Optimisation of Resilient Satellite Communications for Sustainable Digital Connectivity in Remote Rural Africa

ANAS ABUBAKAR BISU

Optimisation of Resilient Satellite Communications for Sustainable Digital Connectivity in Remote Rural Africa

Anas Abubakar Bisu

BSc., MSc., MSc., FHEA

A Thesis presented for the degree of
Doctor of Philosophy

Department of Engineering
Ustinov College
The University of Durham

June, 2020

Abstract

Digital connectivity using telecommunications network infrastructure has become indispensable for the socio-economic development of todays modern society. This digital connectivity is being achieved using terrestrial, satellite or the integration of both to form a more realistic heterogeneous communications network characterise by wireless/wireline, long/short latency, and high/low bandwidth. Satellite and Integrated Satellite-Terrestrial Networks (ISTNs) exhibit unique characteristics such as large bandwidth/capacity and global coverage capabilities, which is being exploited to connect even the remotest and poorest communities in the world in an optimally cost-effective and efficient way.

This thesis investigated the optimum, cost-effective, efficient and sustainable way to help bridge the digital divide between the mainly terrestrially connected developed cities and unconnected remote rural villages, particularly in the isolated remote rural African communities. This was achieved by optimised data transmission using the most widely used (*de facto*) standard transport protocol over the Internet in a pure satellite or hybrid ISTN environment that is characterised by large Bandwidth Delay Product (BDP). Transmission Control Protocol (TCP) accounts for 8090% Internet data traffic and applications over Internet Protocol (IP) nowadays, which makes it one of the most important protocols for data transmission over the Internet.

Experiments were carried out to measure the practical and simulated End-to-End (E2E) latencies of Satellite and ISTN environments from which a framework for E2E latency of ISTN environments was developed for quantifying practical E2E latency of pure satellite and ISTN environments. The practical E2E latency was measured by a passive measurement method using two testbeds of Geostationary satellite network providers that transmit and receive voice over IP signals while the active method was used by the simulation experiments, in which File Transfer Protocol (FTP) traffic over TCP enabled within the designed ISTN topology was created. The measured E2E latencies were used to investigate the impact of long Round-Trip Time (RTT) of Satellite and ISTN environments on large BDP and the standard TCP schemes.

Following the successful E2E measurements and framework development, this thesis also developed and propose an enhanced congestion control algorithm called TCP HYBIC for long RTT and high-speed networks such as ISTN and pure satellite environments. HYBIC is based on large BDP CUBIC and HYBLA TCP algorithms. HYBIC performance analysis and evaluation in comparison to these large BDP schemes were conducted using numerical and simulation methods. The performance analysis and evaluation of HYBIC in comparison to a standard, CUBIC, and HYBLA TCP schemes showed that HYBIC achieves performance in terms of packet delivery ratio of up to 99.96%, jitter of 34 μs, and efficient capacity utilisation of up to 67%. These have overcome the challenges of standard TCP over long RTT and large BDP paths, which form part of the realistic communications network nowadays and the future by improving the utilisation and packet delivery rate.

Finally, as part of future work, this thesis recommended exploring the effect of background traffic on fairness and the effect of more complex network topologies could also be interesting. Experiments with more queue management schemes such as Active Queue Management (AQM) where packets are dropped from the queue before the queue overflows could also be an interesting work for the future. Potential areas of applications such as Tele-medicine and Tele-agriculture could be tested with network topologies designed for resilient and sustainable Satellite and ISTN communications. These would be useful for Telecommunications services and application in remote rural areas, and for emergency/disaster management when terrestrial communications infrastructure failed in the event of a disaster.

Contents

List of Tables

List of Figures

Acronyms and Symbols

Acronym, Symbol	**Meaning**
OWD	One Way Delay
RTT	Round Trip Time
ABW, R_b	Average/Achievable Bandwidth, Throughput
rwnd	Receiver Congestion Cindow
ACK	Acknowledgment
BDP	Bandwidth Delay Product
ACM	Advanced/Adaptive Coding and Modulation
IST(N)	Integrate Satellite Terrestrial (Networks)
AIMD	Additive Increase and Multiplicative Decrease
CWND, W	Congestion Window
BW, C	Bandwidth, Capacity
E2E	End to End
TCP	Transmission Control Protocol
FTP	File Transfer Protocol
IP	Internet Protocol
VoIP	Voice over Internet Protocol
UDP	Userdatagram Protocol
HEO	High Elliptical Orbit
GEO	Geostationary/Geosynchronous Earth Orbit
MEO	Medium Earth Orbit
LEO	Low Earth Orbit
Tx, Rx	Transmit, Receive
CCA	Congestion Control Algorithm
PDR	Packet Delivery Ratio
LFN	Long Fat Networks
SatComs	Satellite Communications

Acronym, Symbol	Meaning
FSS	Fixed Satellite Service
MSS	Maximum Segment Size
MTU	Maximum Transmission Unit
QoS	Quality of Service
QoE	Quality of Experience
ITR, R(t)	Instantaneous Transmission Rate
BER, PER	Bit Error Rate, Packet Error Rate
W(t)	Congestion Window Growth Rate
HTS	High Throughput Satellite
RTO	Retransmission Timeout
SYN	Synchronisation
Seqn	Sequence Number
CBR	Constant Bit Rate
PLMN	Public Land Mobile Network
IW	Initial Congestion Window
ssth, ssthresh, γ	Slow Start Threshold
SaT5G	Satellite and Terrestrial for Fifth Generation
IoE, IoT	Internet of Everything, Internet of Things
eMBB	Enhances Mobile Broadband
mMTC	Massive Machine Type Communications
URLLC	Ultra Reliable and Low Latency Communications
M2M, G	Machine to Machine, Granularity
COT	Commercial-off-the-Shelf
PPP	Public Private Partnership
NRR	Network Resilience and Recovery
RTTVAR/RTTM/SRTT	RTT Variation/RTT Measurements/Smooth RTT
dupACKs/pACK/unACK	Duplicates ACKs/Partial ACK/Not ACK
SS/CA	Slow Start/Congestion Avoidance
GWS	Ground/Gateway Station
SACK	Selective Acknowledgement
ECN	Explicit Congestion Notification
ECT	ECN Capable Transport

Acronym, Symbol	Meaning
FRet/FRec	Fast Retransmit/Fast Recovery
IPDV	Internet Packet Delay Variation
PSTN	Public Switch Telephone Network
SSS	Satellite Space Segment
SUT/UE	Satellite User Terminal/User Equipment
SAS/SCC	Satellite Acces Station/Satellite Control Centre
BGAN	Broadband Global Area Network
PEP	Performance Enhancement Proxies
NS/NAM	Network Simulator/Network Animator
OTCL	Object Oriented Tool Command Language
SNP	Satellite Network Provider
SSNL/SSL	Satellite Satellite Network Link
STNL/STL	Satellite Terrestrial Network Link
B/bps	Byte/Bit Per Seconds
pps	Packet Per Seconds
Hz	Hertz
Eff, η	Efficiency
FSL	Free Space Loss
SNR, dB	Signal to Noise Ratio/Decibel
SDN/CDN	Software Defined Network/Contents Defined Network
T&D	Transmission and Distribution
SGC	Smart Grid Communications
AMI	Advanced Metering Infrastructure
DA	Distribution Automation
DR	Demand Response
SCADA	Supervisory Control and Data Acquisition
DER	Distributed Energy Resources
SM/EV	Samrt Meter/Electric Vehicle
DGM	Distributed Grid Management
FIN	Finish
MIMO	Multiple Input Multiple Output
P2P	Point to Point
FR/FMT	Frequency Re-use/Fade Mitigation Technique

Acronym, Symbol	Meaning
ADSL	Asymetric Digital Subscriber Line
FTTH	Fibre to the Home
RANs	Radio Access/Area Networks
WLAN	Wireless Local Area Network
WiFi	Wireless Fidelity
HAN	Home Area Network
NAN	Neighbourhood Area Network
MAN	Metropolitan Area Network
WAN	Wide Area Network
WASA	Wide Area Situation Awareness
BNL	Bottleneck Link
NOC	Network Operation Centre
WMN	Wireless Mesh Network
NETP	National Emergency Telecommunication Plan
ResISTN	Resilient Integrated Satellite Terrestrial Network
ReSatComs	Resilient Satellite Communications
ISL	Inter Satellite Link
CGC	Complementary Ground Component
ESA	European Space Agency
ARTES	Advanced Research in Telecommunication Systems
SDLC	Software Development Life Circle
pkts/PSZ	Packets/Packet Size
CRC/FEC	Cyclic Redundancy Check/Forward Error Correction
ICT	Information and Communication Technology
IMT	International Mobile Telecommunications
PTR	Packet Transmission Rate
$RTT_{ref} = RTT_0$	Reference RTT
HTCP/STCP	High-speed TCP/Scalable TCP
$t_{ssth} = t_\gamma$	Time to reach Threshold Value γ

Publications

Proceedings

1. A. Bisu, A. Gallant, H. Sun, K. Brigham, and A. Purvis, "Experimental Performance Evaluation of TCP Over an Integrated Satellite-Terrestrial Network Environment," *in Proc. IEEE IWCMC*, pp. 781-786, 2019.

2. A. Bisu, A. Gallant, H. Sun, K. Brigham, and A. Purvis, "Telemedicine via Satellite: Improving Access to Healthcare for Remote Rural Communities in Africa," *in Proc. IEEE HTC'10.*, pp. 1-6, 2018.

3. A. Bisu, A. Purvis, K. Brigham, and H. Sun, "A Framework for End-to-End Latency Measurements in a Satellite Network Environment", *in Proc. IEEE ICC*, pp. 1-6, 2018.

Presentations

4. A. Bisu, K. Brigham, H. Sun, A. Purvis, and A. Gallant, "Telemedicine: Exploring the Potentials of Satellite Communications for Remote African Villages," *WRIHW 3rd Early Career Researcher Conference (ECR)*, Wolfson Research Institute for Health and Wellbeing, Durham, 2018

5. A. Bisu and A. Purvis, "Resilient Digital Technology: Design and Development of Digital Technology Services for Remote Villages of Africa," *Engineering Complexity Resilience Network (ENCORE+) Conference*, Sheffield, 2017.

Declaration

The work in this thesis is based on my research carried out at the Department of Engineering, England. No part of this thesis has been submitted elsewhere for any other degree or qualification and it is all my own work unless referenced to the contrary in the text.

Acknowledgements

I would like to sincerely thank and extend my profound appreciations to my supervisors; Prof. Andrew Gallant, and Prof. Hongjian Sun, as well as my previous supervisors; Emeritus Professor Alan Purvis and Dr. Katharine Brigham. Your immense contributions and guidance from start to completion of this research is enormous, I could not find a word to express how I feel and what is acquired under your guidance. I would like to especially extend my appreciation to my primary supervisor, Prof. Andrew Gallant who led me to the successful completion of this thesis with his sense of humour, expertise, professionalism, and mentorship, I am indeed lucky and grateful for the opportunity to work with you, thank you for always being there to listen, welcoming, and the guides out of difficulties. To Prof. Hongjian Sun, a big thank you for the guidance and funding under EUH2020 that provide me an opportunity to have European Telecommunications industry experience and working on practical problems with the industry expert at Hellenic Telecommunications Organisation (OTE) in Athens, Greece. To Dr. Katharine Brigham, thank you for the guidance and help in paper writing and coding my proposed algorithm. Finally, a big thank to Emeritus Prof. Alan Purvis, the man who initiated and set path to this research by providing initial funding under GCRF for the feasibility study conducted in Nigeria and European Space Agency (ESA) in Nordwijk, Netherlands and short course at Oxford University that provided me with an opportunity to learn more about satellite technology and network with industry experts in Europe and beyond. I would like to thank Ian, Neil and Colin from the electronics workshop for their help with any electronic hardware issues, always ready to help out with new ones or troubleshoot the old device.

My sincere appreciation the constructive criticisms and feedback of my annual

review team and internal examiner, Assoc. Prof. Claudio Balocco and Prof Dagou Zeze. I would also like to sincerely thank my external examiner Professor Xiaoli Chu for her objective and useful feedback and comments during the Viva voce examination.

My profound gratitude and appreciation to my lovely and caring parent, *Alhaji Garba Bisu* and *Late Hajiya Hauwa'u Muhammad Jariri* for their tireless effort in training, guidance, help and support from my childhood to adulthood that made me who I am today.

To my beloved and gorgeous wife *Amina* and my lovely daughter *Amal*, thank you for your patience and understand during the difficult times of this journey, I really appreciate your presence and all the coffee, Tea, food, and entertainment looking at Amal running up/down healthy and strong that kept me on track to achieve what I thought might not be achievable.

Finally, my profound gratitude to my dear nation Nigeria through Petroleum Technology Development Fund (PTDF) for the rare and merited scholarship opportunity under the overseas Scholarship scheme (OSS) of the Agency that brought me here in the first place and carter for all my financial needs to carry out this research, met amazing people from different culture and race. I felt indebted to my nation for providing me with this golden and rare opportunity, I am now prepared and always ready to serve in any capacity that helps in our nation's building.

Dedication

To the loving memory of my beloved Mother **Hajiya Hauwau Muhammad Jariri**, *I pray, may her gentle soul continue to rest in perfect peace may ALLAH grant her the highest place in Jannatul firdaus*

Chapter 1

Introduction

1.1 An Overview

Satellite Communications (SatComs) have unique features such as global coverage, high availability anywhere/anytime, high bandwidth capacity and resilience for ubiquitous digital connectivity [1–4]. These features could be exploited and explored to potentially bridge the digital divide that exists due to lack of access to the telecommunications network infrastructure in remotely isolated rural areas for digital connectivity and services, particularly rural communities in Africa. However, SatComs also exhibit the undesirable attributes of high latency due to long propagation delays, particularly the Geostationary Earth Orbit (GEO) satellites covering large footprints of multiple continents. Intermittent connectivity in lower orbit satellite links with low latency such as Medium/Lower orbit (MEO/LEO) satellites and generally high wireless link errors in satellite networks contribute significant performance degradation when using SatComs for data communications. These key characteristics of SatComs lead to performance degradation and capacity underutilisation in data communications, especially using Transmission Control over the Internet Protocols (TCP/IP) [5–12], which account for 80-90% of the internet traffic nowadays [13–15]. For effective and efficient use of SatComs to connect the isolated remote rural communities in Africa, and beyond, the negative impact of long communication delays need to be investigated and addressed to achieve optimal performance of TCP over satellite channels and sustainable connectivity.

Optimised, resilient and sustainable SatComs networks could be an effective solution for closing the digital gap in the near future. This thesis investigates the impact of SatComs long latency on data communications performance over the Internet using the TCP over high capacity and high delay GEO satellite channels with Fixed Satellite Service (FSS) and an Integrated Satellite-Terrestrial Network (ISTN) environment for real-time voice and Internet applications in remote isolated areas. The key goals were to identify and find how to optimise the performance of this protocol over SatComs or hybrid network environments for better utilisation of the huge available satellite capacity. This has applications in remote rural areas, particularly in Africa for socio-economic development and growth of the communities using an improved digital connectivity.

Mitigating the negative impacts of long latency on TCP transmission rate (Throughput) could optimise the data communications performance and provide good quality of service (QoS) while providing optimum and efficient utilisation of expensive bandwidth/capacity as shown in Fig. 1.1.

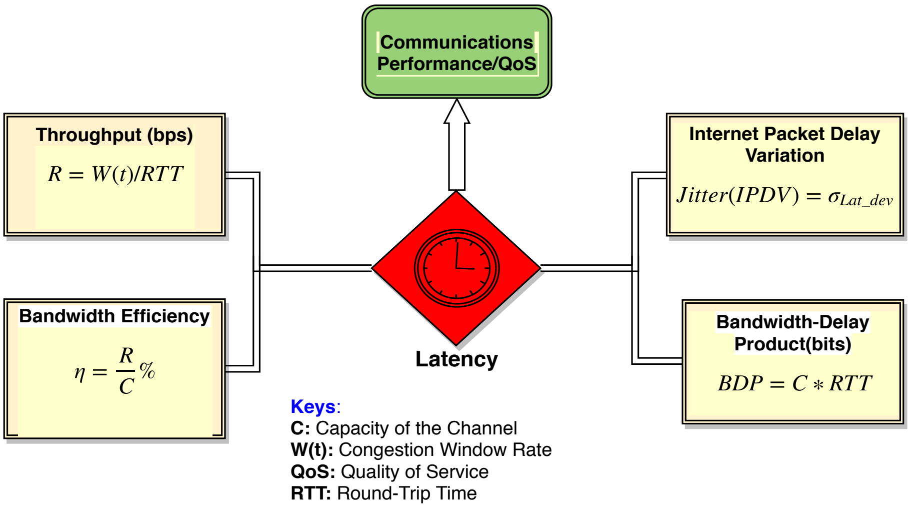

Communications Performance/QoS
Throughput (bps)
$R = W(t)/RTT$
Bandwidth Efficiency
$\eta = \frac{R}{C}\%$
Latency
Internet Packet Delay Variation
$Jitter(IPDV) = \sigma_{Lat\_dev}$
Bandwidth-Delay Product(bits)
$BDP = C * RTT$
Keys:
C: Capacity of the Channel
W(t): Congestion Window Rate
QoS: Quality of Service
RTT: Round-Trip Time

Figure 1.1: Latency Dependent TCP Performance and QoS Parameters

On the other hand, the sustainability of communications could be improved using cheaper and widely available powering options in order to maintain long term sustainable connectivity. Satellite provides resiliency, but more options and use cases scenarios will be investigated by this research to improve resilience and sustainability.

1.2 Motivation

The lack of terrestrial communications infrastructure, poor availability and services in remote rural areas keep about 67% of 1.3 billion Africans digitally disconnected to the rest of the world [16]. About 62% of the African population live in sparse remote villages with extremely low population density and difficult terrain (rocks, mountains, forest, landslides/erosions, rivers etc.), unfriendly climate (extremely hot and rainy) and poverty without much economic activities [17]. These and other factors make deployment and maintenance of terrestrial communications infrastructure near impossible, or not economically feasible, subjecting at least 340 million people to travel at least 50 km to access the Internet in sub-saharan Africa [18]. This digital isolation in most African communities contributed immensely to their underdevelopment with a consequence on socio-economic development opportunities such as e-health, e-learning, e-agriculture, emergency and disaster management enabled by digital services [17].

Careful performance optimisation of Internet data communications over satellite channels and exploration of the unique features of SatComs such as global coverage (anywhere), high capacity, fast/easy deployment, high availability (anytime) of about 99.999%, accessibility and with highly resilience could potentially be utilised to digitally connect the unconnected areas in Africa, and beyond, for socio-economic developments. These include improved healthcare delivery using Telemedicine, Agriculture using Tele-Agriculture and quality education using Tele-Education.

However, the de-facto standard TCP algorithm currently being used by the IP networks penalises wireless and long distance network links with high wireless link errors or Packet Error Rate (PER) and long Round-Trip Time (RTT). These degrade performance, underutilise the available capacity and give an unfair share of available bandwidth in a heterogeneous network environment involving short distance wired or wireless and satellite channels. The performance degradation of TCP due to these attributes of the satellite link needs to be investigated. Moreover, better solutions are needed that could be optimum and efficient over GEO satellite channels which have high capacity and global coverage advantages, and the potential to bridge the digital gap even in the remotest rural areas of the world, except the polar regions.

1.3 Aim

The aim of this thesis is to investigates the practical performance of data communication, particularly IP traffic applications using the TCP/IP over heterogeneous networks involving at least a satellite link. This investigates a way to improve the performance of TCP algorithms based on schemes such as *HYBLA, CUBIC* and *NewReno* for better performance and efficient utilisation of the available capacity of satellite systems, rather than using a de-factor standard of TCP algorithms in most operating systems nowadays, which often results in degraded performance and under-utilisation of capacity over satellite channels due to the attributes mentioned earlier. The research also exploits the advantages such as high capacity and global coverage of satellite networks for bridging the digital divide, which could be achieved using the objectives in the subsequent section.

1.4 Contributions

Standard TCP algorithm implementations depend largely on acknowledgements (ACKs) and RTT of transmitted data packets with blind tagging of any lack of acknowledgement from the packets receiver as a loss event due to network congestion. These assumptions might be wrong and highly penalise and degrade the performance of realistic heterogeneous networks incorporating wireless links with high PER and satellite links with additionally long RTT. Alternative TCP algorithms that remove dependencies or reduced the negative impact of long RTT of satellites links and high PER are investigated for enhanced performance of TCP over heterogeneous networks such as ISTNs. The contributions of this thesis are listed as follows;

1. Study and explore the existing TCP algorithms for Satellite such as HYBLA and CUBIC and develop a new scheme for enhanced performance of TCP over satellite or hybrid networks in terms of throughput, capacity utilisation, Jitter and packet delivery ratio.

2. Develop model and framework for E2E latency measurement in a real satellite

testbed. This was achieved by developing different case study scenarios for pure and hybrid satellite (ISTN) environments. Conduct performance analysis and evaluation of the actual E2E latency measured using the the testbed and investigated its impacts on the standard TCP.

3. Enhanced TCP algorithm for heterogeneous networks incorporating GEO satellite links. My new improved algorithm combines the modifications of slow start and congestion avoidance phases of both HYBLA and CUBIC algorithms while integrating them to benefit from their unique advantages such as stability and RTT-fairness and friendliness to other schemes. This aimed to achieve better throughput and packet delivery performance of TCP over heterogeneous and large BDP networks with efficient utilisation of huge available capacity of GEO satellite links as compared with previous TCP schemes.

4. Designed and developed resilient and sustaible network scenarios for the potential applications in remote rural areas, particularly digitally unconnected African remote rural communities.

1.5 Thesis Structure

Chapter 1: This chapter introduces the thesis background, the motivation, aims, objectives and contributions, and the thesis structure.

