated.[6] This models approximates the handover

[6]The satellite is assumed to pass through the observer's zenith at $t = 6\,min$.

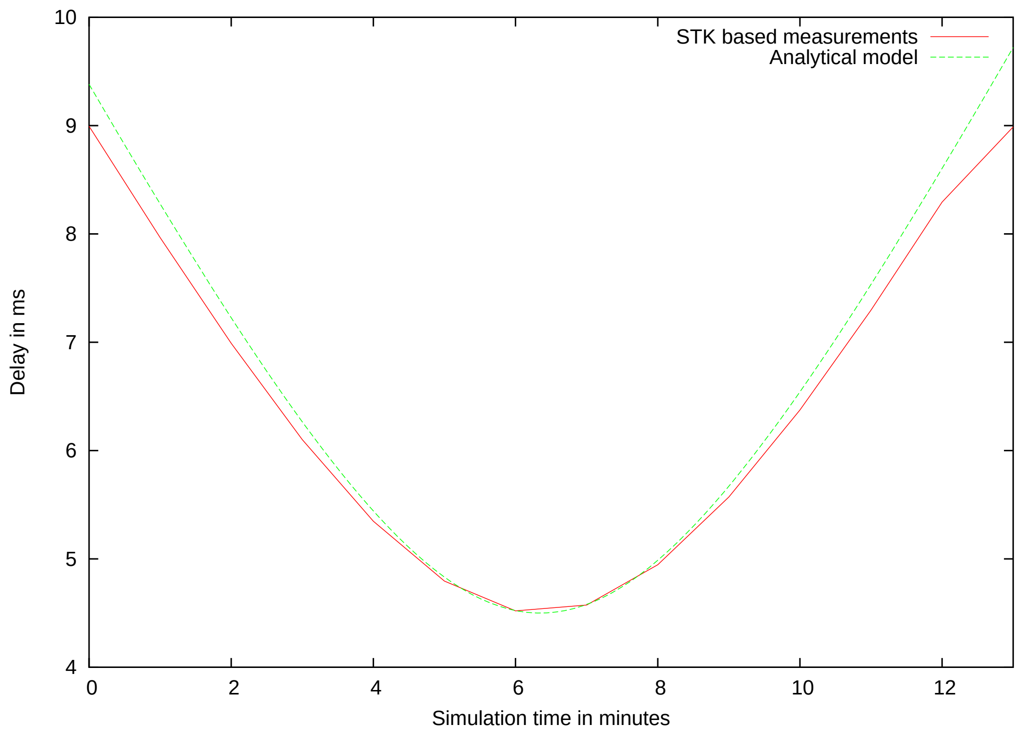


Fig. 5.9 Comparison of Analytically Gained Up- and Downlink Delay with STK based measurements

which occurs when one satellite disappearing at the horizon passes all its connections to the satellite which is just about to appear in the ground station's visibility range. The analytical delay is therefore mathematically described by

$$delay(t) = \frac{\sqrt{R_e^2 + r^2 - 2R_e r\cos\left(\frac{2\pi}{12}((t \bmod 12) - 6)\right)}}{c} \tag{5.20}$$

Figure 5.9 plots the analytical delay model against the STK based delay measurements. Both models return delays which differ less than 7% at any time t. This is almost half of the relative error calculated in Eq. (5.3) for the up- and down-link delay derived from STK. The relation between the two results is given by the fact that Eq. (5.2) is based on a "sample and hold" approximation of intermediate delays whereas the plot in Fig. 5.9 linearly interpolates in between the samples gained from STK's AER analysis. Additionally, STK might employ a rather complete simulation model considering more than the mass of the earth and the satellite, i.e. STK might not limit the analysis to a simple two-body problem as done for the analytical approach.

Lemma: Eq. (5.20) defines the periodic repetition of the delay signal based on the analytical model. As STK's AER analysis reports a sample every minute, we obtain at least 12 samples of STK's delay approximation and can thus use these samples to guess intermediate values. For simplicity, either a "sample and hold" method or linear interpolation is applied. This approximation is sufficient as a better interpolation based on a Fourier series does not yield in significant improvements. The interested reader is referred to Appendix 1 for further details. •

5.3.3 Opnet Models

The evaluation of TCP over an erroneous, variable delay LEO satellite channel is carried out with the Opnet Modeler simulation tool. General modeling concepts and terminologies were previously introduced in Section 3.2.2 which leaves the focus of this section on the simulation models which were developed as a part of this thesis, namely a model describing the satellite channel (Section 5.3.3.1) and a model which is actually used to analyze TCP's performance (Section 5.3.3.2).

The model of the satellite channel consists of a modified point-to-point link imposing the characteristics of a satellite channel on data being forwarded. The paragraph reasons why the simulation may abstract from a wireless link model and explains the modifications of the standard (wired) pipeline stage procedures. Finally, two network nodes which can be used to verify the proper functionality of the modified pipeline stages are introduced.

The simulation model used to analyze TCP will be shown to consist of standard network components (especially network nodes) which are connected via the new developed satellite link model. As the simulation model consists in its whole of several experiments each focussing on a different simulation parameter (e.g. TCP flavor, timer granularity, etc.), experiments are hierarchically grouped in subnetworks.

5.3.3.1 Model of the Satellite Channel Opnet comes with a native interface in order to model satellite channels (ref. to Section 3.2.2): Orbit path parameters for each satellite may be imported from STK thus resulting in "moving network nodes in the sky". These nodes may then be equipped with a wireless transceiver in order to communicate with each other and with (wireless) nodes on the ground. As this modeling approach requires the usage of the radio link transceiver pipeline [3], at least an exact definition of the following network components and characteristics had to be implemented in each corresponding pipeline stage:

- antenna gain of transmitting terminal,
- antenna gain of receiving terminal,
- background noise and interference noise definition,
- error allocation according to the received signal to noise ratio, and
- error correction according to the used FEC scheme.

Additionally, this approach requires an implementation of a "tracking and pointing" algorithm to find a satellite in the sky and to track the latter's movements. The only advantage of this modeling concept is its ability to model an entire satellite network (several nodes in the sky) and to include handovers. This in turn requires an exact description of the underlying MAC protocol in order to gain valuable results with respect to a TCP/IP analysis.

The focus of this simulation is to analyze how prone TCP is to the variability of the propagation delay. STK based simulations of the considered satellite network show that the variability is clearly dominated by the up- and down-link delay. Additionally, the error model described in 2.2.2 provides a constant end-to-end specific bit error rate after applying FEC which does already imply antenna characteristics and channel noise. Therefore, a (wired) point-to-point transceiver pipeline with a modified propagation delay stage is used instead of its wireless counterpart to model TCP connections.

Opnet's Point-to-Point Pipeline Stage Architecture Every packet[7] being transmitted from one node to another is passed to a pipeline stage architecture which simulates the physical characteristics of the transmission channel. For a (wired) point-to-point link, this architecture consists of four stages:

1. transmission delay,
2. propagation delay,
3. error allocation, and
4. error correction

Figure 5.10 illustrates the execution sequence of each different stage.

The transmission delay stage is invoked immediately upon beginning transmission of a a packet, in order to calculate the amount of time required for the entire packet to complete transmission. This result

[7]Even though Opnet speaks of packet oriented communication, a bit- or byte-stream can also be transmitted for a packet length of one bit / byte.

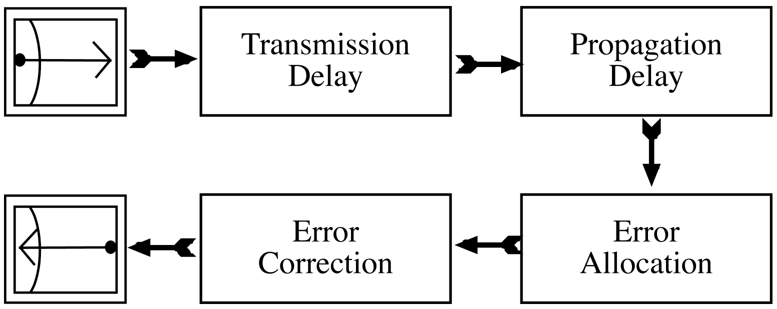


Fig. 5.10 Opnet's Point-to-Point Link Transceiver Pipeline Execution Sequence for one Transmission [3]

can be thought of as the simulation time difference between the beginning of the transmission of the first bit and the end of transmission of the last bit of the packet. Thus, the transmission delay d_{trans}depends only on the packet length l and the channel's bandwidth b:

$$d_{trans} = \frac{l}{b} \tag{5.21}$$

After this time has elapsed, the interface may immediately transmit another pending packet.

The propagation delay stage is invoked after the return of the transmission delay stage with no simulation time in between. The purpose of this stage is to calculate the amount of time required for the packet's signal to travel from the link's source to its destination. As this time depends on the slant range in between ground station and satellite at the time of transmission, this pipeline stage is later on modified to reflect the delay characteristics of the satellite link.

The error allocation stage is called in third place; its execution does not consume simulation time. The error allocation stage estimates the number of bit errors based upon the implemented error scheme and corrupts the packet accordingly. Hence, this pipeline stage will later on embody the error model described in Section 2.2.2.

The error correction stage corrects any corrupted packets and forwards them afterwards to other modules in the receiving node. Thus, this stage depends on the ability of an employed FEC scheme to correct corrupted packets. [3]

Modification of Standard Pipeline Stage Procedures For the simulation of the given satellite channel, the propagation delay stage shipped with Opnet (txdel model) is kept. During the simulation, the standard error allocation model (error model) is initialized to corrupt packets with a constant bit error rate. This method is valid for an appropriately chosen FEC scheme if the ground terminal is not shadowed. Shadowing is not considered, as it cannot be compensated with FEC and would therefore correspond to a link failure making any kind of communication impossible. [38, 39] The error correction stage is simply initialized to a null-function as the error allocation model does already reflect the capability of the implied FEC scheme to correct errors. This leaves the focus on a new implementation of the propagation delay model.

The new point-to-point propagation delay model is able to calculate

- a constant propagation delay based on the distance in between sender and receiver,
- a variable propagation delay based on a mathematical model of the satellite link's slant range, and
- a variable propagation delay based on a previous analysis of the communication path with STK.

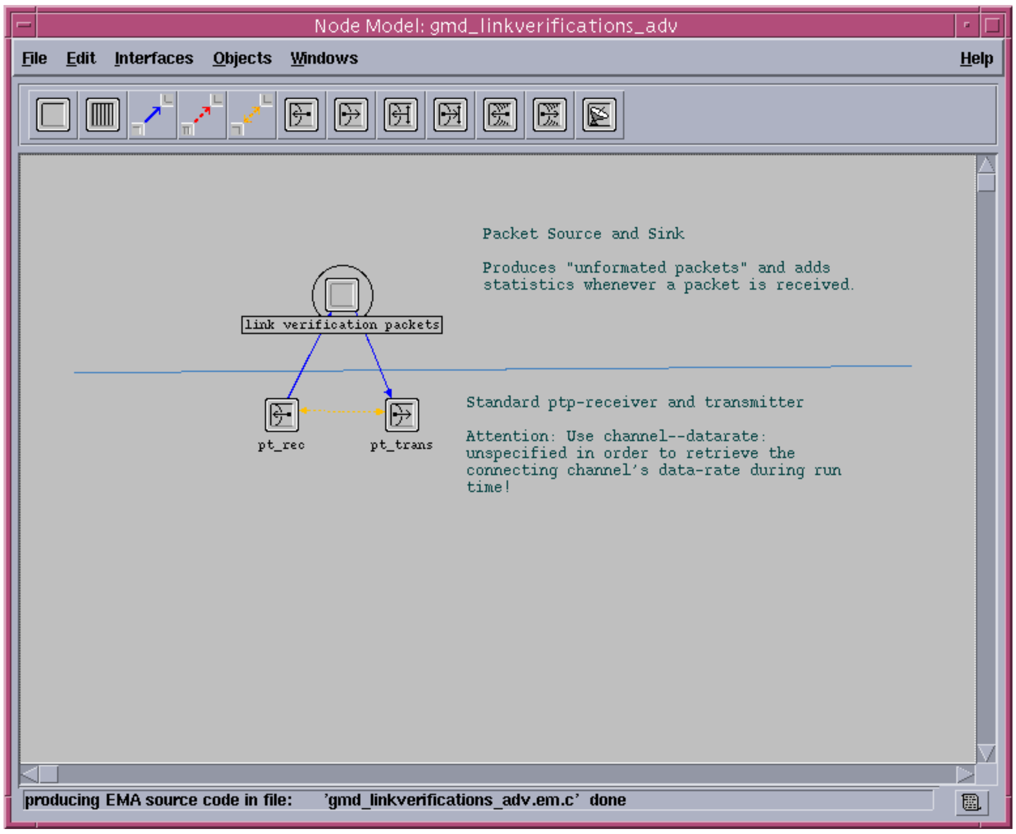
Opnet's simulation kernel requires that the propagation delay stage procedure accepts a packet address as its sole argument. When the propagation delay pipeline stage exits, the simulation kernel expects the computed propagation delay to be assigned to the packet's transmission data attribute (TDA) represented by the symbolic constant OPC_TDA_PT_PROP_DELAY. [3] In order to calculate the propagation delay, the object ID of the link carrying the packet is extracted from the packet's TDA. Afterwards, the link is probed to determine the delay that the user has specified during simulation runtime. The value of the delay may indicate that the pipeline stage has to compute the actual value (i.e. if the value equals to a special constant which was previously defined) or is considered as a fixed delay associated with the link.[8] If the user chooses a non-constant delay type, the pipeline stage model either invokes the standard Opnet function link_delay() to calculate the distance based delay, calculates the delay based on a mathematical model, or calls the function get_stk_delay() to determine the delay according to a STK analysis.

```
#define DISTANCE_BASED    1E+15
#define ISS_MAX_VAR_CST    999999003
#define ATMSAT_LEO        1E+16
void gmd_vardelay_dpt_propdel (Packet *pk) {
    double prop_delay;
    /*
     * extract required information from packet
    */
       /* Obtain object id of link carrying transmission.  */
       link_objid = op_td_get_int (pkptr, OPC_TDA_PT_LINK_OBJID);
       /* Obtain propagation delay associated with that link.  */
       op_ima_obj_attr_get (link_objid, "delay", &prop_delay)
    /*
     * perform calculation of propagation delay (override the
     * delay if it contains a special value)
    */
       if (prop_delay == DISTANCE_BASED) {
          prop_delay = link_delay(link_objid);
       }
       if (prop_delay == ATMSAT_LEO) {
          prop_delay = op_sim_time();
          /* save the "type conversion" difference */;
          x = prop_delay - (long) prop_delay
          prop_delay = (long) prop_delay % 720;
          /* add the difference lost due to type conversion */
          prop_delay += x;
          /* phase shift to assure user in satellite's nadir at t=0)
          prop_delay -= 360;
          x = 2 * M_PI * prop_delay / 6762 ;
          prop_delay = sqrt(6.37*6.37+7.72*7.72-2*6.37*7.72*cos(x))/3E2;
       }
       if (prop_delay == ISS_MAX_VAR_CST) {
          prop_delay = get_stk_delay( op_sim_time(), iss_maxVar );
       }
       /*
        * Other STK based delay patterns are omitted here.
        * The entire source code can be found
        * in Appendix F.
       */
    /*
     * place result in TDA to return to simulation
     * kernel
    */
         op_td_set_dbl (pk, OPC_TDA_PT_PROP_DELAY, prop_delay);
}
```

Some uncommon calculation methods were necessary to determine the propagation delay according to the mathematical model given in Eq. (5.20). Opnet's simulation kernel is unable to perform a modulo operation on a variable of the type double. Therefore, the simulation time has to be converted to type long and afterwards converted back to its original type, i.e. double. The conversion process is not loss-less. Thus, the difference between the original and converted simulation time is saved and in the end added to the result of the modulo operation.

In addition to the variable propagation delay of the considered LEO satellite constellation, the pipeline stage procedure may return the delay of various other satellite constellations based upon previous STK analyzations. These constellations are implemented to verify the simulation model against the results obtained by Allman et al. [14] In order to return a delay according to the AER data as in Eq. (5.5),

[8]The graphical user interface (GUI) hides this handling from the user. The latter may either choose a symbolic name if a variable or distant based delay is wished, or may specify a constant delay associated with the link. As Opnet's simulation kernel is not able to pass symbolic names in between the GUI and the pipeline stage model, the definition of special values for each type of variable delay is needed.


Node Model: gmd_linkverifications_adv
File Edit Interfaces Objects Windows Help
link verification packets
pt_rec
pt_trans

Packet Source and Sink

Produces "unformated packets" and adds statistics whenever a packet is received.

Standard ptp-receiver and transmitter

Attention: Use channel--datarate: unspecified in order to retrieve the connecting channel's data-rate during run time!

producing EMA source code in file: 'gmd_linkverifications_adv.em.c' done

Fig. 5.11 Node Model of the LinkVerification Node

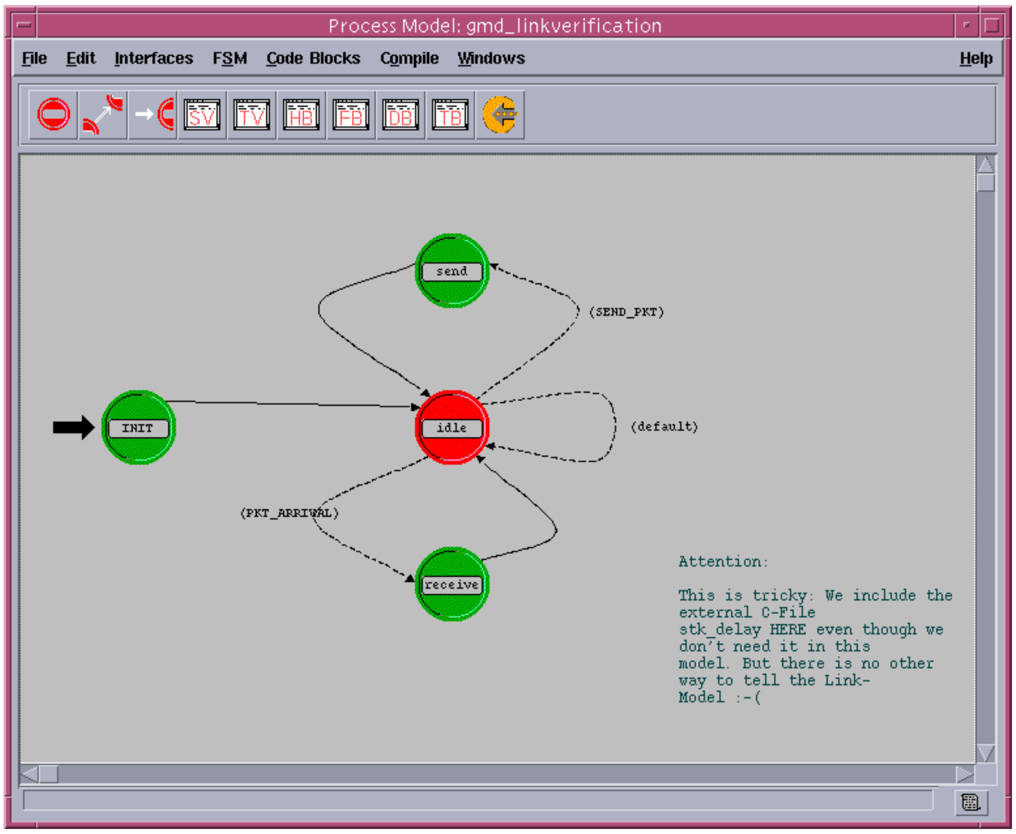
the current simulation time and a structure with the pre-computed delays are passed to the function get_stk_delay(). The latter reduces the simulation time to be within the delays' period and retrieves the current delay from the lookup table. This behavior corresponds to a "sample and hold" interpolation of the pre-calculated delays and is therefore affected with the error given in Eq. (5.3). The source code of the lookup function as well as the definition of the delay tables passed to get_stk_delay() are given in Appendix F.

Supporting Network Nodes for Verificaiton Purposes The implementation of the link model to reflect the variable propagation delay behavior of a satellite channel consists of several development steps and is thus, by nature, prone to implementation error. Therefore, the simulation process includes a method to verify the behavior of the final link model.

In order to test the just developed satellite link, packets with the least possible protocol overhead are forwarded over the link. The delay in between sending and receiving the packet as well as the utilized bandwidth can be measured accordingly. In order to conduct these tests, a network node type *gmd_linkverification* is implemented.

The internal structure of the network node, depicted in Fig. 5.11, consists of a point-to-point receiver and transmitter (pt_rec and pt_trans) which are directly connected to the node's main process. The latter's mere porpuse is to transmit and receive (raw) data packets.[9] The transmission may be characterized during simulation runtime in terms of the packet length, transmission rate, and the duration of packet transmission. The process instantaneously accepts any packet arriving on the receiver. Upon arrival, the packet's creation time stamp is evaluated to calculate the encountered transmission delay and the packet is destroyed.

[9]The term "raw data packet" indicates that the actual format, i.e. the meaning of the bits within the packet, is irrelevant at this time and is therefore not model except for the packet 'slength.



Attention:

This is tricky: We include the external C-File stk_delay HERE even though we don't need it in this model. But there is no other way to tell the Link-Model :-(

Fig. 5.12 Process Model of the LinkVerification Node

Figure 5.12 illustrates the internal stages of the main process. As soon as the simulation starts, the INIT state is entered and internal variables as well as pointers to record any statistics associated with this node are initialized. If the user configured the node to start generating traffic, the first interrupt of the name GEN_NEW_PKT is scheduled.

```
/* Define constands used in the process model */
#define GEN_NEW_PKT   10
/* Function Declarations */
static void gmd_linkverification_sv_init ();
/** state (INIT) enter executives **/
FSM_STATE_ENTER_FORCED_NOLABEL (1, "INIT",
                         "gmd_linkverification () [INIT enter execs]")
    {
    /* Initilaize the statistic handles to keep */
    /* track of traffic sinked by this process.  */
    gmd_linkverification_sv_init () ;
      /* schedule the first packet to be sent */
    /* start_time is set to -1 if we do not want to generate traffic */
    if (start_time >= 0.0)
        {
        op_intrpt_schedule_self (op_sim_time () + start_time, GEN_NEW_PKT);
        }
    }
```

Afterwards, the finite state machine (FSM) enters its IDLE state and waits for an interrupt to occur. The following events may at least trigger an interrupt: the receiver causes an interrupt to indicate a packet's arrival, the node sends itself an interrupt after the timer elapses, or another node directly addresses an interrupt to this node. If an interrupt occurs, the FSM exits the IDLE state and determines upon exit the type and the code of the interrupt. Afterwards, the transition conditions are evaluated to determine the next state. Due to the order in the switch-statement, priority is given to sending packets.

```
/* State transition macro definitions */
```

```
#define   SEND_PKT   (intrpt_type == OPC_INTRPT_SELF
                         && intrpt_code == GEN_NEW_PKT)
#define   PKT_ARRIVAL   (intrpt_type == OPC_INTRPT_STRM)
/** state (idle) exit executives **/
FSM_STATE_EXIT_UNFORCED (3, "idle",
                 "gmd_linkverification () [idle exit execs]")
    {
    /* Determine the type of interrupt.  */
    intrpt_type = op_intrpt_type ();
    intrpt_code = op_intrpt_code ();
    }
/** state (idle) transition processing **/
FSM_INIT_COND (SEND_PKT)
FSM_TEST_COND (PKT_ARRIVAL)
FSM_DFLT_COND
FSM_TEST_LOGIC ("idle")
FSM_TRANSIT_SWITCH
    {
    FSM_CASE_TRANSIT (0, 2, state2_enter_exec, ;, "SEND_PKT",
                       "", "idle", "send")
    FSM_CASE_TRANSIT (1, 0, state0_enter_exec, ;, "PKT_ARRIVAL",
                       "", "idle", "receive")
    FSM_CASE_TRANSIT (2, 3, state3_enter_exec, ;, "default",
                       "", "idle", "idle")
    }
}
```

As the definition of the transition macros shows, the event SEND_PKT occurs if the process sends itself an interrupt and if this specific interrupt is of the kind GEN_NEW_PKT; the receiver indicates a pending packet in its queue with the interrupt type OPC_INTRPT_STRM.

As soon as the FSM enters the SEND state, a new packet is created. The size of the packet is determined by the user's specification at simulation run time. As this specification might be the definition of a packet size distribution with a given probability, the size has to be calculated for each new packet. Finally, the packet is send to the stream connected to this module (i.e. to the transmitter, refer to Fig. 5.11) and a new interrupt is scheduled to create the next packet. Again, the inter-arrival time is determined according to a time distribution specification provided by the user during runtime.

```
/** state (send) enter executives **/
FSM_STATE_ENTER_FORCED (2, state2_enter_exec, "send",
                         "gmd_linkverification () [send enter execs]")
    {
    /* Create a packet using the outcome of the loaded */
    /* distribution.  */
      pk_size = oms_dist_outcome (packet_size_dist_handle);
    pkptr  = op_pk_create (pk_size*8);
      /*  This is the point at which to create statistic about */
    /*  sent packets */
    /* to be filled */
      /* Send the packet on the stream connected to this */
    /* module.  */
    op_pk_send (pkptr, 0);
      /* schedule an interrupt in order to create the next */
    /* packet if the stop time is not reached */
    if ((stop_time < 0.0) || (stop_time  > op_sim_time () ))
        {
        intarrvl_time = oms_dist_outcome (intarrvl_time_dist_handle);
        op_intrpt_schedule_self (op_sim_time () + intarrvl_time, GEN_NEW_PKT);
        }
    }
```

Finally, if a packet has arrived and the RECEIVE state is entered, the process obtains the incoming packet from the stream that caused the interrupt and calculates the size of the received packet as well as the end-to-end delay. Afterwards, the process records local and global statistics (i.e. the end-to-end delay, erroneousness of the packet, etc.) and finally destroys the packet.

```
/** state (receive) enter executives **/
FSM_STATE_ENTER_FORCED (0, state0_enter_exec, "receive",
                         "gmd_linkverification () [receive enter execs]")
    {
```

```
/* Obtain the incoming packet.  */
pkptr = op_pk_get (op_intrpt_strm ());
  /* Caclulate metrics to be updated.  */
pk_size = (double) op_pk_total_size_get (pkptr);
ete_delay = op_sim_time () - op_pk_creation_time_get (pkptr);
  /* Update local and globel statistics.  Source code is omitted here.
     Please refer to Appendix  F */
/* Destroy the received packet.  */
op_pk_destroy (pkptr);
}
```

5.3.3.2 TCP Simulation Model The Opnet simulation model consists of several hierarchically placed network-clouds. These networks are not relevant to the simulation itself but are merely used to group the experiments by their simulation parameters. Figure 5.13 illustrates the network hierarchy: At its top level, the definition and parameter of TCP based applications and their execution profile is placed.This definition is later on simultaneously used by all nodes in this simulation to start TCP-based data transfers. The network could in the top level merely groups all experiments. The first sub level divides the experiments in groups with the same minimum retransmission timeout (RTO_min). For each of this group, the second level sets the size of the TCP connection's receive buffer, and finally, the third sub-level assigns different timer granularities for the specific experiment. Thus, each simulation model contains

$$n_{total} = n_{RTOs} \cdot n_{recBuffer} \cdot n_{timerGran} \tag{5.22}$$

different experiment sets where n_{RTOs}, $n_{recBuffffer}$, and $n_{timerGran}$ denote correspondingly to the number of different RTOs, receive buffer sizes, and timer granularities in each sub-network. As all experiments in a single simulation model refer to the same TCP flavor and all links in the model impose the same bit errors on the traffic, the simulation model is duplicated for each TCP flavor and bit error rate to be evaluated.

Network Nodes The actual simulation takes place in the most bottom network level which contains two client nodes, a server node, and an IP network cloud (Fig. 5.14). The client node is configured to execute the application profile defined in the top-level network hierarchy, i.e. to run the actual experiments. It communicates via a point-to-point link with the network cloud to which the server is connected using the modified pipeline stage architecture representing the LEO satellite link. The IP network could in the middle is necessary in order to connect other computers to the network. These additional computer may start competing TCP connections to evaluate the fairness of TCP over the satellite link.[10] It should be noted that the links in between the IP network cloud and the two clients do not impose an additional delay on the traffic nor does the IP could cause traffic to be lost or corrupted.

Application and Profile Definition The experiment, i.e. the kind of data transfer in between client and server, is defined for all bottom-level networks via the application and profile definition functionality (Fig. 5.15). The application definition contains only a single description, namely *FTPtransfer*. This definition describes a single FTP transfer of a 250 MB file from the server to the client.

The profile definition contains information on how often and at which time a certain application is invoked. For this simulation, a profile is specified which starts the application *FTPtransfer* right after the simulation has started.