Chapter 2: Reviews the potential and unique characteristics of SatelliteComs, which provides an insight as to why this research focused on use of SatComs instead of its terrestrial counterpart. The chapter also highlights the research and development frontiers of the satellite industry, as well as future of SatComs for emerging technologies and how these could be effectively applied in isolated remote rural areas that are digitally disconnected to the rest of the world for socio-economic development of these areas and communities.

Chapter 3: The chapter presents an overview of transport protocol, particularly the standard TCP/IP, TCP over SatComs channels and the User Datagram Protocol (UDP). This chapter discusses the original (de-facto) standard TCP algorithms and relevant schemes of TCP aimed at solving performance disparity for different network environments such as GEO satellite link in a more realistic heterogeneous networks nowadays and also called ISTNs environment.

Chapter 4: In this chapter, the experiments for channel Measurements, Simulations, Emulation, Performance Evaluation and Analysis using the results obtained is presented. The chapter discusses real-world testbeds and use cases scenarios used for the practical measurements of the performance of realistic heterogeneous networks. The scenarios developed for the channel measurement and framework proposed for End-to-End (E2E) latency measurement in a satellite network environments are also presented and discussed in relation to performance of real heterogeneous networks (ISTNs) incorporating at least a leg of GEO satellite channel.

Chapter 5: The chapter presents the mathematical modelling for an optimised TCP algorithm over heterogeneous (IST) networks called TCP HYBIC, numerical analysis of performance, and simulated implementation, testing and evaluation of the new HYBIC proposal. This chapter also compared the performance of the new improved proposal (HYBIC) with HYBLA and CUBIC schemes, these formed the basis of HYBIC proposal for improving performance of TCP over satellite communications channels.

Chapter 6: This chapter draws the conclusions, recommendations and directions for future work.

References

[1] S. Kota, and M. Marchese, ”Quality of Service for Satellite IP Networks: A Survey”, *International Journal of Satellite Communications and Networking*, Vol. 21, pp. 303-349, John Wiley, 2003.

[2] T. Gayraud, and P. Berthou, ”A QoS Architecture for DVB-RCS Next Generation Satellite Networks”, *EURASIP Journal on Wireless Communications and Networking*, Vol. 2007, pp. 1-9, Hindawi Publishing Co., 2007.

[3] G. Maral and M. Bousquet, *Satellite Communications Systems: Systems, Techniques and Technology*, 5th Ed. John Wiley, West Sussex, UK, 2009.

[4] D. Minoli, *Innovations in Satellite Communications Technology: The Industry Implications of DVB-S2X, High Throughput Satellites, Ultra HD, M2M and IP*, John Wiley, 2015.

[5] V. Jacobson, and R. Braden, ”TCP Extensions for Long-Delay Paths”, *Network Working Group*, RFC 1072, 1988.

[6] B. Barakat, E. Altman, and W. Dabbous, ” On TCP Performance in a Heterogeneous Network: A Survey”, *IEEE Communications Magazine, Telecommunications at the Start of the New Millennium*, pp. 40-46, 2000.

[7] T. R. Henderson, and R. H. Katz, ”TCP Performance over Satellite channels”, *University of California Berkeley, Report No.*, UCB/CSD-99-1083, California, 1999.

[8] J. S. Stadler, and J. Gelman, "Performance Enhancement for TCP/IP on a Satellite channel", *In Proc. of IEEE Military Communications Conference (MILCOM)*, Vol. 1, pp. 270-276, 1999.

[9] C. Metz, "TCP over Satellite...The Final Frontier", *IEEE Internet Computing, On the Wire*, pp. 76-80, 1999.

[10] N. Ghani, and S. Dixit "TCP/IP Enhancement for Satellite Networks", *IEEE Communications Magazine*, pp. 64-72, 1999.

[11] M. Allman, D. Glover, and L. Sanchez, "Enhancing TCP Over Satellite Channels Using Standard Mechanisms", *The Internet Society/IETF*, RFC 2488 , 1999.

[12] M. Allman, et al., "Ongoing TCP Research Related to Satellites," *Internet Society Network Working Group, RFC*, 2760, BCP. 28 2000.

[13] J. N. Pelton, S. Madry, and S. Camacho-Lara, *Handbook of Satellite Applications*, 2nd Ed., Vol. 1, pp. 629-647, Springer, 2017.

[14] R. Prasad, C. Dovrolis, M. Murray and K. Claffy, "Bandwidth Estimation: Metrics, Measurement Techniques, and Tools", *IEEE Network*, Vol. 17, Issue: 6, 2003.

[15] T. Braun, M. Diaz, J. E. Gabeiras, and T. Staub, *End-to-End Quality of Service Over Heterogeneous Networks*, Springer, Berlin, 2010.

[16] Hootsuit and We are Social, Digital in 2018: Essential Insights Into Internet, Social Media, Mobile and E-Commerce Use Around the World, accessed 17/04/2018 via https://wearesocial.com/uk/blog/2018/01/global-digital-report-2018. Jan 2018.

[17] M. Franci, " High Throughput Satellites (HTS)...An Opportunity For All", SatMagazine, Worldwide Satellite Magazine, pp. 46, Satnews Publishers, CA USA, 2016

[18] A. Hall, " Future-Sat Africa...To Connect All", SatMagazine, Worldwide Satellite Magazine, pp. 38, Satnews Publishers, CA USA, 2016

[19] C. Caini, and R. Firrincieli, "TCP HYBLA: A TCP Enhancement for Heterogeneous Networks", *International Journal of Satellite Communications and Networking (IJSCN)*, Vol. 22, pp. 547-566, John Wiley, 2004.

[20] S. Ha, I. Rhee, and L. Xu, "CUBIC: A New TCP-Friendly High-Speed TCP Variant", *ACM SIGOPS Operating Systems Review - Research and Developments in the Linux Kernel*, Volume 42 Issue 5, pp. 64-74, 2008.

[21] C. Caini, R. Firrincieli, and D. Lacamera, "Comparative Performance Evaluation of TCP Variants on Satellite Environments", *Proceedings of IEEE ICC*, IEEE, 2009.

Chapter 2

Satellite Communications and Potentials in Remote Rural Africa

2.1 Satellite Communications

Telecommunications has become indispensable for the socio-economic development of humankind. This has being the key enabler of the Information and Communications Technology (ICT) that drive the digital information age. Developments in electronics led to the discovery of effective terrestrial and satellite telecommunication systems with vast applications in almost all areas of life nowadays. As mentioned in chapter 1, satellite communications (SatComs) features like global coverage, rapid/cheaper deployment, high availability, scalability, reliability, broadcast and multicast support [1–4], can be exploited to provide connectivity and enable services even in the remotest rural areas of the world and help bridge the digital divide between the connected and unconnected communities for making the world a truly global village. However, terrestrial networks have advanced faster [5] in recent years, offering cheaper solutions in developed, and accessible in urban environments of the developing nations. This is not economically feasible in remote areas with extremely difficult harsh terrain and not reliable during natural disasters that cause damage to the ground infrastructures [2]. In the past, SatComs technologies did not advance at the same pace as terrestrial communications due to the service cost and interoperability challenges [6]. However, the potential of SatComs and heterogeneous ISTNs in the socio-economic growth of isolated rural areas and the future generation of communications networks such as 5G New Radio (NR) led to the full commitment of key industry players to revamp SatComs technologies [5]. In recent years, significant research efforts are being made to increase the capacity of satellite systems using High Throughput Satellites (HTS), low-cost Nano/Micro satellites, flexible payload, portable and cheaper ground terminals to prepare for the anticipated new services opportunities as an integral part of next generation networks like ISTNs shown in Fig. 2.1 [1, 6, 7].

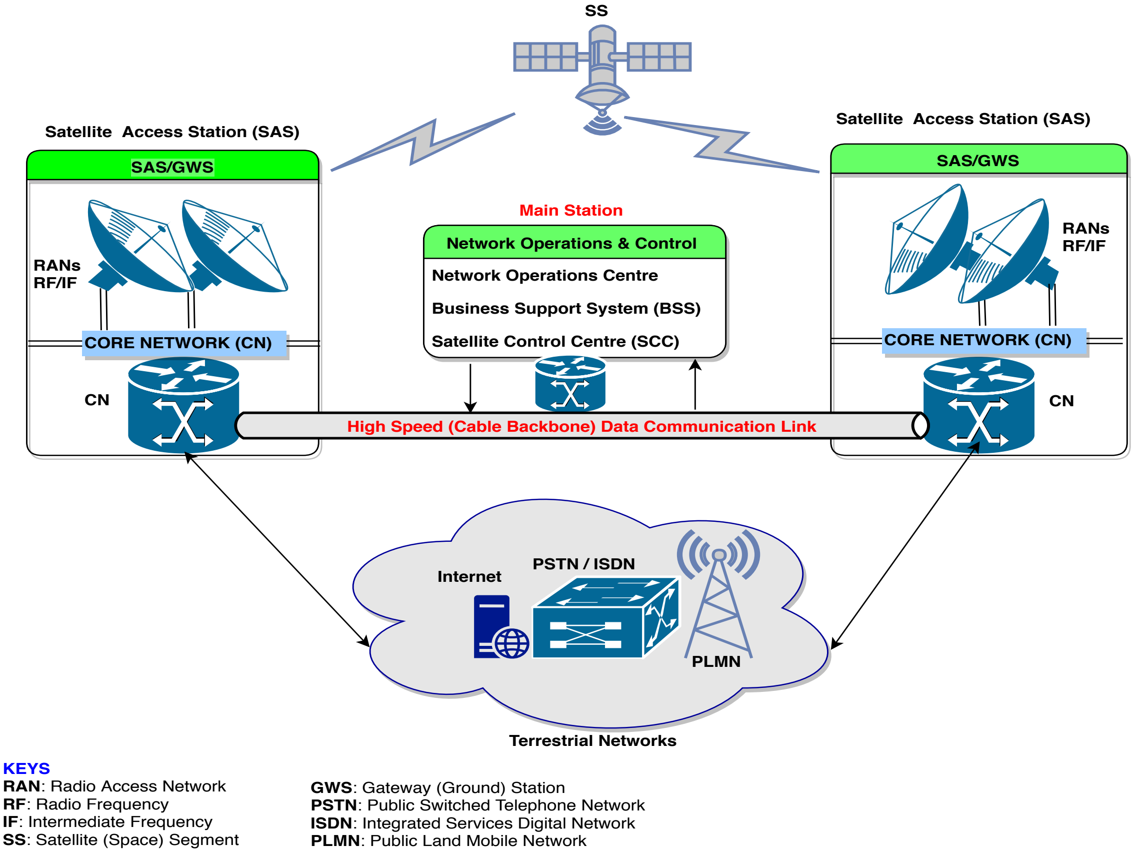

SS
Satellite Access Station (SAS)
SAS/GWS
RANs
RF/IF
CORE NETWORK (CN)
CN
Main Station
Network Operations & Control
Network Operations Centre
Business Support System (BSS)
Satellite Control Centre (SCC)
High Speed (Cable Backbone) Data Communication Link
Satellite Access Station (SAS)
SAS/GWS
RANs
RF/IF
CORE NETWORK (CN)
CN
Internet
PSTN / ISDN
PLMN
Terrestrial Networks
KEYS
RAN: Radio Access Network
RF: Radio Frequency
IF: Intermediate Frequency
SS: Satellite (Space) Segment
GWS: Gateway (Ground) Station
PSTN: Public Switched Telephone Network
ISDN: Integrated Services Digital Network
PLMN: Public Land Mobile Network

Figure 2.1: A Typical Topology of Integrated Satellite-Terrestrial Network

High availability (99.999%) of SatComs everywhere at anytime including during natural disasters is an advantage to explore for solving the digital divide in the African remote villages [8, 9]. We proposed an optimised Internet data transmission using pure SatComs and hybrid ISTN as the best candidate that could provide the novel solution. The key challenges for the optimum performance of SatComs include the negative effect of long latency, degraded achievable throughput and underutilisation of the satellite high capacity.

Digital services value chain includes climate corporations using satellite data for agricultural projections, which can be a means to pay for sustained connectivity in remote poor rural communities [10]. Terrestrial communications networks closed the digital divide between developed countries and urban areas in African, but not in remotely isolated rural Africa, where satellite and ISTNs will play a vital role. Africa's vastness, resources and sector strengths are poised to reap more benefits from SatComs with its 45% mobile penetration projected to reach 54% by 2020 with SatComs help to mobile operators achieve this goal [10].

2.2 Recent Development in SatComs

In recent years, SatComs witnessed a remarkable increase in research and industrial efforts to improve technologies at a global scale due to ever-increasing demands of SatComs services and applications to complement terrestrial communication networks. The renewed research and industrial efforts towards advancing SatComs technology focused on equipment cost reduction (in ground, launch and space segments), size, capacity improvement, quality of service, and protocol performance enhancement in an heterogeneous network environment with hybrid satellite, terrestrial, wired and wireless channels such as ISTNs shown in Fig. 2.1. These also help in the effective integration, performance, and coverage expansion for the future 5G networks [7, 11, 12]. These developments have been summarised in the subsequent sub-sections.

2.2.1 Capacity Expansion: High Throughput Satellites

The recent demand for high data rate, wider coverage and advanced capacity communication systems led to the development of a new generation of satellites in K-Bands at 10-18 GHz/18-30 GHz (Ku/Ka) [9], most prominent are KA-SAT and High Throughput Satellites (HTS). Developed to offer broadband access services comparable to terrestrial Asymmetric Digital Subscriber Line (ADSL 2+) at the same cost to millions of users globally using smaller terminals [13]. The KA-SAT and HTS can offer data rate up to 90 Gbps and expected to increase tremendously in the near future to rival the capacity and quality of Fibre-To-The-Home (FTTH) terrestrial systems [13, 14].

By 2020, the next generation operational satellites are envisioned to have up to 1 Tbps capacity with reduced cost per MB using higher frequency Q, V and W bands (40-110 GHz), and smart MIMO antennas, larger beams and transponders [13–15]. The capacity provided by the 1st and 2nd generations of Ka-band like iPSTAR, WildBlue I, SpaceWay 3, KA-SAT, ViaSat1 and other HTS is about 20-to-100 Gbps [16, 17]. HTS/KA-SAT experienced variable attenuation due to rain and employ Adaptive Coding and Modulation (ACM) as a Fade Mitigation Technique (FMT) [16]. Spectral efficiency is achieved by multiple narrow antenna beams (multi-Spot beams of up 82-spots) that allow a higher (x 20) Frequency Re-use (FR) factor [13, 16, 18].

The quest for more capacity in the future requires more available satellite bandwidth; this needs exploitation of higher bands beyond Ka-band. Recent research efforts have focused on exploiting Q/V-Band (40-75 GHz), E-band (71-86 GHz) and W-band (75-110 GHz) for higher capacity in the next generation satellites for 5G network applications [19, 20]. Key research focus in the future on high-capacity satellites systems will be on the effects of weather and power efficiency at Q/V/E/W-bands. The use of Medium Earth Orbit (MEO) Ka-band HTS systems reduces the long latency problems of Geostationary Earth Orbit (GEO) in L-band systems [9]. Further research is needed to expand the capacity and portability of satellite user/ground terminals to avoid bottlenecks and ease of use in communications to compete with terrestrial systems. Design of future high-capacity satellite systems require critical considerations of dependent parameters like capacity, data rate; channel bandwidth, spectrum efficiencies and power. Interdependence of these design parameters are given by Eqns. (2.2.1) to (2.2.5) [13, 21].

$$C = B_w log_2(1 + \frac{S}{N}) \tag{2.2.1}$$

where C (in bps) is the Shannon's theoretical maximum capacity limit of the system, B_w is the channel Bandwidth (in Hz), and $\frac{S}{N}$ is the dimensionless Signal-to-Noise Ratio (SNR in dB). The achievable bit rate (throughput) $R_b \leq C$ can be computed from Shannon-Hartley theorems in Eqn. (2.2.2) and spectral efficiency η (in b/Hz) determined by (2.2.3) below.

$$R_b \leq B_w log_2(1 + \frac{S}{N}) \tag{2.2.2}$$

$$\eta = \frac{R_b}{B_w} \leq log_2(1 + \frac{S}{N}) \tag{2.2.3}$$

Therefore, η is inversely proportional to the B_W and directly proportional to the SNR of the transmission system give by:

$$SNR = \frac{P_{EIRP}}{FSL} XG/T \tag{2.2.4}$$

Simplifying 2.2.4 gives SNR as:

$$SNR = \frac{P_t G_t G_r}{(\frac{4\pi D}{\lambda}) K T_s B_w} \tag{2.2.5}$$

where P_{EIRP} is an effective isotropic radiated power in watts (W), FSL is the free space loss, G/T is the figure of merit, D signal path length (in km), λ is the wavelength, k is the Boltzmann constant, T_s system temperature (in kelvin), G_t and G_r are dimensionless transmitter and receiver antenna gains respectively.

The theoretical maximum channel capacity C and transmission rate, R_b (achievable throughput) directly depend on the channel bandwidth, B_w and SNR as shown in Eqn. (2.2.1) and (2.2.2). On the other hand, the channel efficient utilisation, η (efficiency) depends on capacity and throughput as given in Eqn. (2.2.3) or the SNR, which is derived using Eqn. (2.2.4) and (2.2.5).

Therefore, HTS will revolutionise communications by offering the needed high capacity systems to serve the accelerating growth in high data rate demands of the era by the end of decade for the next decades. These will provide cost effective high-speed/throughput Internet access and connectivity anywhere in the world, be it isolated remote areas, at sea or in the air. This is a key potential for 5G and Internet of Things (IoT) connected devices as discussed in the subsequent sections.

2.2.2 Size, Weight and Cost Reduction

Research in miniaturisation of SatComs technologies (space and ground segments) has gained much interest in recent years due to the institutional and individual user demands for smaller, portable and cheaper satellite systems [22, 23].

Satellites (spacecraft) are classified as: large, medium or small (mini to femto classes) based on the in-orbit wet mass (payloads and fuel) [23]. Miniature or Small Satellites (SmallSats) have launch weight (or wet mass) of less than 500 kg with different weight categories as mini or SmallSat (100-500 kg), MicroSat (10-100 kg), NanoSat (1-10kg), PicoSat (0.1-1 kg) and FemtoSat (less than 0.1 kg) [23–25]. NanoSats and PicoSats (1-1.33 kg and 10 cm^3 or "1U" Units form factor) assembled using commercial-off-the-shelf (COTS) components are called CubeSats. These could be standalone with limited functionality or as a block for a larger NanoSat such as "3U" NanoSat contains three CubeSats shown in Fig. 2.2 and other SmallSats family characteristics [23, 26, 27].

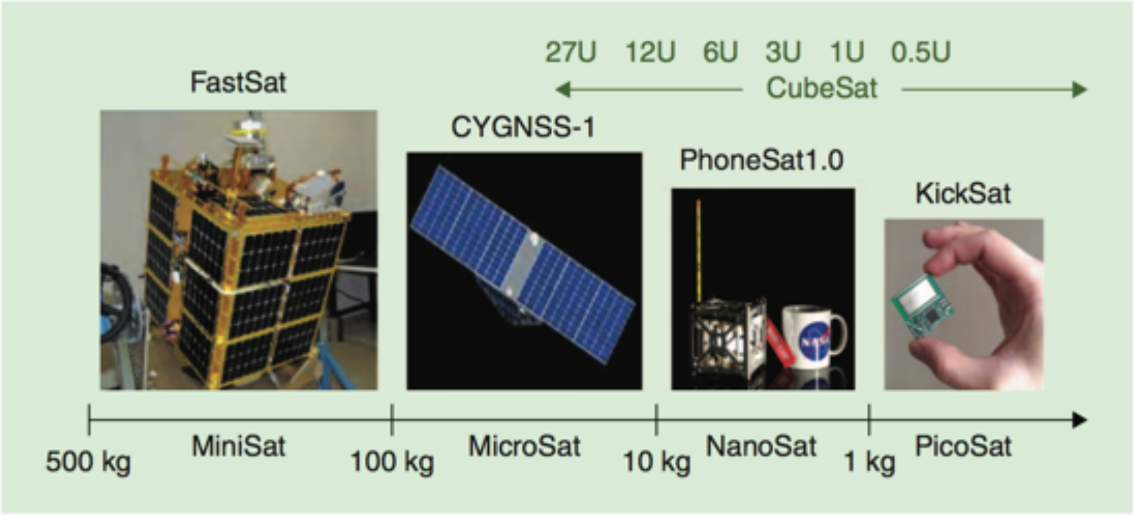


Figure 2.2: SmallSat Family Dimensions and Weights (adapted from [26])

The major benefit of SmallSats is cost reduction in both development and deployment, which makes them attractive for future satellite applications, designed and developed by educational/research institutions and small companies [25]. To save the launch cost, SmallSats can be launched in multiple numbers (swarm) and as piggybacks by sharing a ride (ride-sharing) on larger launch vehicles and from international space stations (ISS) [27–29]. Low manufacturing cost, easy mass production and faster building times makes SmallSats a testbed for new technologies in mainstream space science and education.

The research and development of SmallSats was more pronounced in early 2000s, researchers and industries considered NanoSats (CubeSats) more attractive and has become famous due to the developed standard by California Polytechnic State University and Stanford University in 2001 [24, 28]. Over a thousand SmallSats were developed/launched by Universities and non-US space agencies from 1998 to 2018 (Fig. 2.3), with rapid increase of design and development over 2000 expected to be deployed/launch by 2023 [23].

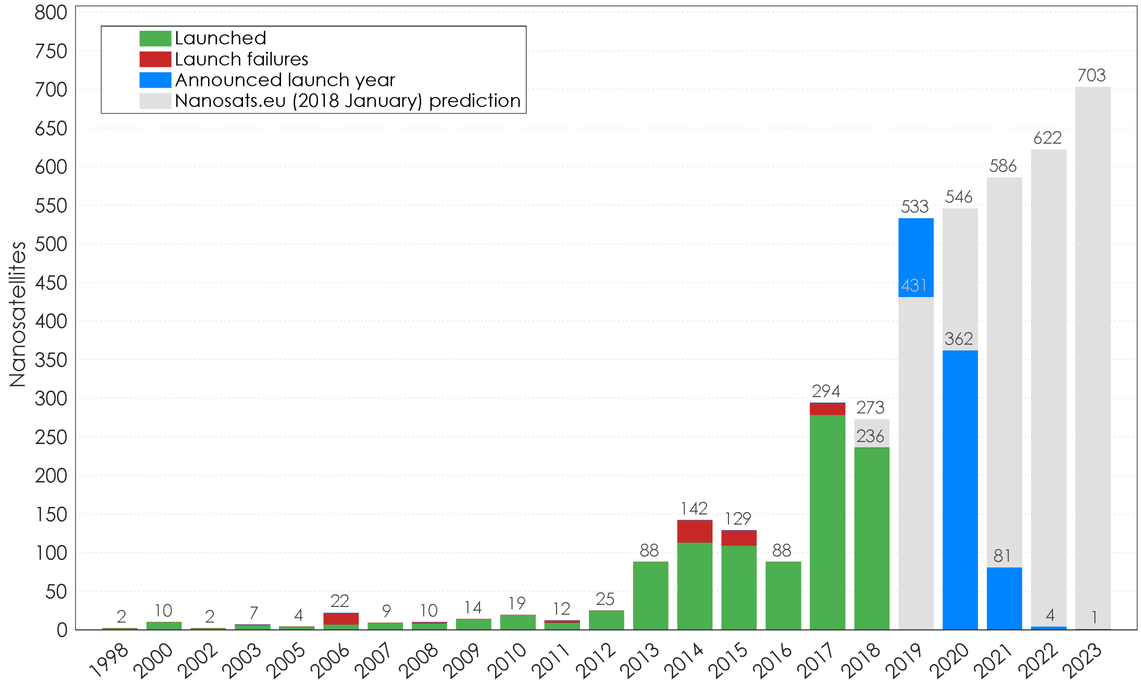


Figure 2.3: SmallSats Development and Launch 1998-2023 (adapted from [30])

SmallSats missions are typically launched into Low Earth Orbit (LEO) [31, 32], operating with low power and frequency bands. There is a growing interest in small spacecraft for missions beyond LEO and at higher frequency bands [25, 26]. The key challenges of SmallSats are limited payload, operation in LEO, high failure rate (historically 50%), short lifetime and low performance due to size and power constrains [26, 33]. These limit the functionality of SmallSats and footprint coverage, which can only be extended by using satellites constellation.

The SmallGEO platforms initiated by ESA will use smaller (reduced weight) satellites on GEO with capabilities and functions of traditional GEO satellites while using some features of SmallSats such as low power and weight [34]. This will deliver wider coverage and functionality at Ku/Ka-band beyond that of SmallSats. The first flight mission of this platform is Hispasat 36W-1, launched in first quarter of 2017 [34]. Launched into GEO with launch weight of 3200 kg, chemical/electrical propulsion, up to 3 kW payload power and 15 years lifetime [34].

The SmallGEO mission is to deliver a telecommunications satellite platform capable of accommodating multiple band commercial payloads and missions such as TV broadcasting to multimedia applications, Internet access and mobile/fixed services in Ku/Ka-Bands. This was the first telecommunications satellite to use the SmallGEO platform, part of ESA Advanced Research in Telecommunication Systems (ARTES) program and will cover Europe, the Canary Islands and South America with faster multimedia services, better signal quality and flexible land coverage [34].

SmallSats is characterised by small size, lower development cost and time, low energy consumption and limited on-board transponders. These characteristics limit the SmallSats lifespan, functionality and applications [35]. New ways to improve the functionality and applicability are evolving from the research and industry such as exploring the new concepts of Small satellite in geostationary orbit (SmallGEO) at higher frequency bands (Ku/Ka-bands) and using satellites constellation in the form of network of satellites with Inter Satellite Link (ISL) for wider coverage and performance. Recent research efforts to improve the modulation scheme using software define radio (SDR) for small satellites in next generation 5G systems is also proposed in [31]. Further research efforts are required to bridge the gaps and open more potential benefits and competitive rapid development in SatComs and related technologies.

2.2.3 Satellite and Next Generation of New Radio Networks

Communications systems and technologies experience dramatic changes in recent years with sophisticated features which continue to evolve by exploring higher bandwidth, increased capacity, ubiquity with access coverage everywhere and availability every time. Substantial growth in demand for more data and coverage by different applications and services posed a challenge to communications researchers and the industry. The report by ESA COSAT project [36], predicted the explosion in data traffic in which the volume of digital video traffic grow by over x4 in 2018 (see Fig 2.4) with mobile video traffic contributing 69% followed by 12% web and 11% audio traffics [36, 37].

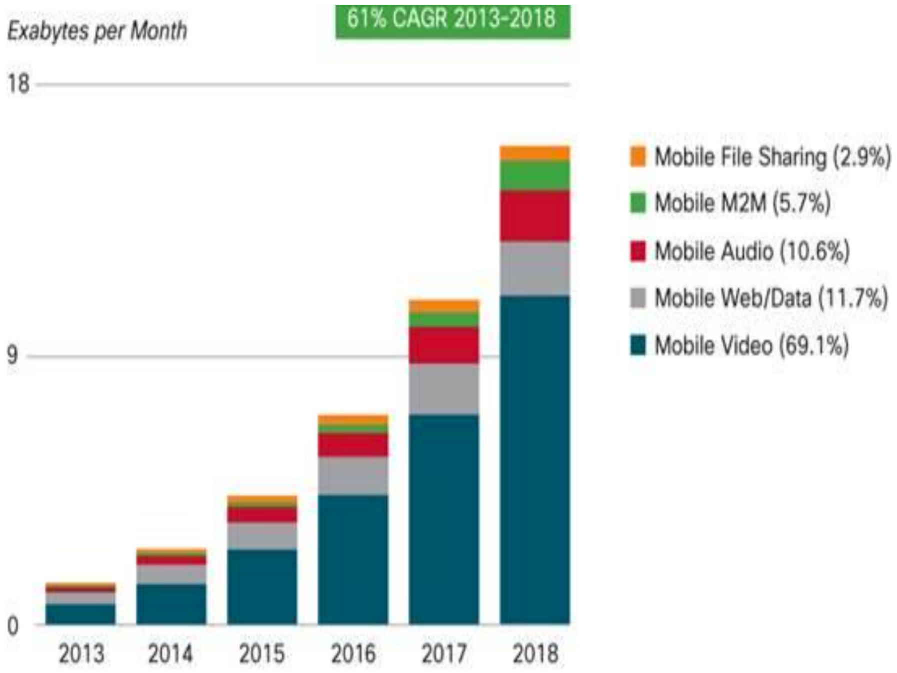


Figure 2.4: Data Traffic Growth Evolution 2013-2018 (adapted from [36])

The growth of global mobile data traffic is expected to reach 50-petabytes (10^{15}) per month by 2021 [38], therefore research to improve system capacity is ongoing to cope with the projected increase in data traffic and users demands. However, using terrestrial networks alone for Enhanced Mobile Broadband (eMBB), Massive Machine Type Communications (MMTC), and Ultra-Reliable and Low Latency Communications (URLLC) usage scenarios of the future networks as shown in Fig. 2.5 would be very expensive and unrealistic, thus more research is required to explore the potential benefits of SatComs systems falling costs to complement the terrestrial networks of the future.

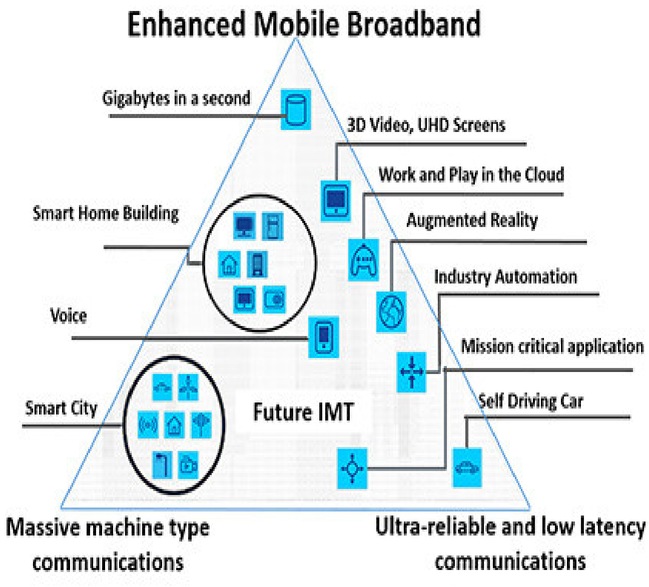

Enhanced Mobile Broadband
Gigabytes in a second
3D Video, UHD Screens
Work and Play in the Cloud
Smart Home Building
Augmented Reality
Industry Automation
Voice
Mission critical application
Smart City
Future IMT
Self Driving Car
Massive machine type communications
Ultra-reliable and low latency communications

Figure 2.5: NR (5G) IMT Usage Scenarios for 2020 & Beyond (adapted from [41])

The key advantages to explore in hybrid integration of Satellite and Terrestrial (ISTN) for supporting the key requirements and capabilities of the future communication networks shown in Fig. 2.6 includes global coverage, spectrum efficiency, enhance capacity, backhaul link, and resilience.

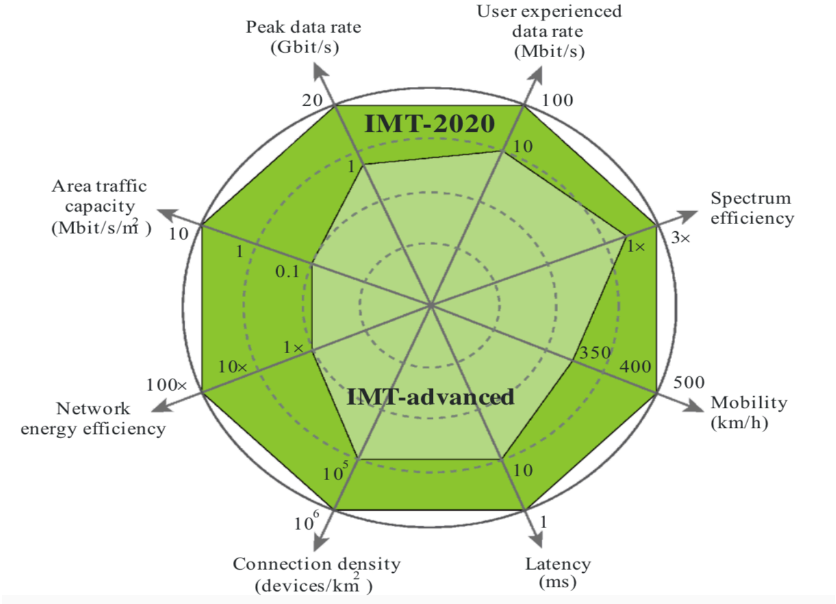


Figure 2.6: Key Capabilities Enhancements from IMT-Advanced (4/4.5G) to IMT-2020 (adapted from [41])

Next generation communications networks known as New Radio (NR) or Fifth Generation (5G) networks are more than just a cellular mobile, but instead a wireless network, combining new technologies and capabilities, working side-by-side with advanced existing technologies. For several, if not all of the 5G objectives, SatComs will play a vital role in 5G networks as shown in Fig. 2.7 with its unique features. Satellite and Terrestrial networks for 5G (SaT5G) is one of the European Commission Horizon 2020 (EC H2020) through its 5G Public Private Partnership (5G PPP) phase 2 project initiated to explicitly address the hybrid integration of SatComs and 5G systems [39].

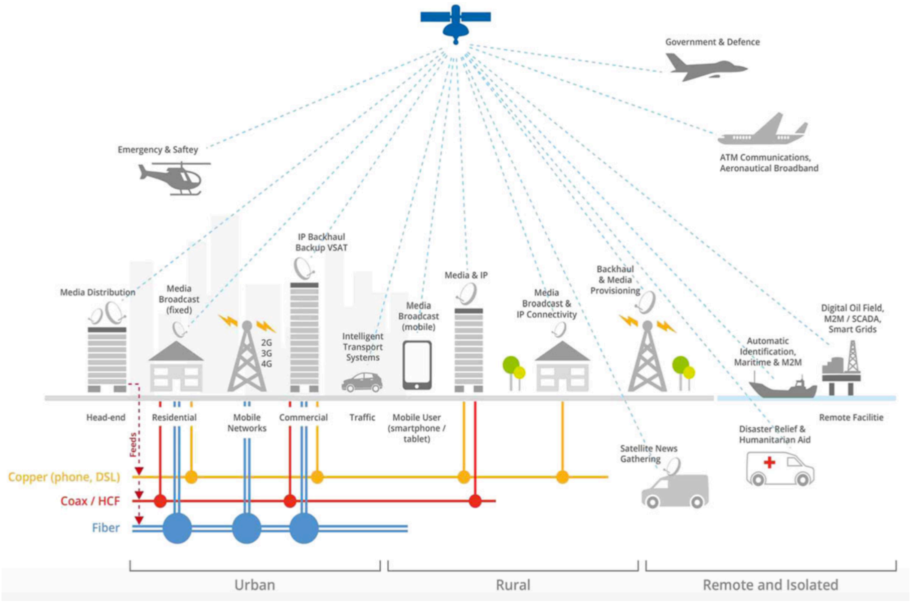


Figure 2.7: SatCom Roles and Use cases Scenarios in 5G Networks [39]

The vision and ambition of SaT5G phase 2 project initiative is to develop cost effective "plug and play" SatCom solutions for 5G that will enable service providers and operators to speed up 5G deployment everywhere (see Fig. 2.7), make SatCom technologies ready for integration in 5G and create new and growing market opportunities for the SatCom industry with a primary focus on backhaul via satellite [39]. The different use cases are: edge delivery and media content offload, 5G fixed backhaul, 5G premises, and 5G moving platform backhaul, more details of several architecture options can be found in [40].

Nowadays, SatComs delivers mobile backhaul, converged media, linear/non-linear TV broadcast, broadband and Machine-to-Machine (M2M) services that will become part of the 5G ecosystem. SatComs can provide indispensable redundancy to terrestrial networks and improve resilience of the overall 5G networks in a hyper-connected world of Internet of Everything (IoE), Smart Cities and Transport Systems of the future using high bandwidth/capacity satellite bands to deliver current and planned next generation 5G services. The key features of the 5G eco-system are *Resilience* and *Ubiquity*, which are known characteristics of SatComs. Therefore, to achieve 5G vision of *"Internet Everywhere"*, at the heart of 5G services deployment that will enable connecting *everyone* and *everything* anywhere/anytime, 5G eco-system needs satellite connectivity using over 100 GEO and non-GEO HTS by 2020 to contribute Terabits of data connectivity worldwide [42, 43].

Researchers and Industry identified challenges of current communication networks required to be addressed effectively by future 5G networks; are higher capacity (x1000), higher data rate (x10-100 or 1-10 Gbps of throughput), lower end-to-end latency (1-5 ms), massive device connectivity (x10-100), reduced cost (sustainable), increase availability (99.999%), coverage (100%), efficient energy consumption (90% reduction) and consistent Quality of Experience (QoE) [44, 45].

A critical study of these challenges leads to conclusions that; SatComs must be part of 5G networks to meet the availability and coverage requirements cost-affordably [6]. This will provide connectivity to difficult areas not covered by terrestrial infrastructure, as a cost-effective backup connection, multimedia distribution (Content Delivery Networks) and backhauling. Part of ARTES 1 studies conducted recently by the European Space Agency (ESA) [7, 36], identified key roles of satellites in the future 5G networks. The study optimised the roles in the support of terrestrial networks and maximises the likelihood of take-up, through identifying synergies where satellite could enhance terrestrial networks. This also analysed and forecasted the development of terrestrial mobile market, outlined lessons from past satellite initiatives, developed four scenarios of increasing harmonisation and convergence for analysis, calculated the coverage and examined the costs of adding satellite Complementary Ground Components (CGC) to existing terrestrial infrastructure [7]. Four scenarios proposed by ESA ARTES 1 reported to successfully achieve ISTNs objectives are; 1) Terrestrial only access complemented by satellite based backhaul 2) Hybrid terrestrial and satellite optimised access 3) Partial harmonised terrestrial and satellite access 4) Converged terrestrial and satellite access [7].

A key enabling technology trend in the next generation networks (ISTN and 5G) is software define networks (SDN) that reference the service delivery architecture that translates into selected satellite-terrestrial integration scenarios. Lack of prevalent standards exists today and much functionality is mainly deployed on vendor-specific network appliances, which execute specific functions in satellite ground segment network architectures. Research in SDNs and Content Defined Networks (CDNs) is required to ensure the north-bound (uplink) interface of satellite network management system conforms to future 5G standards. This will also bridge the developmental lag between satellite and terrestrial networks [6, 46]. SDNs technology consistency with the trends and developments in terrestrial networks should allow overcoming several existing limitations in operational flexibility, evolvability, interoperability and end-to-end QoS over ISTNs to form the key architecture of heterogeneous ISTN [1, 6].

Resilient Backhaul will provide 5G connection to remote Radio Area Networks (RANs) in special and emergency situations due to human or natural disasters, improve availability and bandwidth at peak demand using high capacity and low cost future GEO and non-GEO HTS [1, 42, 43, 46].

Finally, the cost of communications coverage using only 5G terrestrial infrastructure might become unbearable with increasing capacity needs, particularly for isolated rural, remote, and even urban areas. SatCom will be key player in 5G as a complementary solution for ubiquitous coverage, broadcast/multicast provision, and emergency/disaster recovery as shown in Fig. 2.7 [47, 48]. Satellites will have unique opportunities for providing 5G services in rural areas, support MMTC and eMBB that will enable new applications such as smart agriculture, smart grid, telemedicine (e-health) environmental protection, and transportation. These will be achieved and supported using more than 100 GEO HTS and mega-constellations of LEO satellites systems (see Fig. 2.8) delivering ubiquitous connectivity of terabits per second (Tbps) capacity accessible anytime by 2020-2025 [47]. The evolving SatCom systems, satellite RANs integrated into the 5G system and other terrestrial wireless technologies will provide seamless connectivity and simultaneous radio access technologies among heterogeneous network environment to improve availability, capacity and reliability [47].

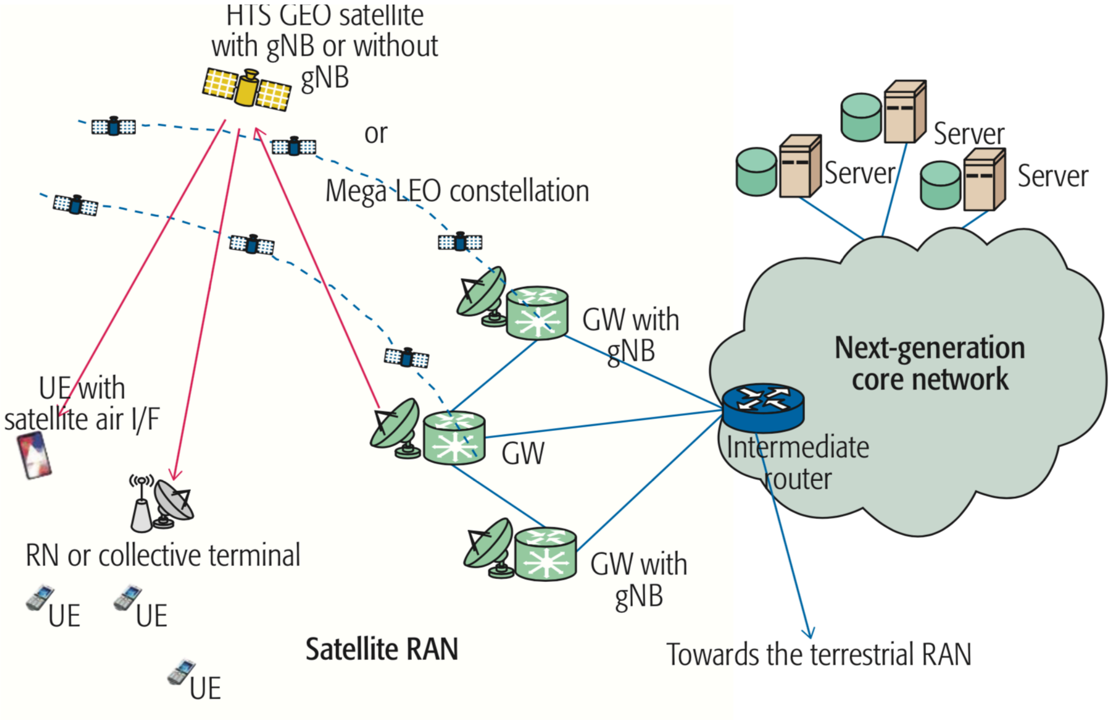

HTS GEO satellite with gNB or without gNB
or
Mega LEO constellation
Server
Server
Server
GW with gNB
Next-generation core network
UE with satellite air I/F
GW
Intermediate router
RN or collective terminal
GW with gNB
UE
UE
Satellite RAN
Towards the terrestrial RAN
UE

Figure 2.8: SatCom RAN Architecture using eMBB Scenario (adapted from [47])

2.3 Potential Features for Remote Rural Areas

SatComs applications and services span almost all aspects of human socio-economic development through distinctive features like global coverage, high bandwidth/capacity, broadcast ability, multiple access capability, availability and resilience even in the remotest rural areas of the world excluding the south and north poles of the globe. These can be exploited to provide global digital connectivity capabilities in the form of internet access, high-quality (HD, Ultra HD, 4K etc.) entertainment, earth observations/mapping, Smart Grid data communications, and emergency response communications in difficult terrain or remote rural areas [9].

SatComs bridges the digital communication gap using over 1450 satellites orbiting the earth as of 2017 [49], that cannot be met by the terrestrial communications infrastructure due to terrain difficulties or harsh environment that makes it impossible or not economically feasible to deploy and maintain such as optical fibre and public land mobile infrastructure.