5.4 SIMULATION SETTINGS

The scope of the simulations is limited to TCP's ability to reflect a variable propagation delay with its internal timer structures and how a reduced timer granularity G effects this ability. In addition, the simulations should reveal if TCP's performance in terms of throughput can be improved by reducing the timer granularity in an erroneous environment. Besides, the effect of different advertised receive buffer sizes on the RTT measurement and throughput are subject of this analysis. Therefore, the following sections

[10]This simulation aspect is not considered within this work. Therefore the BG_Traffic node is simply disabled.

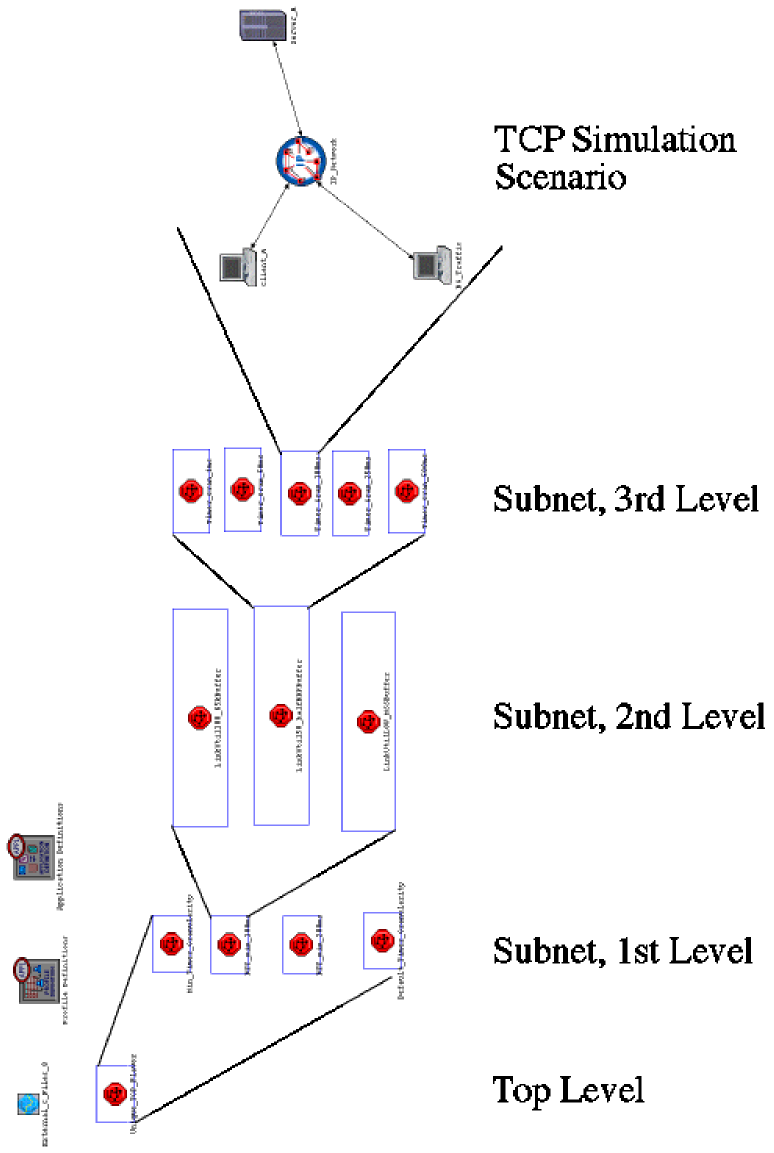


Fig. 5.13 Network Hierarchy of Opnet Simulation Model

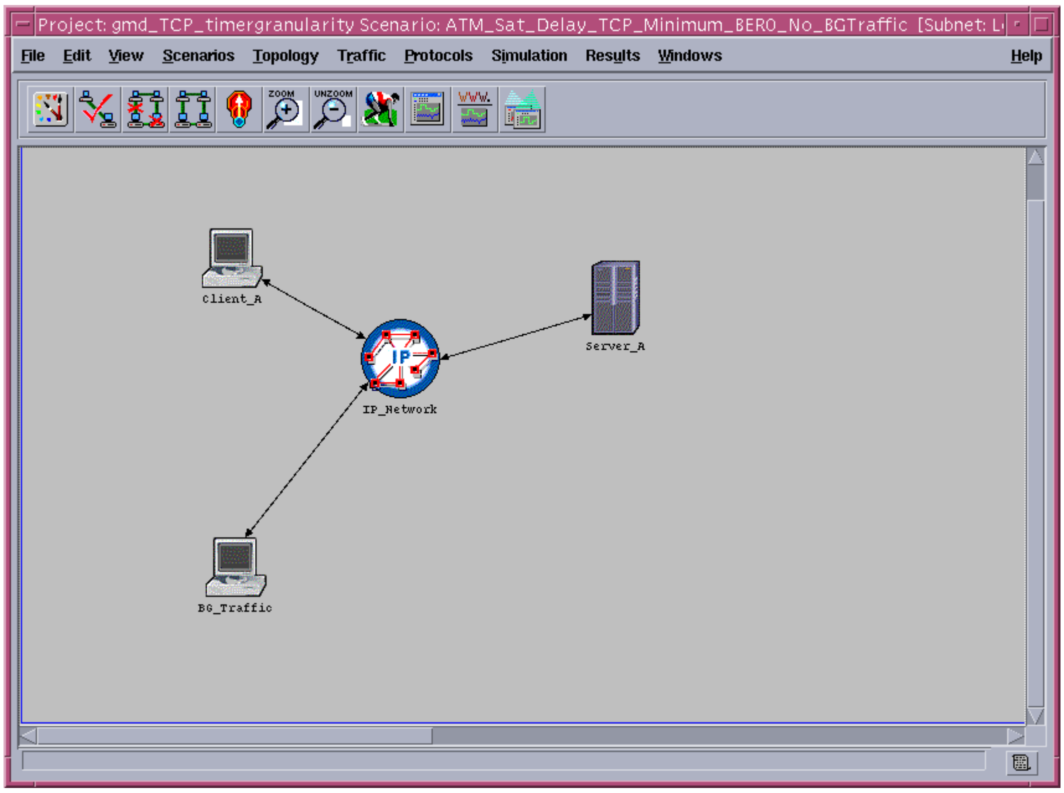
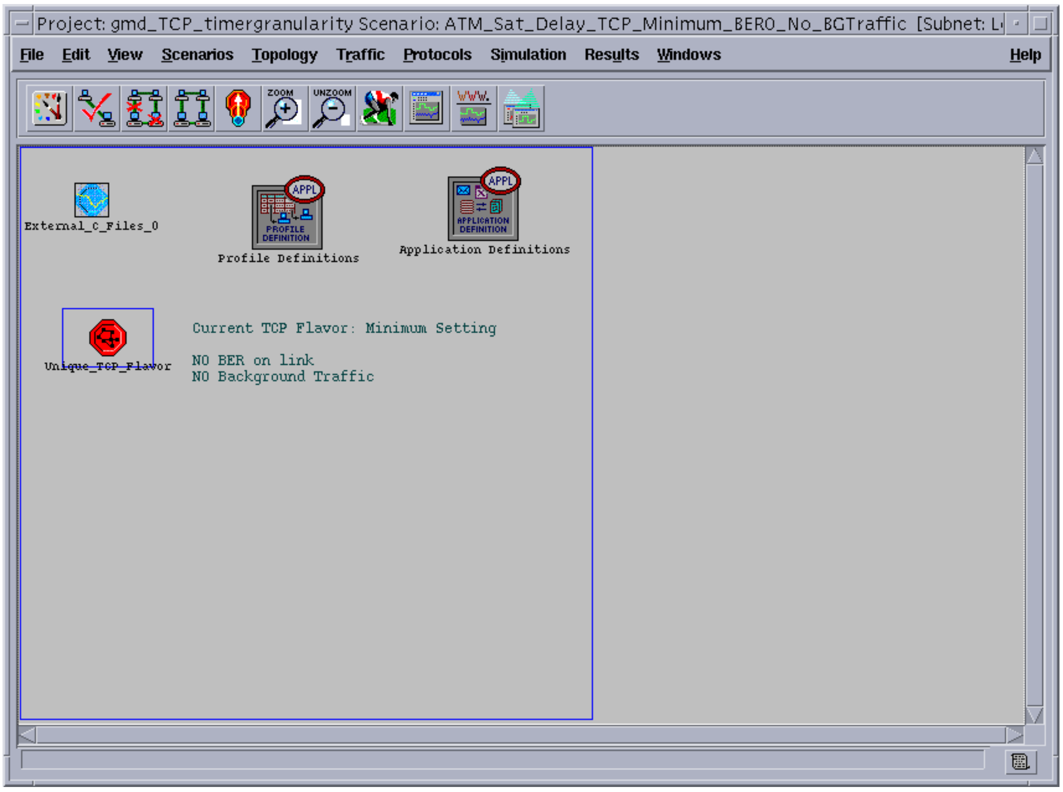
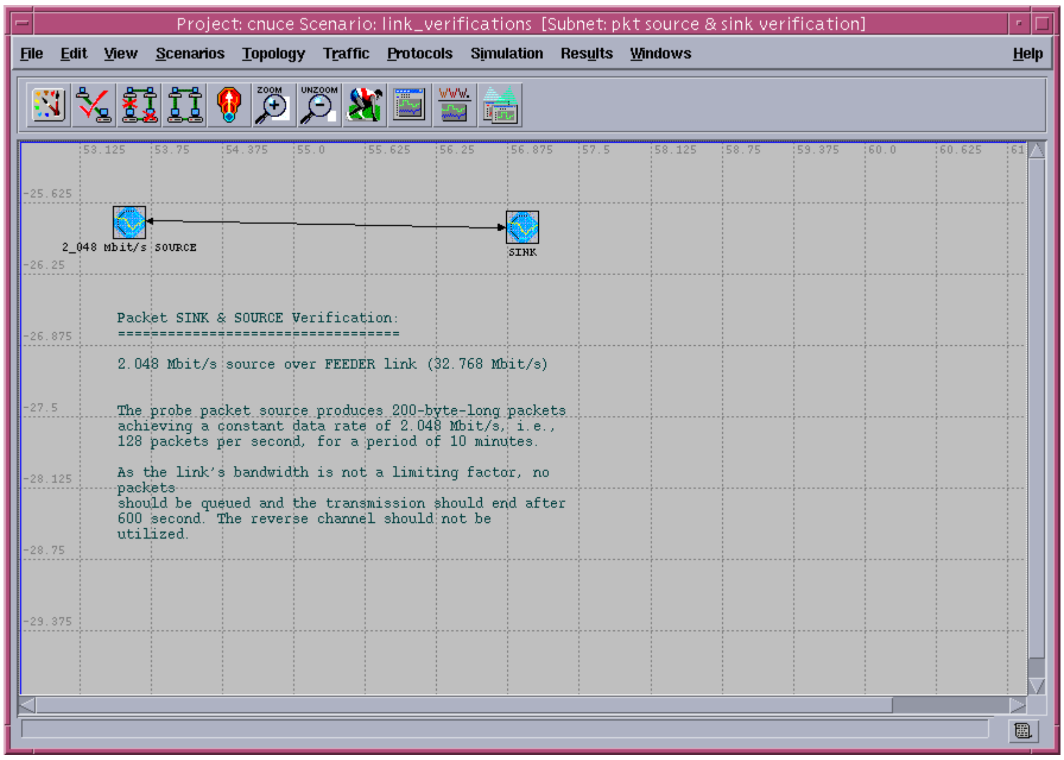

Project: gmd_TCP_timergranularity Scenario: ATM_Sat_Delay_TCP_Minimum_BER0_No_BGTraffic [Subnet: L
File Edit View Scenarios Topology Traffic Protocols Simulation Results Windows Help
ZOOM
UNZOOM
Client_A
Server_A
IP
IP_Network
BG_Traffic

Fig. 5.14 Network Nodes for TCP Analysis


Project: gmd_TCP_timergranularity Scenario: ATM_Sat_Delay_TCP_Minimum_BER0_No_BGTraffic [Subnet: L
File Edit View Scenarios Topology Traffic Protocols Simulation Results Windows Help
ZOOM
UNZOOM
External_C_Files_0
APPL
PROFILE DEFINITION
Profile Definitions
APPL
APPLICATION DEFINITION
Application Definitions
Unique_TCP_Flavor

Current TCP Flavor: Minimum Setting
NO BER on link
NO Background Traffic

Fig. 5.15 Profile and Application Definition


Project: cnuce Scenario: link_verifications [Subnet: pkt source & sink verification]
File Edit View Scenarios Topology Traffic Protocols Simulation Results Windows Help
ZOOM
UNZOOM
2_048 Mbit/s SOURCE
SINK

```
Packet SINK & SOURCE Verification:
=================================

2.048 Mbit/s source over FEEDER link (32.768 Mbit/s)

The probe packet source produces 200-byte-long packets
achieving a constant data rate of 2.048 Mbit/s, i.e.,
128 packets per second, for a period of 10 minutes.

As the link's bandwidth is not a limiting factor, no
packets
should be queued and the transmission should end after
600 second. The reverse channel should not be
utilized.
```

Fig. 5.16 Testing Scenario of the Packet-Source-Sink Node

will specify the simulation settings used to answer these questions and to verify the used simulation models themselves.

Section 5.4.1 focuses on the simulation settings used to assure a proper functionality of the new implemented satellite link model (modified point-to-point pipeline stage) by transmitting raw data packets over the link. This simulation setting is apart from the actual TCP evaluation. Afterwards, Section 5.4.2 names the simulation parameters applied to evaluate TCP with respect to the framework of this thesis.

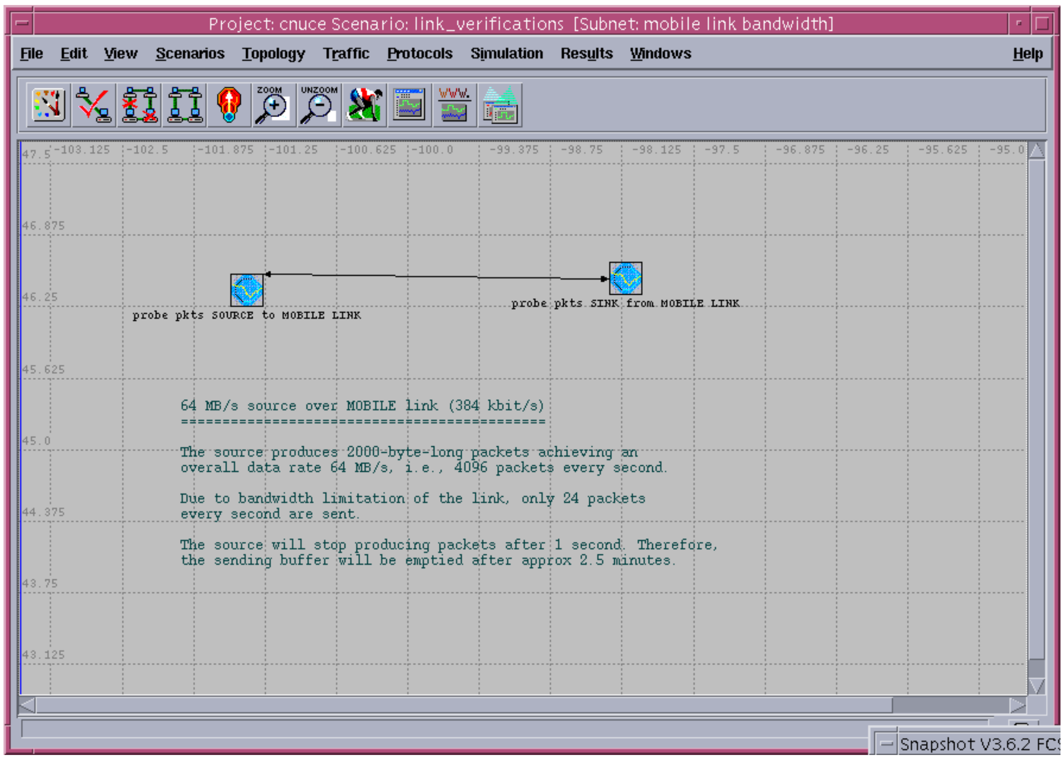
5.4.1 Verification of Simulation Model

Five experiments are conducted to verify the proper behavior of the developed point-to-point pipeline stage. For these experiments, the newly developed "supporting network node" (ref. to Section 5.3.3.2) were used. The first experiment checks the functionality of the node itself; other verify the bandwidth limitations and the variable delay of the newly developed point-to-point link model.

Verification of Supporting Network Nodes Two instances of the network node described in Section 5.3.3.1 are connected with each other via a 32.768 Mbit/s, point-to-point link (Fig. 5.16). The first node (source) sends 200-byte-long packets with a data rate of 128 packets/s. After ten minutes, the source node stops sending packets. The second node (sink) is configured to receive any arriving packets and to destroy them, i.e. the backward channel is not utilized.

As the source node sends with 2.048 Mbit/s, the following behavior is expected if the model works properly:

- The throughput on the forward channel should be 2.048 Mbit/s (6.25% utilization).
- The throughput on the reverse channel should be zero.
- The utilization of the forward channel should drop to zero right after the source stops its transmission (i.e. after ten minutes).


Project: cnuce Scenario: link_verifications [Subnet: mobile link bandwidth]
File
Edit
View
Scenarios
Topology
Traffic
Protocols
Simulation
Results
Windows
Help
ZOOM
UNZOOM
probe pkts SOURCE to MOBILE LINK
probe pkts SINK from MOBILE LINK
Snapshot V3.6.2 FC

```
64 MB/s source over MOBILE link (384 kbit/s)
============================================

The source produces 2000-byte-long packets achieving an
overall data rate 64 MB/s, i.e., 4096 packets every second.

Due to bandwidth limitation of the link, only 24 packets
every second are sent.

The source will stop producing packets after 1 second. Therefore,
the sending buffer will be emptied after approx 2.5 minutes.
```

Fig. 5.17 Testing Scenario of Mobile User Bandwidth

Bandwidth Verification According to the target system specification (ref. to Appendix D), the satellite channel provides a bandwidth of either 384 kbit/s, 2.048 Mbit/s, or 32.768 Mbit/s depending on the kind of link, i.e. a link to a mobile user terminal, a fixed terminal, or to the network provider. [18] To verify the bandwidth limitations, the source is configured to send 2000-byte-long packets with a rate of 4096 packets/s (64 Mbit/s required bandwidth). The transmission is stopped after one second for the mobile link, after ten seconds for the fixed terminal link, and after five minutes for the provider link.

The link is expected to show an utilization of 100% throughout the transmission. The node configured as a packet sink should record packets arriving at a constant rate which should correspond to the bandwidth limitation of the link.

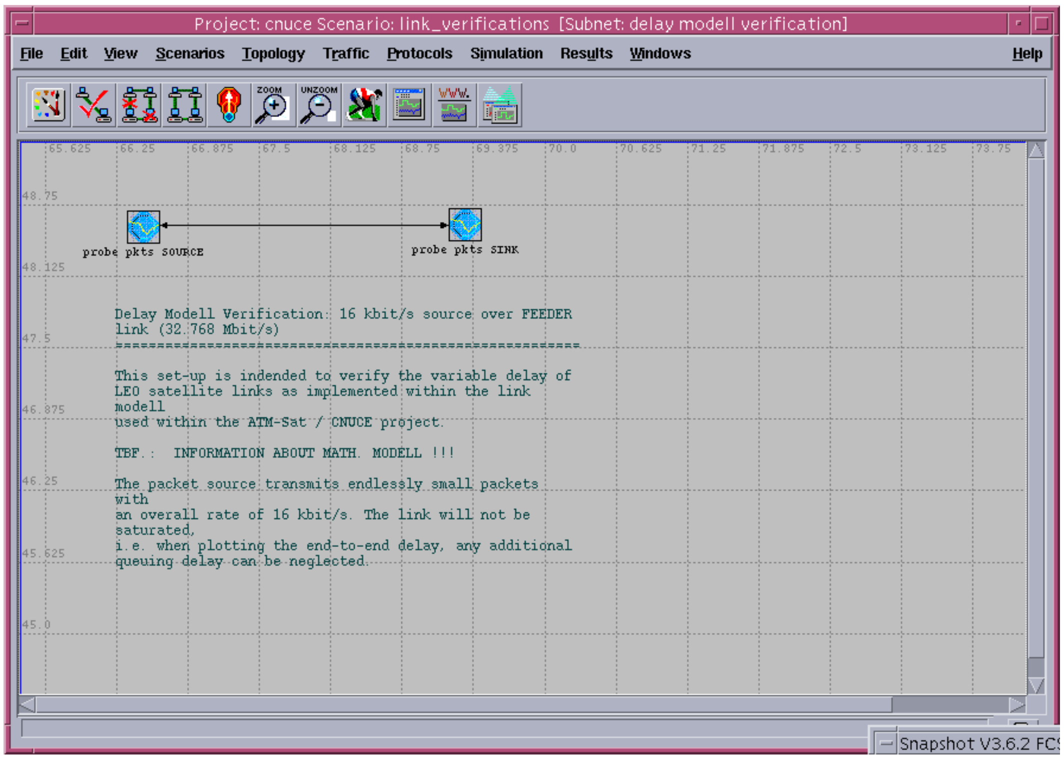
Figure 5.17 illustrates the scenario for the mobile link. Illustrations of the scenario checking on the fixed and the provider link are omitted.

Variable Propagation Delay Finally, the source node is configured to send a constant-rate bit-stream of 16 kbit/s over a provider link (32.768 Mbit/s available bandwidth, refer to Fig. 5.18). The time in between the packet's creation and reception is recorded by the sink node and should thus reflect the variable propagation delay of the link.

Additional delay measurements are conducted. For these measurements, the link is configured to impose variable propagation delays referring to different satellite constellations. These constellations correspond to those used by Allman et al; they are evaluated to verify that the simulation model used for this work is comparable with the one described in [14]. Table 5.1 provides an overview of the analyzed delays; measurement results are depicted in Appendix B.

5.4.2 Analysis of TCP

The simulation settings to evaluate TCP are threefold: First experiments focus on TCP's ability to reflect a variable delay of a satellite link. Accordingly, Section 5.4.2.1 describes the applied algorithm of TCP to determine its RTO. Afterwards, the parameters of file transfer experiments utilizing different receive


Project: cnuce Scenario: link_verifications [Subnet: delay modell verification]
File Edit View Scenarios Topology Traffic Protocols Simulation Results Windows Help
ZOOM
UNZOOM
probe pkts SOURCE
probe pkts SINK
Snapshot V3.6.2 FCS

```
Delay Modell Verification: 16 kbit/s source over FEEDER
link (32.768 Mbit/s)
==========================================================

This set-up is indended to verify the variable delay of
LEO satellite links as implemented within the link
modell
used within the ATM-Sat / CNUCE project.

TBF.:  INFORMATION ABOUT MATH. MODELL !!!

The packet source transmits endlessly small packets
with
an overall rate of 16 kbit/s. The link will not be
saturated,
i.e. when plotting the end-to-end delay, any additional
queuing delay can be neglected.
```

Fig. 5.18 Testing Scenario of Variable Propagation Delay

Table 5.1 Summary of Analyzed ISL Propagation Delays

ISL[a]	Period	Illustration
ATM-Sat[b]	12 minutes	Fig. B.1
Cobe[c]	62 minutes	Fig. B.2
Cobe[d]	3.7 hours	Fig. B.3
International Space Station ISS[c]	54 mintutes	Fig. B.4
International Space Station ISS[d]	60 minutes	Fig. B.5
Lageos[c]	3.5 hours	Fig. B.6
Lageos[d]	8.4 hours	Fig. B.7
Navstar	n/a[e]	Fig. B.8
Radcal[c]	61 minutes	Fig. B.9
Radcal[d]	3.77 hours	Fig. B.10

[a] All ISLs are named with respect to TDRSS-5.
[b] The ATM-Sat LEO satellite delay not an ISL delay but the associated up- and down-link delay.
[c] Consider is the access interval which results in the maximal variation of the delay.
[d] Consider is the access interval which results in the minimal variation of the delay.
[e] Periodic repetition not necessary as Navstar has a 24-hour visibility from TDRSS-5.

Table 5.2 Verification of TCP Timer Computation — Simulation Parameters

Scope	Parameter	Value
Application Profile and Definition	FTP	100% get-commands
	Inter-request time	90,000 seconds
	Transferred File Size	400 MByte
TCP Settings	Timer Granularity	"not used" (i.e. none)
	Max. Segment Size	536 Byte
	Rec. Buffer Size	3300 Byte
	Max. ACK Delay	200 msec
	Slow Start Initial Count	1
	Initial RTO	1 sec
	Min. RTO	1 msec
	Max. RTO	64 sec
	RTT Gain	1 (1st experiment)
		0.125 (2nd experiment)
	Deviation Gain	0.25
	RTT Deviation Coefficient	4

buffer sizes are given in Section 5.4.2.2. Finally, Section 5.4.2.3 specifies the bit error rates that are used to repeat all experiments in an erroneous environment.

5.4.2.1 TCP Timer Analysis Opnet's TCP implementation tracks the variance of the RTT measurements in addition to the smoothed RTT in order to calculate the RTO (ref. to Section 4.1.3.1). As described by Jacobson [50], each of the following equations is applied to the current RTT measurement M

$$Err = M - A \tag{5.23}$$
$$A \leftarrow A + g \cdot Err \tag{5.24}$$
$$D \leftarrow D + h \cdot (|Err| - D) \tag{5.25}$$
$$RTO = A + 4 \cdot D \tag{5.26}$$

where A is the smoothed RTT and D is the smoothed mean deviation. Err is the difference between the last recent segment's RTT measurement and the current RTT estimator. [73]

The first simulations with respect to TCP focus on the latter's ability to measure the variable propagation delay. As Opnet does only report the smoothed RTT A, the experiment is twofold: In the first experiment, the gain g for the smoothed RTT is set to 1 in order to have Opnet report the measured segment's RTT. In a second experiment, g and h are set to their standard values of 0.125 and 0.25 correspondingly. The timer granularity is set to none, i.e. Opnet does exactly calculate the segment RTT.

The experiments include the measurement of the RTO as well. Therefore, the lower bound of the RTO is set to 1 msec to have the measurements reflect the calculated RTO. Table 5.2 summarizes the simulations parameters. The file to be transferred via FTP is large enough to allow TCP's timer an estimation of an entire delay's period. The receive buffer is reduced to 3300 byte to avoid that the queuing delay dominates the time measurements which might be the case for a saturated link.

5.4.2.2 Variation of Receive Buffer Size and Timer Granularity A second set of simulations focuses on different advertised receive buffer sizes and different TCP timer granularities. Both variables might effect TCP's ability to estimate the variable link delay: The buffer size determines the link's saturation which in turn might result in different queuing delays, whereas the timer granularity determines the accuracy of TCP's RTT estimation. An inaccurate estimation might cause TCP's retransmission timeout mechanism to trigger prematurely which in turn imposes unnecessary network load (ref. to Section 4.1.3.1).

Several receive buffer sizes are evaluated ranging from the maximum segment size (MSS) up to a multiple of the bandwidth delay product (BDP). For each of these buffer sizes, TCP's timer granularity

Table 5.3 Effects of Receive Buffer Size and TCP Timer Granularity — Simulation Parameters

Scope	Parameter	Value
Application Profile and Definition	FTP	100% get-commands
	Inter-request time	90,000 seconds
	Transferred File Size	250 MByte
TCP Settings	Receive Buffer Sizes	1460 Byte (MSS Size)[a]
		3300 Byte (1/2 BDP)
		6600 Byte (BDP)
		64000 Byte
	Timer Granularity	adjusted to avoid false retransmissions[b]
	Max. Segment Size	1460 Byte[a]
	Max. ACK Delay	200 msec
	Slow Start Initial Count	1
	Initial RTO	1 sec
	Min. RTO	1 msec
	Max. RTO	64 sec
	RTT Gain	0.125
	Deviation Gain	0.25
	RTT Deviation Coefficient	4

[a] Typical LAN based MSS minus 40 Bytes for the IP header.
[b] The timer granularity is adjusted in steps of 1 ms.

is altered in steps of 1 ms in order to determine the lowest possible granularity which does not cause premature retransmissions. Table 5.3 summarizes the simulation parameters.

5.4.2.3 Introduction of Bit Errors A fine timer granularity makes TCP more sensitive to changes of the segment RTT and thus allows a best possible calculation of the RTO. The RTO should approximate the arrival time of a segment's ACK in a best possible manner to determine the earliest possible retransmission time of a lost segment but must not cause TCP to trigger premature retransmissions if a segment is slightly delayed due a regular change in the RTT (ref. to Section 4.1.3.1). Therefore, further experiments concentrate on the effects of various bit error rates on TCP's ability to reflect the variable link delay, to avoid false retransmissions, and to uphold good performance.

The first set of experiments focus on TCP's ability to reflect the link's back- and forward propagation delay in the presence of bit errors. The second set of experiments evaluates several TCP flavors in combination with various timer granularities and buffer sizes. The goal of the second experiment set is to measure TCP performance in terms of FTP-throughput which allows later on to evaluate the benefits of finer (hand-tuned) timer granularities.

All experiments impose either an BER of $2 \cdot 10^{-8}$ and $2 \cdot 10^{-5}$. The first rate is the worst error rate which is allowed with the ATM-Sat project[11] [17] whereas the second rate is a worst case approximation of a typical BER[12] found in today's satellite systems. [13]

Effects on TCP's Ability to Reflect the Variable Link Delay A large-file FTP transfer is conducted to check on TCP's ability to reflect the link's propagation delay in the presence of BERs. The parameters of the experiment are given in Table 5.3 except for the buffer size. The latter has been set only to 3300 Byte to avoid additional delays imposed by a saturated link and its filled queues. The timer granularity is set to 4 ms, i.e. the lowest possible value which does not cause (false) retransmission in the error-free case.

[11]The adaptive FEC scheme described by Emmelmann and Bischl can even guarantee lower error rates. [38] The used error rate of $2 \cdot 10^{-8}$ was the upper bound required by the system's specifications.
[12]RFC 2488 does even state that current satellite radio systems shall be able to guarantee a BER of 10^{-7} or less.

Table 5.4 Experiment Set Parameters to Evaluate TCP Throughput Performance[a][b]

Parameter	Variations
TCP Flavor	TCP Reno
	TCPmin
Advertised Receive Buffer Size	64000 Byte
	3300 Byte ($\approx 1/2$ BDP)
	1460 Byte (MSS)
Bit Error Rate	none
	$2 \cdot 10^{-8}$
	$2 \cdot 10^{-5}$

[a] Each of the resulting experiment set contains a number of experiments with varying timer granularities and lower bounds of the RTO according to Table 5.5.

[b] Omitted TCP settings as well as the Application Profile and Definition can be deduced from Table 5.3

Table 5.5 Experiment Number For a Given TCP Flavor, Buffer Size, and BER [35]

Lower Bound of RTT Timer Granularity	None[a]	100 ms	200 ms	Default[b]
G = 1 ms		2	7	
G = 50 ms		3	8	
G = 100 ms		4	9	
G = 200 ms		5	10	
G = 500 ms		6	11	12
Minimal G to avoid false retransmissions (varying)	1			

[a] A lower bound of 1 ms was used.

[b] Opnet modeler implements a lower bound of 500 ms for the RTO. This is feasible since the default timer granularity G is 500 ms as well.

Effects of Timer Granularity on TCP Throughput An optimized or even hand-tuned TCP timer granularity promises the retransmission of lost segments at an earliest possible stage. Final experiments evaluate this degree of performance improvement.

Two TCP flavors, namely TCP Reno and "TCPmin" are evaluated. TCP Reno[13] employs the Fast Retransmit and Fast Recovery Algorithm [11,73] whereas "TCPmin" corresponds to Vanilla TCP which has the two enhancements disabled and is therefore compliant to RFC 793. [46] The simulation consists of a single FTP transfer of a 250 MB-large file using either one of the two TCP flavors. The transfer is repeated for different advertised receive buffer sizes and different error rates. Table 5.4 summarizes the possible parameter settings.

Finally, each of the resulting experiment consists of numerous FTP file transfers using different TCP timer granularities G and different lower bounds of the RTO (ref. to Table 5.5) thus resulting in a total of 216 unique file transfers. It should be noted that different timer granularities are evaluated without limiting possible values to those avoiding false retransmissions. The focus of the measurement is on the time to transmit a file. Every possible experiment is run several times to assure higher confidence in the simulation results.

[13]Please refer to Section 4.1 for an introduction for TCP mechanisms.

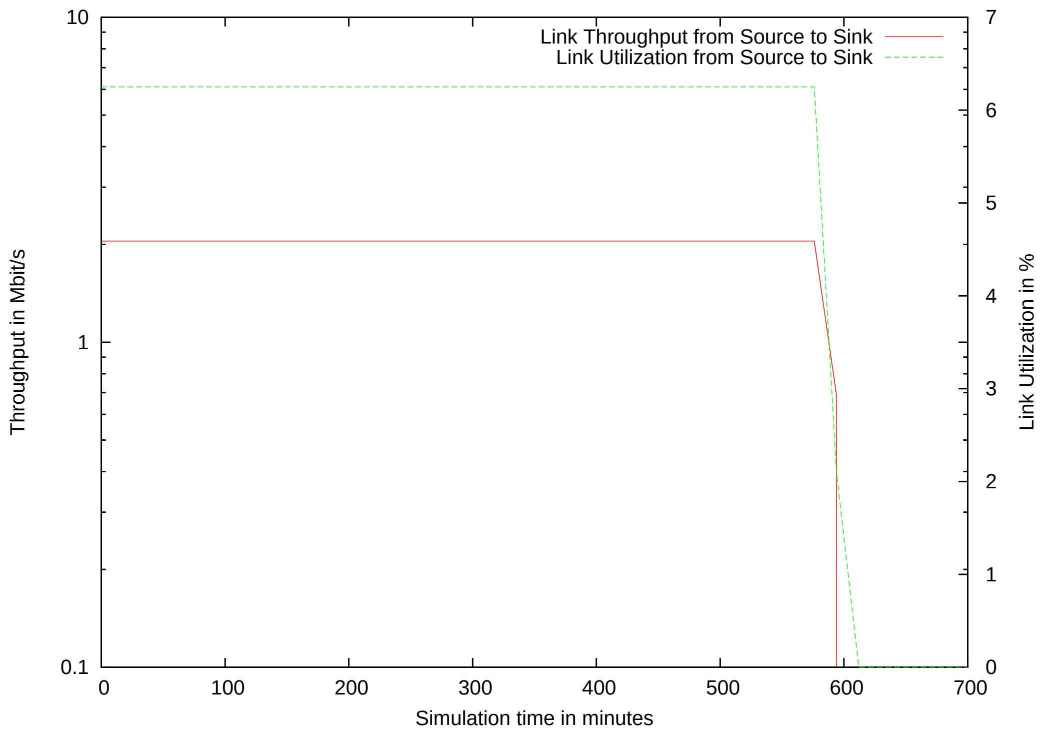


Fig. 5.19 Verification of Packet-Source-Sink Node

5.5 SIMULATION RESULTS AND EVALUATION

The presentation of the simulation result is twofold: Section 5.5.1 presents the data which assures that the developed simulation environment (i.e. the satellite channel link model) works correctly. Afterwards, Section 5.5.2 evaluates the simulation results related to TCP.

5.5.1 Link Model Verification

The simulation results to verify the adapted point-to-point pipeline stage model representing the satellite link indicate that the network node used to conduct this verification works properly itself. As expected, the link has the predefined capacity and imposes a variable delay to the forwarded traffic.

5.5.1.1 Verification of Supporting Network Nodes The measurements indicate the proper functionality of the Packet-Source-Sink node. The link's bandwidth is not limiting the data transfer (expected link utilization of exactly 6.25%) and the allocated bandwidth equals to 2.048 Mbit/s (Fig. 5.19). The transmission ends after the specified duration of ten minutes and the reverse channel is never utilized (result not plotted). The shape of the plot at the end of the measurement is explained by the nature of how Opnet accumulates measurement samples. During the simulation, several samples of the link utilization and bandwidth allocation are taken. Opnet averages the samples taken over the last 12 minutes and reports this result.[14] Therefore, the last average interval around the 600 minute mark on the x-axis contains samples of high and zero link utilization. Thus, the reported utilization is less than the expected value. The two plots do not overlap around the 600 minute scale as the time intervals over which samples are averaged are not synchronous.

[14]The period of the interval can be changed or even set to zero in order to report every single measurement. The latter is not feasible for large and long simulations as the analysis data would grow up to several Gigabyte.

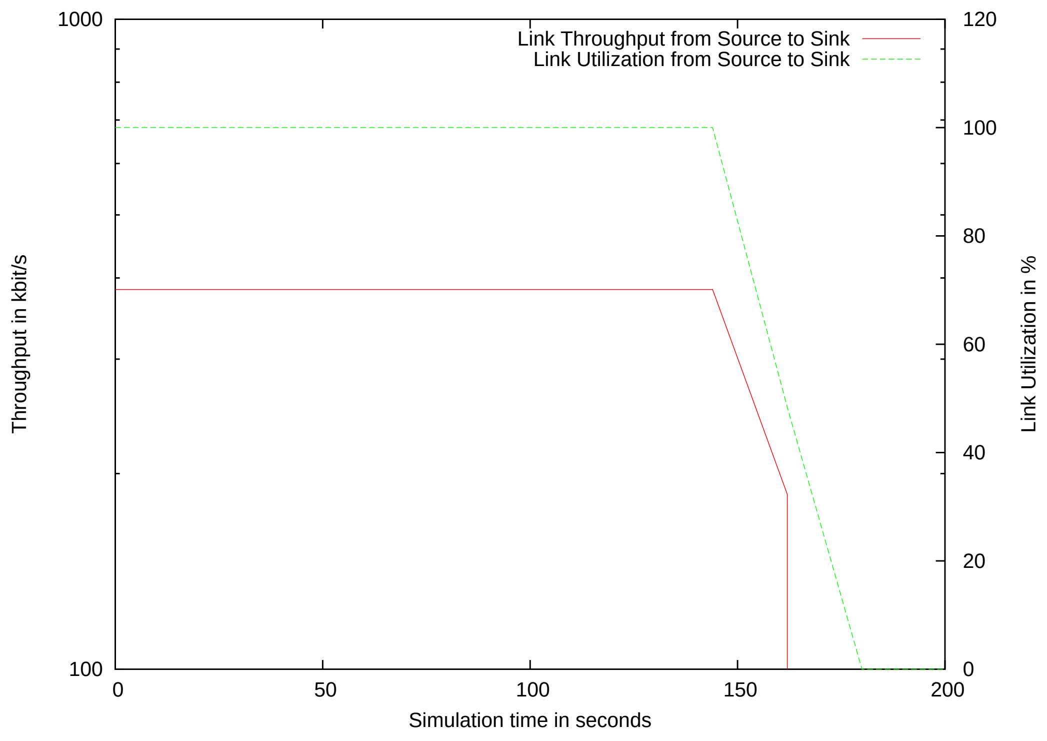


Fig. 5.20 Verification of Bandwidth Limitations of Mobile Link

5.5.1.2 Bandwidth Verification All three types of links limit the available bandwidth according to their specification. The link utilization is permanently 100% and, for the mobile link, the reported bandwidth allocation equals 384 kbit/s (Fig. 5.20).[15] The source produces 2000-byte long packets at a rate of 4096 packets/s (i.e. 64 Mbit/s). After one second, no further packets are forwarded to the transmitter of the "link verification node" (refer to Fig. 5.11); thus, 8 MByte of data are pending for transmission. The receiving node reports arriving data for approximately 170 seconds which is the time to transmit 8 MByte over a 384 kbit/s link. This behavior indicates that the transmitter (refer to Fig. 5.10) does not drop any packets but has an unlimited queue to store accepted packets until the link is either idle or the transceiver pipeline stage function responsible for the transmission delay returns.

5.5.1.3 Variable Propagation Delay For an unsaturated link, the pipeline stage model is well able to impose the variable propagation delay on transmitted packets. Figure 5.21 plots the end-to-end delay as seen by the sink node and compares it with the theoretical delay discussed in Section 5.3.2. The difference between the measured and analytical delay is 5 ms which corresponds to the reported queuing delay. Opnet reports the queuing delay to be the time a packet is hold in the transmitter node of a pipeline execution sequence (see Fig. 5.10) before it can be transmitted.

5.5.2 TCP Analysis

The analysis of TCP shows that TCP's internal timer are well capable to reflect the link's variable propagation delay (Section 5.5.2.1). This ability is not influenced for advertised receive buffers having a size less than the bandwidth delay product while still not limiting the data-rate to approximately one segment per RTT. For these advertised buffers, the timer granularity can only be reduced on a hand-tuned bases without causing premature retransmissions (Section 5.5.2.2). Finally, the introduction of bit errors (Section 5.5.2.3) will show that only low BERs allow TCP to accurately guess the variable RTT. In terms

[15] Illustrations and a detailed analysis of the fixed and provider link can be found in Appendix C.

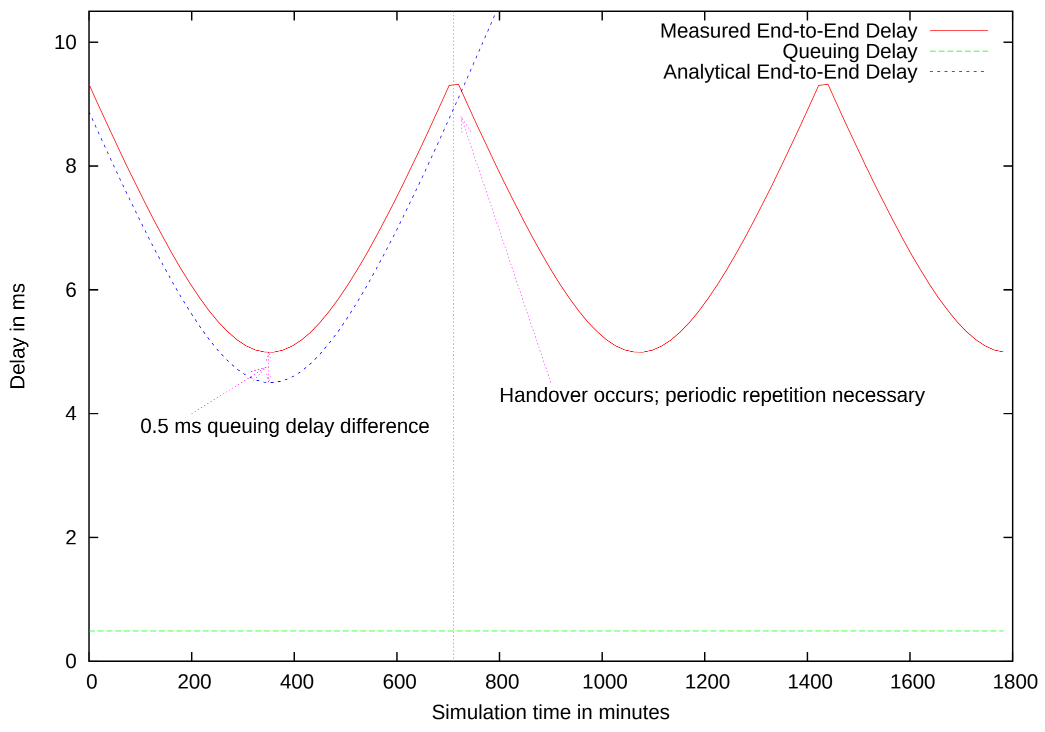


Fig. 5.21 Verification of Variable End-to-End Delay

of throughput performance, the TCP flavor itself rather than "hand-tuned" timer granularities is shown to be the most significant factor.

5.5.2.1 TCP Timer Analysis

TCP's measurement of a segment's round-trip time reflects the expected delay almost perfectly. Both graphs vary in between 13.5 and 22 ms. The following calculation is used to compare the measured segment RTT with the expected delay value d_{exp}:

$$d_{exp} = 2d_{propagation} + d_{queuing} \tag{5.27}$$

where $d_{propagation}$ is the one way propagation delay and $d_{queuing}$ is the sum of the forward and backward queuing delay (Fig. 5.22). The difference in between the expected value and the measurement cannot be explained by TCP mechanisms. It can be considered as a processing delay (e.g. needed for memory allocation) as, for the given bandwidth of 2.048 Mbit/s, is corresponds almost exactly to the time to receive 256 MByte.

Figure 5.23 illustrates the forward and backward queuing delay which are applied to Eq. (5.27). The application profile defines FTP 100% get-operation modus, i.e. the link from the client to the server should only carry ACKs whereas the link from the server to the client carries data packets. This definition explains the observed queuing delays of 0.15 ms and 3.36 ms. The first delay corresponds to the time, a pure TCP/IP ACK remains in the queue (transfer of 40 Byte over a 2.040 Mbit/s link). The second delay corresponds to approximately 1.5 data segments in the queue (segment size 536) and can be explained by the client's ACK behavior. The client does either acknowledge a received segment right away or implies the delayed acknowledgment scheme. Thus, an ACK arriving at the server may result in either sending a single or two new data segments which results on the average in 1.5 segments pending in the queue.

The final measurements compare the measured segment round-trip time with the smoothed RTT and the retransmission timeout (Fig. 5.24). Almost all samples of the measured and smoothed RTT plot directly over each other; the retransmission timeout follows the anticipated RTT throughout the file transfer.

5.5.2.2 Variation of Receive Buffer Size and Timer Granularity

So far we have seen that TCP's timer algorithms are well capable to reflect the variable propagation delay of the LEO satellite

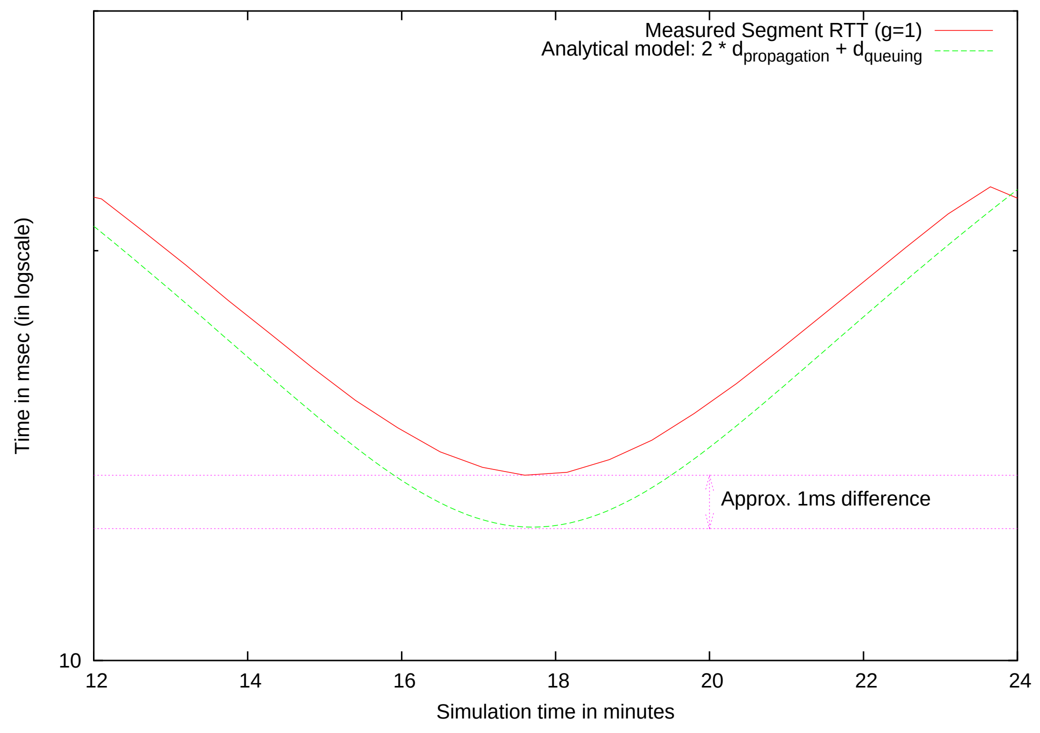


Fig. 5.22 Difference between Measured Segment RTT and Analytically Calculated RTT

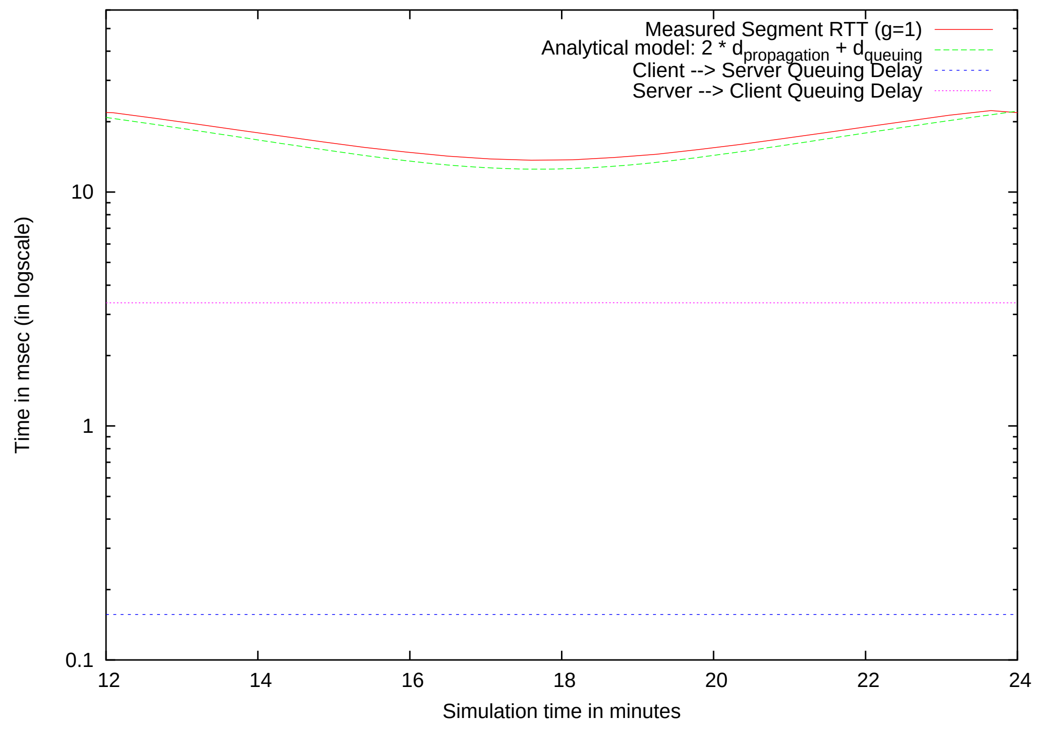


Fig. 5.23 Queuing Delay, Measured Segment RTT, and Analytically Calculated RTT

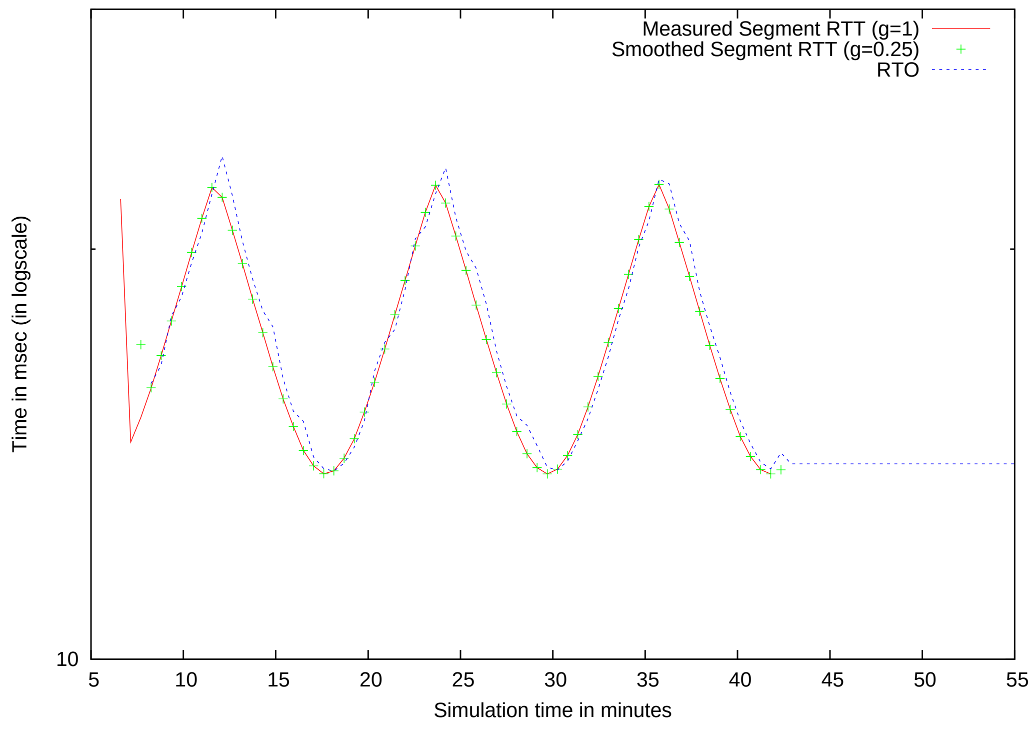


Fig. 5.24 Measured Segment RTT, Smoothed RTT, and RTO

link for a buffer which does not saturate the available bandwidth and for an infinite fine timer granularity G. The following results will show the effect of different buffer sizes and timer granularities on this ability.

Effects of Receive Buffer Size on Propagation Delay Approximation The advertised receive buffer size effects TCP's ability to reflect the variable link delay. A receive buffer which permanently saturates the link, prevents TCP to accurately guess the current variable link delay. The segment round-trip time, as Fig. 5.25 illustrates, is dominated by the constant queuing delay:

$$\begin{aligned} d_{queuing} &= \frac{64\ kByte}{2048\ kbit/s} \\ &= 250\ ms \end{aligned} \tag{5.28}$$

which reflects the number of unacknowledged data on the forward link. The average propagation delay of the ACK on the reverse channel (7 ms) has to be added to the queuing delay to come up with the experienced segment RTT. A timer granularity of 1 ms did not cause any false retransmissions.

On the other side of the spectrum, a low link utilization, effects on link delay approximation does not necessarily result in a good approximation of the variable propagation delay. Figure 5.26 plots the measures smoothed SRTT for an advertised TCP receive buffer size of 1460 Byte. The size is equivalent to the maximum segment size[16] and limits TCP to transmit only a single segment at a time. The receiver employs the delayed ACK option and thus acknowledges the segment only if a second one arrives or its local time-out expires. As the buffer size allows only one segment of unacknowledged data, ACKs are only emitted due to the time-out's expiration and the client measures a constant 200 ms segment RTT. The careful reader would expect a SRTT in between 210 and 220 ms to take the forward and backward propagation delay into account. This delay is not measured as a TCP timer granularity of less than 26 ms results in needless retransmissions caused by premature expirations of the sender's RTO. Therefore, the

[16]The MSS is 1500 Byte, but TCP does not have to allocate memory for the 40-byte IP header.

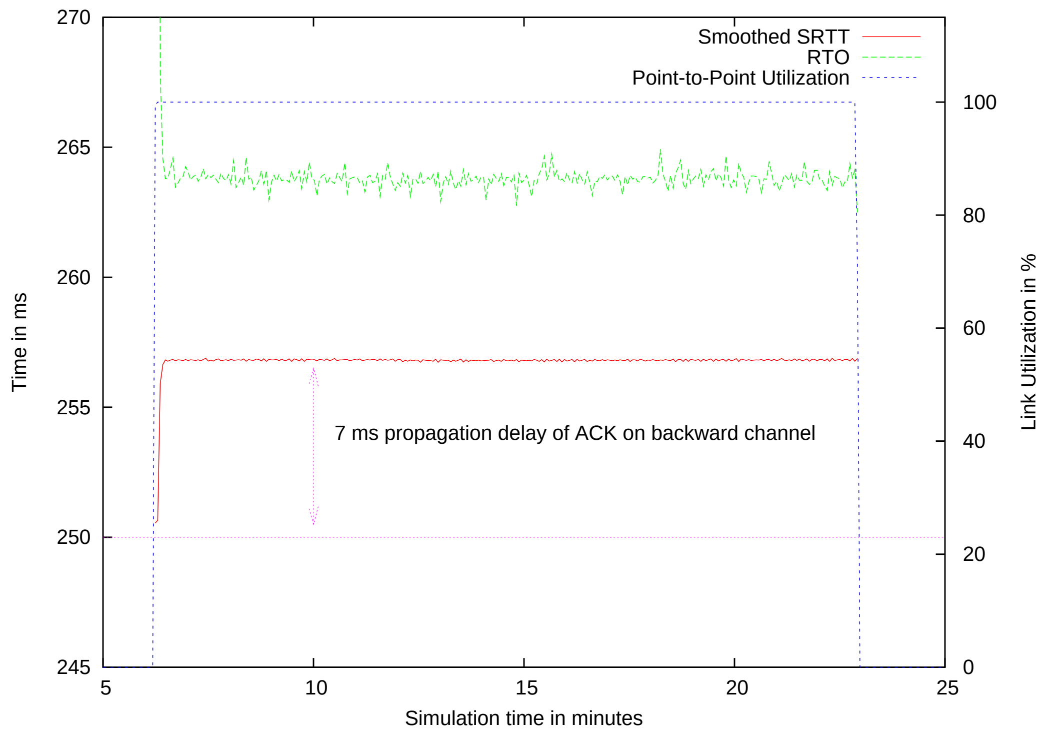


Fig. 5.25 Effect of 64kB Receive Buffer on Delay Estimation (G=1 ms)

measured SRTT can only be a multiple of 26 ms which ignores SRTTs in between 200 and 220 ms and leaves the smoothed average around 200 ms (ref. to Section 4.1.3.1).