2.3.1 Smart Grid Communications Networks

The current electric grid is being transformed into a more dynamic, resilient and adaptable Smart Grid (SG) of the evolving future electric utilities [50]. The challenge of transforming the electric utilities and units involved in the *generation*, *transmission*, *distribution* and *consumption* of electricity to a *smarter* grid is enormous and requires understanding of evolving communications technology requirements and development due to inherent complexity [51]. Incorporating advanced communications and network technologies in the future electric power systems to form Smart Grid Communications (SGC) is the key to achieving *smarter* grid networks that will make the information and data exchange reliable, secure, and sustainable. In addition, ubiquitous connectivity is vital for connecting intelligent electronics in both urban and remote rural locations. Stable duplex communications of data, control and monitoring instructions to utilities central control centres of SGC networks are required. This will revolutionise electricity generation, delivery and use by integrating two-way flow of both electricity and information capable of monitoring and responding to changes from power generation plants to consumer preferences and appliances with potential benefits of increase reliability and energy efficiency, better harnessing/integrating renewable energy, reduces CO_2 emissions, better consumer control to electricity demand/usage and cost-effective energy system from generation to consumption [50–53]. Therefore, the goal of SGC is to allow utility companies to generate and distribute energy efficiently and also allow consumers to optimise energy consumption will only be enabled by SGC networks and technologies, which rely on successful design and implementation of reliable, secure, robust, and cost-effective communications infrastructure.

This is challenging since SGC need integration of different network elements for maintaining the communications among expansive number of homogeneous industrial devices scattered over large geographical locations (see Fig. 2.9) and diverse applications needing highest availability of data communications with different QoS constraints [52, 53].

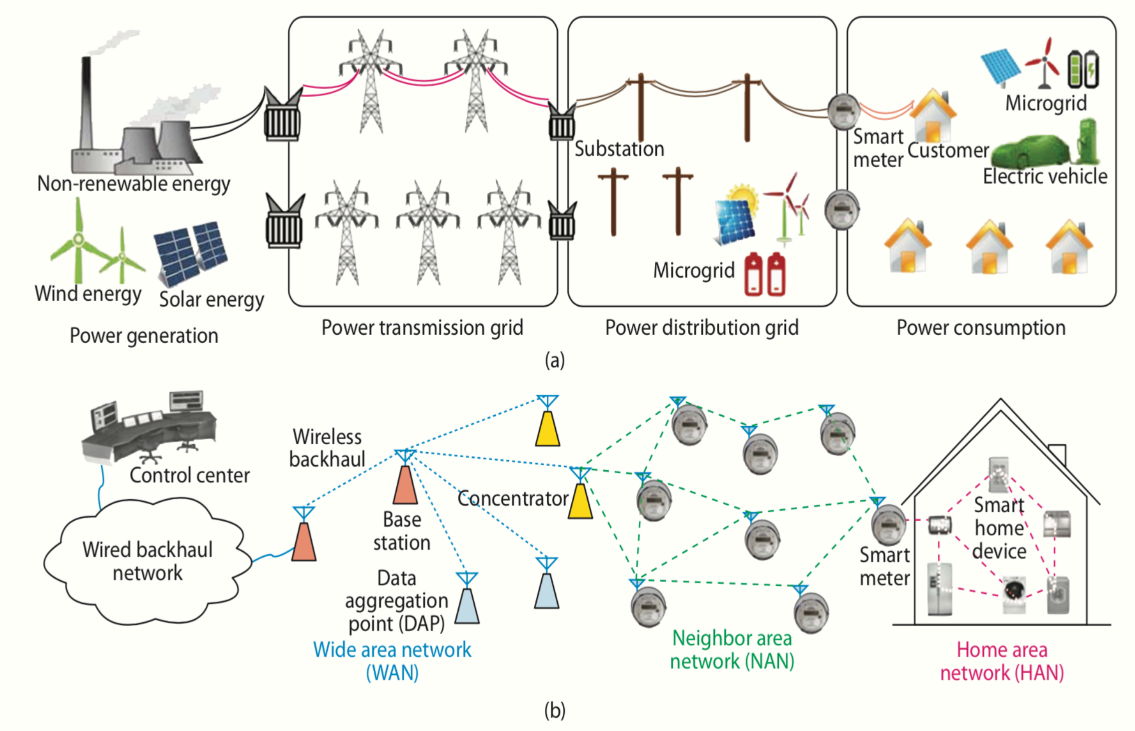


Figure 2.9: Layered Architecture of SG and SGC (a) Power System Hierarchy (b) Communications System Hierarchy (adapted from [52])

The typical layered architecture of SG networks, consist of two layers; (1) the power system layer (Fig. 2.9a) that integrates different power generation, transmission grid, distribution grid, substations and customers and (2) the communication system layer (Fig. 2.9b). This is the unique feature that formed the SGC networks responsible for intelligent monitoring, controlling and automating the grid compared to the traditional electrical grid, [52]. The focus here is on SGC networks, the communication system layer represented in three segments, namely Home Area Network (HAN), Neighbour Area Network (NAN) and Wide Area Network (WAN) as shown in Fig. 2.9b, which can only be realised by integrated technologies such as SatComs, wireless, wired and optical communications systems technologies [52, 54, 55].

SatComs among other terrestrial communication technologies like WiMAX, ZigBee, WiFi, cellular and optical fibre networks will play an indispensable role in achieving the SGC networks of the future. SatComs, in particular, has unique features of high capacity, global (ubiquitous) coverage, availability, and resilience. These will provide high bandwidth, high data rate and extended coverage using its ubiquitous connectivity to remote isolated substations and customer locations, which are beyond the reach of terrestrial networks to help ensure 100% network availability. Utilities in SG applications require high availability communications that are cost-effective and highly redundant connections at critical sites where terrestrial communications might not cope or be impossible due to severe man made disruptions or damage on both fixed and wireless infrastructure due to natural disaster, SatCom could be the preferred candidate in these cases [53]. SatCom providers already provide more services exclusive to Machine-to-Machine (M2M) communications that are essential to core SG applications/ (unctionalities) such as Substation Automation (SA) with Supervisory Control and Data Acquisition (SCADA), telemetry, Advanced Metering Infrastructure (AMI) backhaul and Distribution Automation (DA) among others, which covers Generation, Transmission and Distribution (T&D), and Distributed Energy Resources (DER) domains [50, 56] as shown in Fig 2.10 and 2.11.

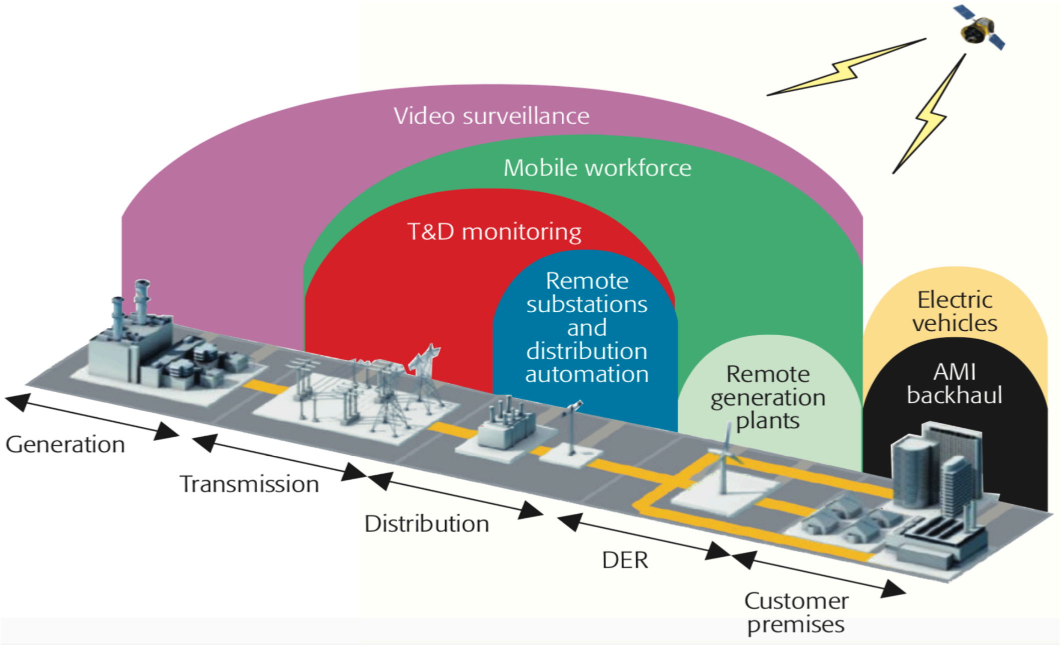
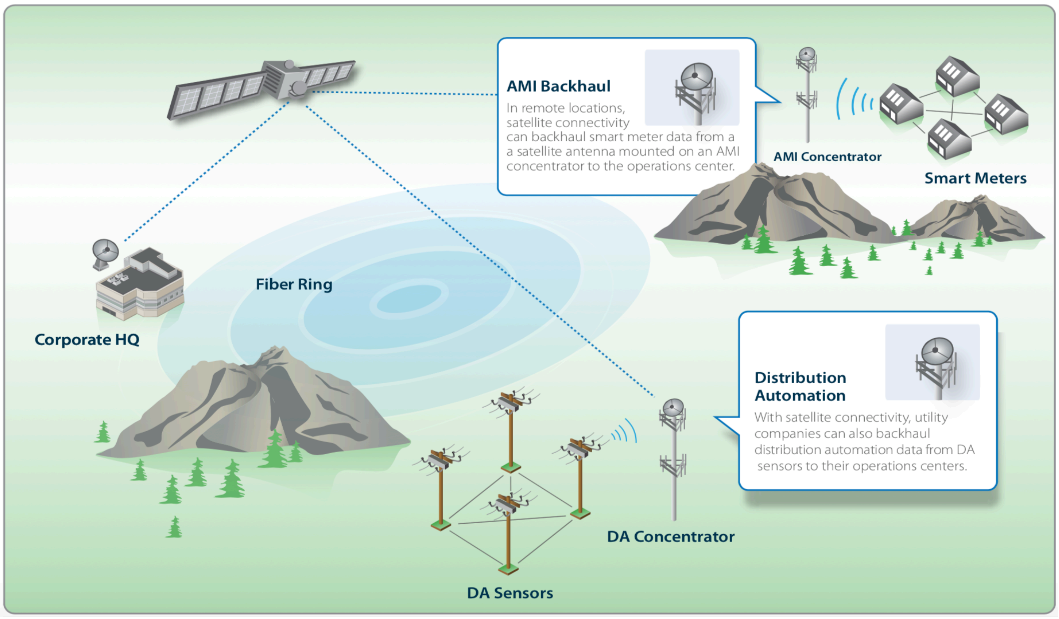


Figure 2.10: Smart Grid Applications and Domain for SatCom (adapted from [56])



In remote locations, satellite connectivity can backhaul smart meter data from a a satellite antenna mounted on an AMI concentrator to the operations center.

With satellite connectivity, utility companies can also backhaul distribution automation data from DA sensors to their operations centers.

Figure 2.11: SatCom Support for SG (adapted from [54])

The key SG applications including AMI/Smart Meter (SM), Distribution Grid Management (DGM), Demand Response (DR), DER and storage, Wide-Area Situational Awareness (WASA), electric vehicles/transportation, and Distribution Grid

Management (DGM) have different levels of communications QoS requirements (low/loose (L), medium (M), and high/tight (H/T)) in terms of Bandwidth, Latency/Jitter (uses loose and tight levels), Reliability, Security, Cost and Backup Power network parameters as given in the Table 2.1 and Fig. 2.12 [50, 56].

Table 2.1: SG Applications and Communications QoS Requirements

	Key Applications and QoS Requirement					
Net Parameter	**AMI/SM**	**DR**	**DERS**	**EV**	**WASA**	**DGM**
Bandwidth (kbps)	10-500	14-100	9-56	9-100	600-1500	9-100
Latency	2-15 s	$\geq 500\ ms$	20 ms-15s	2s-5 min	20-200 ms	100 ms-2 s
Reliability(%)	99-99.99	99-99.99	99-99.99	99-99.99	$\geq$ 99.999	99-99.99
Security	H	H	H	Rel. H	H	H
Cost	L					M-H
Redundancy(hrs)	NN	NN	1	NN	24	24-72

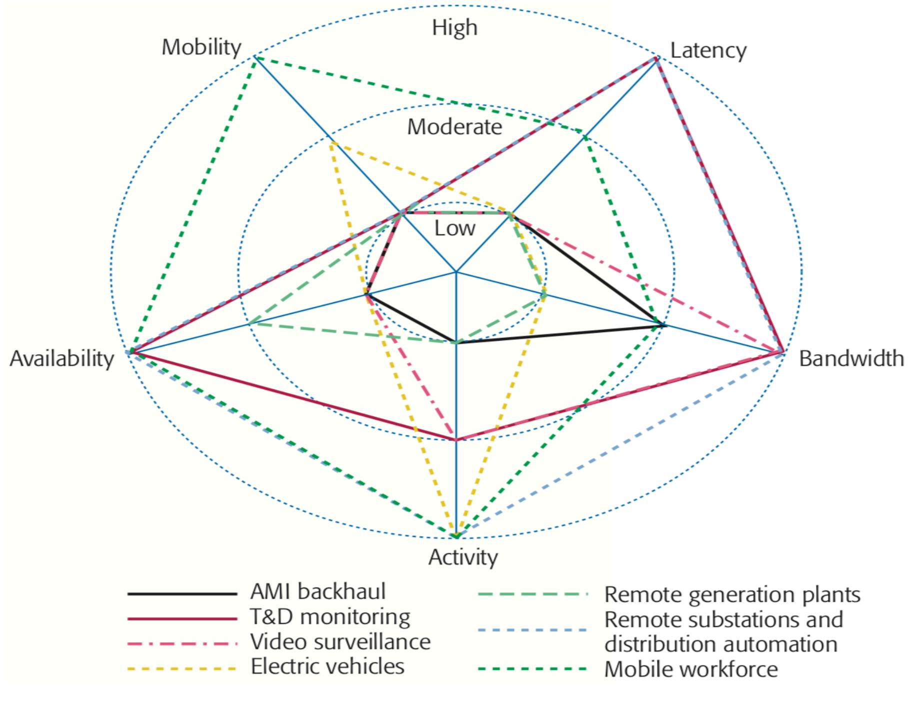


Figure 2.12: QoS Requirements Map for Key SG Applications (adapted from [56])

The DGM consisted of sub-applications like DA, SA, Video Surveillance (VS) and fleet management by Automatic Vehicle Location (AVL). Levels (L, M and H/T) of communications QoS requirement for SG applications can be described as below and map as shown in Fig. 2.12 [56].

1. **Bandwidth:** L is 10 kbps to 250 kbps, M is 250 kbps - 1 Mbps and H is $\geq$ 1 Mbps

2. **Latency/Jitter:** Loose latency and high jitter tolerant applications used L, application's operation impacted, but unlikely loss of service during connectivity lost for significant time (minutes to few hours) such as those assigned M, there is some limits to E2E latency, and T/H means strict requirements for amount of E2E latency and significant harm is likely during connectivity lost for significant time.

3. **Reliability:** L is no significant operational harm would results if connectivity lost for significant time, M operations impacted, unlikely to results in loss of service if connectivity is lost for significant time, and H means significant harm is likely if connectivity is lost for significant time.

4. **Security:** L means no significant operational harm if link were compromised by spoofing or data intercepted, M significant, but limited harm were link is compromised, and H means highly visible and widespread harm when link were compromised.

5. **Cost:** L is relatively low infrastructure and operational cost, M relatively moderate operational and infrastructure costs, and H means high operational cost and/or infrastructure.

Therefore, looking at the SG applications communications QoS requirement in Table 2.1, and descriptions above, SatCom will be an excellent candidate for realising the next generation of power grid using intelligent SG aim of achieving more efficient, reliable, secure, intelligent, and cost-effective power system from generation to consumption phases. SatCom has the capabilities to provide extensive coverage, fast deployment, high bandwidth and capacity, and high availability of up to 99.999% for reliable communications where terrestrial communications infrastructures are not economically viable, inadequate for the domain-specific requirements or completely not feasible [56].

2.3.2 Emergency and Disaster Response Communications

In the event of natural disaster, essential communications services and applications using terrestrial infrastructure breakdown and are interrupted due to damage and power supply disrupted in the terrestrial communications infrastructure. Significant communication outages occur in a wide area for hierarchical networks such as Public Land Mobile Networks (PLMN) infrastructure that become congested or completely damaged. A fast deployment of alternative communications and recovery is vital to any emergency and disaster management in order to provide the first response team, rescue team and command centres with the required network to coordinate relief efforts across wide areas. SatCom is an ideal candidate for emergency and disaster management that can provide communications capabilities for the wide area regardless of terrestrial communications network availability. The disaster recovery phase of any natural disaster required faster communications to support applications and services such as email, Internet access and voice communications.

For Instance, during the great *Tohoku Earthquake and Tsunami in Japan*, outages in communications using PLMN occurred in wide area although the Base Stations (BS) were not damaged, the core networks were congested due to the explosion in user traffic, PLMN operators experienced over fifty times (x50) more call attempts than normal, while many BS stopped functioning due to the power supply outage [2]. Satellite operators allocated 500MHz of additional bandwidth to disaster areas as a response to communication demand explosion for disaster management and recovery [2].

Features such as wider coverage and resilience made SatCom an ideal candidate and backbone to provide quick and effective communication relief in the event of natural disaster that caused damage to terrestrial communications network infrastructure [2]. Additionally, the extreme capacity (Gbps/Tbps system throughput) and multi-beam/steerable antenna ability of recently developed HTS can provide solutions to the huge traffic demands in the event of disaster and emergency response using resilient network access to public networks over satellite as a backbone [2, 9].

References

[1] R. Ferrus, H. Koumaras, O. Sallent, *et al.*, "On the Virtualization and Dynamic Orchestration of Satellite Communication Services", *In Proc. of IEEE 84th Vehicular Technology Conference (VTC)*, 2016.

[2] T. Shigenori, M. Katsuyuki, S. Hiroyasu, O. Atsushi, N. Hiroki, and K. Nei, "Flexibility-Enhanced HTS System for Disaster Management: Responding to Communication Demand Explosion in a Disaster," *IEEE Transactions on Emerging Topics in Computing*, 2017.

[3] K. Kaneko, H. Nishiyama, N. Kato, A. Miura, and M. Toyoshima, "Construction of a Flexibility Analysis Model for Flexible High-Throughput Satellite Communication Systems With a Digital Channelizer", *IEEE Transaction on Vehicular Technology*, Vol. 67 (3), pp. 2097-2107, 2018.

[4] Y. Abe, H. Tsuji, A. Miura, and S. Adachi, "Frequency Resource Allocation for Satellite Communications System Based on Model Predictive Control and Its Application to Frequency Bandwidth Allocation for Aircraft", *In Proc. IEEE Conf. on Control Technology and Applications*, pp. 165-170, Aug 2018.

[5] S. Watts and O. G. Aliu, "5G Resilient Backhaul Using Integrated Satellite Networks", *in Proc. IEEE Conf. on 7th Advanced Satellite Multimedia Systems and 13th Signal Processing for Space Communications (ASMS/SPSC)*, pp. 114-119, 2014.

[6] R. Ferrs et al., "SDN/NFV-Enabled Satellite Communications Networks: Opportunities, Scenarios and Challenges", *Elsevier Physical Communication*, Vol. 18, no. 2, pp. 95-112, 2016.

[7] S. Phil et al., "Study of Satellite-Terrestrial Integration," *in ARTES 1 of the European Space Agency (ESA)*, Vol. 1, no. UK4000102650/11/NL/NR, 2013.

[8] R. Dhaou, B. Escrig, B. Paillassa, and C. Bes‘, "Extending Satellite Service Availability Through Energy Efficient Cooperation", *in Proc. IEEE 25th International Symposium on Personal, Indoor and Mobile Radio Communications*, pp. 1758-1762, 2014.

[9] D. Minoli, *Innovations in Satellite Communication and Satellite Technology: The Industry Implications of DVB-S2X, High Throughput Satellites, Ultra HD, M2M, and IP . John Wiley & Sons, Inc.*, New Jersey, USA, 2015.

[10] A. Hall, and M. Rao "Future-Sat Africa... To Connect All ", *in Worldwide Satellite Magazine (SatMagazine) Nov. 2016*, pp. 44-47. Available: www.satnews.com/magazines.php.

[11] C. Niephaus, M. Kretschmer, and G. Ghinea, "QoS Provisioning in Converged Satellite and Terrestrial Networks: A Survey of the State-of-the-Art," *IEEE Communications Surveys and Tutorials*, Vol. 18, no. 4, pp. 2415-2441, 2016.

[12] X. Feng and R. Yang, "Research on the Management and Control Method of Small Satellite IUR Cooperative Innovation," *in Proc. IEEE 35th Chinese Control Conference (CCC)*, pp. 9799-9804, 2016.

[13] H. Fenech et al., "Future High Throughput Satellite Systems", *in Proc. IEEE 1st AESS European Conference on Satellite Telecommunications (ESTEL)*, pp. 1-7, 2012.

[14] H. Fenech, S. Amos, and A. Tomatis, "KA-SAT and Future HTS Systems", *in Proc. IEEE 14th International Vacuum Electronics Conference (IVEC)*, 2013.

[15] G. Giovanni, K. L. Doanh, A. L. Van, and D. C. Tomaso, "Gateway Handover Implications on Transport Layer Performance in Terabit Satellite Networks", *in Proc. IEEE 7th Advanced Satellite Multimedia Systems Conference and the 13th Signal Processing for Space Communications Workshop (ASMS/SPSC)*, pp. 9-16, 2014.

[16] O. Vidal, G. Verelst, J. Lacan, E. Alberty, J. Radzik, and M. Bousquet, "Next Generation High Throughput Satellite System", *in Proc. IEEE 1st AESS European Conference on Satellite Telecommunications (ESTEL)*, pp. 1-7, 2012.

[17] H. Fenech, S. Amos, A. Tomatis, and V. Soumpholphakdy, "High throughput satellite systems: An analytical approach," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 51, no. 1, pp. 192-202, 2015.