A link utilization of roughly 50% allows TCP's SRTT measurement mechanism to approximate the shape of the variable propagation delay. Figure 5.27 plots the Smoothed SRTT, RTO, and point-to-point utilization for an advertised receive buffer size of 3300 kB which is roughly half of the bandwidth delay product. In addition, the graph includes the expected SRTT. It is calculated from the measured queuing delays $D_{queuing}$ and the measured one-way propagation delay $d_{propagation}$ according to Section 5.5.1.3.

$$SRTT_{expected} = 2d_{propagation} + d_{queuing} \tag{5.29}$$

Measurements indicate that the queuing in the client-server direction is determined by the transmission of ACKs (40-byte long) where as the server-client direction reveals a queuing delay of approximately 8.8 ms which corresponds to the transmission delay imposed by 1.5 TCP/IP segments.

$$\begin{aligned} d_{queuing} &= d_{queuingClient2Server} + d_{queuingServer2Client} \\ &\approx \frac{40Byte + 1.5MSS}{2048kbit/s} \\ &= \frac{40Byte + 1.5 \cdot 1500Byte}{2048kbit/s} \\ &\approx 0.156ms + 8.789ms = 9.945ms \end{aligned} \tag{5.30}$$

The average number of bytes in the reverse channel's queue ($1.5 \cdot MSS$) is caused by the acknowledge behavior of the client. The delayed ACK mechanism causes only every 2nd segment to be acknowledged and the server may transmit up to two new segments upon the arrival of an ACK. As the advertised receive buffer size is not a multiple of the MSS, either one or two segments may be sent to the client which, on the average, places 1.5 segments in the queue. There is still a 1 ms gap in between the smoothed and expected SRTT. It is not possible to tell if this gap is either caused by memory allocation procedures in order to receive a packet or by the timer granularity of 4 ms which is the finest granularity for which TCP does not unnecessarily re-transmit segments.

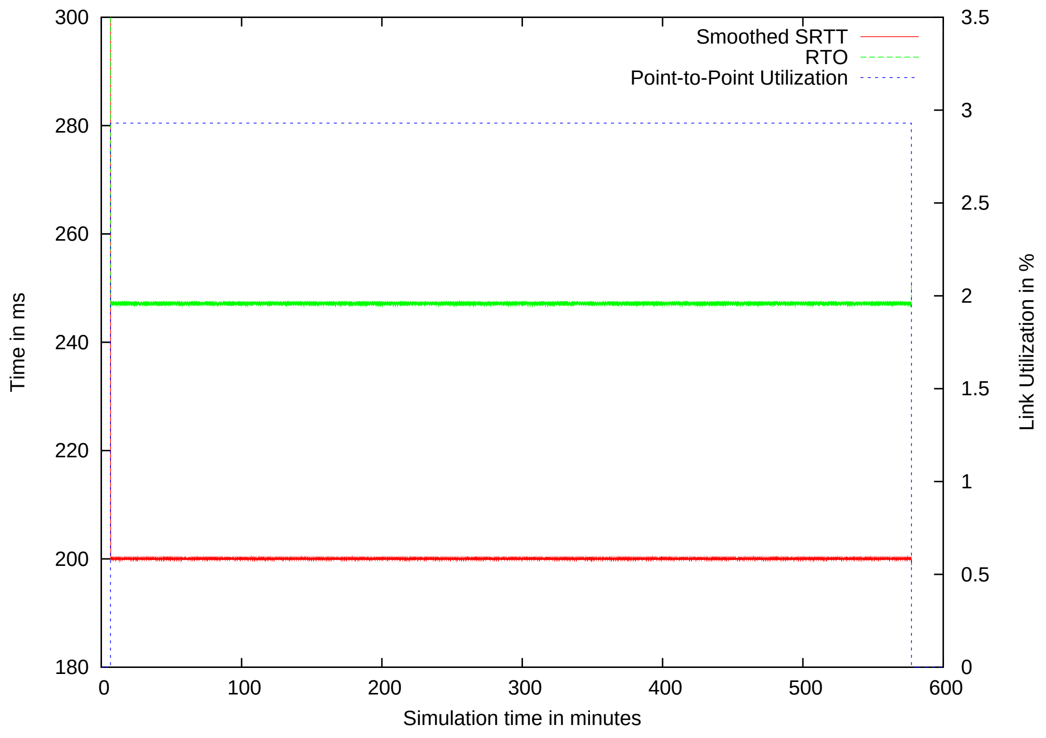


Fig. 5.26 Effect of 1460 Byte (MSS) Receive Buffer on Delay Estimation (G=26 ms)

The anti-cyclical change of the link utilization with respect to the SRTT can be intuitively explained: simulation results, TCP analysis, anti-cyclical change of link utilization wrt. SRTT The number of unacknowledged data on the link is limited by the advertised receive buffer size ($size_{advRecBuffer}$). A link utilization ($linkUtil$) of 100% is reached when this limit is equal or larger than the bandwidth delay product (BDP) and anti-proportional to the BDP for smaller advertised receive buffers.[17]

$$(size_{advRecBuffer} \geq BDP) \quad \Rightarrow \quad (linkUtil = 100\%) \tag{5.31}$$

$$(size_{advRecBuffer} \leq BDP) \quad \Rightarrow \quad \left(linkUtil = \frac{size_{advRecBuffer}}{BDP}\right) \tag{5.32}$$

Equation (5.32) holds for the entire duration of the FTP transfer and yields for a constant advertised receive buffer size and a constant bandwidth to

$$linkUtil \sim \frac{1}{SRTT} \tag{5.33}$$

The final experiment with an advertised receive buffer size of 6600 kByte illustrates the situation when the link is only saturated from time to time (Fig.5.28). For periods of low link utilization, TCP's timer is well able to approximate the expected SRTT whereas for a saturated link, the queuing delay dominates. A timer granularity of 1 ms is sufficient to avoid false retransmissions for the conducted experiment.

Minimization of Timer Granularity The value of TCP's timer granularity G influences the latter's retransmission behavior (Section 4.1.3.1). Table 5.6 lists the lowest possible value of G which avoids TCP to trigger its retransmission timeout prematurely. As an accurate estimation of the SRTT favors quick retransmissions in the case of lost segments, these values are later on used to compare standard timer granularities with these hand-tuned values to evaluate FTP transfers with regard to their throughput.

[17] A perfect growth of the congestion window CWND is assumed as it is the case for an error-free transmission.

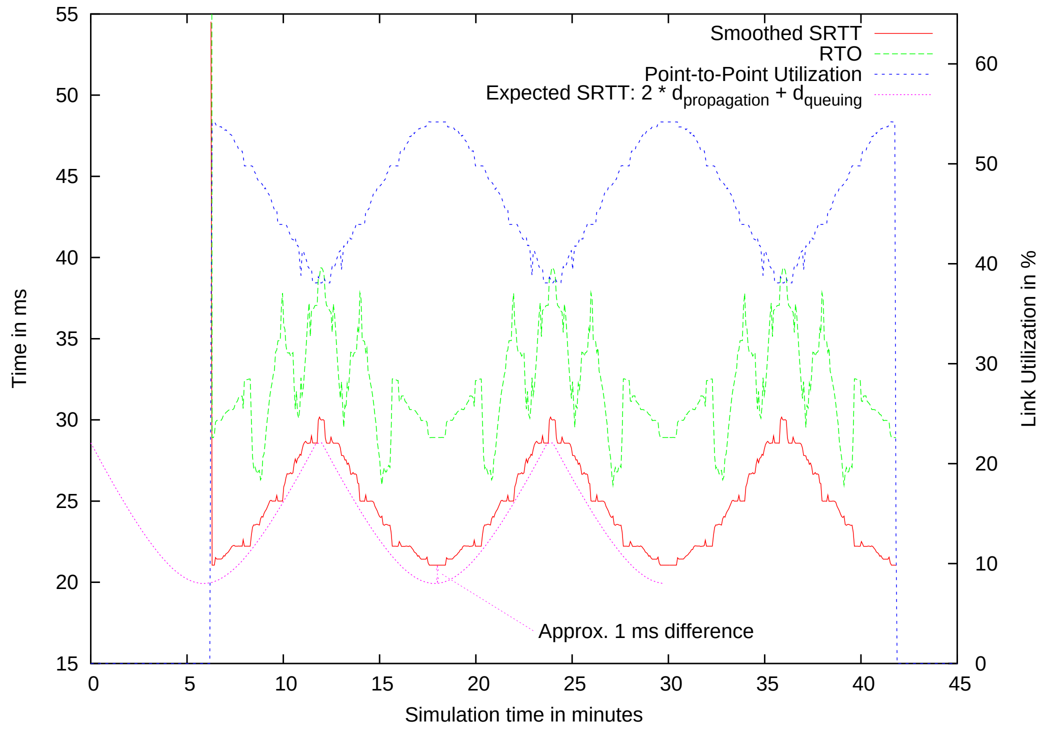


Fig. 5.27 Effect of 3.3 kByte (0.5 BDP) Receive Buffer on Delay Estimation (G=4 ms)

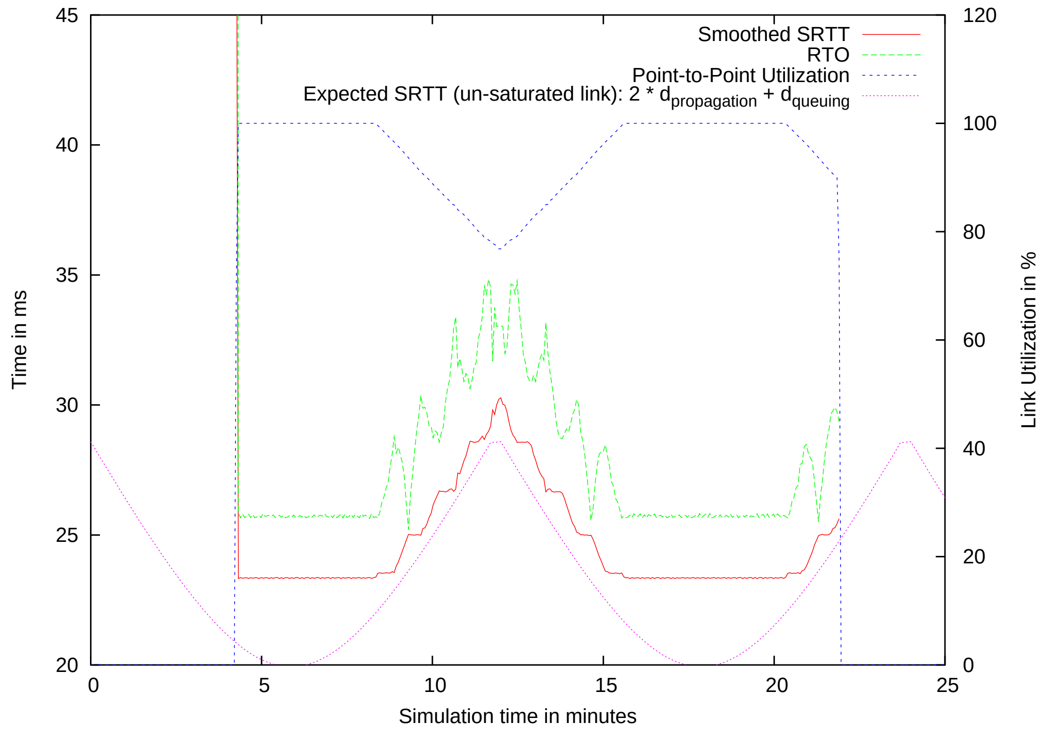


Fig. 5.28 Effect of 6.6 kByte (BDP) Receive Buffer on Delay Estimation (G=1 ms)

Table 5.6 Lowest Possible Value of TCP Timer Granularity G To Avoid Needless Retransmissions

Advertised Receive Buffer Size	Mininum Timer Granularity (G) to avoid false retransmission
64000 Byte	1 ms
6600 Byte ($\approx$ BDP)	1 ms
3300 Byte ($\approx 1/2$ BDP)	4 ms
1460 Byte (= MSS)	26 ms (250 ms)[a]

[a] Only consecutive values larger than 200 ms avoid needless retransmissions without regard to the time the FTP transfer is started. $G = 26\ ms$ is a single occurrence.

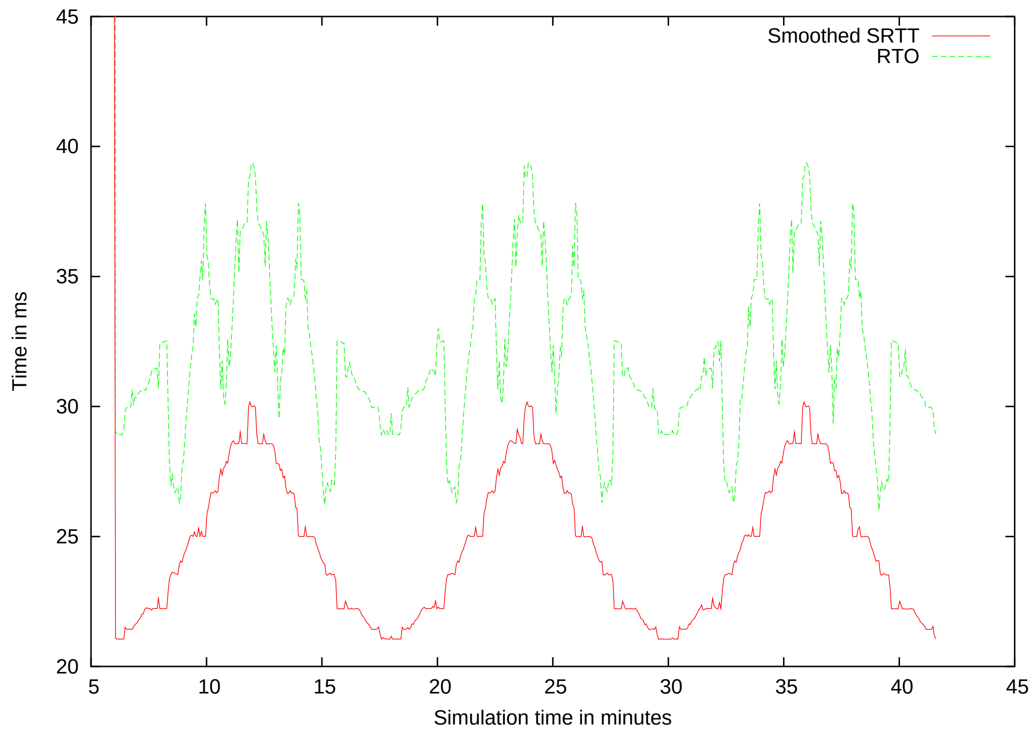


Fig. 5.29 Smoothed SRTT and RTO for a BER of $2 \cdot 10^{-8}$ and a Buffer Size of 3.3 kB

5.5.2.3 Introduction of Bit Errors Up to now, simulation results have indicated that TCP is well able to reflect the variable propagation delay for buffers not saturating the link while still being large enough to transmit a reasonable number of packets simultaneously. False retransmission are avoided for lower timer granularities than the standard 500 ms setting even though they require an accurate hand-tuning with respect to the satellite link. Now, these results are reassured in the presence of an erroneous link.

Effects on TCP's Ability to Reflect the Variable Link Delay TCP is well able to reflect the variable link's propagation delay in the presence of low bit error rates. The measurements of the current value of the Smoothed SRTT and the RTO for an error rate of $2 \cdot 10^{-8}$ (Fig. 5.29) do hardly differ from the error-free case (Fig. 5.27). Retransmissions are solely caused by corrupted segments. The retransmission timer does never trigger prematurely, i.e. it does never tell TCP to retransmit a segment which is not lost but only delayed. Figure 5.29 does also visualize this behavior: The RTO curve is always well above the SRTT curve.

In contrast to low bit-error rates, TCP's timer measurement mechanisms is not able to reflect the link's variable propagation delay for a high BER of $2 \cdot 10^{-5}$. A segment being timed can only be acknowledged if all preceding segments are received and acknowledged. Thus, severe segment loss and

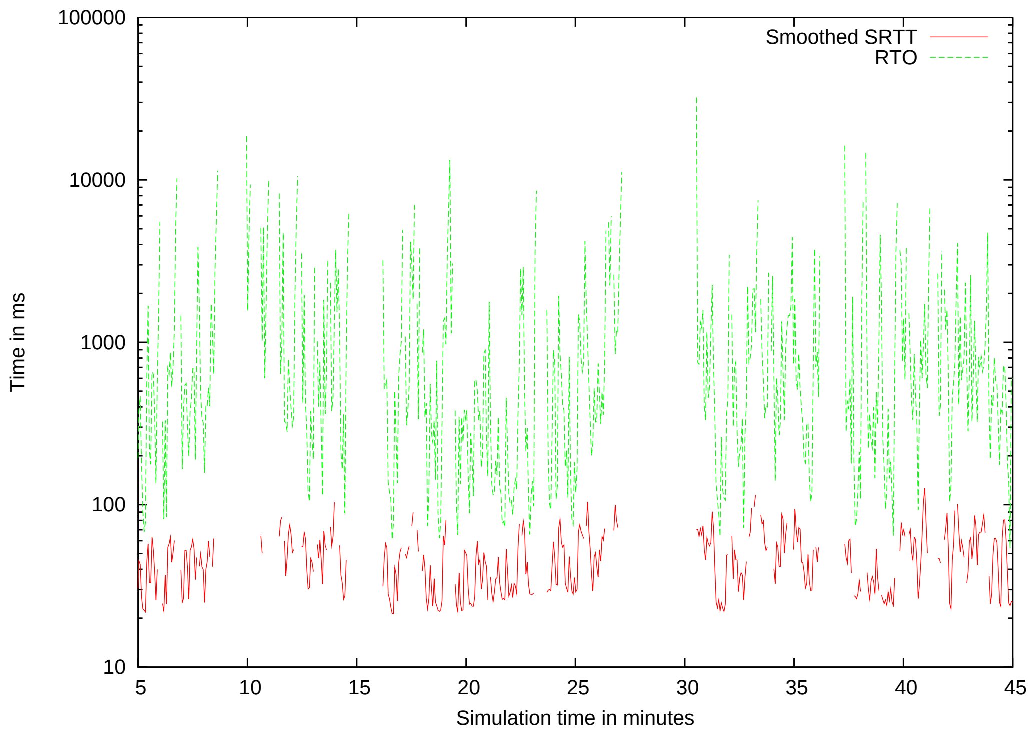


Fig. 5.30 Smoothed SRTT and RTO for a BER of $2 \cdot 10^{-5}$ and a Buffer Size of 3.3 kB

the retransmission of the missing segments have a strong influence on the SRTT measurement (which is completed when sender receives the corresponding ACK). Figure 5.30 depicts the Smoothed SRTT and reveals its high variability which in turn causes, according to Eq. (4.6), extremely high values of the RTO.

Even though neither the Smoothed SRTT nor the RTO values resemble to the link's variable propagation delay, TCP does not prematurely retransmit any segments as the RTO is always well above the current SRTT (Fig. 5.31).

Effects of Timer Granularity on TCP Throughput The assumption that the introduction of bit errors does not yield to false retransmissions is only valid for timer granularities which do not cause the RTO estimator to trigger prematurely in the error-free case. In order to compare the effects of different timer granularities, the normalized FTP response time is calculated for each experiment. This does partially eliminate the influence of the advertised receive buffer size on the results.

The normalized FTP response time t_n is defined as

$$t_n \overset{def}{=} \frac{t}{t_0} \tag{5.34}$$

where t is the FTP response time reported for the experiment applying a given timer granularity, receive buffer size, TCP flavor, and BER. The FTP response time t_0 is determined for a transmission over an error-free channel. Experiments show its independence of the timer granularity G and lower bound of the RTO. FTP response times are determined depending on the advertised receive buffer size (see Table 5.7).[18] The FTP response time for a 64 kB buffer is approximately twice as high as the one for the 3.3 kB Buffer as the BDP is approximately 6.6 kB thus limiting the throughput for the 64 kB experiments.

The experiments run over a link with a low bit error rate of $2 \cdot 10^{-8}$ show the dominant influence of the TCP flavor and the advertised receive buffer size over the timer granularity G with respect to the measured FTP response time (refer to Fig. 5.32).

[18]Values of G which result in false retransmissions are not considered to determine t_0.

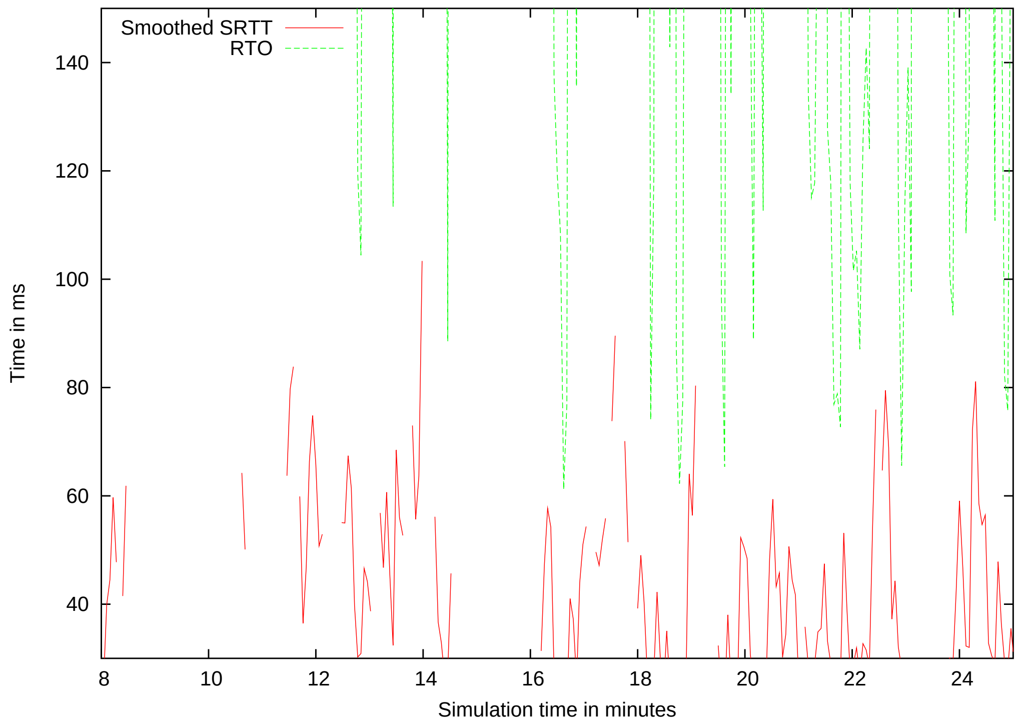


Fig. 5.31 Magnification of Fig. 5.30: Smoothed SRTT and RTO for a BER of $2 \cdot 10^{-5}$ and a Buffer Size of 3.3 kB

Table 5.7 FTP Response Times for various Receive Buffer Sizes

Advertised Receive Buffer Size	FTP Response Time^{ab} [min]
64000 Bytes	20.89
3300 Bytes	39.70
MSS	574.30

^{a} Reported is the time to successfully transfer a 250 MB file over an error-free link.

^{b} Results are independent of used TCP flavor and TCP timer granularity G as long as G is coarse enough to avoid false retransmissions.

The worst degradation of FTP performance can be observed for the TCPmin flavor advertising a receive buffer of 64 kB. As this TCP flavor lacks of the the Fast Retransmit and Recovery algorithm (see Section 4.1.3.2), TCP recognizes lost segments only after the RTO timer expires. The congestion window is entirely closed and the transmission is restarted with the lost segment onwards. As every closure of the congestion window directly results in (short) periods of low link utilization, the FTP response time is naturally larger than the one encountered by TCP Reno which avoids closing the congestion window. A tremendous performance improvement is reached by reducing TCPs timer granularity to 250 ms. As the finer granularity allows TCP to better approximate the current SRTT, a loss may be detected earlier. Further reducing the timer granularity improves FTP performance only marginally. Timer granularities of 100 ms and below may even result in a degradation. The finer the timer granularity, the more sensitive is TCP to sudden changes in the RTT which occur for every collapse of the congestion window after a segment loss. Thus, these extremely fine timer granularities cause several premature retransmissions as the RTO elapses before the first transmitted segment arrives.

In contrast to TCPmin with an advertised receive buffer of 64 kB, TCP Reno (with the same buffer size) and any TCP flavor announcing an receive buffer size equivalent to the MSS are hardly effected by the low bit error rate. For both cases, the experienced RTT is determined by the advertised receive buffer size: A buffer size of 1460 Byte (MSS) does result in a rather constant SRTT which is determined by TCP's delayed ACK mechanism; opening the CWND larger than one segment is therefore useless and any possible reduction algorithm of the CWND in case of segment losses has no effects on the FTP response time. The TCP Reno flavor in turn employs the Fast Retransmit and Recovery algorithm. As the rather low BER does only occasionally cause a segment to be lost, TCP Reno detects these losses far in advance before the RTO elapses. Only the lost segment is retransmitted and the congestion window is not entirely closed. This results in an almost optimal performance of nearly 1 without regard to the used timer granularity. It should be noted, that, for the advertised receive buffer of 64 kB, the timer granularity has no effect on TCP's ability to reflect the variable propagation delay as the latter is dominated by the rather constant queuing delay.

FTP connections which advertise a receive buffer size of 3.3 kB and thus utilize approximately half of the available bandwidth are degraded for both TCP flavors: Reno and "Min". Again, TCP Reno's Fast Retransmit and Recovery Algorithm outperforms TCP Min for all probed timer granularities. Occasionally, more than one segment is lost per RTT which causes TCP Reno to collapse its CWND as well. In these cases, an accurate estimation of the current (variable) RTT increases FTP performance which can be deduced from larger normalized FTP response timer for a timer granularity of 500 ms. Again, extreme fine granularities below 100 ms may cause false retransmissions and degrade TCP performance.

In the presence of high bit error rates, e.g. $2 \cdot 10^{-5}$ as illustrated in Fig. 5.33, performance does not any more depend on the used TCP flavor. Due to severe segment loss (i.e. more than one segment per RTT), TCP Reno cannot prevent the congestion window to entirely collapse. Thus, TCP Reno and TCPmin behave the same and their performance does only depend on the advertised receive buffer size and applied timer granularity. Fine timer granularities below 100 ms result in a high variance of the measured FTP response time as they cause the RTO to prematurely elapse several times. Even though the RTO triggers retransmissions before the corresponding ACK could have arrived, the retransmissions are (by accident) necessary as the severe BER caused the timed segment to be lost. Even though this behavior results partially in better results than a rather coarse timer granularity would, it is not desired as it is unpredictable.

The reader should explicitly notice that these experiments do only reveal how prone a TCP connection is to different BERs for different buffer sizes and timer granularities. Still, the total transmission time of the 250 MB file over the erroneous link (BER of $2 \cdot 10^{-5}$) is optimized for large receive buffers as they utilize the link's bandwidth the best.

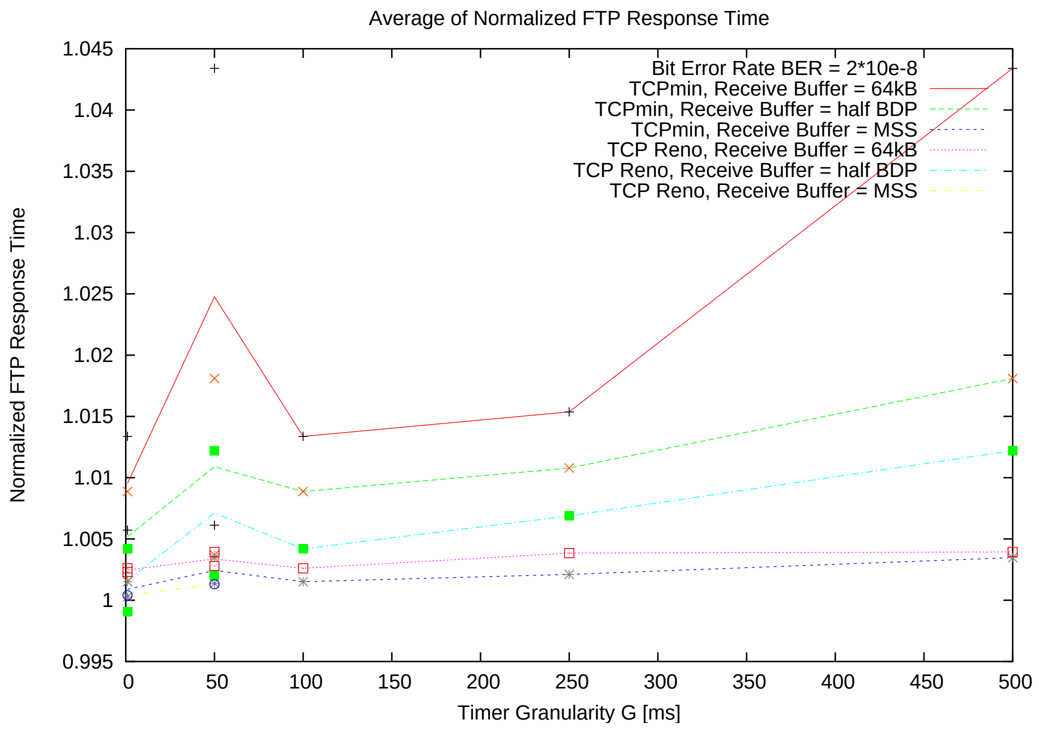


Fig. 5.32 Normalized FTP Response Time ($BER = 2 \cdot 10^{-8}$, various receive buffer sizes and TCP flavors)

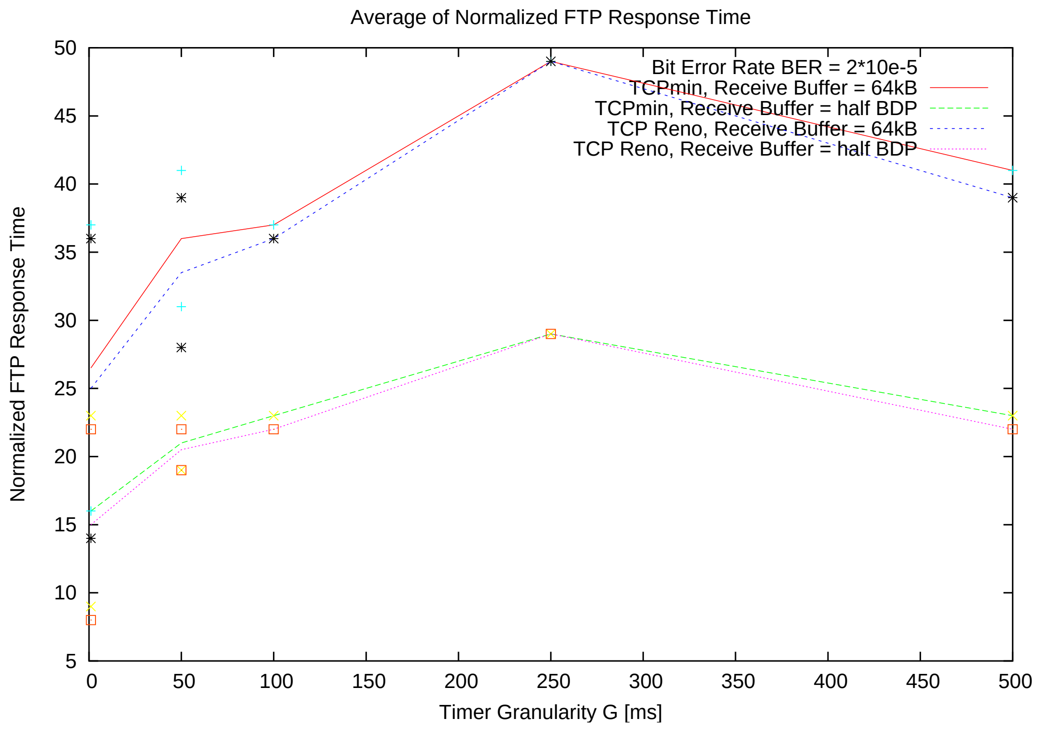


Fig. 5.33 Normalized FTP Response Time ($BER = 2 \cdot 10^{-5}$, various receive buffer sizes and TCP flavors)

6 Conclusions

6.1 CONCLUSIONS

The presented analysis of TCP performance over an erroneous LEO satellite link answered the research problems stated in Section 5.1:

1. TCP is well able to reflect the variable propagation delay of the LEO satellite link in the presence of low BERs only. For high BERs, neither the smoothed SRTT nor the RTO reflect the variable delay. Premature retransmissions of presumingly lost segments do not occur.

2. A reduced timer granularity G has no influence on this ability as long as the reduction does not cause premature collapses of the RTO for an error-free environment.

3. Reducing the TCP timer granularity G to 250 ms results in a justifiable performance improvement. Finer timer granularities require a link-specific reduction. For the latter, the gained performance improvement is negligible with regard to the effort to find these hand-tuned timer granularities.

4. TCP's performance is dominated by the employed retransmission mechanisms, i.e. the TCP flavor, rather than by the advertised receive buffer size or TCP's timer granularity.

5. For a given TCP flavor, large advertised receive buffers result in the best throughput but experience still the most relative performance degradation (in terms of normalized FTP response time) in the presence of an erroneous link.

6. Receive buffers larger or equal to the BDP result in a rather constant SRTT which is dominated by the large queuing delay. The domination of the queuing delay over the variable propagation delay of the LEO link has no influence on TCP performance.

These results neglect any proposals to modify the equations applied to the measured RTT in order to calculate a RTO. The focus on improving TCP performance over the network specified within the ATM-Sat project should be on large enough buffer sizes which can easily be assured by simple socket options. As TCP Reno is rather compiled into commercial off the shelf operation systems than Vanilla TCP, the effort to manually tune the timer granularity is not worth the possible performance improvements. Additionally, some operation systems couple the timer granularity to the system clock's frequency which makes an isolated tuning of the timer granularity complicated.

6.2 SUMMARY OF CONTRIBUTIONS

In order to analyze the effects of advertised receive buffer sizes and timer granularities on TCP performance over an erroneous LEO satellite link, a TCP simulation model has been developed (Section 5.3.3.2). This model includes over 200 experiments specified by different TCP flavors, advertised buffer sizes, timer granularities, and BERs (Section 5.4.2). It allows to measure the FTP response time (from which the throughput can be deduced) for a given file size and includes a new pipeline stage architecture which allows to impose a satellite specific variable propagation delay on packets being transmitted via a point-to-point duplex link (Section 5.3.3.1).

This new contributed propagation delay pipeline stage function is based upon the simulation of the analyzed satellite constellation (Section 5.3.1). A description of the end-to-end propagation delay was derived and further analyzed with respect to its variability. The up- and down-link delay was proven to have the most predominant influence on the delay's variability for all possible ISLs occurring in a shortest-path routing based network. Thus, the final propagation delay model could be restricted to the mere up- and down-link delay. Due to this analysis, an Opnet model for a point-to-point link imposing satellite specific propagation delays has been implemented and an interface in between STK and Opnet was provided to allow an easy exchange of propagation delay descriptions from STK to Opnet. The proper functionality of the simulation model has been verified for six different satellite constellations including ten propagation delay characteristics with different variability constrains (Sections 5.4.1 and 5.5.1).

The simulation results contributed to a deeper understanding of TCP's timer mechanisms. Experiments (Section 5.5.2) showed that a rather fine timer granularity has only a minor influence on TCP's ability to reflect a variable (satellite link) propagation delay. The performance improvement expected to come along with fine timer granularities was rather low for values of G below 250 ms which in turn does not justify a hand-tuned TCP implementation with respect to the project framework of this work. Instead, the results induce that the best possible performance improvement of TCP can be derived from optimized congestion control and retransmission algorithms which possible include ways to differ in between congestion and corruption-based losses.

6.3 FUTURE PROSPECTS AND OPEN ISSUES

Future work should back up these simulation based results by conduction real-time experiments, i.e. connecting computers over a satellite link or a satellite link simulator in order to conduct FTP transmissions. These experiments might reveal some aspects of real-world, implementation specific characteristics of the TCP protocol stack. One of these aspects to consider might be, e.g., the TCP timestamp option which is always enabled for FreeBSD systems. [7] First experiments indicate, that the internal "tick rate" to which the timer granularity is directly coupled has no influence on a possible premature segment retransmission. [37] Besides, the simulations should be extended to include competing traffic which might be TCP- or non-TCP-based.

Additionally, a more accurate model of hand-overs should be included: Up to now, handover is assumed to be caused by satellites disappearing at the horizon which results in a periodic repetition of the up- and downlink delay. If handover is caused by shadowing, the change of the propagation delay might be by far more abrupt. Besides, a handover does usually imply to switch existing connections from one satellite to another which results on possible additional queuing delays if packets have to be buffered (during the handover) to prevent losses. Otherwise, packet loss occurs frequently due to hand-overs and will most likely influence TCP performance.

Up to now, several TCP flavors have been proposed which differ from congestion and packet loss or probe the link for the available bandwidth (Section 1.1.3). These modified algorithms do still have to be carefully evaluated in a wider network environment to evaluate the impact of these modifications and enhancements on other traffic flows on the internet.

Finally, the author envisions a possible extension of the presented simulation workflow (Section 5.2) to allow an exchange of simulation models (i.e. protocol stack descriptions) in between Opnet and a measurement environment including real (FreeBSD-based) systems communicating over a (possible emulated) satellite link. [36]

Appendix A Fourier Series Based Approximation of Propagation Delay using STK Measurements

The simulation model described in Section 5.3.2 approximates intermediate values of the propagation delay in between two STK measurements either by a "sample and hold" method or linear interpolation. As the period of the propagation delay is known to be $T = 112.13 \quad minutes$, one can use the delay samples reported by STK as an input for a Fast Fourier Analysis to gain Fourier Coefficients for a better approximation.

Let $x[n], \; n \in [0, N-1]$ be the n-th sample of the delay reported by STK and assume the delay to be periodic with a period T, then the k-th Fourier Coefficient a_k is defined by

$$a_k = \frac{1}{N} \sum_{k=0}^{N-1} x[n] e^{-jk(2\pi/N)n} \tag{A.1}$$

As all samples $x[n]$ are real values, $a_k = a^*_{-k}$. The absolute values are plotted in Fig. A.1 and reveal the major influence of the first three harmonics which suggests that an approximation using a Fourier Series with all N components might not be necessary.

To prove this assumption, the analytical propagation delay is compared with a linear interpolation of the STK measurements and an approximation of the delay using a Fourier Series. The latter gives intermediate delay values according to

$$x[n] = \sum_{k=0}^{N-1} a_k e^{-jk(2\pi/N)n} \tag{A.2}$$

which are illustrated in Fig. A.2. The difference in between the linear interpolation and the Fourier Series based approximation is neglect-able. [61, 65]

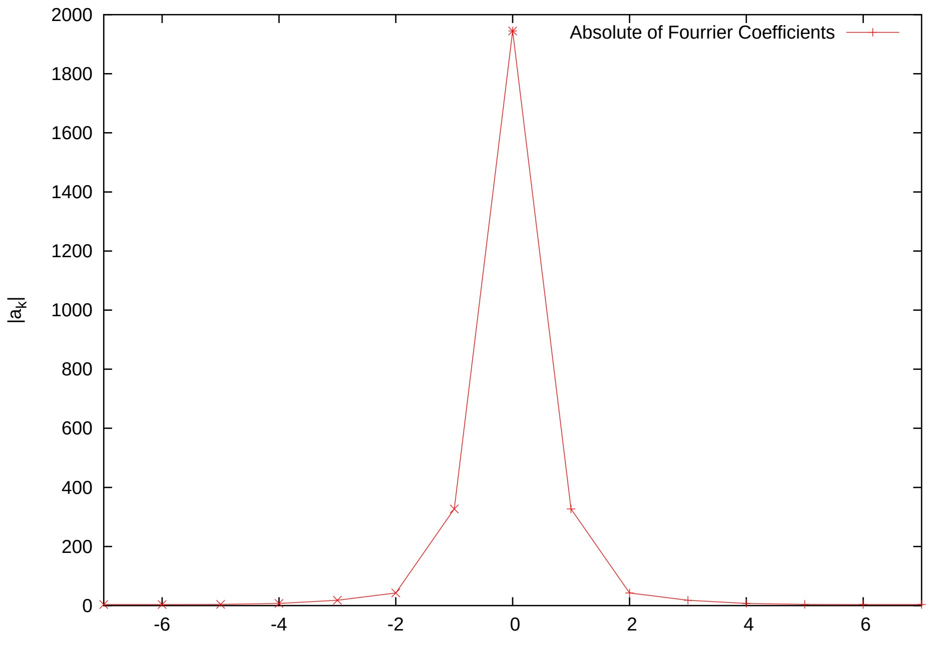


Fig. A.1 Fourier Coeffcients for STK based delay measurements (absolute values)

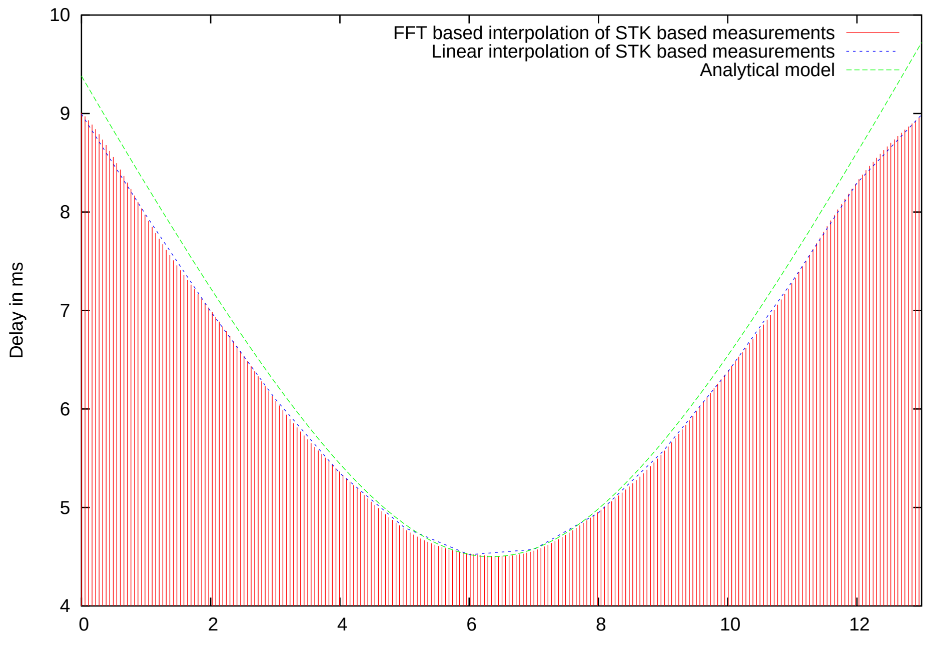


Fig. A.2 Comparison of analytical delay with linear interpolation of STK based measurements and an Fourier based interpolation

Appendix B Delay Analysis of Various Satellite Constellations

Allman et al analyzed in [14] TCP over various variable delay satellite channels. Their analysis included only TCP's ability to reflect the variable delay in its internal timer structures. Erroneous links and various receive buffer sizes were not discussed.

In order to verify the simulation model used for this work, the satellite constellation used in [14] were analyzed with STK to gain corresponding AER descriptions. The latter were transferred into the point-to-point transceiver pipeline stage for the propagation delay. The delay was then measured in Opnet. This work was only included into this simulation process to have a set of well known results in order to verify the functionality of the underlying simulation model. Table 5.1 summarizes the analyzed satellite constellations and their corresponding period.

It should be noted that in addition to the measured delay, Fig. B.1 includes the analytical delay of the up- and down-link for the first period.