[18] K. T. Murata et al., "An Application of Novel Communications Protocol to High Throughput Satellites", *in Proc. IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pp. 1-7, 2016.

[19] M. Ruggieri, C. Riva, M. De Sanctis, and T. Rossi, "Alphasat TDP#5 Mission: Towards Future EHF Satellite Communications", *in Proc. IEEE 1st AESS European Conference on Satellite Telecommunications (ESTEL)*, pp. 1-6, 2012.

[20] P. Harati et al., "E-Band Downlink Wireless Data Transmission for Future Satellite Communication", *in Proc. IEEE Topical Workshop on Internet of Space (TWIOS)*, pp. 1-4, 2017.

[21] C. E. Shannon, "A Mathematical Theory of Communication", *The Bell System Technical Journal*, Vol. 27, no. 4, pp. 623 - 656, 1948.

[22] J. Berk, J. Straub, and D. Whalen, "The Open Prototype for Educational NanoSats: Fixing the Other Side of the Small Satellite Cost Equation", *in Proc. IEEE Aerospace Conference*, pp. 1-16, 2013.

[23] K. M. Anil and A. Varsha, *Introduction to Satellites and their Applications, in Satellite Technology: Principles and Applications*, 3rd ed. pp. 3-35, John Wiley & Sons, 2014, .

[24] R. N. Simons and K. Goverdhanam, "Applications of Nano-Satellites and Cube-Satellites in Microwave and RF Domain," *in Proc. IEEE MTT-S International Microwave Symposium*, 2015.

[25] R. Radhakrishnan, et al. "Survey of Inter-Satellite Communication for Small Satellite Systems: Physical Layer to Network Layer View", *IEEE Communications Surveys & Tutorials*, Vol. 18, no. 4, pp. 2442-2473, 2016.

[26] Y. Rahmat-Samii, V. Manohar, and J. M. Kovitz, "For Satellites, Think Small, Dream Big: A Review of Recent Antenna Developments for CubeSats", *IEEE Antennas and Propagation Magazine*, Vol. 59, no. 2, pp. 22-30, 2017.

[27] C. K. Pang, A. Kumar, C. H. Goh, and C. V. Le, "Nano-satellite Swarm for SAR Applications: Design and Robust Scheduling", *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 51, no. 2, pp. 853-865, 2015.

[28] R. Rose, J. Dickinson, and A. Ridley, "CubeSats to NanoSats; Bridging the Gap between Educational Tools and Science Workhorses", *in Proc. IEEE Aerospace Conference*, 2012.

[29] Y. Ma, X. Zou, and F. Weng, "Potential Applications of Small Satellite Microwave Observations for Monitoring and Predicting Global Fast-Evolving Weathers", *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, pp. 1-11, 2017.

[30] E. Kulu, "Nanosatellite & Cubesat Database", *NanoSats Database*, accessed via www.nanosats.eu/index.html#info on 06/01/2019.

[31] R. K. Miranda et al., "Implementation of Improved Software Defined Radio Modulation Scheme and Command and Telemetry Software Interface for Small Satellites in 5G Systems", *in Proc. IEEE 19th International Conference on OFDM and Frequency Domain Techniques (ICOF)*, 2016.

[32] B. Lim, "SMALL SATELLITE DEVELOPMENTS", *in Proc.IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, 2016.

[33] W. Su, J. Lin, and T. Ha, "Global Communication Coverage Using Cubesats", *in Proc. IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*, 2017.

[34] ESA, ”Smallgeo: SmalGeostationary Platform”, *European Space Agency ARTES Programme*, available at www.esamultimedia.esa.int/docs/telecom/smallGEO_factsheet_sp_web.pdf, accessed on January 4, 2019.

[35] D. Zhou, et al. ”Mission Aware Contact Plan Design in Resource-Limited Small Satellite Networks”, *IEEE Transactions on Communications*, Vol. 65, no. 6, pp. 2451-2466, 2017.

[36] W. Zia et al., ”The Role of Satellite in Converged Mobile/Fixed/Broadcasting Environments”, *in European Space Agency ARTES 1*, Vol. 1, no. 4000104993/11/NL/CLP, 2014.

[37] K. *Ádám*, C. Stefan, V. Christopher, S. Vasilios, and G. Maria, ”Advanced Topics in Service Delivery over Integrated Satellite Terrestrial Networks”, *in Proc. IEEE 7th Advanced Satellite Multimedia Systems Conference and the 13th Signal Processing for Space Communications Workshop (ASMS/SPSC)*, 2014.

[38] Z. Lin et al. ”Robust Secure Beamforming for 5G Cellular Networks Coexisting with Satellite Networks”,*IEEE Journal on Selected Areas in Communications*, Vol. 36, No. 4, pp. 932-945, 2018.

[39] K. Liolis, et al ”Use Cases and Scenarios of 5G Integrated Satellite-Terrestrial Networks for Enhanced Mobile Broadband: The Sat5G Approach”,*International Journal of Satellite Communication Network*, pp. 1-22, John Wiley, 2018.

[40] B. T. Jou, et al, ”Architecture Options for Satellite Integration into 5G Networks”, *in Proc. European Conference on Networks and Communications (EuCNC)*, pp. 398-402, IEEE, 2018.

[41] ITU, ”ITU-R. IMT Vision -Framework and Overall Objectives of the Future Deployment of IMT for 2020 and Beyond: Mobile, Radiodetermination, Amateur

and Related Satellite Services", *Radiocommunication Sector of ITU (ITU-R)*, ITU-R Rec. M.2083-0, 2015. Available: www.itu.int/ dms_pubrec/itu-r/rec/m/R-REC-M.2083-0-201509-!!PDF-E.pdf.

[42] ESOA, "Satellite: Bringing Resilience & Ubiquity to the 5G Ecosystem", *EMEA Satellite Operators Association*, Accessed via www.esoa.net/Resources/NEW-2016-5G-infographic-FINAL.pdf, on 19/11/2016.

[43] ESOA, "Satellite and 5G: Ubiquity, Resilience and Reliability", *EMEA Satellite Operators Association*, Accessed via https://esoa.net/cms-data/positions/Satellite_5G_Spectrum_052017.pdf, on 06/09/2019.

[44] X. Artiga et al., "Terrestrial-Satellite Integration in Dynamic 5G Backhaul Networks", *in Proc. IEEE 8th Advanced Satellite Multimedia Systems Conference and the 14th Signal Processing for Space Communications Workshop (ASMS/SPSC)*, 2016.

[45] A. Gupta and R. K. Jha, "A Survey of 5G Network: Architecture and Emerging Technologies",*IEEE Access*, Vol. 3, pp. 1206-1232, 2015.

[46] W. Simon and G. A. Osianoh, "5G Resilient Backhaul Using Integrated Satellite Networks", *in Proc. IEEE 7th Advanced Satellite Multimedia Systems Conference and the 13th Signal Processing for Space Communications Workshop (ASMS/SPSC)*, pp. 114-119, 2014.

[47] G. Giambene, S. Kota, and P. Pillai, "Satellite-5G Integration: A Network Perspective", *IEEE Network*, pp. 25-31, 2018

[48] 3GPP, "Study on Using Satellite Access in 5G", *3GPP Technical Report (TR)*, 22.822 V16, Stage 1, Rel. 16, 2018.

[49] SIA, "State of the Satellite Industry Report", 20th Ed., *Satellite Industry Association*, accessed via www.sia.org/wp-content/uploads/2017/10/SIA-SSIR-2017-full-2017-10-05-update.pdf, 09/01/2019.

[50] USA DOE, "Communications Requirements of Smart Grid Technologies", *Report by Department of Energy, USA*, October, 5, 2010.

[51] Z. Fan, *et al.*, "Smart Grid Communications: Overview of Research Challenges, Solutions and Standardization Activities", *IEEE Communications Surveys & Tutorials*, Vol. 15, no. 1, pp. 21-38, 2013.

[52] Q. Ho, Y. Gao, and T. Le-NGoc, "Challenges and Research Opportunities in Wireless Communication Networks for Smart Grid", *IEEE Wireless Communications*, pp. 89-95, 2013.

[53] M. De Sanctis,*et al.*, "Satellite Communications Supporting Internet of Remote Things", *IEEE Internet of Things*, Vol. 3, no. 1, pp. 113-123, 2016.

[54] B. Gohn, and C. Wheelock "Smart Grid Network Technologies and Role of Satellite Communications", *Pike Research Report*, pp. 1-26, PikeResearch LLC, 2010. accessed via https://vdocuments.mxsection-1-idirect-mediafileswhite-paperspike-researchsmart-gridsmart.html

[55] K. Wang,*et al.*, "A Survey on Energy Internet Communications for Sustainability", *IEEE Transactions on Sustainable Computing*, Vol. 2, no. 3, pp. 231-254, 2017.

[56] A. Meloni, and L. Atzori, "The Role of Satellite Communications in the Smart Grid", *IEEE Wireless Communications*, pp. 50-56, 2017.

Chapter 3

Transport Protocols Over Satellite Channels

This chapter presents an overview of transport protocols including Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) for data communications over the Internet Protocol (IP), with focus on TCP due to its wider use over the Internet. The chapter discusses the standard TCP algorithms and other modified schemes that aimed to address performance issues for different network environments, in particular, GEO satellite links as part of a more realistic heterogeneous network environment.

3.1 Transmission Control and Internet Protocols

The Transmission Control Protocol and Internet Protocol (TCP/IP) suite is one of the network standards and technologies for the global internet that was developed and funded under the Defence Advanced Research Projects Agency (DARPA) within the Department of Defence (DoD) of the US government in the early years of the Internet, which help to specify how computers communicate, set conventions for interconnecting networks and forwarding traffic [1–3]. As the Internet evolve and grow with new unique technologies, services and applications, the protocols also needed to be enhanced to accommodate new changes for efficient and effective performance of data communications over high capacity and long distance networks of today and the future.

Protocols such as TCP/IP provide the syntactic and semantic rules for communication and form the basis for the internet data communications by over $80-90\%$ of 4 billion connected machines and things today. The standard TCP carries the details of the message formats, how machines respond to message reception in the form of acknowledgement to ensure reliability, and specifies how errors, link congestions and other abnormalities are handled. TCP allows computer communication to be discussed independently of a vendor's network hardware, while the IP was designed for interconnected systems of packet-switched communication networks that transmit blocks of data known as datagrams or packets from sources (Tx) to destinations (Rx) hosts of fixed length IP addresses, and if necessary, fragment and reassemble long datagrams for transmission via small packet networks [1, 2, 4].

3.1.1 Standard TCP Scheme

The original TCP was designed based on [4], as a highly reliable E2E protocol to support information and resource sharing in different packet-switched networks. This provides packet sizes, transmission failures, sequencing, flow control, E2E error checking, and the creation/destruction of logical process-to-process connections [1, 4]. The high E2E reliability of TCP enhances robustness in the presence of unreliable data communication and improves the availability in the presence of congestion in packet-switched communication networks and interconnected systems of these networks [1]. This connection-oriented protocol is intended to fit into layered hierarchical protocols that support many host-to-host (E2E) packet switching network applications reliably. The standard TCP is an acknowledgement (ACK) based protocol. This (ACK) determines the success or failure of data transmission, and strongly impacted by the time delay (latency) in accepting, delivering, and transporting the data. Therefore, there is a need for careful development of timing procedures in order to achieve successful data delivery with improved performance in different network environments.

Ideally, TCP should be able to operate over a wide spectrum of communication systems from hard-wired connections to packet-switched or circuit-switched networks and use basic IP to send and receive variable-length segments of data encapsulated in envelopes of Internet datagrams [1]. The IP datagram provides a way for addressing the TCP connection's source and destination, and the fragmentation/reassembly of segments needed to accomplish transport and delivery via different multiple networks and interconnected gateway systems. Security, precedence, and TCP segment compartmentalisation is encapsulated within the IP packet (datagram) for E2E communication of information over multiple networks [1, 2]. In the layering of the Open System Interconnection (OSI) model of the TCP/IP suite, TCP interfaces with high level user application (layer) processes and lower level IP layer.

The high reliability of TCP allows it to recover from lost, duplicated, damaged, and out of order data delivered over the Internet. This is accomplished by using a sequence number (used to correct segment ordering, to eliminate duplicates segments and for flow control) and ACK (use for data retransmission within the timeout interval) from the receiver while damaged data are found using a *checksum* on each transmitted segment. However, standard TCP assumed wired networks like Local Area Networks (LAN or Ethernet) or large ARPANET networks based on packet switching technology [1]. This assumption limits the performance of standard TCP in other network environments such as wireless and SatCom networks as will be discussed in the following sections.

3.1.2 Mathematical Models of the Algorithms

The key parameters in the implementation and evaluation of the performance of TCP are the *Congestion Window (CWND/CWND or simply W), window growth rate (evolution)* $W(t)$*, Initial CWND (IW), Slow Start Threshold (ssth or* γ*), time to reach the threshold value (*t_γ*), transmission elapsed time (t), and Round-Trip-Time (RTT).* These parameters are required to implement the TCP scheme algorithm and performance can be measured by evaluating parameters such as *Instantaneous Transmission Rate (*$R(t)$*), total segment (data) transmitted (*T_D*), and goodput (*G_p*).*

Establishing a TCP connection starts with the three-way handshake procedure, which is initialised by the TCP source and replied to by the destination node or initialised simultaneously by two TCP connections where each of the two TCP nodes receives a Synchronise (SYN) segment that carries no ACK. The arrival of an old duplicate SYN segment helps the receiver note that a simultaneous connection initiation is in progress. The algorithm is implemented in two phases; Slow Start (SS) and Congestion Avoidance (CA) using standard mathematical model of CWND growth rate function, W(t) given by (3.1.1), CWND update rule, W (3.1.2), Instantaneous transmission rate, R(t) (3.1.3) and total data segments transmitted, T_D (3.1.4). Equations (3.1.2) and (3.1.3) were derived from (3.1.1) while (3.1.4) was derived by integrating (3.1.3).

$$W^S(t) = \begin{cases} 2^{\frac{t}{RTT}} & 0 \le t < t_\gamma \quad SS\ (Exponent) \\ \frac{t-t_\gamma}{RTT} + \gamma & t \ge t_\gamma \quad CA\ (Linear) \end{cases} \tag{3.1.1}$$

where $t_\gamma = RTT\log_2(\gamma)$, is the time that the Slow Start Threshold (ssthresh) value γ is reached with RTT. The standard TCP window is updated according to the rule in (3.1.2) below.

$$W_{i+1}^S = \begin{cases} W_i^S + 1 & SS \\ W_i^S + \frac{1}{W_i^S} & CA \end{cases} \tag{3.1.2}$$

The TCP instantaneous transmission rate $R(t) = W(t)/RTT$ can now be derived from (3.1.1) and simplified in (3.1.3) below:

$$R^S(t) = \begin{cases} \frac{2^{t/RTT}}{RTT}; & 0 \le t < t_\gamma \quad SS \\ \frac{1}{RTT}(\frac{t-t_\gamma}{RTT} + \gamma); & t \ge t_\gamma \quad CA \end{cases} \tag{3.1.3}$$

Total data segments transmitted throughout the duration of the TCP flow, T_D is determined by $\int_0^1 R(\tau)d\tau$ and the simplified version given by (3.1.4).

$$T_D^S(t) = \begin{cases} \frac{2^{t/RTT}-1}{\ln(2)} & 0 \le t < t_\gamma \\ \frac{\gamma-1}{\ln(2)} + \frac{(t-t_\gamma)^2}{2RTT^2} + \frac{\gamma(t-t_\gamma)}{RTT} & t \ge t_\gamma \end{cases} \tag{3.1.4}$$

3.1.3 Retransmission Timer Computations Algorithm

The TCP sender employs a standard algorithm to compute and manage an important retransmission time to ensure data delivery in the absence of any feedback from the TCP remote receiver, as a mechanism for detection and recovering from loss.

The duration of this timer is known as Retransmission Timeout (RTO) [5–8]. The algorithm used by the TCP sender to calculate the *RTO* of an established and active connection is based on two state variables; the Smoothed RTT (SRTT) and RTT Variation (RTTVAR), in addition to the assumed variable called the Clock Granularity (G) in unit of seconds. The order and procedure of computing the *SRTT, RTTVAR* and *RTO* itself is as follows using the given equations [5–8].

1. Before any measurement of the actual *RTT* made for a segment sent between TCP source and destination, the TCP source sets the value of the initial $RTO_{Initial} = 1 - 3$ sec, TCP implementations may use any $RTO \geq 1$ sec, but any value of $RTO < 1$ sec is approximated to 1 sec.

2. After the first *RTT* measurement (RTT_i) is made the TCP sender should the set the other variables as follows. $SRTT_i = RTT_i$ and $RTTVAR_i = RTT_i/2$, which then used to compute the current RTO_i value as:

$$RTO_i = SRTT + max(G, KXRTTVAR); K = 4 \; and \; i = 1. \qquad (3.1.5)$$

3. Subsequently, when the RTT_{i+1} measurement is made, the sender must set the new RTO_{i+1} value strictly in the following order.

$$RTTVAR_{i+1} = (1 - \beta)X(RTTVAR_i + \beta)X|(SRTT_i - RTT_{i+1})| \qquad (3.1.6)$$

$$SRTT_{i+1} = (1 - \alpha)X(SRTT_i + \alpha)XRTT_{i+1} \qquad (3.1.7)$$

Where $\alpha = 1/8$ and $\beta = 1/4$ should be used in the above equations as in most implementations [8]. Therefore, more generalised and up to date form of *RTO* value can be derived from Eq. (3.1.5) above.

$$RTO_{i+1} = SRTT_{i+1} + max(G, KXRTTVAR_{i+1}) \qquad (3.1.8)$$

As mentioned earlier, any updated *RTO* value computed and found to be less than 1 sec should be approximated to 1 sec.

Conservatively, TCP implementations employ coarse grain clocks to measure the *RTT* and to activate the *RTO*. This imposes a large minimum amount on the *RTO*, which is required to keep TCP conservative and avoid bogus retransmissions, although this still needs further research [5–8]. Finer *G* values of ≤ 100 msec perform better than more coarse granularities [8].

3.1.4 Algorithms and Mechanisms Implementations

The implementations of standard TCP includes four intertwined algorithms and mechanisms namely; Slow Start (SS), Congestion Avoidance (CA), Fast Retransmit (FRet), and Fast Recovery (FRec). These four algorithms are fully documented in [9], for the Internet as standards. Key algorithms and mechanisms implemented have been highlighted and explained below:

1. **Slow Start (SS) Algorithm:** The standard TCP connection starts with the sender transmitting segments to the receiver's advertised window (RWND) size into the network. This is fine for hosts (sender/receiver) on the same Local Area Network (LAN). However, problems may arise when there are slower links and routers (bottlenecks) between the source and the destination, because some intermediate nodes such as routers need to queue the packets, and it is likely that router space (buffer) is filled and runs out of space to accept more packets [9]. This implementation decreases the throughput of a TCP flow significantly. Therefore, SS is employed as the more effective algorithm to prevent this performance degradation in TCP.

   SS phase of the algorithm is implemented by ensuring the rate at which new packets should be injected into the network is the rate at which acknowledgements are returned to the sender by the receiver. This is achieved by adding a new congestion window (CWND) to the sender's TCP packet and is initialised to one segment (size announced by the receiver) or set to the default Maximum Segment Size (MSS) value of 1024, 536 or 512 bytes if a new connection is established with the host on a different network. The CWND is then increased by one MSS on every ACK received by the sending host. This is referred to as Additive Increase (AI) as expressed mathematically by Eqn. (3.1.2). The TCP sender can transmit up to the minimum of the CWND and RWND, i.e $min(CWND, RWND)$, where the flow control is imposed by the sender using CWND based on the perceived assessment of network congestion, while the receiver uses RWND to impose flow control based on the available buffer space at the receiver for that connection [9].

The key objective of the SS algorithm phase is to probe the network for the available capacity in order to prevent network congestion [10]. In some algorithm implementations, the initial CWND (IW) value, must be $\leq 2MSS$ bytes, while other non-standard TCP algorithms allow larger value limited by Eq. (3.1.9).

$$IW = min(4MSS, max(2MSS, 4380\ bytes)) \tag{3.1.9}$$

Usually, the TCP sender begins by sending one data segment and waits for ACK from the TCP receiver. When the sender receives an ACK, the CWND is increased from one to two, and now two segments can be sent. When each of the segments is ACKed, the CWND is incremented from two to four. This implementation doubles the CWND (AI) when ACK is received by the senders, which leads to exponential window growth as shown by Eq. (3.1.1). Delayed ACKs from the receiver side and one ACK for every two segments received can break the exponential growth of the CWND. The maximum capacity of the network may be reached and the intermediate router start to discard packets, which informs the sender that the CWND become too large relative to the network/link capacity.

Most TCP schemes implemented, execute SS algorithm phase when CWND ¡ ssthresh (γ) by randomly selecting high initial value of *ssthresh* and decreasing that value in response to congestion SS phase is exited when CWND $\geq ssthresh(\gamma)$ or congestion is detected [9, 10]. Prior TCP algorithm implementations executed the SS phase only when the sender and receiver are on different networks [9].

2. **Congestion Avoidance (CA) Algorithm:** In standard TCP, congestion events occur when there is a bottleneck with data arriving in a larger pipe (faster link) and exiting in a smaller pipe (slower link) or when multiple input data streams arrive at a router with an output capacity less than the sum of the inputs to the router [9]. During the congestion event, TCP assumes packet is lost and trigger the CA algorithm phase. The standard TCP algorithm as-

sumed the packet loss by corrupt or damage packet is very small ($\ll 1\%$), thus packet loss only indicated congestion somewhere along the network connecting the source and destination and the CA algorithm is implemented to deal with TCP flow lost packet situations [9].

The two key indicator parameters for packet loss are a timeout event and duplicate ACKs (dupACKs) received. The CA slows down transmission rate of packets in the network using Multiplicative Decrease (MD) of halving the current window (0.5CWND), and then triggers SS again, but has different objectives and is independent of the SS phase, although the two algorithms (SS and CA) are practically implemented together as in Eqs. (3.1.1) and (3.1.2) [9]. Like SS, the CA needs to maintain two state variables for each connection; *ssthresh* size and CWND. The CA algorithm is employed by the TCP sender when CWND > *ssthresh* until congestion is detected, while either of the CA or SS could be employed when CWND = *ssthresh* [10].

During the CA algorithm phase, as mathematically expressed by Eqs. (3.1.1) and (3.1.2), CWND (W_{i+1} or W(t)) is incremented by one full-sized segment after receiving multiple ACKs equivalent to the value of CWND/MSS, that is; each (CWND/MSS) ACKs is updated to a new value of CWND = CWND + MSS. To update CWND for each non-dupACK (segment arrived the receiver in order of their sequence number) received during the CA phase Eq. (3.1.10) is usually used, which indicates a linear growth rate of W(t) as in Eqns. (3.1.1) and (3.1.2) [9, 10].

$$CWND_{new} = CWND_{prior} + (MSSXMSS)/CWND_{prior} \tag{3.1.10}$$

The general principle for CA algorithm implementation is to increase the CWND by one full-sized segment segment per RTT and not after receiving number of ACKs. Implementations that maintained CWND in full-sized segment have difficulty using Eq. (3.1.10), therefore employ the general principle of CWND = CWND + MSS as an alternative [10]. In the event of detecting packet loss by the source using RTO, the size of γ must be set to a value less than or equivalent to Eq. (3.1.11) [10].

$$\gamma = max(FlightSize/2, 2XMSS) \tag{3.1.11}$$

The Flight Size (FS) is the quantity of UnACKed data in the network, which is a vital parameter to use here instead of CWND as was mistakenly used in some TCP scheme implementations. Additionally, during RTO, the CWND value must not be set to be greater than the window at the loss event (one full-sized segment) regardless of the size of IW, which indicates switching back to SS phase. Thus, after the retransmission of the lost/dropped packet, the TCP source employs the SS algorithm to increment the CWND from one full-sized segment to the new value of γ where the CA algorithm phase starts again. Generally, the combined implementation of SS and CA algorithms in standard TCP connection operates as given below:

- Initialise a TCP connection by setting CWND to one or two MSS and γ to 65535 bytes or 128 seg (MSS = 512 bytes/segments; 65535/512 = 128).
- The TCP flow output procedure must not send data above the minimum of CWND and RWND
- At the event of congestion, indicated by timeout or arrival of dupACKs, halve the current window, CWND/2 (MD), the min((CWND,RWND), 2MSS) value is then set as new *ssthresh*. The congestion event resulting from a timeout executes the SS phase again by setting the CWND = MSS.
- New ACK is received by the sender, increment CWND based on the algorithm (SS or CA) being executed by the TCP at that time according to mathematical expression in Eqns. (3.1.1) and (3.1.2).

The SS algorithm is executed when CWND $\leq \gamma$, otherwise the CA algorithm is executed, if CWND ¿ γ. The SS phase continues until CWND is updated and reaches halfway to when congestion event happened, because it already started recording from half of the CWND size that led to the congestion. Therefore, the key differences of the SS and CA algorithm implementations can be summarised as follows; SS increases the CWND exponentially (AI) by the amount of ACKs received within each RTT, whereas the CA increments the CWND at most ($\leq$) one full-sized segment each RTT regardless the amount of ACKs (can be multiple ACKs) received within that single RTT. The mistake made by many TCP implementations is the incorrect addition of a small fraction of the segment size (usually MSS/8) in the CA phase, and this should be avoided in the future proposals for better stability [9].

The early version of TCP (Tahoe), implemented only the SS and CA algorithms and employed the timeout (RTO) and retransmission mechanisms for loss detection and recovery respectively [3, 9, 10]. Tahoe waits for the timer (RTO) to expire to indicate loss before retransmitting the missing segment. This scheme continued until 1990 when heuristic Fast Retransmit and Fast Recovery algorithms were introduced in TCP Reno as another implicit mechanism to detect loss and congestion using duplicate ACKs (dupACKs) instead of waiting for the timer (RTO) to expire. These implicit techniques, that is utilising timeout (Tahoe) and dupACKs (Reno) to detect loss, and variations in RTT to detect congestion formed the basis of earlier versions of standard TCP described in subsequent sections.

3. **Fast Retransmit Mechanism:** The Fast Retransmit (FRet) algorithm is a mechanism employed by the TCP sender to detect and fix data (packet/segment) loss based on the receiving at least three dupACKs that indicate the segment was lost before reaching the receiver instead of out-of-order segments at the receiver end [9, 10]. The TCP sender retransmits the lost segment after receiving the 3 dupACKs from the TCP receiver before the retransmission timer expires, i.e without waiting for the RTO to occur [9, 10]. Prior to the modification of CA algorithm in 1990s and introduction of FRet/FRec [9, 11, 17], dupACK is immediately generated when an out-of-order segment is received at the TCP receiver end for the purpose of experimenting with the FRet algorithm and to notify the TCP sender that an out-of-order segment was received, and what

sequence number is expected, this dupACK should not be delayed [9]. The TCP sender distinguishes between out-of-order and loss events by the number of dupACKs it receives. The assumption is there will be only one or two dupACKs before the reordered segment is processed at the receiver, which generates a new non-dupACK, thus the out-of-order segments are considered, whereas receiving at least three dupACKs in a succession strongly indicates segment was lost and the retransmission of the lost segment is performed before the expiry of the retransmission timer. The sender uses dupACKs to detect network problems such as reordering, replication and dropped/lost of segments along the network path [9, 10, 17].

4. **Fast Recovery Mechanism:** After retransmission of lost segment using the FRet algorithm, CA phase is performed by using Fast Recovery (FRec) algorithm as a mechanism to govern the continuous transmission of the new data until a non-dupACK arrives at the sender end. This enhancement provides high throughput under moderate congestion for large windows in particular. The reception of dupACKs not only indicates network problems like lost segments, but also that other segments, at least three (if 3 dupACKs are received), most likely arrived at the destination, since the receiver can only generate the dupACKs when segments were received (n dupACKs, imply n segments received), which indicates that such segments have left the network and are now in the receiver's buffer [9, 10]. Therefore, dupACKs also tell the TCP sender that, there is still flow of data (network not congestion) to the TCP receiver and should not reduce the flow abruptly by performing the SS phase, thus this is the reason CA/FRec are used instead of going back to SS phase [9, 10]. Usually, the FRet and FRec algorithms are implemented together as given below, just like CA and SS algorithms are implemented together as mentioned above [9, 10]:

   - After receiving the third dupACK in a sequence, the TCP sender sets the new value of *ssthresh* to one-half the current CWND (i.e the FlightSize), but at least two segments (2MSS) as expressed by Eq. (3.1.11).

- After the lost/missing segment is retransmitted, set the new value of the CWND = ssthresh + 3MSS. This fills up the CWND by the amount of segments that already left the network and are stored in the receiver's buffer.
- For each dupACK received in addition to the previous, increase the CWND by one MSS (CWND = CWND + MSS). This inflates the CWND for the added segment that left the network. If allowed by the new value of CWND/RWND, a segment is transmitted.
- On the arrival of next ACK (non-dupACK) for the new data segment, set CWND = ssthresh; this value is determined by Eq.(3.1.11). This ACK should be for the start of retransmission (step 1 above), one RTT after the retransmission. The ACK should also acknowledge all the intermediate segments sent between the lost segment and the reception of first dupACK. TCP CWND is now down to one-half (CWND = ssthresh) the rate it was when the segment lost occurred, thus it is now in CA phase.

These two implicit mechanism (FRet and FRec) enhancements for loss detection and recovery were implemented for the first time in the standard TCP schemes Tahoe 4.3BSD (FRet) followed by the SS, and Reno 4.3BSD (FRec) releases. However, these mechanisms considered the network as a black-box and packet/segment loss as an indication of congestion in the network with little or no regards to delay and multiple packet losses in a single RTT. The techniques may work effectively for purely best-effort TCP data communication with less or no sensitivity to delay or loss of individual packets, but are not effective or aimed to help applications and interactive traffic like audio, video, telnet, and web-surfing that are sensitive to the multiple packet losses or increasing delay (latency) due to retransmissions of lost packets [12]. The network as a black-box notion of some TCP schemes, the state of the congestion is determined by the sender's probing for the network state, through progressive load (CWND, outstanding/unACKed data) increment until the network become loaded or congested indicated by segment loss, which is ineffective for some networks. Therefore, other explicit feedback (detection and recovery) mechanisms like Selective Acknowledgement (SACK) option and Explicit Congestion Notification (ECN) that provide information explicitly to detect loss and congestion by specifying the exact sequence number of the lost segment or marking packets by the routers as notification to indicate network congestion instead of packet drops were proposed in the newer TCP schemes as described below.

Selective Acknowledgement Option

Performance degradation also arises when TCP connections experience multiple segment loses/drops within one window (CWND) of data with catastrophic consequence on throughput due to the lost of ACK-based clock in a cumulative ACK scheme, which leads to multiple packet losses [13, 18]. FRet/FRec feedback mechanisms use cumulative ACKs, which do not ACK the received segments not at the left edge of the receive window. This implicit information is limited and only allows TCP sender to detect single lost segment per RTT using FRet/FRec mechanisms [3, 13]. The sender is also forced to wait one RTT in order to learn about each missing segment, or an aggressive TCP source retransmit segments unnecessarily, which may have already been delivered successfully to the receiver. To help overcome these limitations of the cumulative ACKs based schemes, the Selective Acknowledgement (SACK) mechanism option integrated with Selective Repeat Transmission policy were proposed [3, 13]. The receiver sends SACK packets to the source with explicit information about the received data segments, which the source can rely on for retransmitting the lost data segments only.

Nowadays, many TCP schemes including CUBIC, HYBLA and other LFN charactarised by high BDP used SACK options in their implementations with evidence on improved and better performance than disabling SACK option [13, 18]. The SACK is usually implemented/enabled in two simple steps, firstly by enabling a SACK-permitted option, which may be included as two-bytes in a SYN segment to show that the SACK option is possible when connection is established [18]. Secondly, employing the SACK option by the TCP receiver when sending ACKs with explicit information over an established connection to inform the sender of the segment received and queued after permission has been granted from SACK-permitted [18].

SACK is optional in TCP implementations and enabled when the SACK-permitted option is received for that connection in the SYN segment. This enables TCP data receiver to use SACK option in an ACK segment whenever it has queued an un-ACKed data in its buffer. The SACK option includes additional explicit information from the receiver, which the sender uses to optimise retransmissions [18]. The SACK is an optional alternative to TCP's cumulative ACK mechanism. It does not completely replace the cumulative ACK-based mechanism neither is it mandatory. The explicit information for each data block contains the first sequence number in a block called the left-edge and the sequence number immediately behind that block is called the right-edge, more details on Implementation can be found in [13, 18]. Unfortunately, SACK has never been used in the Internet due to the debate about how it should be employed concurrently with the window shift option of the TCP [13, 18], and precisely how the TCP sender responds to SACK was not specified in the SACK documentation, which made most implementations to retransmit all lost packets blocks [3].

Explicit Congestion Notification

Explicit Congestion Notification (ECN) is another explicit feedback mechanism or technique proposed to solve the limitations of the implicit measurement aimed at tackling congestion in a network with established TCP connection. This mechanism needs routers all through the Internet to notify TCP/IP connections as congestion occurs [3]. The ECN mechanism helps the Internet provide a congestion indication for an incipient congestion as congestion starts to develop through packet marking instead of dropping them to notify TCP about the developing congestion [12]. The concept of ECN is as simple as marking a TCP segment as it flows over the internet through the routers along the flow path. These routers use a pair of bits in the IP header to register congestion status to help notify the TCP receiver if the segment experienced congestion at any point along the network path. The TCP receiver then uses next ACK to inform the TCP source about the congestion event that occurred, which the sender responds by decreasing its CWND accordingly. The ECN mechanism uses a two bit IP header to record congestion at the routers and two bits in the TCP header from the reserved area to allow the sender and receiver to communicate. One of the two bits of the TCP header is used by the receiver to notify the sender about the congestion events while the other bit is used by the sender to inform the receiver that congestion notification has been received. The two IP header bits, called Congestion Experienced (CE) code-point taken from the type of service field of an ECN-capable TCP are used for making the ECN mechanism more robust and a router can use any of the two bits to inform the sender that congestion occurred [3, 12].

The ECN field in the IP header of ECN-Capable Transport (ECT) protocol with two bits, making four ECN codepoints combination from 00 to 11 (decimal 0 to 3). The sender sets ECT codepoints of 10 and 01 to indicate that the end-points are ECT called; ECT(0) for 10 and ECT(1) for 01 and routers treat both codepoints as equal, senders could use either codepoints to indicate ECT, on a packet-by-packet basis. Packets not using ECN are indicated by not-ECT codepoint of 00 while routers use CE codepoint of 11 to indicate congestion to the end nodes (receivers) more details on ECN can be found in [12]. Unlike SACK, ECN has been deployed over the Internet and non-compatible IP tunnels would have to be upgraded to conform to it [12].

3.1.5 Establishing and Closing TCP Connection

The TCP connection between sender and receiver is established using a procedure called three-way handshake initiated by the TCP sender. This is achieved by sending and receiving three segments containing Synchronisation (SYN), ACKs and Sequence Number (*Seqn*) data. The first handshake segment sent by the TCP sender to the TCP receiver is identified by the SYN bit set in the code field. The second segment sent back by the TCP receiver to the sender contains SYN and ACK bits set to acknowledge the first SYN received, and another SYN from the TCP receiver for completing the handshake procedure as shown in Fig. 3.1. The final handshake segment sent by the TCP sender is only an ACK of the previous SYN + ACK reply to inform the destination that TCP connection is agreed and established between the end systems [3].

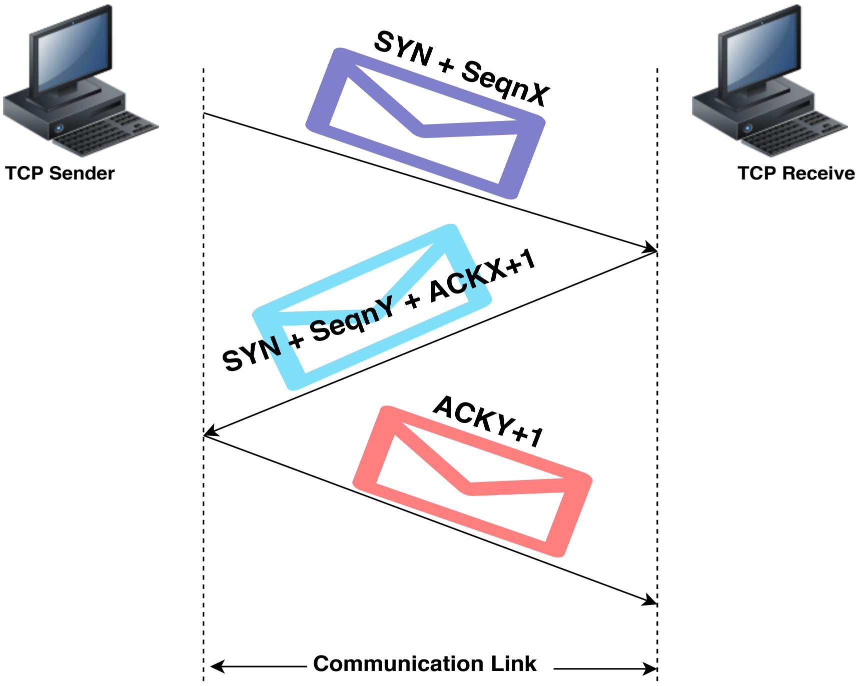


Figure 3.1: A Typical TCP Three-Way Handshake Procedure

TCP software on the end system usually waits passively for the Three-Way Handshake (3WH) initiated by the TCP software of another end system. The procedure was carefully designed to work effectively even when both end systems attempt to initiate the handshake concurrently such that the TCP connection can be established from either or both ends at the same time. Data flows in both directions equally without priority of being master or slave once the connection has been established.

The 3WH procedure is necessary and sufficient for accurate SYN between the two TCP connection points, since segments can be delayed, lost/dropped, duplicated or arrive out-of-order. Therefore, TCP has a rule of ignoring additional connection requests after a connection has been established [3].

The handshake procedure achieves two vital functions; first it guarantees both end nodes agreed and ready to exchange data with both ends knowing their readiness, secondly it provides both end points to choose and agree on initial *Seqns* with each end system randomly selecting initial *Seqn* that will be use to identify bytes in the data stream it is sending, which is vital.

Terminating TCP Connection

The end systems using TCP flow to communicate data ends the conversation using a close operation procedure. This is a modified version of the 3WH procedure to terminate TCP connections. The connections are full duplex that contain two independent data stream transfers on each direction, thus, the application program on the sender's side informs the receiver to terminate the connection in one direction when it has no more data to send. The other half of the connection is terminated when the sender finishes transmitting the final data, waits for that to be ACKed by the receiver and then sends a segment containing Finish (FIN) bit set. The TCP receiver ACK the FIN segment and then notifies its application software through OS's end-of-file mechanism that no more data to be received as shown in Fig. 3.2.

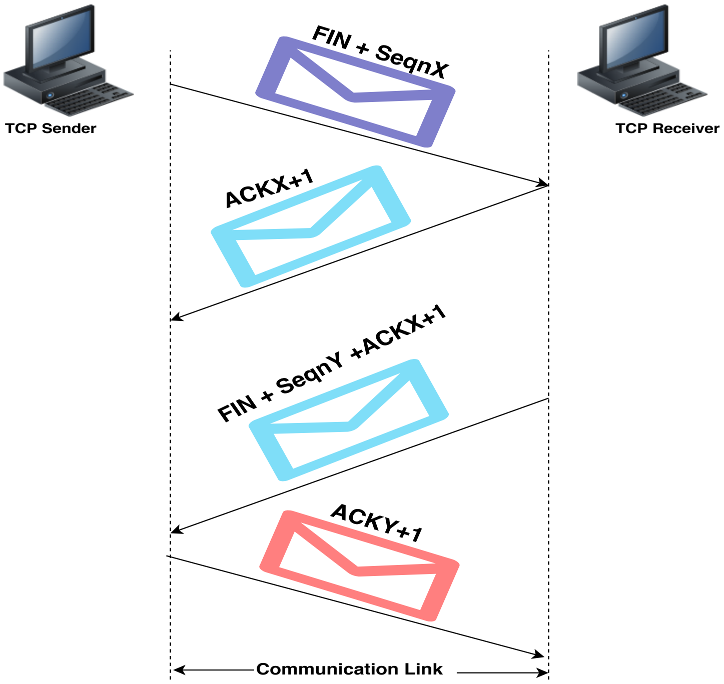


Figure 3.2: A Typical TCP Close Operation Procedure

After closure of a connection in a specified direction, the TCP receiver will not accept any more data stream from that direction, but data flow with ACKs from receiver can continue in the opposite (sender) direction until the sender closes the conversation in its direction. The TCP software on each end system deletes record of its connection when conversations on both directions are closed.

The difference between the 3WH for establishing the connection and its modified version for connection closing operation (terminate the conversations) is the reception of the initial FIN segment. After the FIN is received, the receiver sends back only an ACK to inform the application program to terminate without generating another FIN segment instantly. The ACK prevents retransmission of the initial FIN segment by the sender during the waiting period. Finally, the application software instructs TCP to terminate connection completely and sends the second FIN segment to the TCP sender, which will then reply with the third ACK segment to conclude the process as shown in Fig. 3.2.

3.1.6 Standard TCP Variants

TCP/IP and the Internet technologies are evolving and growing exponentially, with new modified proposals and old being revised to cope with the demands and dynamics of communications technologies. The significant demand is connectivity that brings additional traffic, new Internet uses bring new applications and dynamics in traffic patterns. Internet data traffic is expected to grow dramatically with the deployment of 5G networks, new applications and requirements are also expected which requires the use of different network and communications technologies. Therefore, TCP implementations need to be revisited and redesigned for better performance and to accommodate future heterogeneous networks and communications technologies.

TCP is the most widely used transport protocol in the Internet nowadays, and accounts for $80 - 90\%$ of the Internet data traffic. This has caught the attention of many researchers investigating the performance of TCP over different communications channels and heterogeneous network environments. This has resulted in different TCP variants being proposed, here we review three key standard TCP variants schemes, which include *Tahoe, Reno and New Reno* because of their relevance to how the standard TCP evolved.

Tahoe

Tahoe is the first standard TCP version to implement the congestion control algorithm proposed by Van Jacobson [5], which is based on the CWND regulating the number of segments sent (transmission rate) over the network, and the TCP sender estimation of losses as a mechanism for controlling congestion [19]. The CWND increment in Tahoe follows the SS exponential increment phase and CA linear increment phase as described earlier.

TCP Tahoe uses Go-back-N error-recovery and timeout loss detection/recovery mechanisms. These mechanisms are inefficient and suffer from quite a few drawbacks including the limitation of Go-back-N error-recovery that it can only retransmit packets already received by the TCP receiver because it uses cumulated ACKs. Another drawback of Tahoe is its loss detection and recovery mechanism using a timer that is triggered for every packet and remains active until the reception of a corresponding ACK or FRet. occurs. Therefore, at every packet loss, it waits for timeout and the pipeline to be emptied, which is costly in high BDP links. After loss detection, Tahoe goes back to SS phase with a value ssthresh (γ) = CWND/2 to recover from the loss. New mechanisms were proposed to overcome these drawbacks; including selective repeat mechanism to solve the limitations of error-recovery, while FRec. and FRet. mechanism for loss detection and recovery issues in Tahoe [19]. These modifications and many more were implemented in the other TCP versions such as Reno as discussed below.