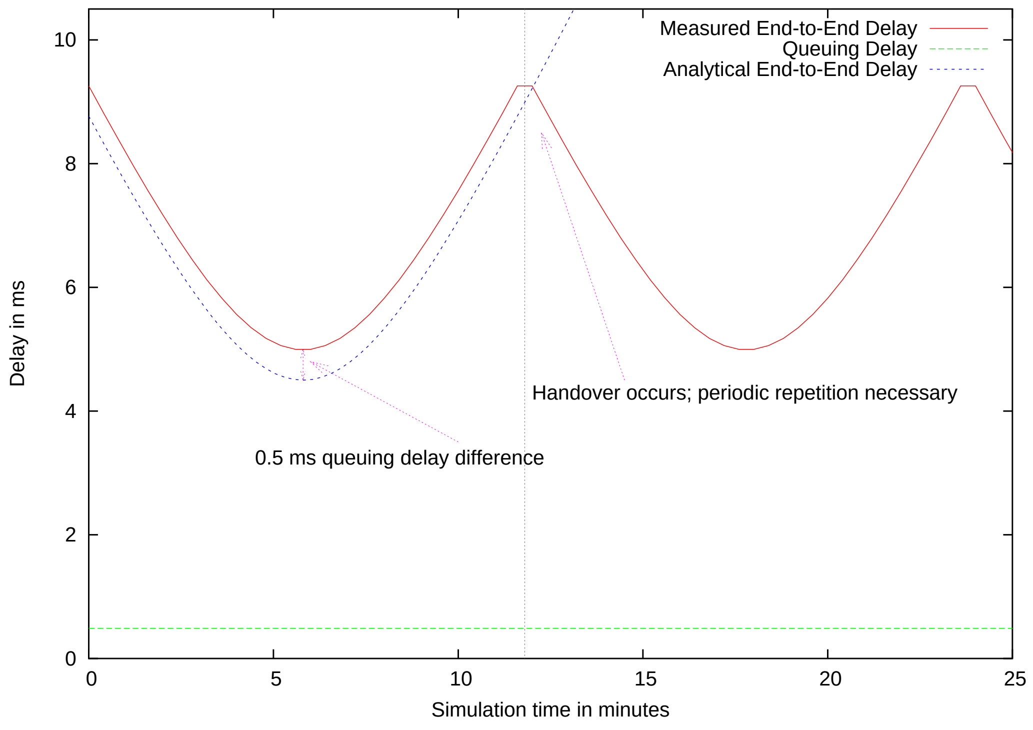


Fig. B.1 Delay ATM-Sat LEO Constellation

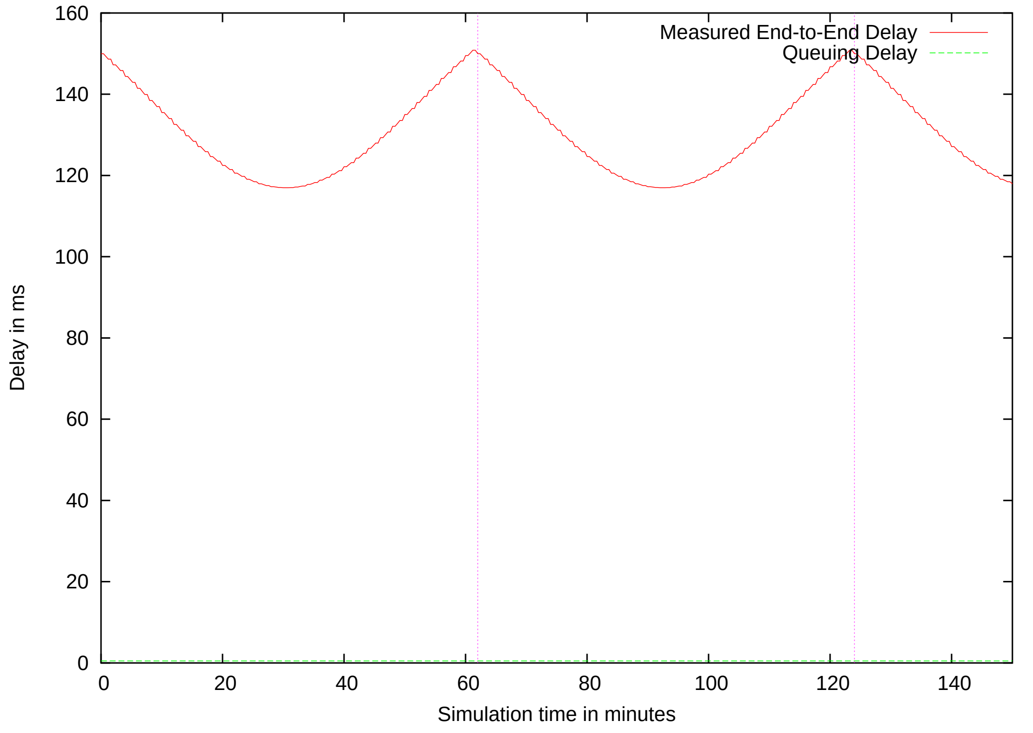


Fig. B.2 Cobe Delay with Maximal Variation

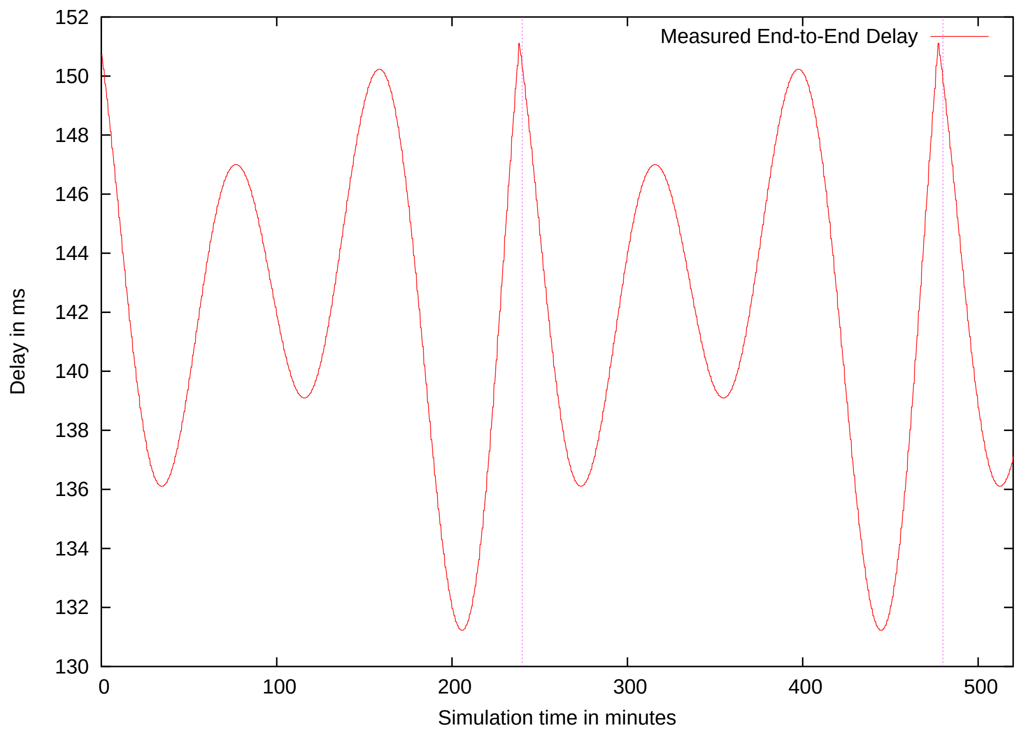


Fig. B.3 Cobe Delay with Minimum Variation

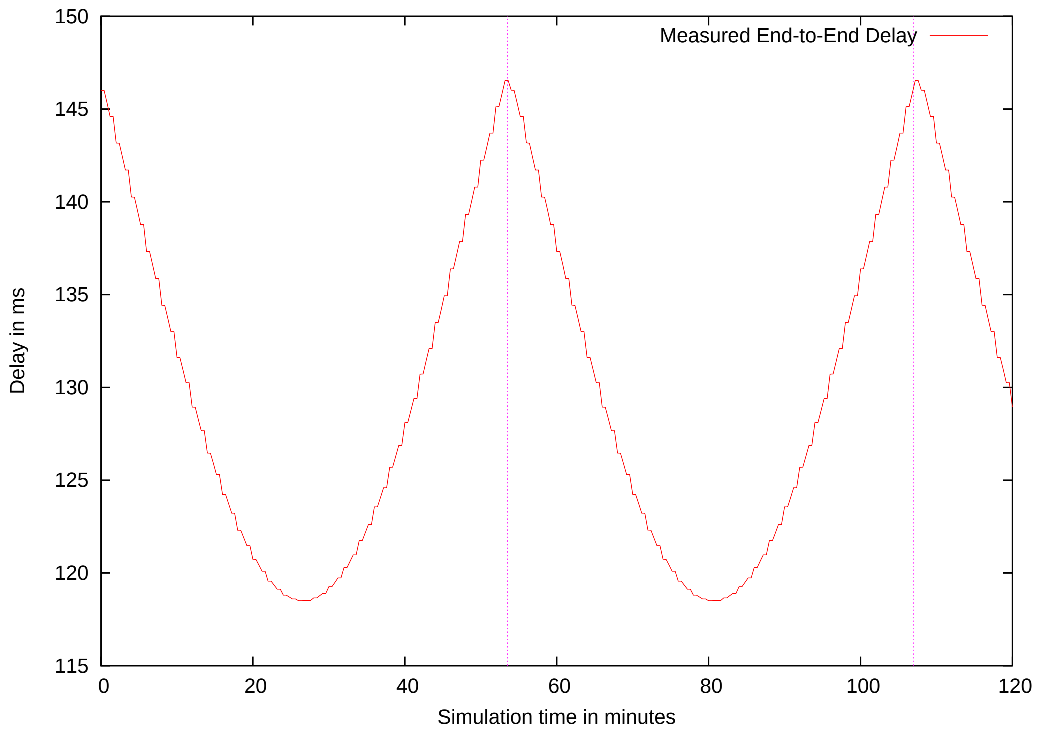


Fig. B.4 ISS Delay with Maximum Variation

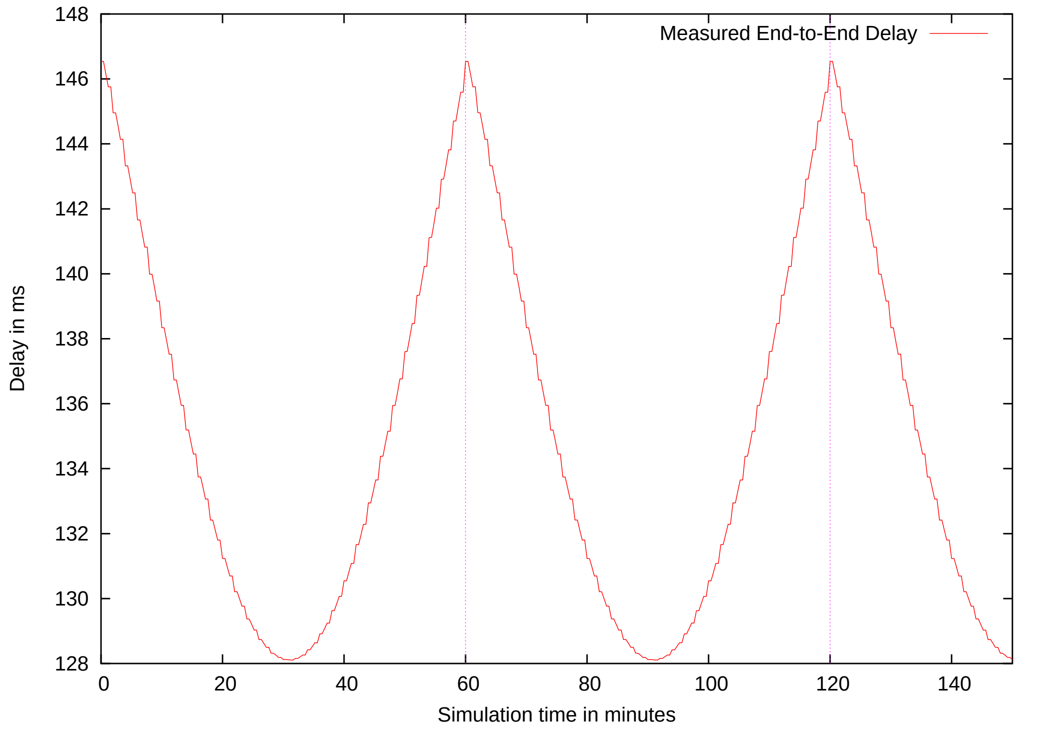


Fig. B.5 ISS Delay with Minimum Variation

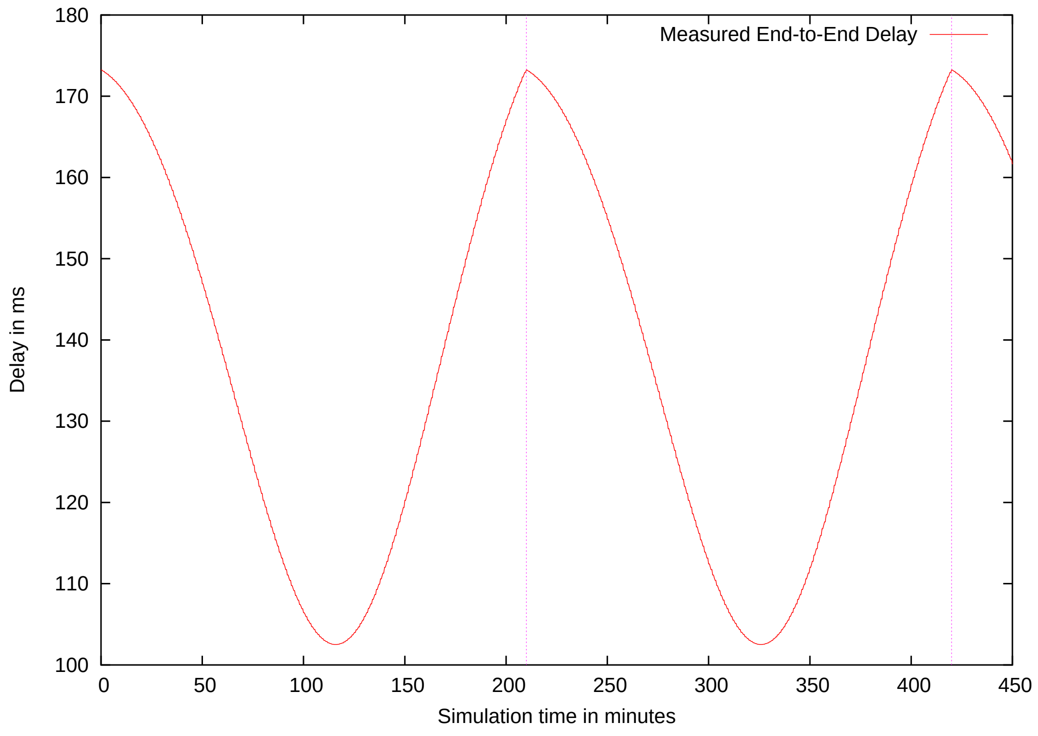


Fig. B.6 Lageos Delay with Maximum Variation

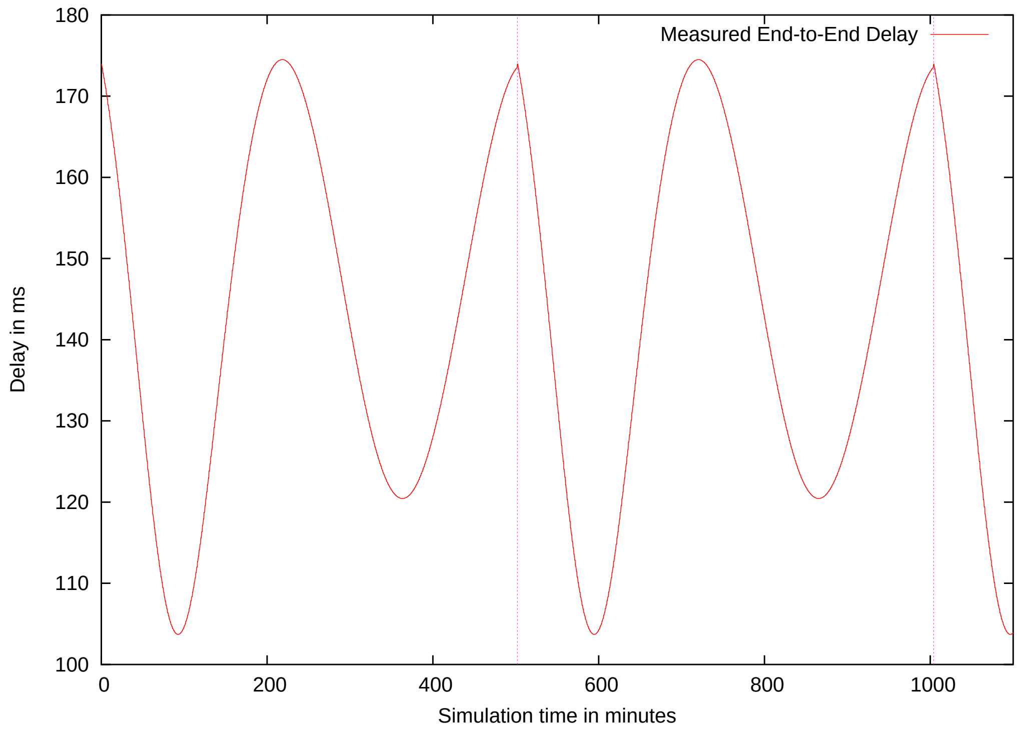


Fig. B.7 Lageos Delay with Minimum Variation

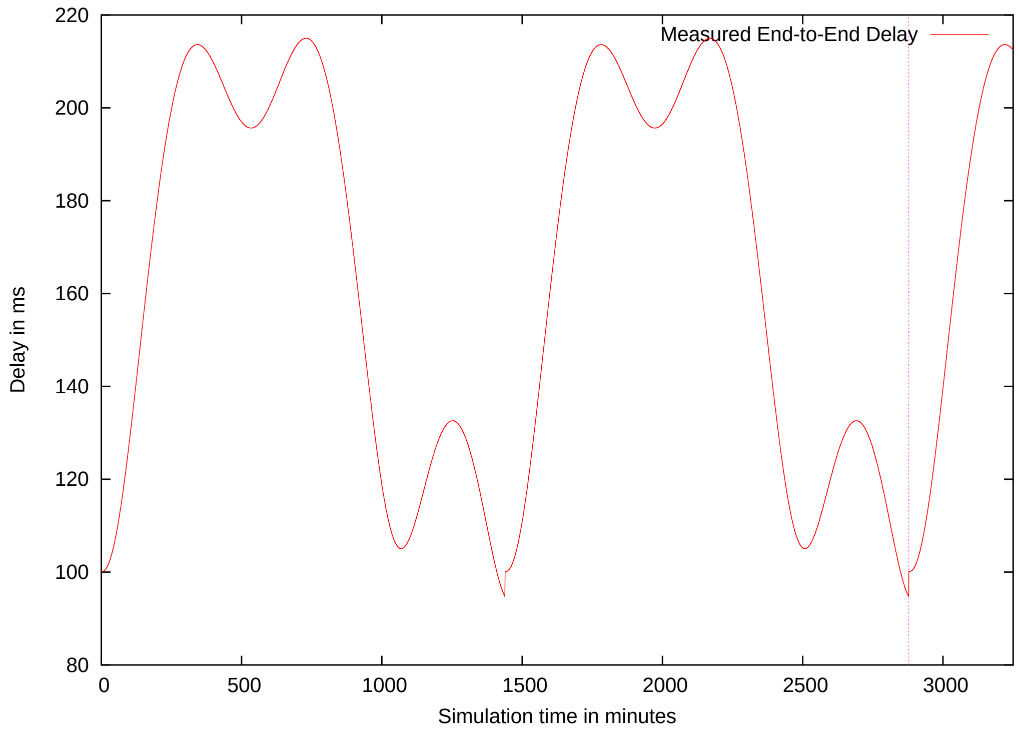


Fig. B.8 Navstar Delay

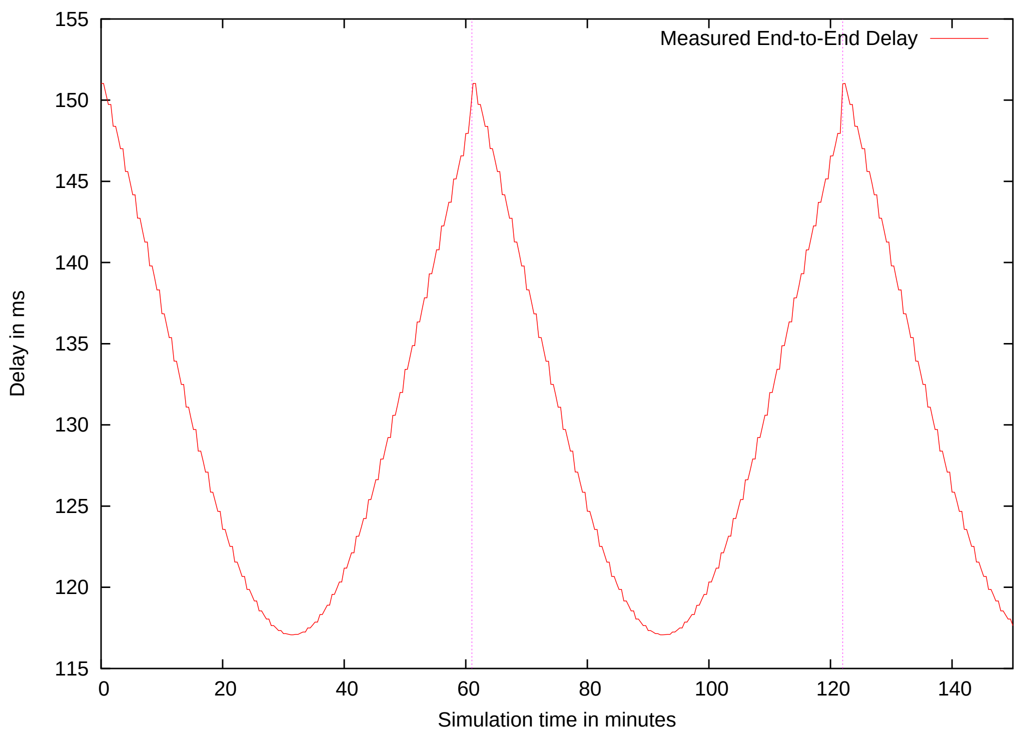


Fig. B.9 Radcal Delay with Maximum Variation

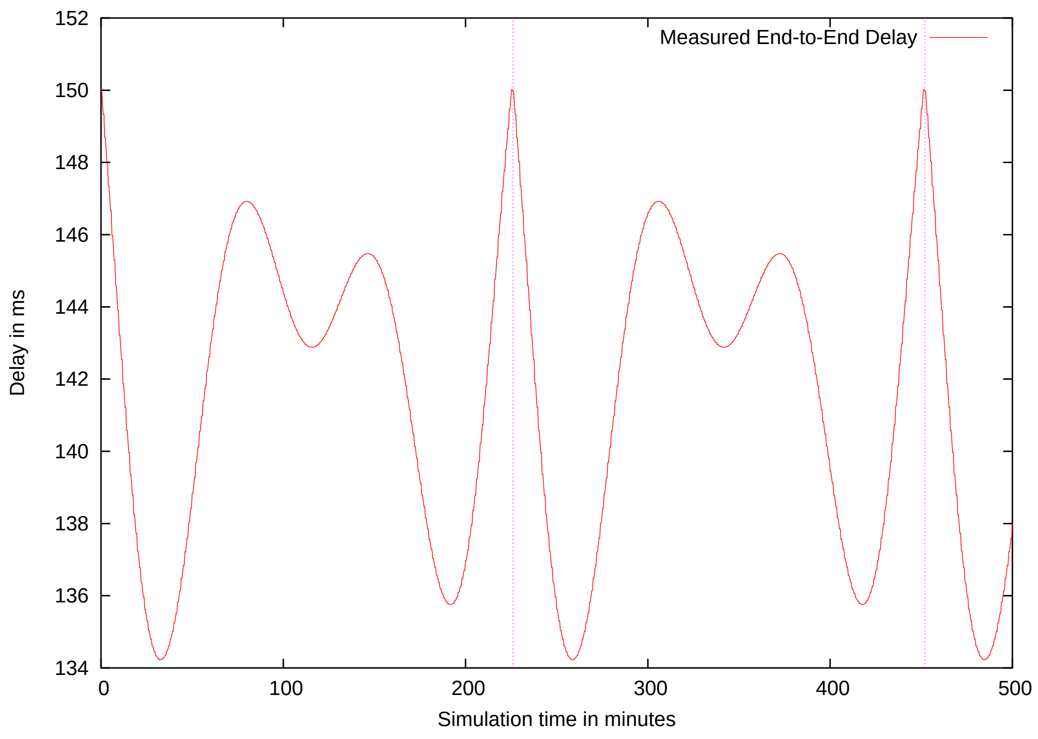
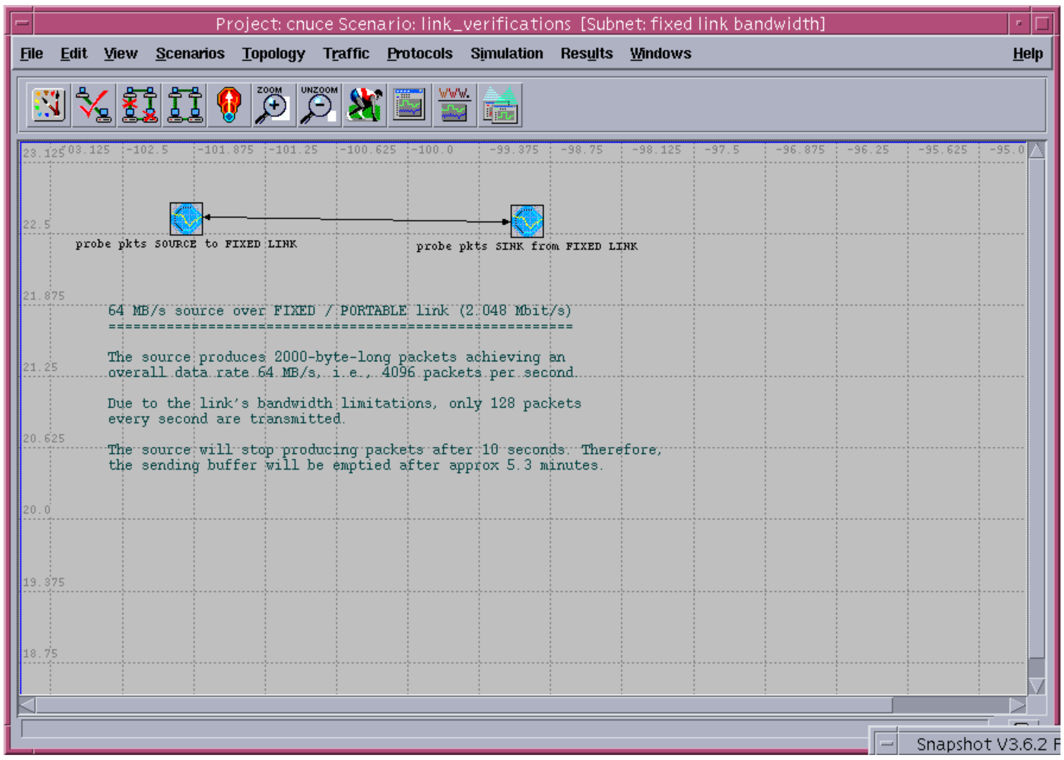


Fig. B.10 Radcal Delay with Minimum Variation

Appendix C
Auxiliary Measurement Results

The ATM-Sat Project [6] specifies the satellite link to provide either 384 kbit/s, 2.048 Mbit/s, or 32.768 Mbit/s as available bandwidth. [16] In order to conduct the simulations of TCP, an appropriate simulation model of the satellite channel had to be developed and verified with respect to the imposed propagation delay and bandwidth limitations. The verification of the mobile user link (384 kbit/s) is presented in Sections 5.4 and 5.5. The verification of with respect to the fixed terminal is presented here. The simulation results with respect to the provider link resemble do not yield to further insight into the simulation model as they merely verify its proper functionality, and are therefore omitted here.

The supporting network nodes (refer to Section 5.3.3.1) are connected via the developed pipeline stage architecture (link model) representing the satellite channel. The latter is configured to limit the bandwidth to the one associated with a fixed terminal. The network node acting as a sink (see Fig. C.1) sends 200-byte-long packets at a data rate of 4096 packets/s achieving a data volume production of 64 MB/s. The production of packets is stopped after 10 seconds which leaves a total amount of 640 MB to be transferred via the link. As expected, the link limits the bandwidth to 2.048 Mbit/s as Fig. C.2 illustrates: The link utilization is constantly reported to be 100% for a period of approximately 5.3 minutes.


Project: cnuce Scenario: link_verifications [Subnet: fixed link bandwidth]
File Edit View Scenarios Topology Traffic Protocols Simulation Results Windows Help
ZOOM
UNZOOM
probe pkts SOURCE to FIXED LINK
probe pkts SINK from FIXED LINK
Snapshot V3.6.2

```
64 MB/s source over FIXED / PORTABLE link (2.048 Mbit/s)
========================================================

The source produces 2000-byte-long packets achieving an
overall data rate 64 MB/s, i.e., 4096 packets per second.

Due to the link's bandwidth limitations, only 128 packets
every second are transmitted.

The source will stop producing packets after 10 seconds. Therefore,
the sending buffer will be emptied after approx 5.3 minutes.
```

Fig. C.1 Testing Scenario of Fixed Terminal Bandwidth Limitations

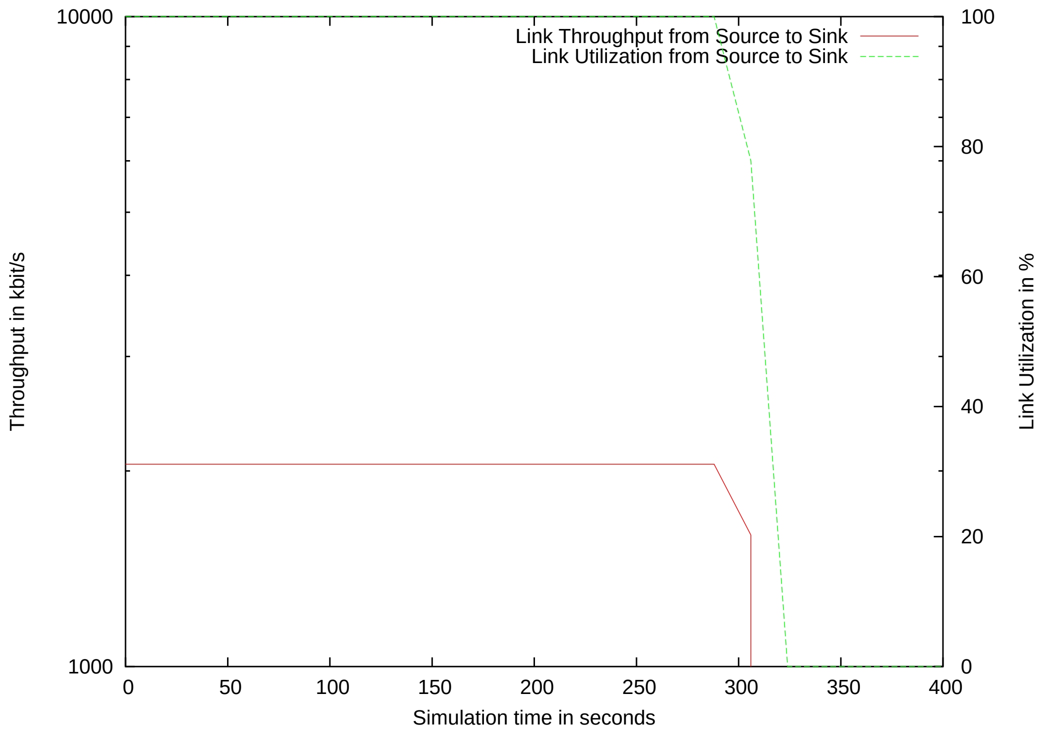


Fig. C.2 Verification of Bandwidth Limitations of Fixed Terminal Link

Appendix D
Target System Parameters

D.1 SATELLITE CONSTELLATION

The ATM-Sat project envisioned an orbit altitude of 1300 – 1400 km and an minimum elevation angle of 20° – 30°. For this constellation, the required number of satellites to ensure a continuous coverage is at least 20. [76] Nevertheless, a number of 70 – 100 satellites is considered to achieve multiple satellite visibility which in turn improves service availability. To guarantee a good coverage apart from the polar regions, a Walker constellation with a typical inclination in the range of 45° to 55° is considered. Table D.1 summarizes the satellite constellation parameters as they are agreed on by the ATM-Sat consortium. [18]

Table D.1 Constellation Parameter for the ATM-Sat Target System

Number of Satellites	72
Orbit Altitude	1350 km
Orbit Period	112.7 min.
Inclination	47°
Phasing	25°
Minimum Elevation Angle	20°
Satellite Visibility	$\approx$ 12 min.

D.2 LINK BUDGET

The following tables show the link budget for the uplink and downlink of the ATM-Sat target system. The satellite Tx and Rx antenna gain has been adjusted to a spotbeam size in the order of 100 – 500 km (depending on the elevation angle). As a rain attenuation (atmospheric losses) of 4.3 dB in the uplink and of 2.8 dB in the downlink is already included in the link budget, a low link margin is acceptable. [18]

Table D.2 Basic assumptions for link budget calculation

Effective maximum distance	2700 km
Elevation angle	20°
Earth station antenna efficiency	0.6
On board antenna efficiency	0.6
Link availability	99%

Table D.3 Linkbudget for uplink (high bit rate fixed and portable terminal)

Bit Rate (with ATM encapsulation)	2447 kbit/s
Coding RS(65,53)	1.226
Up link data rate (with coding)	3000 kbit/s
Up link frequency	30 GHz
Earth station transmitted power	8 W
Output back-off	0 dB
Transmitted power	9.03 dBW
Tx antenna gain	35 dBi
Pointing error	0.3 deg
Pointing losses	0.35 dB
Output losses	1 dB
EIRP	42.68 dBW
Free space losses	190.60 dB
Atmospheric losses (99%)	4.3 dB
Mobile channel losses	2 dB
Polarisation losses	0.5 dB
Interference allowance	0.3 dB
Satellite Rx ant. gain at -4.3 dB	31.64 dBi
Rx losses	0.6 dB
Temperature of connection	290 $^\circ$K
Antenna temperature	290 $^\circ$K
Receiver noise figure	1.9 dB
Receiver noise temperature	159.15 $^\circ$K
System noise temperature	449.15 $^\circ$K
Satellite Ts	26.52 dB $^\circ$K
Satellite G/T	4.52 dB/$^\circ$K
C/N0	77.5 dBHz
Data Rate (2447 kbit/s)	63.89 dBHz
Implementation losses	1.5 dB
Eb/N0	12.11 dB
Req. Eb/N0 at CER 10^{-10}	7 dB
Link Margin	5.11 dB

Table D.4 Linkbudget for uplink (low bit rate portable and mobile terminal)

Bit rate (with ATM encapsulation)	245 kbit/s
Coding RS(65,53)	1.226
Up link data rate	300 kbit/s
Up link frequency	30 GHz
Earth station transmitted power	4 W
Output back-off	0 dB
Transmitted power	6.02 dBW
Tx antenna gain	25 dBi
Pointing error	0.3 deg
Pointing losses	0.35 dB
Output losses	1 dB
EIRP	29.66 dBW
Free space losses	190.60 dB
Atmospheric losses (99%)	4.3 dB
Mobile channel losses	2 dB
Polarisation losses	0.5 dB
Interference allowance	0.3 dB
Satellite Rx ant. gain at -4.3 dB	31.64 dBi
Rx losses	0.6 dB
Temperature of connection	290 °K
Antenna temperature	290 °K
Receiver noise figure	1.9 dB
Receiver noise temperature	159.15 °K
System noise temperature	449.15 °K
Satellite Ts	26.52 dB°K
Satellite G/T	4.52 dB/°K
C/N0	65.08 dBHz
Data Rate	53.89 dBHz
Implementation losses	1.5 dB
Eb/N0	9.69 dB
Req. Eb/N0 at CER 10^{-10}	7 dB
Link Margin	2.69 dB

Table D.5 Link budget for downlink (high bit rate fixed and portable terminal)

Bit rate (with ATM encapsulation)	30000 kbit/s
Coding RS(70,56)	1.25
Down link data rate	37500 kbit/s
Down link frequency	20 GHz
On board power per carrier	40 W
Output back-off	2 dB
Transmitted power	14.02 dBW
Tx antenna gain	31.5 dBi
Output losses	1.5 dB
EIRP/carrier (peak)	44.02 dBW
EIRP/carrier at -4.3 dB	39.72 dBW
Free space losses	187.08 dB
Atmospheric losses (99%)	2.8 dB
Mobile channel losses	2 dB
Polarisation losses	0.5 dB
Interference allowance	0.3 dB
Earth Station antenna gain	36.24 dBi
Pointing error	0.3 deg
Pointing losses (with tracking)	0.15 dB
Rx losses	0.2 dB
Sky temperature	30 °K
Antenna temperature	153.55 °K
Receiver noise figure	1.3 dB
Receiver noise temperature	101.20 °
Earth terminal system temperature	260.89 °
Earth terminal Ts	24.16 dB°
Terminal G/T	11.72 dB/°K
C/N0	87.01 dBHz
Data Rate (30 Mbit/s)	74.77 dBHz
Implementation losses	1.7 dB
Eb/N0	12.24 dB
Req. Eb/N0 at CER 10^{-10}	7 dB
Link Margin	5.24 dB

Table D.6 Link budget for downlink (low bit rate portable and mobile terminal)

Bit rate (with ATM encapsulation)	3000 kbit/s
Coding RS(70,56)	1.25
Down link data rate	3750 kbit/s
Down link frequency	20 GHz
On board power per carrier	40 W
Output back-off	2 dB
Transmitted power	14.02 dBW
Tx antenna gain	31.5 dBi
Output losses	1.5 dB
EIRP/carrier (peak)	44.02 dBW
EIRP/carrier at -4.3 dB	39.72 dBW
Free space losses	187.08 dB
Atmospheric losses (99%)	2.8 dB
Mobile channel losses	2 dB
Polarisation losses	0.5 dB
Interference allowance	0.3 dB
Earth Station antenna gain	26.24 dBi
Pointing error	0.3 deg
Pointing losses (with tracking)	0.15 dB
Rx losses	0.2 dB
Sky temperature	30 °K
Antenna temperature	153.55 °K
Receiver noise figure	1.3 dB
Receiver noise temperature	101.20 °K
Earth terminal system temperature	260.89 °K
Earth terminal Ts	24.16 dB°K
Terminal G/T	1.72 dB/°K
C/N0	77.01 dBHz
Data Rate (3 Mbit/s)	64.77 dBHz
Implementation losses	1.7 dB
Eb/N0	12.24 dB
Req. Eb/N0 at CER 10^{-10}	7 dB
Link Margin	5.24 dB

Appendix E
Auxiliary Tools for AER Data Conversion from STK to Opnet

The following C-code simply reads slant ranges from a text file (which were previously reported by STK as part of the AER report) and automatically generates an h-file with the predefined propagation delay structure as expected by the $get_stk_delay()$ function (ref. to Section 5.3.3.1 and Appendix F.2.1).

```
#include <stdio.h>
#include <math.h>

main()
{
        int lines =0; /* produced lines of delay-entries */

        float range, delay;

        /* include file header */
        printf("#ifndef FILENAME_H \n");
        printf("#define FILENAME_H \n");
        printf("#ifdef  __cplusplus \n");
        printf("extern \"C\"{ \n");
        printf("#endif \n");


        /* anything we have to include ... */
        printf("\n\n");
        printf("#include \"stk_delay_types.h\"\n");

        printf("\n");

        /* print first lines of delay array ...*/

        printf("const DELAY_TYPE spesify_array_name [] = {\n");
        printf("\t_specify_orbit_period_T,\n");
        printf("\t_specify_granularity_of_vektor_elements");
                /* attention: not ,\n in last printf here !! */

        while( scanf("%f", &range) != EOF ) {
                delay = range / 3e5;
                printf(",\n");
                printf("\t%f", delay);
                lines ++;
        }

        printf("\n};\n");

        printf("/* produced %d lines with delay entires */", lines);

        /* include file trailer */
        printf("#ifdef  __cplusplus\n");
        printf("}\n");
        printf("#endif\n");
        printf("#endif\n");

}
```