Reno

TCP Reno is an improved version of Tahoe, which includes a modification of the FRet. algorithm to integrate FRec. algorithm and employs selective repeat mechanism for the packet loss recovery [19]. These enhancements bring some level of intelligence in Reno that helps to detect packet loss earlier while the pipeline (buffer/link) is not emptied. The modification of FRet. to incorporate FRec. halves the CWND (set CWND and ssthresh to $CWND_{loss}/2$) without going back to SS phase. During the FRet. period, CWND is incremented by the number of dupACKs not the usual ACKs. The TCP sender using Reno needs an immediate ACK whenever a segment is received, and the sender then assumes that receiving a dupACK indicates the next segment in the sequence has been delayed in the network and some segments arrived at the receiver out-of-order or lost due to network congestion. Therefore, when the sender receives three dupACKs, it assumes the segment was lost and activates the FRet. mechanism so that the segment is re-transmitted before the timeout timer expires and the pipe is still almost full. After the packet loss, returning to the SS algorithm phase like in Tahoe, it empties the pipe and that is the drawback solved by integrating FRet., FRec. and selective repeat mechanisms instead of returning to the SS algorithm.

Nonetheless, the Reno version also has some limitations that required improvements for better performance, which include successive FRet. or the false FRet. followed by a false-recovery problems. Another drawback is performance degradation in the presence of multiple packet drops within the same transmission window or single RTT. This makes Reno performance the same as Tahoe under high packet losses since it can only detect one packet loss. To solve these limitations, modifications were proposed and implemented in other TCP versions such as NewReno as discussed below.

NewReno

TCP NewReno scheme is a proposed enhancement to the behaviour of Reno that limits performance at the event of multiple packets loss in one window. In New Reno, a modification was made in FRec. algorithm, which used partial ACKs (pACKs) to indicate multiple losses in one window [19]. This modified FRec. and changes that followed have been described in [14–16], and found to be adequately efficient for wired networks with low to moderate BDP, but inefficient with poor performance over high BDP and high link error wireless network environments. NewReno performs poorly in wireless network environment, because it assumes and interprets packet loss as a result of network congestion [16, 19]. These losses could be due to wireless link error or bad transmission due to corruption. In the absence of SACK, the sender has little implicit information available to make retransmission decisions during FRec., thus NewReno algorithm responds to pACKs, which are ACKs that cover new data, but not all of the data outstanding when the loss was detected [16].

In the NewReno FRet/FRec algorithm modification, the sender can infer, from the received dupACKs, whether multiple losses in one window of data likely occurred, and avoid retransmission time-out or making multiple CWND reductions due to such event. The NewReno applies to the FRec. mechanism, starting when three dupACKs arrive at the TCP sender and finishes when either retransmission timeout occurs or ACK that acknowledges all of the data including the outstanding data when the FRec. started is received [14–16].

NewReno attempted to solve the problem of multiple packet losses in one window by responding to pACKs in the absence of SACK due to either the SACK option is not locally supported or TCP connection at the other end is unwilling to use the explicit SACK option [14–16]. Moreover, as mentioned earlier, NewReno performance becomes degraded over high *BDP* and wireless network environments such as SatCom and hybrid ISTNs environments. This poor performance over particular network environments motivated new and improved TCP schemes presented in the following sections and this thesis.

3.1.7 TCP Performance Over Heterogeneous Networks

The performance of standard TCP depends on link capacity and transfer rate (achievable throughput), RTT and their product, known as BDP. This product measures the data segments/packet that would fill the pipe (link/buffer) needed at the source and destination for maximum transfer rate on the TCP connection over the network path. This is simply the amount of unACKed that must be accommodated by TCP to keep the pipe full [17]. The performance of TCP becomes degraded with large BDP value network environments such as SatCom, Fibre Optic and heterogeneous networks environment incorporating one or all of them. Nowadays, real communication networks are heterogeneous. The Internet communication path with a large BDP is called Long, Fat Pipe and networks containing such a path is referred to as LFN, pronounced elephan(t).

High latency and high capacity satellite channels, particularly the next generation and current GEO-HTS exhibit very large BDP and are categorised as LFN.

Another category of LFN are terrestrial fibre-optical paths, which have low delay of about 30 ms, and exhibit very high capacity [17]. These LFN environments are attributed to very high outstanding unACKed (BDP) segments of at least MSS, usually 1024 bytes each.

The three fundamental issues of standard TCP leading to performance degradation over LFN paths and some heterogeneous network networks environments are window size limit, multiple losses recovery and RTT measurement (RTTM) [17]. Attempts were made to propose solutions to some of these performance problems and research is still going on to identify other problems and solutions.

1. **Window Size Limit:** Standard TCP uses a sixteen (16) bit header field for its receive window (RWND) to the sender, which specifies the largest window (CWND) allowed as $2^{16} = 65536$ bytes $\approx$ 65 KB containing 64 segments with MSS=1024 B, 128 segments with MSS=512 B in each CWND. TCP window scale option was proposed as a solution to this limitation [17], this allows CWND to be larger than 65 KB and defines an implicit scale factor used to multiply the window size in a TCP header to compute the true CWND size value [17]. The window scale option extends the TCP window from 16 to 32 bits and employs a scale factor to convey this new 32 bit value in the standard TCP header of 16 bits window, more details can be found in [17].

2. **RTT Measurement:** Reliable data delivery in TCP is implemented by retransmission of unACKed segments within a specified retransmission timeout (RTO) period. Accurate dynamic computation of RTO is crucial to TCP performance and it is computed by estimating the mean and variance of the measured RTT, the time interval between sending and receiving ACK a data segment. To solve the problem and determine the RTT accurately for achieving high performance, TCP options such as Timestamps and RTTM mechanism were proposed [17]. Mnemonic RTTM was used for the mechanism to differentiate it from other uses of the Timestamps option, RTTM mechanism is another option that utilised Timestamps, which allows almost every segment including retransmissions to be timed at insignificant computational cost.

3. **Loss Recovery:** Prior to recent modifications of standard TCP, the previous implementations drained the data pipeline with every segment loss, and activated the SS phase for the loss recovery. The modifications using algorithms such as FRet and FRec. were made to recover from single packet loss in one window without draining the data pipeline. In an event of more than one packet loss per window, retransmission timeout occurs with resulting pipeline drain and activation of SS phase [17].

   Inflating the size of the CWND to equalise the capacity of an LFN leads to increase of the likelihood of more than one packet dropped in one window, which could have a catastrophic impact on the achievable throughput of TCP over LFN such as Satellite and 5G networks environment of the future. Explicit feedback mechanisms for detection and recovery from multiple losses/drops per window such as SACK and ECN options were then introduced. These modifications provide the TCP sender with explicit information on which packets are queued at the receiver buffer and which are yet to arrive at the receiver site. However, in the non-LFN environments, these explicit mechanisms such as SACK reduce the amount of segments retransmitted, but do not improve the performance. Although there are technical issues in both format and semantics of the SACK option questioning the complexity, it has become vital option in the LFN environment [13, 17].

3.2 TCP Over Satellite Channel

Over the years, standard TCP was effective, efficient and robust over terrestrial networks especially wireline [24], but as mentioned in the previous sections, TCP performance becomes degraded and inefficient over heterogeneous networks involving satellite and wireless networks characterised with high link error rate, long latency and larger BDP (LFN) [17, 18, 36]. Performance of standard TCP protocol variants like Tahoe, Reno, and NewReno were designed to depend largely on network latencies (delays) and ACKs for congestion control and management with little or no regards to the peculiar characteristics of satellite and wireless radio links such as long/variable RTT, large BDP, and high Bit/Packet Error Rate (BER/PER) that may contribute to high segment losses and inefficient utilisation of available bandwidth through a catastrophic effect on achievable throughput. These made the original TCP algorithm design not fit for SatCom and wireless radio channels [17, 22, 23, 25, 36]. The standard TCP congestion control mechanism was designed to increase congestion window (CWND) and segment transmission rate based on RTT. These mechanisms degrade the performance of TCP connections over satellite channel, characterised by long RTT, large BDP, and variable RTT that lead to high Packet Delay Variation (PDV) or jitter [22, 24, 36]. Moreover, the original TCP algorithm attributed all the segment losses as due to congestion. This is reasonable for wired connections with considerably limited errors [23, 25, 36], but indirectly degrade performance in the presence of satellite channels with significantly higher BER and Intermittent link in the case of MEO and LEO satellite links [22, 27, 28, 36]. In the presence of link losses, the original TCP algorithm reduced the size of the CWND by half persistently, which has devastating effects on Satellite link performance with high RTT, BDP and high BER, which hindered fast re-opening of the CWND [22, 26, 36]. Other characteristics of satellite channels that have significant impacts on the performance of standard TCP are Asymmetry of down/up links (due to ground equipment cost) and intermittency of non Geostationary Earth Orbit (GEO) satellite links [22].

Because of these unique attributes of SatCom channels, that impact the performance of transport protocol like TCP, recommendations [27, 28] and proposals [13, 18, 20, 22, 23, 27–29, 36] were made to enhance TCP over networks with at least one satellite link in order to achieve better performance and efficiently utilise the huge available capacity of satellite systems [21, 35, 37]. Some of the key TCP enhancements proposals relevant to the satellite and other large BDP or hybrid network environments are discussed in the following sections.

3.2.1 TCP BIC

Binary Increase Congestion Control (BIC) was proposed to solve a very vital constraint of many TCP enhancement algorithms proposed in the past, this constraint is RTT unfairness where competing TCP flows, with different RTT (low and high), take an unfair share of the available bandwidth [29, 30]. This action limits high RTT flows such as LFN and satellite paths, because many previous TCP enhancements proposals focused on scalability by making CWND increase rate, $W(t)$ larger as CWND size, W_{i+i} grows and ironically that is what makes them scalable while severely having RTT unfairness problem [29]. The TCP BIC proposed a congestion control algorithm that reduces unfairness due to different RTT while maintaining the bandwidth scalability and friendliness of the TCP flows by using two CWND control policies known as additive and binary search window increase function [29]. In TCP BIC, additive increase with a large window increment is employed when CWND is large to ensure RTT fairness and good scalability, while binary search increase is used when the CWND is small in order to support TCP friendliness [29, 39]. As mentioned, network environments with high-speed and large delays such as satellite present a unique environment in which the original TCP suffers from capacity or bandwidth underutilisation problem, many congestion control suggested to alleviate this problem consider mainly bandwidth scalability and TCP friendliness properties at the expense of RTT fairness property tackled by BIC proposal [18, 22, 23, 29]. The RTT unfairness (RTT_{UFS}) of two flows with different RTTs given by RTT_i and RTT_j such that $RTT_i \leq RTT_j$ is the ratio of their average throughputs, R_{avg} computed from average window size, W_{avg} as follows.

$$W_{avg} = (RTT/(tX\sqrt[d]{c}))^{d/d-1} \tag{3.2.1}$$

where c and d ($0.5 \leq d \leq 1$) are protocol-dependent constants, t is the time interval between two consecutive loss events of a flow during steady state. The constant d values for AIMD, HSTCP, and STCP are $0.5, 0.82$ and 1 respectively and the average sending rate (throughput), R_{avg} is given by Eq. (3.2.2).

$$R(w)_{avg} = \frac{W_{avg}}{RTT} \Leftrightarrow R(p)_{avg} = \frac{c}{RTTX(p^d)} \tag{3.2.2}$$

Thus, RTT_{UFS} is computed by substitution and simplification of Equations (3.2.1) and (3.2.2) in the R_{avg} ratio of two flows with different RTTs as in (3.2.3) and (3.2.4).

$$RTT_{UFS} = \frac{R(w)_{avgi}}{R(w)_{avgj}} = \left(\frac{W_i X RTT_j}{W_j X RTT_i}\right) \tag{3.2.3}$$

The simplified version of RTT unfairness is given by;

$$RTT_{UFS} = \left(\frac{RTT_j}{RTT_i}\right)^{1/(1-d)} \tag{3.2.4}$$

TCP BIC proposal achieves better and improved performance compared with High Speed TCP (HSTCP), Scalable TCP (STCP) and AIMD based TCP like NewReno considering the three properties namely bandwidth scalability, TCP friendliness and RTT fairness [29]. Although designing congestion control algorithms to support all the three properties for LFN environment is challenging, BIC satisfied these criteria better than other TCP schemes mentioned [29]. For instance, HSTCP and STCP are extremely scalable in low-loss rates and TCP friendly under high-loss rate, but are not RTT fair to other flows with different RTT, the RTT_{UFS} of two flows were found to be $(RTT_2/RTT_1)^{5.56}$ for HSTCP flows and $(RTT_2/RTT_1)^{\infty}$ for STCP flows using Equation (3.2.4) and substituting different values of d as given above. Moreover, AIMD TCP schemes with square $RTT_{UFS} = (RTT_2/RTT_1)^2$ achieve scalability by increasing its additive increase factor, which is not friendly with other TCP flows [29]. The two phases of the BIC algorithm are binary search and additive increase as shown in Fig. 3.3 and explain below.

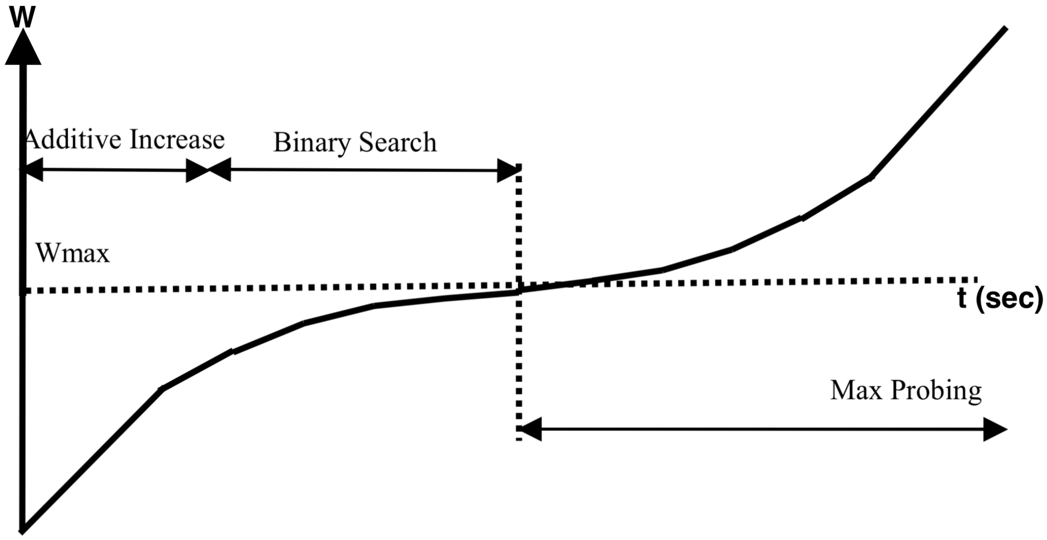


Figure 3.3: BIC-TCP Window Growth Function $W^B(t)$ [29]

- **Binary Search Increase:** In this phase, congestion control is viewed as searching problem where the system provides an explicit Yes/No feedback response at the event of packet loss, explicit to whether the current sending rate (average achievable throughput), R_{avg} is greater than the network capacity, C, i.e $R_{avg} > C$. The search problem starts with current minimum window size W_{min}, a window size just after the FRec phase (i.e a window size just after reduction at the event of loss) and maximum W_{max}, a window size just before the last FRec. phase (a window size just before reduction) where the last packet loss event took place [29, 31].

  The BIC binary search algorithm calculates frequently the midpoint W_{mid} between W_{min} and W_{max} and sets the current window size $W_{current} = W_{mid}$ value and checks for feedback in the form of packet losses. The W_{mid} is considered the new W_{max} (W_{new_max}) when packet loss is detected through the feedback received, or set to new W_{min} (W_{new_min}) when packet loss is not detected. This search process is repeated until the difference between W_{max} and W_{min}

drops to a value less than the preset threshold, known as minimum increment S_{min} [29]. The search ended when $W_{max} - W_{min} < S_{min}$.

The binary search method allows more aggressive bandwidth probing at the start when the difference from the $W_{current}$ to the target window size W_{target} is large, and less aggressive as the $W_{current}$ approaches the target W_{target}. This gives the protocol a distinctive feature of a logarithmic increase function, which reduces its increase rate as window size approaches the threshold or saturation point. Protocols that are scalable increase their rates at the saturation point to achieve maximum increment between two successive loss event (epoch). Usually, the amount of loss packets is proportional to the size of the last increment before the loss event. Therefore, binary search increase can decrease packet loss with key benefit of concave response that connects well with that of additive increase [29].

- **Additive Increase:** The combination of an additive increase strategy and the binary search increase guarantees fast convergence and RTT-fairness of the BIC. If the distance to the W_{mid} from the current W_{min} is too large, increasing the window size directly to W_{mid} may add too much load to the network. When the distance from the $W_{current}$ to the W_{target} in binary search increase is greater than a preset maximum step known as maximum increment S_{max}, window size is incremented by S_{max} until the distance becomes less than S_{max} where the window increases directly to the W_{target}. Therefore, after a large window decrease, the strategy initially starts with linear window increase, and then a logarithmic window increase [29]. The combination of additive and binary search increase is called binary increase, and when combined with multiplicative decrease strategy, the binary increase tends to a pure additive increase under large window values. Larger window values lead to a larger decrease in multiplicative decrease and a longer additive increase duration [29]. However, small window size tends to a pure binary search increase and a shorter additive increase duration [29].

The uniqueness of TCP BIC from other high-speed (high bandwidth) TCP algorithm proposals is its stability property that is achieved by employing the binary search algorithm described earlier, in which the CWND grows to the midpoint W_{mid} between the last window size (i.e W_{max}) where packet loss occurred and last window size (i.e W_{min}) without packet loss for one RTT duration. The search into the midpoint is intuitive and an ingenious technique of finding the current capacity of the path, which must be somewhere between the W_{min} and W_{max} values provided the network conditions remains the same since the last congestion detected through the last packet loss. This window growth by midpoint (W_{mid}) found through the search is more effective and efficient than the previous method used by other TCP algorithms like Reno, NewReno and SACK to grow their CWND one per RTT. After the BIC window grows to W_{mid} without detection of packet loss in the network, which indicates the network can handle even more data traffic, BIC sets the new $W_{min} = W_{mid}$ and performs another search to obtain new W_{mid}. This helps to grow the CWND really fast when the $W_{current}$ size is much less than the available capacity of the network, and slowly reduces the window increment when $W_{current}$ size gets closer to the available capacity, i.e where previous packet loss occurred [20, 29, 39]. The entire window growth function of BIC-TCP is a logarithmic concave function that keeps the CWND much longer at the saturation point (equilibrium) than convex or linear functions that have the largest CWND increment at the saturation point and therefore have the largest overshoot in the event of packet losses. These unique features made BIC highly scalable and very stable [20, 29].

Although BIC was found to have better performance in terms of scalability, stability and RTT unfairness properties, it still has some dependence on RTT to estimate the appropriate CWND by probing the available capacity and also to measure the RTT unfairness as in (3.2.3) [29, 31]. Moreover, BIC window growth function can be aggressive under short RTT or low speed networks and the multiple different phases of window control added extra complexity in the protocol analysis [31]. The TCP CUBIC described below attempted to solve these issues, improve fairness, TCP friendliness and simplify the window growth function of the BIC at the same time retaining its scalability and stability properties for better performance and capacity utilisation for high BDP network paths such as a satellite network environment.

3.2.2 TCP CUBIC

CUBIC is another TCP scheme for large BDP network environments, which is based on BIC described in the previous section. This is an enhancement to the sender's side congestion control algorithm, which simplifies the original BIC window control, enhances TCP-friendliness and RTT-fairness while maintaining the stability and scalability properties of BIC [31–33]. CUBIC as the name implies, replaced the slow linear window growth after congestion/loss events with a CUBIC function in terms of the elapsed time t since the last loss event occurred and is independent of RTT. The modification of window growth to a CUBIC function improved the scalability, and stability over high BDP network paths and also achieved better RTT-fairness with fair bandwidth sharing among competing connections with different RTT since the window growth is independent of RTT and all flows in the same bottleneck network paths grow their CWND at the same rate [31, 32].

Tackling RTT-fairness has been the most challenging issues of many versions of TCP protocols, because the issue involved friendliness to existing TCP flows (TCP-friendliness) and the fair share of bandwidth among the competing flows with different RTT values, which include both inter and intra protocol fairness [31, 32]. Protocol friendliness property determines whether a protocol is being fair to standard TCP, which is crucial to the safety of the protocol by making sure that its use does not unfairly affect the most common network flows using standard TCP. Some of the famous large BDP TCP proposals such as BIC, HSTCP, and STCP achieve TCP-friendliness using TCP-modes or regions where they behave as standard TCP. These algorithms normally enter TCP-mode when their CWND is below a small cutoff constant of around 30 pkts typically, i.e CWND ¡ 30 pkts [31, 32].

The CUBIC algorithm observed that the TCP-region as the regime where TCP performs well should be defined by the congestion epoch time called *epoch_start* (**the real-time period between two successive losses**) and not by the CWND size. Although BDP means the available network capacity by CWND size or packet count, CWND size is not sufficient to characterise the performance of TCP since its growth rate depends on RTT. Therefore, it is appropriate and better to define the TCP-region in real-time to keep the window growth rate independent of RTT, this is the main attribute that makes CUBIC TCP-friendly under both short and long RTT network paths [31, 32]. The CUBIC congestion epoch period (*epoch_start*) and throughput are determined by packet loss rate only. Therefore, CUBIC can operate in a TCP-mode during high loss rate, and short or high RTT with guaranteed RTT-fairness [31, 32]. However, since the CWND grow is fixed and independent of the RTTs, shorter RTT CUBIC flows could experience slower window growth rate compared to standard TCP flows. This is overcome by the use of TCP-mode since TCP like SACK work well under short RTTs [32].