Appendix F
Opnet Models

F.1 POINT-OT-POINT PIPELINE STAGE PROPAGATION DELAY PROCEDURE

```
/* gmd_vardelay_dpt_propdel.ps.c */
/* Default propagation delay model for pt-to-pt link Transceiver Pipeline */
/* modified for the ATM SAT Project */

/****************************************/
/*         Copyright (c) 1993-2000      */
/*       by OPNET Technologies, Inc.      */
/*       (A Delaware Corporation)      */
/*    3400 International Drive,  N.W.      */
/*       Washington, D.C., U.S.A.      */
/*          All Rights Reserved.      */
/****************************************/

#include "opnet.h"
#include <math.h>
#include "link_delay.h"
#include "gmd_stk_delay.h"

#define DISTANCE_BASED   1E+15

/* Constants to use if stk_delay files are used */
#define   COBE_MIN_VAR_CST      999999001
#define COBE_MAX_VAR_CST      999999002
#define ISS_MAX_VAR_CST         999999003
#define ISS_MIN_VAR_CST         999999004
#define LAGEOS_MAX_VAR_CST      999999005
#define LAGEOS_MIN_VAR_CST      999999006
#define NAVSTAR_MAX_MIN_VAR_CST   999999007
#define RADCAL_MAX_VAR_CST      999999008
#define RADCAL_MIN_VAR_CST      999999009


/* Constant to use if the approximation function of delay as
   experienced within the ATM-Sat Project is added */
#define ATMSAT_LEO      1E+16

#if defined (__cplusplus)
```

```
extern "C"
#endif
void
gmd_vardelay_dpt_propdel (Packet * pkptr)
   {
   Objid      link_objid;
   double      prop_delay;
   double      x; /* tmp var */


   /** Compute propagation delay associated with a      **/
   /** packet transmission on a point-to-point link.   **/
   FIN (gmd_vardelay_dpt_propdel (pkptr));

   /* Obtain object id of link carrying transmission. */
   link_objid = op_td_get_int (pkptr, OPC_TDA_PT_LINK_OBJID);

   /* Obtain propagation delay associated with that link. */
   if (op_ima_obj_attr_get (link_objid, "delay", &prop_delay) == OPC_COMPCODE_FAILURE)
      op_sim_end ("Error in point-to-point propagation delay pipeline stage (dpt_propdel):",
         "Unable to get propagation delay from link attribute.", OPC_NIL, OPC_NIL);

   if (prop_delay == DISTANCE_BASED)
       {
       prop_delay = link_delay(link_objid);
       }


   /* The following section sets the delay according to the look-up table
      derived via stk.
      Please refer to the gmd_stk_delay.h file.
   */

   if (prop_delay == COBE_MIN_VAR_CST)
      {
      prop_delay = get_stk_delay( op_sim_time(), cobe_minVar );
      }

   if (prop_delay == COBE_MAX_VAR_CST)
      {
      prop_delay = get_stk_delay( op_sim_time(), cobe_maxVar );
      }

   if (prop_delay == ISS_MAX_VAR_CST   )
      {
      prop_delay = get_stk_delay( op_sim_time(), iss_maxVar );
      }

   if (prop_delay == ISS_MIN_VAR_CST)
      {
      prop_delay = get_stk_delay( op_sim_time(), iss_minVar );
      }

   if (prop_delay == LAGEOS_MAX_VAR_CST)
      {
      prop_delay = get_stk_delay( op_sim_time(), lageos_maxVar );
      }

   if (prop_delay == LAGEOS_MIN_VAR_CST)
      {
      prop_delay = get_stk_delay( op_sim_time(), lageos_minVar );
      }

   if (prop_delay == NAVSTAR_MAX_MIN_VAR_CST)
      {
      prop_delay = get_stk_delay( op_sim_time(), navstar_maxMinVar );
      }

   if (prop_delay == RADCAL_MAX_VAR_CST)
      {
```

```
      prop_delay = get_stk_delay( op_sim_time(), radcal_maxVar );
      }

   if (prop_delay == RADCAL_MIN_VAR_CST)
      {
      prop_delay = get_stk_delay( op_sim_time(), radcal_minVar );
      }

   /* The following section sets the delay according to an approximative function
      describing the current (i.e., time-variiying) delay as experienced within
      the ATM-SAT Project */
   if (prop_delay == ATMSAT_LEO)
       {
      prop_delay = op_sim_time();
      x = prop_delay - (long) prop_delay; /* save the "type conversion" difference */
      prop_delay = (long) prop_delay % 720;
      prop_delay += x; /* add the difference lost due to type conversion */
      prop_delay -= 360;

      x = 2 * M_PI * prop_delay / 6762 ;
      prop_delay = sqrt( 6.37 * 6.37 + 7.72 * 7.72 - 2 * 6.37 * 7.72 * cos( x )) / 3E2;
       }

   /* Place propagation delay in packet transmission data attribute. */
   op_td_set_dbl (pkptr, OPC_TDA_PT_PROP_DELAY, prop_delay);

   FOUT;
   }

```

F.2 INTERFACE TO STK (EXTERNAL PIPELINE STAGE CODE)

F.2.1 Lookup Function get_stk_delay()

F.2.1.1 Interface Definition (h-file)

```
#ifndef GMD_STK_DELAY_H
#define GMD_STK_DELAY_H
#ifdef  __cplusplus
extern const "C"{
#endif

#include "stk_delays/stk_delay_types.h"

/*
 * Vektors with delays:
 *
 * 1st element: function period T
 * 2nd element: granularity of vektor, i.e., every xy time-tics, we switch
 *      to the next vector entry
 * 3rd-end: delays
 *
 * Example:
 *         const DELAY_TYPE my_delay[] = {
 *                   60,
 *                   10,
 *                   1,
 *                   2,
 *                   3,
 *                   4
 *         };
 *
 * Attention: Base time unit is SECONDS !!!
*/


/*
**  prototypes
*/
int calc_index( DELAY_TYPE, const DELAY_TYPE [] );
DELAY_TYPE get_stk_delay( DELAY_TYPE, const DELAY_TYPE []);

/*
** extern constal declarations for delay tables
*/

extern const DELAY_TYPE cobe_minVar [];
extern const DELAY_TYPE cobe_maxVar [];

extern const DELAY_TYPE iss_maxVar [];
extern const DELAY_TYPE iss_minVar [];

extern const DELAY_TYPE lageos_maxVar [];
extern const DELAY_TYPE lageos_minVar [];

extern const DELAY_TYPE navstar_maxMinVar [];

extern const DELAY_TYPE radcal_maxVar [];
extern const DELAY_TYPE radcal_minVar [];



#ifdef  __cplusplus
}
#endif
#endif

```

F.2.1.2 Interface Implementation (external C-code)

```
#include "gmd_stk_delay.h"
#include "gmd_stk_delay_definitions.h"

#include <opnet.h>
#include <math.h>

DELAY_TYPE get_stk_delay( DELAY_TYPE c_time, const DELAY_TYPE stk_delay_array[] )
{
        DELAY_TYPE      d = 0;

        FIN (get_stk_delay(c_time, stk_delay_array))

        d = stk_delay_array [calc_index( c_time, stk_delay_array )];

        FRET (d);
} /* of get_stk_delay */

int calc_index( DELAY_TYPE t, const DELAY_TYPE delays[])
{
        DELAY_TYPE t_temp;
        const DELAY_TYPE T = delays[0];
        const DELAY_TYPE delta = delays[1];
        int             idx;

        FIN (calc_index( t, delays))

        /* reduce given time to be within period T */
        t_temp = ( (t / T) -  (long) (t / T) ) * T;

        /* return index of array */
        idx = (int) (t_temp / delta) + DELAY_INDEX_OFFSET;

        FRET (idx);
}
```

F.2.2 STK based Delay Tables

F.2.2.1 Definition of Available Delay Patterns

```
#ifndef GMD_STK_DELAY_DEFINITIONS_H
#define GMD_STK_DELAY_DEFINITIONS_H
#ifdef  __cplusplus
extern "C"{
#endif

#include "stk_delays/stk_delay_types.h"

/*
 * Vektors with delays:
 *
 * 1st element: function period T
 * 2nd element: granularity of vektor, i.e., every xy time-tics, we switch
 *      to the next vector entry
 * 3rd-end: delays
 *
 * Example:
 *         const DELAY_TYPE my_delay[] = {
 *                   60,
 *                   10,
 *                   1,
 *                   2,
 *                   3,
 *                   4
 *         };
 *
 * Attention: Base time unit is SECONDS !!!
*/

#include "stk_delays/cobe_minVar.h"
#include "stk_delays/cobe_maxVar.h"

#include "stk_delays/iss_maxVar.h"
#include "stk_delays/iss_minVar.h"

#include "stk_delays/lageos_maxVar.h"
#include "stk_delays/lageos_minVar.h"

#include "stk_delays/navstar_max_min.h"

#include "stk_delays/radcal_maxVar.h"
#include "stk_delays/radcal_minVar.h"


#ifdef  __cplusplus
}
#endif
#endif

```

F.2.2.2 Internal Type Definitions

```
#ifndef STK_DELAY_TYPES_H
#define STK_DELAY_TYPES_H
#ifdef  __cplusplus
extern "C"{
#endif

/*
 *      Defines the offset to the first delay in the delay table.
 *      Please refer also to the layout of the delay table
 *
 *      Also define the type of the array elements holding the delays.
*/
#define DELAY_INDEX_OFFSET      2
#define DELAY_TYPE              double

#ifdef  __cplusplus
}
#endif
#endif

```

F.2.2.3 Cobe Delay with Maximum Variance

```
#ifndef COBE_MAX_VAR_H
#define COBE_MAX_VAR_H
#ifdef  __cplusplus
extern "C"{
#endif


#include "stk_delay_types.h"

const DELAY_TYPE cobe_maxVar [] = {
        3720,
        60,
        0.149494,
        0.148142,
        0.146755,
        0.145338,
        0.143896,
        0.142434,
        0.140957,
        0.139470,
        0.137979,
        0.136489,
        0.135006,
        0.133537,
        0.132087,
        0.130663,
        0.129272,
        0.127920,
        0.126614,
        0.125360,
        0.124165,
        0.123035,
        0.121978,
        0.120998,
        0.120101,
        0.119294,
        0.118581,
        0.117966,
        0.117454,
        0.117048,
        0.116751,
        0.116565,
        0.116491,
        0.116529,
        0.116680,
        0.116942,
        0.117313,
        0.117792,
        0.118374,
        0.119056,
        0.119834,
        0.120702,
        0.121656,
        0.122690,
        0.123797,
        0.124971,
        0.126207,
        0.127496,
        0.128834,
        0.130213,
        0.131626,
        0.133067,
        0.134529,
        0.136007,
        0.137494,
        0.138983,
        0.140471,
        0.141949,
        0.143414,
        0.144860,
        0.146283,
        0.147676,
        0.149037,
        0.150360
};
#ifdef  __cplusplus
}
#endif
#endif
```

The delay tables of other analyzed configurations are omitted here as they do not convey further understanding of implementation methods.

F.3 PROCESS MODEL OF THE LINKVERIFICATION NODE

```
/* Process model C form file: gmd_linkverification.pr.c */
/* Portions of this file copyright 1992-2002 by OPNET Technologies, Inc. */



/* This variable carries the header into the object file */
static const char gmd_linkverification_pr_c [] =
                  "MIL_3_Tfile_Hdr_ 81A 30A modeler 7 3DBEA9BF\
                    3DBEA9BF 1 castoralphagem mem 0 0 none none\
                    0 0 none 0 0 0 0 0 0";
#include <string.h>



/* OPNET system definitions */
#include <opnet.h>

#if defined (__cplusplus)
extern "C" {
#endif
FSM_EXT_DECS
#if defined (__cplusplus)
} /* end of ’extern "C"’ */
#endif


/* Header Block */

/* Include files. */
#include "oms_dist_support.h"

/* Define constands used in the process model */
#define GEN_NEW_PKT   10

/* State transition macro definitions */
#define SEND_PKT   (intrpt_type == OPC_INTRPT_SELF && intrpt_code == GEN_NEW_PKT)
#define PKT_ARRIVAL   (intrpt_type == OPC_INTRPT_STRM)

/* Function Declarations */
static void   gmd_linkverification_sv_init ();

/* End of Header Block */


#if !defined (VOSD_NO_FIN)
```

```
#undef BIN
#undef BOUT
#define BIN  FIN_LOCAL_FIELD(last_line_passed) = __LINE__ - _block_origin;
#define BOUT BIN
#define BINIT FIN_LOCAL_FIELD(last_line_passed) = 0; _block_origin = __LINE__;
#else
#define BINIT
#endif /* #if !defined (VOSD_NO_FIN) */



/* State variable definitions */
typedef struct
 {
 /* Internal state tracking for FSM */
 FSM_SYS_STATE
 /* State Variables */
 Stathandle        bits_rcvd_stathandle;
 Stathandle        bitssec_rcvd_stathandle;
 Stathandle        pkts_rcvd_stathandle;
 Stathandle        pktssec_rcvd_stathandle;
 Stathandle        ete_delay_stathandle;
 Stathandle        bits_rcvd_gstathandle;
 Stathandle        bitssec_rcvd_gstathandle;
 Stathandle        pkts_rcvd_gstathandle;
 Stathandle        pktssec_rcvd_gstathandle;
 Stathandle        ete_delay_gstathandle;
 double         start_time;
 OmsT_Dist_Handle      packet_size_dist_handle;
 OmsT_Dist_Handle      intarrvl_time_dist_handle;
 OmsT_Dist_Handle      start_time_dist_handle;
 double         stop_time;
 OmsT_Dist_Handle      stop_time_dist_handle;
 } gmd_linkverification_state;

#define pr_state_ptr      ((gmd_linkverification_state*) SimI_Mod_State_Ptr)
#define bits_rcvd_stathandle    pr_state_ptr->bits_rcvd_stathandle
#define bitssec_rcvd_stathandle   pr_state_ptr->bitssec_rcvd_stathandle
#define pkts_rcvd_stathandle    pr_state_ptr->pkts_rcvd_stathandle
#define pktssec_rcvd_stathandle   pr_state_ptr->pktssec_rcvd_stathandle
#define ete_delay_stathandle    pr_state_ptr->ete_delay_stathandle
#define bits_rcvd_gstathandle   pr_state_ptr->bits_rcvd_gstathandle
#define bitssec_rcvd_gstathandle  pr_state_ptr->bitssec_rcvd_gstathandle
#define pkts_rcvd_gstathandle   pr_state_ptr->pkts_rcvd_gstathandle
#define pktssec_rcvd_gstathandle  pr_state_ptr->pktssec_rcvd_gstathandle
#define ete_delay_gstathandle   pr_state_ptr->ete_delay_gstathandle
#define start_time       pr_state_ptr->start_time
#define packet_size_dist_handle   pr_state_ptr->packet_size_dist_handle
#define intarrvl_time_dist_handle  pr_state_ptr->intarrvl_time_dist_handle
#define start_time_dist_handle   pr_state_ptr->start_time_dist_handle
#define stop_time       pr_state_ptr->stop_time
#define stop_time_dist_handle   pr_state_ptr->stop_time_dist_handle

/* This macro definition will define a local variable called */
/* "op_sv_ptr" in each function containing a FIN statement. */
/* This variable points to the state variable data structure, */
/* and can be used from a C debugger to display their values. */
#undef FIN_PREAMBLE
#define FIN_PREAMBLE gmd_linkverification_state *op_sv_ptr = pr_state_ptr;


/* Function Block */

enum { _block_origin = __LINE__ };
static void
gmd_linkverification_sv_init ()
 {
 Objid    my_id;
/* object ID of surrounding processor or queue */
 Objid    traf_gen_comp_attr_objid;
```

```
/* object ID of compound traffic attribute */
 Objid    traf_conf_objid;
/* ...... subcomponent of compund traffic attrib */
 Objid    pkt_gen_comp_attr_objid;
/* object ID of compunt packet attribute, i.e., part of the traffic attribute */
 Objid    pkt_gen_args_objid;
/* ....... subcomponent of compound packet attrib */
 char    start_time_string [128];
 char    stop_time_string [128];
 char    intarrvl_rate_string [128], packet_size_string [128];

 /** Initializes state variables associated with  **/
 /** this process model.        **/
 FIN (gmd_linkverification_sv_init ());

 /* Initializes statistic handles associated with this */
 /* process model             */
 bits_rcvd_stathandle = op_stat_reg ("GMD Linkverifications.Traffic\
 Received (bits)",
                                     OPC_STAT_INDEX_NONE,
                                     OPC_STAT_LOCAL);
 bitssec_rcvd_stathandle = op_stat_reg ("GMD Linkverifications.Traffic\
 Received (bits/sec)",
                                        OPC_STAT_INDEX_NONE,
                                        OPC_STAT_LOCAL);
 pkts_rcvd_stathandle = op_stat_reg ("GMD Linkverifications.Traffic\
 Received (packets)",
                                     OPC_STAT_INDEX_NONE,
                                     OPC_STAT_LOCAL);
 pktssec_rcvd_stathandle = op_stat_reg ("GMD Linkverifications.Traffic\
 Received (packets/sec)",
                                        OPC_STAT_INDEX_NONE,
                                        OPC_STAT_LOCAL);
 ete_delay_stathandle = op_stat_reg ("GMD Linkverifications.End-to-End\
 Delay (seconds)",
                                     OPC_STAT_INDEX_NONE,
                                     OPC_STAT_LOCAL);

 bits_rcvd_gstathandle   = op_stat_reg ("GMD Linkverifications.Traffic\
 Received (bits)",
                                        OPC_STAT_INDEX_NONE,
                                        OPC_STAT_GLOBAL);
 bitssec_rcvd_gstathandle  = op_stat_reg ("GMD\
 Linkverifications.Traffic Received (bits/sec)",
                                          OPC_STAT_INDEX_NONE,
                                          OPC_STAT_GLOBAL);
 pkts_rcvd_gstathandle   = op_stat_reg ("GMD\
 Linkverifications.Traffic Received (packets)",
                                        OPC_STAT_INDEX_NONE,
                                        OPC_STAT_GLOBAL);
 pktssec_rcvd_gstathandle  = op_stat_reg ("GMD\
 Linkverifications.Traffic Received (packets/sec)",
                                          OPC_STAT_INDEX_NONE,
                                          OPC_STAT_GLOBAL);
 ete_delay_gstathandle  = op_stat_reg ("GMD\
 Linkverifications.End-to-End Delay (seconds)",
                                       OPC_STAT_INDEX_NONE,
                                       OPC_STAT_GLOBAL);

 /* Determine the object ID of the containing node */
  my_id = op_id_self ();

 /* Read the traffic generation parameters. */
 op_ima_obj_attr_get (my_id, "Traffic Generation Parameters",
                      &traf_gen_comp_attr_objid);
 traf_conf_objid = op_topo_child (traf_gen_comp_attr_objid,
                                  OPC_OBJTYPE_GENERIC, 0);

 /* Determine the start time for traffic generation. */
 op_ima_obj_attr_get (traf_conf_objid, "Start Time", start_time_string);
```

```
  if (strcmp (start_time_string, "Never") != 0)
   {
   start_time_dist_handle = oms_dist_load_from_string (start_time_string);
   start_time = oms_dist_outcome (start_time_dist_handle);
   }
  else
   {
   start_time = -1.0;
   }

  /* Determine the stop time for traffic generation. */
  op_ima_obj_attr_get (traf_conf_objid, "Stop Time", stop_time_string);
  if (strcmp (stop_time_string, "Never") != 0)
   {
   stop_time_dist_handle = oms_dist_load_from_string (stop_time_string);
   stop_time = oms_dist_outcome (stop_time_dist_handle);
   }
  else
   {
   stop_time = -1.0;
   }


  /* Load the distribution used to determine the packet */
  /* interarrivals.          */
  op_ima_obj_attr_get (traf_conf_objid,
                      "Packet Generation Arguments",
                       &pkt_gen_comp_attr_objid);
  pkt_gen_args_objid = op_topo_child (pkt_gen_comp_attr_objid,
                                     OPC_OBJTYPE_GENERIC, 0);

  op_ima_obj_attr_get (pkt_gen_args_objid, "Interarrival Time",

 intarrvl_rate_string);
  intarrvl_time_dist_handle
               = oms_dist_load_from_string (intarrvl_rate_string);

  /* Load the distribution used to determine the size of */
  /* the packets being generated.       */
  op_ima_obj_attr_get (pkt_gen_args_objid,
                      "Packet Size", packet_size_string);
  packet_size_dist_handle
                  = oms_dist_load_from_string (packet_size_string);

   FOUT ;
  }

 /* End of Function Block */

 /* Undefine optional tracing in FIN/FOUT/FRET */
 /* The FSM has its own tracing code and the other */
 /* functions should not have any tracing.   */
 #undef FIN_TRACING
 #define FIN_TRACING

 #undef FOUTRET_TRACING
 #define FOUTRET_TRACING

 #if defined (__cplusplus)
 extern "C" {
 #endif
  void gmd_linkverification (void);
  Compcode gmd_linkverification_init (void **);
  void gmd_linkverification_diag (void);
  void gmd_linkverification_terminate (void);
  void gmd_linkverification_svar (void *, const char *, char **);
 #if defined (__cplusplus)
 } /* end of 'extern "C"' */
 #endif
```

```



/* Process model interrupt handling procedure */


void
gmd_linkverification (void)
 {
 int _block_origin = 0;
 FIN (gmd_linkverification ());
 if (1)
  {
  /* used to store the type of interupt and its code  */
  /* they both need to be set in the EXIT EXECUTIVE block */
  /* of the state that is LEFT upon an interrupt   */
  int   intrpt_type;
  int   intrpt_code;

  Packet*  pkptr;
  double  pk_size;
  double  ete_delay;
  double  intarrvl_time;


  FSM_ENTER (gmd_linkverification)

  FSM_BLOCK_SWITCH
   {
   /*---------------------------------------------------------*/
   /** state (receive) enter executives **/
   FSM_STATE_ENTER_FORCED (0, state0_enter_exec, "receive",
                          "gmd_linkverification ()\
 [receive enter execs]")
    {
    /* Obtain the incoming packet. */
    pkptr = op_pk_get (op_intrpt_strm ());

    /* Caclulate metrics to be updated.  */
    pk_size = (double) op_pk_total_size_get (pkptr);
    ete_delay = op_sim_time () - op_pk_creation_time_get (pkptr);

    /* Update local statistics.    */
    op_stat_write (bits_rcvd_stathandle,   pk_size);
    op_stat_write (pkts_rcvd_stathandle,   1.0);
    op_stat_write (ete_delay_stathandle,   ete_delay);

    op_stat_write (bitssec_rcvd_stathandle,  pk_size);
    op_stat_write (bitssec_rcvd_stathandle,  0.0);
    op_stat_write (pktssec_rcvd_stathandle,  1.0);
    op_stat_write (pktssec_rcvd_stathandle,  0.0);

    /* Update global statistics. */
    op_stat_write (bits_rcvd_gstathandle,   pk_size);
    op_stat_write (pkts_rcvd_gstathandle,   1.0);
    op_stat_write (ete_delay_gstathandle,   ete_delay);

    op_stat_write (bitssec_rcvd_gstathandle,  pk_size);
    op_stat_write (bitssec_rcvd_gstathandle,  0.0);
    op_stat_write (pktssec_rcvd_gstathandle,  1.0);
    op_stat_write (pktssec_rcvd_gstathandle,  0.0);

    /* Destroy the received packet. */
    op_pk_destroy (pkptr);
    }


   /** state (receive) exit executives **/
   FSM_STATE_EXIT_FORCED (0, "receive", "gmd_linkverification ()\
 [receive exit execs]")
```

```
     {
     }


    /** state (receive) transition processing **/
    FSM_TRANSIT_FORCE (3, state3_enter_exec, ;, "default",
                       "", "receive", "idle")
     /*--------------------------------------------------------*/



    /** state (INIT) enter executives **/
    FSM_STATE_ENTER_FORCED_NOLABEL (1, "INIT",
                       "gmd_linkverification () [INIT enter execs]")
     {
     /* Initilaize the statistic handles to keep */
     /* track of traffic sinked by this process. */
     gmd_linkverification_sv_init () ;

     /* schedule the first packet to be sent */
     /* start_time is set to -1 if we do not want to generate traffic */
     if (start_time >= 0.0)
      {
      op_intrpt_schedule_self (op_sim_time () + start_time, GEN_NEW_PKT);
      }
     }


    /** state (INIT) exit executives **/
    FSM_STATE_EXIT_FORCED (1, "INIT",
                           "gmd_linkverification () [INIT exit execs]")
     {
     }


    /** state (INIT) transition processing **/
    FSM_TRANSIT_FORCE (3, state3_enter_exec, ;,
                       "default", "", "INIT", "idle")
     /*--------------------------------------------------------*/



    /** state (send) enter executives **/
    FSM_STATE_ENTER_FORCED (2, state2_enter_exec, "send",
                            "gmd_linkverification () [send enter execs]")
     {
     /* Create a packet using the outcome of the loaded */
     /* distribution.         */

     pk_size = oms_dist_outcome (packet_size_dist_handle);
     pkptr = op_pk_create (pk_size*8);

     /* This is the point at which to create statistic about */
     /* sent packets           */
     /* to be filled */

     /* Send the packet on the stream connected to this */
     /* module.          */
     op_pk_send (pkptr, 0);

     /* schedule an interrupt in order to create the next */
     /* packet if the stop time is not reached     */
     if ((stop_time < 0.0) || (stop_time > op_sim_time () ))
      {
      intarrvl_time = oms_dist_outcome (intarrvl_time_dist_handle);
      op_intrpt_schedule_self (op_sim_time () + intarrvl_time,
                               GEN_NEW_PKT);
      }
     }


    /** state (send) exit executives **/
    FSM_STATE_EXIT_FORCED (2, "send",
                         "gmd_linkverification () [send exit execs]")
     {
     }


    /** state (send) transition processing **/
    FSM_TRANSIT_FORCE (3, state3_enter_exec, ;,
                       "default", "", "send", "idle")
     /*---------------------------------------------------------*/



    /** state (idle) enter executives **/
    FSM_STATE_ENTER_UNFORCED (3, state3_enter_exec, "idle",
                             "gmd_linkverification () [idle enter execs]")
     {
     }


    /** blocking after enter executives of unforced state. **/
    FSM_EXIT (7,gmd_linkverification)


    /** state (idle) exit executives **/
    FSM_STATE_EXIT_UNFORCED (3, "idle",
                          "gmd_linkverification () [idle exit execs]")
     {
     /* Determine the type of interrupt.  */
     intrpt_type = op_intrpt_type ();
     intrpt_code = op_intrpt_code ();
     }


    /** state (idle) transition processing **/
    FSM_INIT_COND (SEND_PKT)
    FSM_TEST_COND (PKT_ARRIVAL)
    FSM_DFLT_COND
    FSM_TEST_LOGIC ("idle")

    FSM_TRANSIT_SWITCH
     {
     FSM_CASE_TRANSIT (0, 2, state2_enter_exec, ;,
                      "SEND_PKT", "", "idle", "send")
     FSM_CASE_TRANSIT (1, 0, state0_enter_exec, ;,
                      "PKT_ARRIVAL", "", "idle", "receive")
     FSM_CASE_TRANSIT (2, 3, state3_enter_exec, ;,
                      "default", "", "idle", "idle")
     }
     /*---------------------------------------------------------*/



    }


   FSM_EXIT (1,gmd_linkverification)
   }
  }

#if defined (__cplusplus)
 extern "C" {
#endif
 extern VosT_Fun_Status Vos_Catmem_Register (const char * , int ,
                              VosT_Void_Null_Proc, VosT_Address *);
 extern VosT_Address Vos_Catmem_Alloc (VosT_Address, size_t);
 extern VosT_Fun_Status Vos_Catmem_Dealloc (VosT_Address);
#if defined (__cplusplus)
```

```
 }
#endif


Compcode
gmd_linkverification_init (void ** gen_state_pptr)
 {
 int _block_origin = 0;
 static VosT_Address obtype = OPC_NIL;

 FIN (gmd_linkverification_init (gen_state_pptr))

 if (obtype == OPC_NIL)
  {
  /* Initialize memory management */
  if (Vos_Catmem_Register ("proc state vars (gmd_linkverification)",
   sizeof (gmd_linkverification_state),
           Vos_Vnop, &obtype) == VOSC_FAILURE)
   {
   FRET (OPC_COMPCODE_FAILURE)
   }
  }

 *gen_state_pptr = Vos_Catmem_Alloc (obtype, 1);
 if (*gen_state_pptr == OPC_NIL)
  {
  FRET (OPC_COMPCODE_FAILURE)
  }
 else
  {
  /* Initialize FSM handling */
  ((gmd_linkverification_state *)(*gen_state_pptr))->current_block = 2;

  FRET (OPC_COMPCODE_SUCCESS)
  }
 }



void
gmd_linkverification_diag (void)
 {
 /* No Diagnostic Block */
 }




void
gmd_linkverification_terminate (void)
 {
 int _block_origin = __LINE__;

 FIN (gmd_linkverification_terminate (void))

 Vos_Catmem_Dealloc (pr_state_ptr);

 FOUT;
 }


/* Undefine shortcuts to state variables to avoid */
/* syntax error in direct access to fields of */
/* local variable prs_ptr in gmd_linkverification_svar function. */
#undef bits_rcvd_stathandle
#undef bitssec_rcvd_stathandle
#undef pkts_rcvd_stathandle
#undef pktssec_rcvd_stathandle
#undef ete_delay_stathandle
#undef bits_rcvd_gstathandle
```

```
#undef bitssec_rcvd_gstathandle
#undef pkts_rcvd_gstathandle
#undef pktssec_rcvd_gstathandle
#undef ete_delay_gstathandle
#undef start_time
#undef packet_size_dist_handle
#undef intarrvl_time_dist_handle
#undef start_time_dist_handle
#undef stop_time
#undef stop_time_dist_handle



void
gmd_linkverification_svar (void * gen_ptr,
                           const char * var_name, char ** var_p_ptr)
 {
 gmd_linkverification_state  *prs_ptr;

 FIN (gmd_linkverification_svar (gen_ptr, var_name, var_p_ptr))

 if (var_name == OPC_NIL)
  {
  *var_p_ptr = (char *)OPC_NIL;
  FOUT;
  }
 prs_ptr = (gmd_linkverification_state *)gen_ptr;

 if (strcmp ("bits_rcvd_stathandle" , var_name) == 0)
  {
  *var_p_ptr = (char *) (&prs_ptr->bits_rcvd_stathandle);
  FOUT;
  }
 if (strcmp ("bitssec_rcvd_stathandle" , var_name) == 0)
  {
  *var_p_ptr = (char *) (&prs_ptr->bitssec_rcvd_stathandle);
  FOUT;
  }
 if (strcmp ("pkts_rcvd_stathandle" , var_name) == 0)
  {
  *var_p_ptr = (char *) (&prs_ptr->pkts_rcvd_stathandle);
  FOUT;
  }
 if (strcmp ("pktssec_rcvd_stathandle" , var_name) == 0)
  {
  *var_p_ptr = (char *) (&prs_ptr->pktssec_rcvd_stathandle);
  FOUT;
  }
 if (strcmp ("ete_delay_stathandle" , var_name) == 0)
  {
  *var_p_ptr = (char *) (&prs_ptr->ete_delay_stathandle);
  FOUT;
  }
 if (strcmp ("bits_rcvd_gstathandle" , var_name) == 0)
  {
  *var_p_ptr = (char *) (&prs_ptr->bits_rcvd_gstathandle);
  FOUT;
  }
 if (strcmp ("bitssec_rcvd_gstathandle" , var_name) == 0)
  {
  *var_p_ptr = (char *) (&prs_ptr->bitssec_rcvd_gstathandle);
  FOUT;
  }
 if (strcmp ("pkts_rcvd_gstathandle" , var_name) == 0)
  {
  *var_p_ptr = (char *) (&prs_ptr->pkts_rcvd_gstathandle);
  FOUT;
  }
 if (strcmp ("pktssec_rcvd_gstathandle" , var_name) == 0)
  {
     *var_p_ptr = (char *) (&prs_ptr->pktssec_rcvd_gstathandle);
     FOUT;
     }
    if (strcmp ("ete_delay_gstathandle" , var_name) == 0)
     {
     *var_p_ptr = (char *) (&prs_ptr->ete_delay_gstathandle);
     FOUT;
     }
    if (strcmp ("start_time" , var_name) == 0)
     {
     *var_p_ptr = (char *) (&prs_ptr->start_time);
     FOUT;
     }
    if (strcmp ("packet_size_dist_handle" , var_name) == 0)
     {
     *var_p_ptr = (char *) (&prs_ptr->packet_size_dist_handle);
     FOUT;
     }
    if (strcmp ("intarrvl_time_dist_handle" , var_name) == 0)
     {
     *var_p_ptr = (char *) (&prs_ptr->intarrvl_time_dist_handle);
     FOUT;
     }
    if (strcmp ("start_time_dist_handle" , var_name) == 0)
     {
     *var_p_ptr = (char *) (&prs_ptr->start_time_dist_handle);
     FOUT;
     }
    if (strcmp ("stop_time" , var_name) == 0)
     {
     *var_p_ptr = (char *) (&prs_ptr->stop_time);
     FOUT;
     }
    if (strcmp ("stop_time_dist_handle" , var_name) == 0)
     {
     *var_p_ptr = (char *) (&prs_ptr->stop_time_dist_handle);
     FOUT;
     }
    *var_p_ptr = (char *)OPC_NIL;

    FOUT;
    }
```

References

1. The network simulator – ns-2. [Online]. Available: http://www.isi.edu/nsnam/ns/

2. *The American Heritage Dictonary of the English Language*, 4th ed. Houghton Mifflin Company, 2000.

3. *OPNET Modeler Modeling Concepts*. Maryland, USA: OPNET Technologies, Inc., 2001, no. D00115v4, ch. Communication Mechanisms.

4. *Performance issues in asymmetric TCP service provision using broadband satellite*, vol. 148, no. 2, April 2001.

5. *Handbook on Satellite Communications*, 3rd ed. Wiley-Interscience and International Telecommunication Union ITU, 2002.

6. (2003) ATM–Sat: ATM–based multimedia communication via LEO satellites. Berlin, Germany. [Online]. Available: http://www.fokus.gmd.de/research/cc/cats/satellite/Projects/ATM-Sat

7. (2003) The freebsd homepage. [Online]. Available: http://www.freebsd.org/

8. (2003) OPNET — optimum network performance. [Online]. Available: http://www.opnet.com

9. I. Akyildiz, G. Morabito, and S. Palazzo, "TCP-Peach: a new congestion control scheme for satellite IP netoworks," *IEEE/ACM Transactions on Networking*, vol. 9, no. 3, pp. 307–21, June 2001.

10. I. Akyildiz, X. Zhang, and J. Fang, "TCP-Peach+: enhancement of TCP-Peach for satellite IP networks," *IEEE Communicaitons Letters*, vol. 6, no. 7, pp. 303–5, July 2002.

11. M. Allman, V. Paxson, and W. Stevens, "TCP congestion control," RFC 2581, Internet Engineering Task Force (IETF), p. 14, April 1999, Obosletes RFC 2001 "TCP slow Start, congestion avoidance, fast retransmit and fast recovery algorithm". [Online]. Available: http://www.ietf.org/rfc/rfc2581.txt

12. M. Allman, A. Caldwell, and S. Ostermann, "ONE: The Ohio Network Emulator," Ohio University, Tech. Rep. TR-19972, 1997. [Online]. Available: http://roland.lerc.nasa.gov/ mallman/papers/em.ps

13. M. Allmann, D. Glover, and L. Sanchez, "Enhancing TCP over satellite channels using standard mechanisms," RFC 2488, Internet Engineering Task Force (IETF), p. 14, January 1999. [Online]. Available: http://www.ietf.org/rfc/rfc2488.txt

14. M. Allmann, J. Griner, and A. Richard, “TCP behavior in networks with dynamic propagation delay,” Proc. Globecom Proc. of IEEE Globecom. California, USA: IEEE, October 2000.

15. (2003) Product overview - satellite toolkit (STK). Analytical Graphics, Inc. [Online]. Available: http://www.stk.com/products/explore/products/stk-prod-desc.htm

16. H. Bischl, H. Brandt, A. Dreher, M. Emmelmann, A. Freier, B. Hespeler, F. Krepel, E. Lutz, W. Milcz, L. Richard, P. Todorova, and M. Werner, “ATM-sat: ATM-based multimedia communication via LEO-satellites – system architecture report,” German Aerospace Agency (DLR), Tech. Rep., 2000.

17. ——, “ATM-Sat: ATM-based multimedia communication via LEO-satellites – target system report,” German Aerospace Agency (DLR),” Target System Report, 2001.

18. ——, “ATM-Sat: ATM-based multimedia communication via LEO-satellites – final report,” German Aerospace Agency (DLR),” Final Report, 2002.

19. H. Bischl, H. Brandt, A. Dreher, M. Emmelmann, F. Krepel, E. Lutz, L. Richard, P. Todorova, and M. Werner, “ATM-based multimedia communication via NGSO-satellites,” unpublished.

20. H. Bischl, “Availibility and error control for ATM via satellite at Ka-Band,” presented at the Deutscher Luft– und Raumfahrtkongress 2001, Hamburg, Germany, 2001.

21. E. Blanton, M. Allman, K. Fall, and L. Wang, “A conservative selective acknowledgment (SACK)-based loss recovery algorithm for TCP,” RFC 3517, Internet Engineering Task Force (IETF), April 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3517.txt

22. H. Brandt and F. Krepel, “A simulation system for ATM and packet-based MEO and GEO communication systems,” in *Proc. of the 21th AIAA International Communications Satellite Systems Conference*, Yokohama, Japan, April 2003. [Online]. Available: http://www.emmelmann.org/Library/Papers_Reports/docs/AIAA2003b.pdf

23. H. Brandt, F. Krepel, and C. Tittel, “Multiple access layer and signalling simulator for a LEO satellite system,” in *Proc. of the 20th AIAA International Communications Satellite Systems Conference*, Montreal, Canada, May 2002. [Online]. Available: http://www.emmelmann.org/Library/Papers_Reports/docs/AIAA2002.pdf

24. I. Bronstein, K. Semendjajew, G. Musiol, and H. Mühlig, *Taschenhandbuch der Mathematik*, 1st ed. Frankfurt am Main, Germany: Verlag Harri Deutsch, 1993.

25. C. D. Brown, *Spacecraft Mission Design*, 2nd ed., ser. AIAA Education Series, J. S. Przemieniecki, Ed. VA, USA: AIAA American Institute of Aeronautics and Austronautics, Inc., 1998.

26. ——, *Elemts of Spacecraft Design*, ser. AIAA Education Series, J. S. Przemieniecki, Ed. VA, USA: AIAA American Institute of Aeronautics and Austronautics, Inc., 2002.

27. R. G. Brown. (2001) Philosophical physics – lecture notes for elementary phsics 41. Duke University Physics Department. [Online]. Available: http://www.phy.duke.edu/ rgb/Class/phy41/

28. L. Castanet, “Fade mitigation techniques for new satcom systems operating at Ka&V bands.” Ph.D. dissertation, L’Ecole Nationale Supérieur de L’Aeronautique et de L’Espace, Dec 2001.

29. D.-J. Choi, S.-N. Choi, and N.-S. Kim, “Experiments and analysis of the standard TCP mechanisms using ATM based satellite network,” in *Proc. of IEEE International Conference on ATM and High Speed Intelligent Internet*, April, 22-25 2001, pp. 349–353.

30. Y. Chstikapong, H. Cruickshank, and Z. Sun, “Evaluation of TCP and internet traffic via low earth orbit satellites,” *IEEE Personal Communications*, vol. 8, no. 3, June 2001.

31. Y. Chstikapong, H. Cruickshank, Z. Sun, and B. Evans, “Network architecture and performance evaluation of broadband satellite systems,” in *Proc. of IEEE International Conference on Networks (ICON 2000)*, September, 5-8 2000.

32. M. Emmelmann, “TCP/IP over LEO satellites,” in *Proc. of the ATM-Sat Workshop 2000*, German Aerospace Agency (DLR). Oberpfaffenhofen, Germany: German Aerospace Agency (DLR), Dec, 12 2000.

33. ——, “TCP/IP over satellite,” Jul 24 2000.

34. ——, “Dimensioning of receive buffer size and timer granularity for optimal performance of TCP in a variable delay LEO satellite network,” in *Proc. of the Second NASA Space Internet Workshop (SIW 2)*, NASA Goddard Space Flight Center. Greenbelt, Maryland, USA: National Aeronautics and Space Agency (NASA), May, 21–22 2002. [Online]. Available: http://siw.gsfc.nasa.gov

35. ——, “Effects of receive buffer size and timer granularity on TCP performance over erroneous links in a LEO satellite network,” in *Proc. of IEEE Global Communication Conference (Globecom’02)*, Taipei, Taiwan, 2002, paper SAT-05-3.

36. ——, “An integrated prototyping and simulation architecture for space specific protocol developments and verifications,” in *Proc. of the Third NASA Space Internet Workshop (SIW 3)*, NASA Glenn Research Center. Cleveland, Ohio, USA: National Aeronautics and Space Agency (NASA), June, 4–6 2003. [Online]. Available: http://scp.grc.nasa.gov/siw

37. ——, “Performance of TCP over a LEO satellite channel using freebsd, preliminary experiment results,” 2003.

38. M. Emmelmann and H. Bischl, “An adaptive MAC layer protocol for ATM-based LEO satellite networks,” in *Proc. of IEEE Vehicular Technolgy Conference (VTC ’03 Fall)*, Orlando, Florida, USA, oct 2003, invited paper.

39. M. Emmelmann, H. Brandt, H. Bischl, and S. Scalise, “An access protocol for mobile satellite users with reduced link margins and contention probability,” in *Proc. of the First International Conference on Advanced Satellite Mobile Systems (ASMS Conference 2003)*, European Space Agency ESA. Frascati, Italy: ESA Publication Devision (EPD), Jul 10–11, 2003, Special Publication SP-541.

40. J. Farsserotu and R. Prasad, *IP/ATM Mobile Satellite Networks*, ser. Universal Personal Communication Series. Maryland, USA: Artech House, 2002.

41. S. Floyd, J. Mahadavi, M. Mathis, and M. Podolsky, “An extension to the selective ackinowledgment (SACK) option for TCP,” RFC 2883, Internet Engineering Task Force (IETF), July 2000. [Online]. Available: http://www.ietf.org/rfc/rfc2883.txt

42. F. Hargrave, *Hargrave’s Communication Dictionary*. NY, USA: IEEE, 2001.

43. J. Heidemann and P. Huang. (2002, March) IPAM tutorial: Network modelling and traffic analysis with ns-2. [Online]. Available: http://www.isi.edu/nsnam/ns/ns-tutorial/index.html

44. T. Henderson and R. Katz, “Transport protocols for internet-compatible satellite networks,” *IEEE Journal on Selected Areas in Communications*, vol. 17, pp. 326–44, February 1999.

45. C. Hunscher, R. Mayer, A. Jacob, L. Stange, A. Dreher, and L. Richard, “Aktive Antenne für Multimediakommmunikation über Satellit,” Technische Universität Braunschweig, Germany, Final Project Report BMBF contract no. 50 YB 0004, 2001.

46. “Transmission control protocol,” RFC 793, Internet Engineering Task Force (IETF), September 1981. [Online]. Available: http://www.ietf.org/rfc/rfc0793.txt

47. “Requirements for internet hosts – communication layers,” RFC 1122, Internet Engineering Task Force (IETF), October 1989. [Online]. Available: http://www.ietf.org/rfc/rfc1122.txt

48. J. Ishac and M. Allman, “On the performance of TCP spoofing in satellite networks,” in *Proc. of IEEE Military Communications Conference MILCOM*, vol. 1, October, 28-31 2002, pp. 700–4.

49. V. Jacobson, R. Braden, and D. Borman, “TCP extensions for high performance,” RFC 1323, Internet Engineering Task Force (IETF), May 1992. [Online]. Available: http://www.ietf.org/rfc/rfc1323.txt

50. V. Jacobson, “Congestion avoidance and control,” in *Proc. of SIGCOMM ’88,* Stanford, CA, USA, August 1988. [Online]. Available: http://www.emmelmann.org

51. V. Jacobson and R. Braden, “TCP extensions for long-delay paths,” RFC 1072, Internet Engineering Task Force (IETF), p. 16, October, 1 1988. [Online]. Available: http://www.ietf.org/rfc/rfc1072.txt

52. B. Jenkins. Demonstrating orbits with java. [Online]. Available: http://burtleburtle.net/bob/physics/orbit101.html

53. J. Keum. Conic sections. The University of Georgia, Department of Mathematics Education. [Online]. Available: http://jwilson.coe.uga.edu/EMT668/EMAT6680.2000/Keum/Emat6690/Essay2/essay2.html

54. T. Krout, M. Solsman, and J. Goldstein, “The effects of asymmetric satellite networks on protocols,” in *Proc. of IEEE Military Communications Conference (MILCOM)*, vol. 3, October, 18-21 1998.

55. E. Lutz, M. Werner, and A. Jahn, *Satellite Systems for Personal and Broadband Communications*. Springer, 2000.

56. M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, “TCP selective acknowledgement options,” RFC 2018, Internet Engineering Task Force (IETF), October 1996. [Online]. Available: http://www.ietf.org/rfc/rfc2018.txt

57. V. Mhatre and C. Rosenberg, “Performance improvement of TCP-based applications in a multi-access satellite system,” in *Proceedings of Vehicular Technology Conference VTC 2002-Fall*, vol. 3, September, 24-28 2002, pp. 1530–1534.

58. Y. Miyake, T. Hasegawa, and T. Kato, “Internet access using TCP gateway,” in *Proc. of the 5th IEEE Symposium on Computers and Communications*, July, 3-6 2000.

59. J. C. Mogul, *Internet System Handbook*. Reading, Mass.: Addison-Wesley, 1993, ch. IP Network Performance, pp. 575–675.

60. J. Nagle, “Congestion control in IP/TCP internetworks,” RFC 896, Internet Engineering Task Force (IETF), p. 9, January 1984. [Online]. Available: http://www.ietf.org/rfc/rfc896.txt

61. P. Noll, “Signale und systeme,” Technical University of Berlin, Berlin, Germany, 1996.

62. H. Obata, K. Ishida, J. Funasaka, and K. Amano, “TCP performance analysis on asymmetric networks composed of satellite and terrestrial links,” in *Proc. of the International Conference on Network Protocols 2000,* November, 14-17 2000, pp. 199–206.

63. One - the ohio network emulator. Ohio University Internetworking Research Group. [Online]. Available: http://irg.cs.ohiou.edu/one/

64. (2003) Opnet modeler brochure. OPNET Technologies, Inc. [Online]. Available: http://www.opnet.com/products/modeler/home.html

65. A. V. Oppenhein and A. S. Willsky, *Signale und Systeme,* 2nd ed. VCH Verlagsgesellschaft, 1992.

66. C. Papadopoulos and G. Parulkar, “Experimental evaluation of SunOS IPC and TCP/IP protocol implementaion,” *IEEE/ACM Transactions on Networking,* vol. 1, no. 2, April 1993.

67. S. Philopoulos and K. Ferens, “Proxy-based connection-splitting architectures for improving TCP performance over satellite channels,” in *Proc. of IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, vol. 3, May 12-15 2002, pp. 1430–1435.

68. D. J. Raymond. (2003, May, 14) A radically modern approach to introductory physics. New Mexico Tech, Physics Department. [Online]. Available: http://www.physics.nmt.edu/ raymond/classes/ph13xbook/

69. N. Samaraweera and G. Fairhurst, “Explicit loss indication and accurate RTO estimation for TCP error recovery using satellite links,” in *Proc. of IEEE Communications*, vol. 144, February 1997, pp. 47–53.

70. K. Scott, P. Feighery, B. Crow, and M. Jurik, “TCP congestion control in shared satellite environment,” in *Proc. of MILCOM 2002*, vol. 1, October, 7-10 2002, pp. 46–50.

71. (2001, March) Binding energy. The Spaceguard Foundation and the Spaceguard System. [Online]. Available: http://spaceguard.ias.rm.cnr.it/NScience/neo/dictionary/bound-ene.htm

72. W. Stallings, *Data and Computer Communication*, 6th ed. Prentice-Hall, June 2000.

73. W. R. Stevens, *TCP/IP Illustrated: The Protocols*, ser. Addison-Wesley Professional Computing Series. Massachusetts, USA: Addison Wesley Longman, Inc., 1994, vol. Volume 1.

74. A. S. Tanenbaum, *Computer Networks*, 3rd ed. Prentice-Hall, 1996.

75. E. W. Weisstein. (2003) Vernal equinox. [Online]. Available: http://scienceworld.wolfram.com/astronomy/VernalEquinox.html

76. M. Werner, A. Jahn, E. Lutz, and A. Bötcher, “Analysis of system parameters for LEO/ICO-satellite communication networks,” *IEEE JSAC*, vol. 13, no. 2, pp. 371–381, February 1995.

Glossary

Acknowledgement A control message sent to indicate that a data block just received is okay. [42]

AER An acronym from *Azimuth* Elevation Range. This term is used within the context of STK simulations to denote to ASCII-based simulation results describing the current AER values of an analyzed satellite as a function of time.

Application Definition Node A standard library network node available in Opnet. It allows the definition of certain application types and their characteristics. Such a definition might specify an application "large file transfer" which retrieves a 200-MB-large file from a specific server. Another application definition might specify an IP-telephone call which last 10 minutes.

Ascending Node The point of a satellite's orbit plane which is passed when the satellite enters the northern hemisphere. [55]

ASCII An acronym form American Standard Code for Information Interexchange. A standard code used for information interexchange between data processing equipment and over data communication networks. [42]

Azimuth The horizontal angular distance from a reference direction, usually the northern point of the horizon, to the point where a vertical circle through a celestial body intersects the horizon, usually measured clockwise. Sometimes the southern point is used as the reference direction, and the measurement is made clockwise through 360^o. [2]

Bandwidth Delay Product (BDP) The BDP denotes the number of un-acknowledged bytes which can be transmitted simultaneously on a bandwidth-limited link in order to achieve a link-utilization of 100%. As the propagation delay of the acknowledgement has to be considered when calculation the BDP, instead of the one-way delay, the round trip time (RTT) is used.

Bit Error Rate (BER) The probability that a single bit is corrupted.

Cross correlation function The cross correlation $r_{dd^*}(\tau)$ of two functions $d(t)$ and $d^*(t)$ is defined as $r_{dd^*}(\tau) \overset{\text{def}}{=} \int_{+\infty}^{-\infty} d(t)\, d^*(t+\tau)\, dt$. $r_{dd^*}(\tau)$ can be interpreted as a measurement representing how similar $d(t)$ and $d^*(t)$ are for a given phase shift τ.

Data Link Control A term that refers to the functions provided at the data link layer of the ISO/OSI reference model. The data link layer defines protocols for frames or packets and how they are transmitted to and from each network device. [42]

Delayed ACK scheme TCP does not have to acknowledge every received segment. Instead it may choose to wait for k segments (usually $k = 2$) before an ACK is sent. This mechanism is known as delayed ACK scheme.

Fast Recovery Algorithm A congestion control scheme used in TCP Reno. Instead of closing the current congestion window entirely after a retransmission of a lost segment, TCP sets its congestion window eventually to one half.

Fast Retransmit Algorithm A retransmission algorithm used in TCP Reno. After three duplicate ACKs have arrived, TCP starts the retransmission of the presumingly lost segment right away, i.e., it does not wait for the retransmission timeout to expire.

Elevation angle The angle at which a user can see the satellite above the horizon. [55]

Equivalent Isotropic Radiated Power (EIRP) The EIRP is the main characteristic of a transmitter. It is defined as the same power flux density as a given theoretical antenna. [55]

Finite State Machine (FSM) A machine (real or abstract) consisting of a set of states including an initial state, a constellation of possible input events, a set of output events, and a state transition function. [42]

Forward Error Correction (FEC) In communications, an error-correction method involving the addition of redundant bits into the transmitted date stream. The receiver uses the redundant data to correct errors (where possible) introduced in the communication channel. [42]

FTP An abbreviation of File Transfer Protocol. Generally, a protocol that lets a user on one computer system access and copy files to and from another computer system using a network connection. FTP does also refer to a standard high-level application which provides the described services to a user by utilizing the TCP/IP protocol suite. [42]

Geostationary Orbit (GEO) A circular orbit at $h = 35786km$ and an *inclination* of $i = 0$. The orbit period equals one *sidereal day*. Thus, a satellite on this orbit appears "stationary" with respect to an observer on Earth. [55]

Highly Elliptical Orbit (HEO) An elliptical orbit with a rather large eccentricity. Usually, communication between a HEO satellite and an observer on Earth is only possible when the satellite is near the perigee.

ICO Inter-Circular Orbit, a synonym for *Medium Orbit*

Inclined Walker constellation A satellite constellation for global coverage using N satellites in inclined circular LEOs with equal period. Usually referenced by the *Walker Notation N/P/F* in which P stands for the number of orbit planes and F for the phasing factor that determines the angular offset between the satellites in adjacent orbit planes. [55]

Inclination The inclination defines the angle between the orbit plane and the equatorial plane. It is computed positively with respect to the ascending satellite orbit track. [55]

Internet Protocol Refer to *IP*

IP An abbreviation of Internet Protocol. That part of the TCP/IP protocol that governs how packets of information are addressed for delivery throughout the Internet network. [42]

Line of Nodes The intersection between the equatorial plane and the orbit plane is called line of nodes. [55]

Link Object A link object represents in the context of Opnet modeler a definition of a transmission link. This link can be simplex or duplex and might represent a wireless or wired architecture including busses. It represents the associated pipeline stage procedures which are usually implemented in standard C and which modify or delay transmitted data accordingly to the link's characteristics.

Low Earth Orbit Orbits with an altitude in between 500–5000 km.

Nadir The "sub satellite point", i.e. the point where the perpendicular touches the earth when dropped from the satellite.

Neagle Algorith An TCP algorithm proposed in RFC 896 [60] which accumulates small junks of data if TCP has outstanding (un-acknowledged) segments. The collected data is sent when an ACK arrives.

Network node In the context of Opnet modeler, a network node may represent any kind of network device, i.e. a router, work station, firewall, or server. Network nodes are connected via link (nodes) to form a network.

Opnet Modeler An object-oriented, discrete-event based network simulator.

Orbit The path of a celestial body or an artificial satellite as it revolves around another body. [2]

Profile Definition Node The profile definition node is a standard component of Opnet's model library. It allow the definition of certain user profiles. Each profile might consist of several applications (which are defined with the *application definition node*). Such a profile definition might specify a user behavior of retrieving once a day a large file via FTP and talking randomly during the day with colleagues via an IP-telephony application.

Public Switched Telephone Network (PSTN) A domestic telecommunications network accessed by telephones, key telephone systems, private branch exchange (PABX) trunks, and data arrangements. [42]

Quadrature Amplitude Modulation (QAM) A method of modulating a carries with both amplitude modulation and phase modulation techniques to create a constellation of signal points, each representing one of the data points defined by a multi-bit data sample. [42]

Quadrature Phase Shift Keying (QPSK) A modulation technique whereby 2 bits of signaling data are taken together to cause the carrier's phase to be one of four states. [42]

RAAN See *Right ascension of the ascending node*

Range See *slant range*

Retransmission Time-Out (RTO) The RTO is calculated by TCP to determine the time at which a segment has to be retransmitted if it is not previously acknowledged.

Right Ascension of the Ascending Node (RAAN) The RAAN determines the angle between a reference direction and the line of nodes. The reference direction is given by the direction from the earth's center to the sun at vernal equinox. [55]

Round-Trip Time (RTT) The accumulated time of the forward propagation (and transmission) delay and the corresponding delays associated with the reverse channel.

Sidereal Day Time required by the Earth to rotate once with respect to the stars: $23\,h\,56\,min\,4.09\,s = 86164.09\,s$

Simple Network Management Protocol (SNMP) A standard protocol that provides a means to monitor network-related statistics and error conditions and to set network configuration and runtime parameters across the network. [42]

Slant range The slant range denotes the distance between the user terminal and the satellite. [55]

STK An abbreviation for Satellite ToolKit. A commercial-off-the-shelf analysis and simulation tool for satellite constellations.

Sub-Network Opnet forms networks out of network nodes and links in between them. For a better overview, the user may choose to accumulate a number of nodes in so called sub-networks. The sub-network is represented by a single network node and might be extended to see its contents.

TCP An abbreviation for Transmission Control Protocol. It is defined in RFC 793. [51] It is a transport layer protocol for packetizing data, managing the transmission of the packets, and performing error control. TCP is built on top of the Internet Protocol and is generally seen in the combination TCP/IP (TCP over IP). It adds reliable communication, flow-control, multiplexing, connection-oriented communication, and it provides full-duplex, process-to-process connections. [42]

TCP Reno A *TCP* flavor employing the *Fast Retransmit* and *Fast Recovery* algorithm.

TCP SACK A *TCP* flavor applying the selective acknowledge option. SACKs are used by the receiver to provide exact information about the packets correctly arrived. During the fast retransmission phase, the sender first retransmits all suspectedly lost packets before sending new ones. This allows to recover from several lost segments within one RTT. [16,33]

TCP Vanilla Refer to *TCPmin*

TCPmin *TCPmin* denotes throughout the text to a *TCP* flavor compliant to RFC 793 [46]. Thus, *TCPmin* equals to *TCP Reno* but without the *Fast Retransmit* and *Fast Recovery* algorithm.

Time Devision Multiple Access (TDMA) A form of time division multiplexing in which individual terminals are assigned a time to transmit a burst of information, i.e., the information from each signal source is transmitted sequentially across the common link. [42]

Vernal Equinox The date (near March 21 in the northern hemisphere) when night and day are nearly the same length and Sun crosses the celestial equator (i.e., declination 0) moving northward. In the southern hemisphere, the vernal equinox corresponds to the center of the Sun crossing the celestial equator moving southward and occurs on the date of the northern autumnal equinox. The vernal equinox marks the first day of the season of spring. [75]

Walker Orbit Refer to *Inclined Walker constellation*

Zenith The point on the celestial sphere that is directly above the observer. [2]

Index