TCP CUBIC was designed to be less aggressive and fairer to standard TCP in bandwidth allocation than BIC-TCP while maintaining its strong properties of stability, window scalability and RTT-fairness. This new congestion control algorithm (CUBIC) has already been implemented and since replaced BIC-TCP as the default TCP algorithm in Linux and has also been deployed globally [32–34]. Through extensive testing in different Internet scenarios, CUBIC is believed to be safe for testing and deployment in the global Internet [33, 34]. The window growth function of CUBIC starts at W_{max} considered as origin point as shown in Fig. 3.4 with the window growing very fast on window reduction and slows down when it gets closer to W_{max}, the window increase becomes almost zero around W_{max} giving a *concave* profile of a cubic function, see Fig. 3.4 [31, 32]. When the window size grows above the W_{max}, CUBIC starts *probing* (see Fig. 3.4) for more bandwidth with slow window growth at the start and grows faster as the window moves away from W_{max} to a convex profile of a cubic function. The slow growth around W_{max} (concave region) improves the stability and increases the network capacity utilisation while the fast growth away from W_{max} (convex region) guarantees the scalability of CUBIC [31].

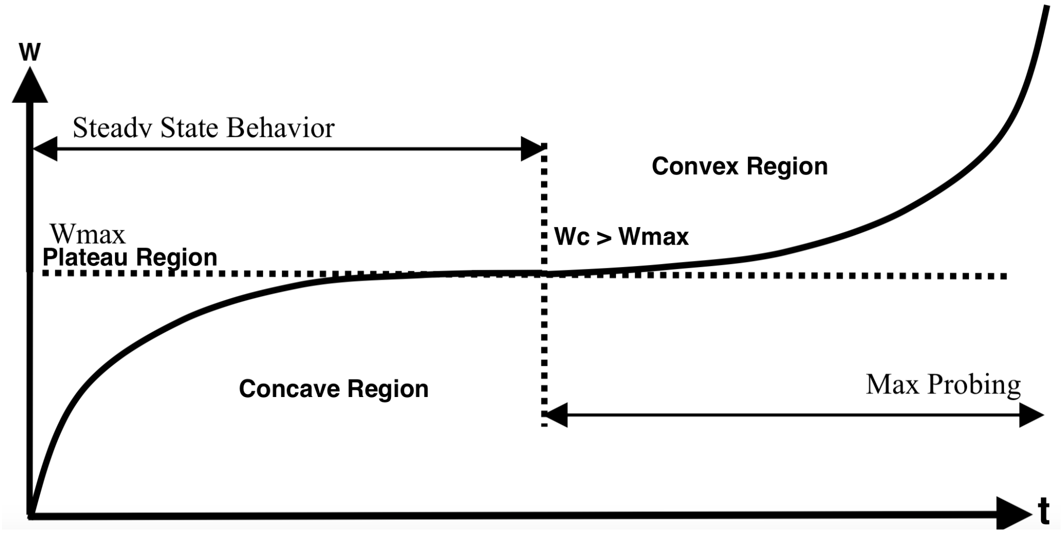


Figure 3.4: TCP CUBIC Window Growth Function $W^C(t)$ [31, 32]

As mentioned, CUBIC was designed to simplify and enhanced the window control of BIC, the congestion window function, independent of the RTT, can be obtained by Eqn (3.2.5) [31, 32].

$$W^C(t) = C(t - K)^3 + W_{max} \tag{3.2.5}$$

$$K = \sqrt[3]{\frac{\beta W_{max}}{C}} \tag{3.2.6}$$

where C is scaling factor set to determined the aggressiveness of window increase in large BDP network environments, t is elapsed time from the last window reduction at the event of packet loss (the beginning of the current CA), W_{max} is window size just before the last window reduction, β is a constant Multiplicative Decrease (MD) factor for window reduction at the time of loss event (replaced the halving window in original TCP), and K computed using equation (3.2.6) is the window growth rate function constant which determine how *slow* or *fast* the window size increases or decreases, i.e the time interval that window function in (3.2.5) takes to increase W^C to W_{max} (origin-point) when no further loss event occurred within that period [31, 32, 34].

At the time of last window reduction following a loss event, CUBIC registers W_{max} as the window size at which the segment/packet loss event occurred and then reduces the CWND by a factor of β i.e the new window size $CWND = \beta W_{max}$. This makes the competing connections on the same E2E path converge to the same window size and a fair share since they decrease by the same multiplicative factor β. This property is controlled by K value in Eq. (3.2.6) that is proportional to W_{max}, and connections with larger W_{max} increase more slowly (reduce more), makes CUBIC ensure intra-protocol fairness among competing flows of the same protocol [32]. After the window reduction, regular FRet and FRec. is performed and then enters CA in which CUBIC starts a window increase using the concave profile by setting its plateau to W_{max} so that the concave growth is maintained until the window value reach W_{max} after which the window growth turns to a convex profile [31, 32]. The concave and then convex window growth style of CUBIC enhances the protocol and network stability at the same time maintaining high network utilisation, because the window size remains nearly constant, creating a plateau around W_{max} where network utilisation is considered to highest and under steady state [31, 32]. Most window sizes of CUBIC are near the W_{max}, which promote high network utilisation and protocol stability. However, protocols with convex growth functions like CUBIC, tend to have the largest window increase around the saturation point that lead to large burst of packet losses [32].

Moreover, the window increment per second was fixed to be at most S_{max} to improve stability and fairness. This attribute also keeps the growth rate linear when far larger than W_{max} and makes it much in harmony with BIC's additive window increment per RTT as it becomes larger than some constant [32]. Although the linear increment per RTT is smaller in CUBIC, it remains constant in real time, making sure the CUBIC's linear window is real-time dependent to enhance its TCP friendliness, particularly for short RTT connections, which makes standard TCP less friendly and more aggressive [32].

CUBIC's window growth is slower than standard TCP in short RTT channels and emulate the TCP window adjustment after a loss event to maintain equal growth rate as that of standard TCP [20]. The average sending rate, Φ of an Additive Increase Multiplicative Decrease (AIMD) protocol is given by (3.2.7).

$$\Phi_{AIMD} = \frac{1}{RTT}\sqrt{\frac{\alpha}{2P}\frac{(1+\beta)}{(1-\beta)}} \tag{3.2.7}$$

$$\alpha = 3\frac{(1-\beta)}{(1+\beta)} \tag{3.2.8}$$

where P is packet loss rate, α as given by (3.2.8) is the TCP-fair additive increment per RTT since the CUBIC window decreases by factor of β after a packet loss event. For standard TCP with $\alpha = 1$ and $\beta = 1/2$ the average transmission (sending) rate is given by (3.2.9).

$$\Phi_{TCP} = \frac{1}{RTT}\sqrt{\frac{3}{2P}} \tag{3.2.9}$$

The same results for TCP in (3.2.9) could also be derived by using α value (3.2.8) in (3.2.7) with random β [32]. The window size of emulated TCP at time t after the last epoch (elapsed) is given by (3.2.10) [32].

$$W_{TCP} = \beta W_{max} + 3\frac{(1-\beta)}{(1+\beta)}\frac{t}{RTT} \tag{3.2.10}$$

When the ACK is received during the CA phase, the CUBIC algorithm calculates the window growth rate $W^C(t)$ during the next RTT using Eq. (3.2.5) by setting $W^C(t + RTT)$ as the candidate *target* value of CWND, where RTT is the weighted average computed by standard TCP algorithm. Based on the value of the current CWND size $W_{current}$, CUBIC is implemented in three different mode regions. The three regions (described below) of the window growth in CUBIC algorithm implementation are; TCP, concave and convex regions of the cubic function.

1. **TCP-Friendly Region:** Upon receiving an ACK during CA phase, CUBIC first checks whether the protocol is within the TCP region or not using the procedure described earlier and details contained in [31, 32, 34]. The standard TCP window size in terms of elapsed time t, $W_{TCP}(t)$ can be analysed [31], and the average window size of AIMD is determined with an Additive Increase

(AI) factor α in (3.2.8) and Multiplicative Decrease (MD) $\beta = 0.5$ of the standard TCP leading to (3.2.9). Now, to achieve the same average sending rate as standard TCP using a random β, α must be equal to $3\frac{(1-\beta)}{(1+\beta)}$, thus when $\beta = 0.5$, we get $\alpha = 1$, which are the same AIMD($\alpha_{aimd}, \beta_{aimd}$) factors in standard TCP AIMD(1, 0.5). Therefore, the window size of emulated TCP at time t after the last epoch is obtained using Eq. (3.2.10) [31, 32, 34].

When the CWND W^C given by Eq. (3.2.5) is less than W_{TCP} given by Eq. (3.2.10) ($W^C < W_{TCP}$), then window size is set to W_{TCP} and CUBIC is in TCP mode (i.e $W_{TCP} \leq W_{max}$), and W^C is set as the current $CWND$ size on each successful ACK received [32].

Based on other analysis [32], CUBIC is TCP-friendly when the congestion elapsed (epoch) time, t and Packet Loss Rate (PLR) are:

$$t < \frac{1}{\sqrt{CRTT}} \tag{3.2.11}$$

$$PLR > 0.36CRTT^3 \tag{3.2.12}$$

Thus, when C = 0.4 and RTT = 100 ms [32], if PLR ¿ 0.000144 CUBIC is TCP friendly with larger friendly region compared to High Speed TCP (HSTCP) with PLR ¿ 0.001. It is also more TCP friendly than HSTCP regardless of PLR when the RTT is extremely small [32].

2. **Concave Region:** After receiving an ACK in CA phase, if the check made above found that the protocol is not in the TCP mode ($W^C < W_{max}$ and then CUBIC is said to be in the concave region as shown in Fig. 3.4. In the concave region, CWND must be increased by $\frac{W^C(t+RTT)-CWND}{CWND}$ for each ACK received and the new *target*, $W^C(t + RTT)$ is computed using (3.2.5) [32, 34].

3. **Convex Region:** CUBIC enters a convex region when its window size is greater than the previous saturation point W_{max} and beyond the plateau, i.e $W^C > W_{max}$. This shows that the network conditions might have been upset since the last loss event, probably indicating more available bandwidth after some connection quitting. This is due to highly asynchronous nature of the Internet where fluctuations in available bandwidth exist at all times. The

convex region is also referred to as *maximum probing* phase where CUBIC searches for new W_{max}. The convex profile ensures the CWND is incremented very slowly at the start, then gradually increases its growth rate. The window growth function for the convex region is the same as that of concave region, which is outside the TCP mode and is incremented by the same value of $\frac{W^C(t+RTT)-CWND}{CWND}$ [31, 34].

The CUBIC congestion control algorithm is triggered when packet loss event is detected using the standard TCP implicit or explicit feedback with FRet. and FRec. mechanisms by setting $\beta = 0.7$, CWND $= W_{max}$, $ssthresh(\gamma) = \beta W_{max}$, $\gamma = max(\gamma, 2MSS)$ and new CWND $W^C = \beta W_{max}$ [33, 34]. When a packet loss event occurs, CUBIC decreases its CWND by a factor of β, which should not be smaller than 0.5, that could lead to slower convergence. Although higher and more adaptive values (i.e $\beta > 0.5$), which result in high convergence can also make protocol analysis harder and impact its stability [32, 34]. Adaptive adjustment of β is an issue under research, but values like 0.2 [32], 0.7 [33, 34] and 0.8 [32] are used to evaluate and analyse the protocol with the latest recommended value of $\beta = 0.7$ [33, 34]. A heuristic approach was added to CUBIC in order to improve the fast convergence speed when the network adds new TCP connections. Existing connections are required to release their bandwidth share to allow new connections room for growth, this release of bandwidth by existing flows is increased using the fast convergence mechanism in CUBIC [32, 34]. In the event of loss with fast convergence, CUBIC stores the value of the last maximum window as W_{last_max} before the window reduction and updates to W_{max} for the current loss event. When a loss event occurs and the two windows values are registered by CUBIC, if the updated $W_{max} < W_{last\_max}$, this means the saturation point experienced by the connection decreased due to the change in the network available bandwidth, thus, allowing this flow to release more bandwidth by decreasing W_{max} further by setting $W_{last\_max} = W_{max}$ and $W_{max} = (1+\beta)W_{max}$ [33, 34].

3.2.3 TCP HYBLA

Long RTT channels such as satellites or heterogeneous network environments with at least one satellite leg are severely disadvantage and under perform compared to their short RTT links competing on the same network connection with TCP/IP Internet traffic [35–38]. This performance degradation in a long RTT network environment is a direct consequence of the intrinsic behaviour of the original TCP algorithm, designed to be a window-based transmission algorithm that relied on the ACK reception at the sender side. The arrival of an ACK from the TCP receiver to the sender depends on the network RTT in which the network environment with long RTT is penalised with reduced CWND growth $W(t)$ that leads to significant achievable throughput degradation and unfair sharing of the achievable bandwidth (RTT-unfairness) among the competing flows sharing the same bottleneck. This performance disparity can be resolved with proper modifications of the standard TCP algorithm [35]. Therefore, a new TCP modification proposal, called HYBLA was designed to cope with these problems and optimise performance over long RTT connections by modifying only the end point without undermining the E2E semantics of standard TCP at the same time removing the dependence of TCP performance on RTT [35]. This serves as an effective mechanism for both high losses due to congestion on links and RTT channels through adopting SACK, use of timestamps and modified CWND increment rule of standard TCP [35, 36]. The additional advantage was the implementation of packet spacing techniques, which remove the bursty transmissions to reduce the probability of buffer overflow at intermediate hops [35].

Performance of Long RTT connections like satellite channels especially GEO, suffer severely by achieving a low instantaneous transmission rate, R(t) even when the CWND evolution, W(t) in time remain constant as given in (3.1.1) and (3.1.3) using standard TCP algorithm, which also experienced longer/slower time scale CWND increase W(t). To make $R(t)$ independent of RTT, HYBLA overcompensates the CWND evolutions, W(t) leading to R(t) dependence on RTT values [35]. Like CUBIC, HYBLA has good TCP-friendliness and RTT-fairness properties with compatibility to other promising TCP enhancement without changing E2E semantics of the standard TCP [35].

Moreover, HYBLA retained the main features of TCP NewReno such as loss recovery phase with necessary implementation of a few additional features [14, 35]. The new proposals in HYBLA included modification of congestion control algorithm based on analytical study of congestion window evolution or growth rate, adoption of SACK option, use of timestamps against inappropriate timeouts and packet spacing against packet burst to minimise the impact of multiple losses [35]. This implementation, substantially alleviates the performance disparity in satellite and heterogeneous networks, especially disparity against satellite and wireless links exhibiting long RTT and high link error rate. This also presents another advantage over standard TCP variants in the presence of congestion and link errors [35]. The modified mathematical model for congestion window growth rate in HYBLA is given by (3.2.13) [35].

$$W^H(t) = \begin{cases} \rho 2^{(\frac{\rho t}{RTT})} & 0 \leq t < t_\gamma \quad SS\ (Exponent) \\ \rho(\frac{t-t_\gamma}{RTT} + \gamma) & t \geq t_\gamma \qquad CA\ (Linear) \end{cases} \tag{3.2.13}$$

where ρ is the normalised RTT for the longer path relative/comparative to fast reference (e.g wired) $RTT_0 = 25$ ms TCP connection given by (3.2.14).

$$\rho = \frac{RTT}{RTT_0} \tag{3.2.14}$$

When $\rho \leq 1$ for faster connections making $RTT \leq RTT_0$, TCP HYBLA behaves like standard TCP [35] as could be deduced also from (3.2.13) and (3.2.16) especially when $\rho = 1$. The time at which the *ssthresh* value $\gamma = 32$ (in HYBLA proposal, but is relatively low value for real links) is reached given by (3.2.15).

$$t_\gamma = RTT \log_2(\gamma) \tag{3.2.15}$$

The expression in (3.2.15) indicates that high RTT values will reach the value of γ in a longer time which results in lower congestion window increase rate $W(t)$ in standard TCP algorithm implementation [35]. Now, based on the standard TCP congestion window update rule in Eq. (3.1.2), HYBLA modification has been rewritten in Eq. (3.2.16).

$$W^H_{i+1} = \begin{cases} W^H_i + 2^\rho - 1 & SS \\ W^H_i + \frac{\rho^2}{W^H_i} & CA \end{cases} \tag{3.2.16}$$

The segment transmission rate (Instantaneous Transmission Rate) for HYBLA is determined from the standard TCP and given by (3.2.18).

$$R^H(t) = \frac{W^S(t)}{RTT} = \frac{W^H(t)}{RTT_0} \tag{3.2.17}$$

Therefore, the final objective of HYBLA to achieve maximum data transmission rate (achievable Throughput) independent of long RTT of channels such as satellite becomes;

$$R^H(t) = \begin{cases} \frac{2^{(t/RTT_0)}}{RTT_0} & 0 \leq t < t_\gamma \quad SS \\ \frac{1}{RTT_0}(\frac{t-t_\gamma}{RTT_0} + \gamma) & t \geq t_\gamma \quad CA \end{cases} \tag{3.2.18}$$

The amount of segments (data) transmitted from the beginning of the HYBLA connection could be determined from (3.2.18) and given in analytical form as:

$$T_S^H(t) = \int_0^t R^H(\tau)d\tau = \begin{cases} \frac{2^{(t/RTT_0)}-1}{\ln(2)} & 0 \leq t < t_\gamma \quad SS \\ \frac{\gamma-1}{\ln(2)} + \frac{(t-t_\gamma)^2}{2RTT_0^2} + \frac{\gamma(t-t_\gamma)}{RTT_0} & t \geq t_\gamma \quad CA \end{cases} \tag{3.2.19}$$

The main goal of HYBLA was to provide a solution to performance degradation due the RTT disparity by modification of the standard CWND evolution $W(t)$ as in (3.2.16) from an analytical study of the CWND dynamics. The key enhancement in HYBLA is the extension of the constant-rate additive increase (AI) policy, it also benefits from adoption of a SACK option and the use of timestamps in the presence of loss events due to congestion or link errors, but this has limited benefit to connections over satellite paths competing with standard TCP flows [35, 36]. However, the key implementation issues of HYBLA are that the topology did not consider the additional RTT of gateway station, absence of wireless link in the network to test heavy error rate link performance in the presence of long RTT of satellite last leg [35].

HYBLA lacks the congestion control mechanism to mitigate packet losses due to high wireless link errors, as a result the algorithm attempts to drastically reduce the congestion window due to its inability to differentiate between link congestion losses and link error losses in high error rate wireless links such as satellite [40]. The window size could grow too aggressive for faster ($RTT_0 \geq 25$ ms) reference connections

3.2.4 Performance Enhancement Proxies

The Performance Enhancement or Enhancing Proxies (PEPs), is one of the mechanisms proposed to enhance the degradation in TCP performance due to the attribute of particular communication channels and network environments such as satellite, wireless and realistic heterogeneous communications network environment. The PEP performance mitigation technique is normally implemented and functions at one or two protocol layers such as application and/or transport layers but, in principle, the PEP implementation may function at any protocol layer and below the network layer, that is, link layer [41]. However, this thesis focuses on transport layer protocol enhancement, thus the review on PEP implementations that operates at the transport layer and interact with TCP, called TCP PEP.

In network environments with large BDP and where ACKs may be bunched together leading to an unwanted data segment burst, TCP PEP modifies the ACK spacing and the behaviour of the TCP flow by producing local ACKs of TCP segments to enhanced the throughput of the flow and to improve the overall performance in network environments such as hybrid ISTN and SatComs networks [41–43]. The TCP spoofing is used synonymously for TCP PEP functionality, but it accurately describes an attribute of intercepting a TCP flow at the middle and terminating the flow pretending to be the intended destination.

PEP Implementations

PEPs are implemented as applications running on communication nodes to enhance the performance of transport layer protocol which degraded due to characteristics of communication links such as satellite. The implementation can be described as Distributed, Integrated, Symmetric, and Asymmetric [41–43].

1. **Integrated and Distributed Implementations:** The integrated implementation of PEP consists of a single PEP component in a single node as shown in Fig. 3.5, usually incorporating one PEP application at the satellite gateway station or user terminal, which represent a single point where performance enhancement is required, a point where satellite and terrestrial link

meet (see Fig. 3.5).

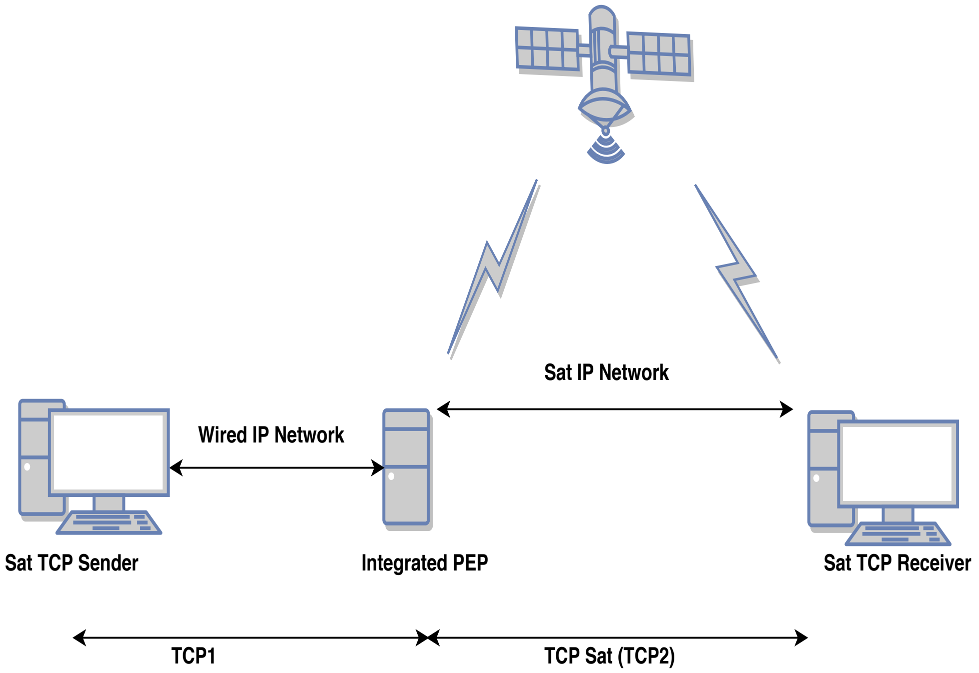


Figure 3.5: Integrated PEP Architecture in ISTN Environment

This provides an impedance matching of the heterogeneous networks. On the other hand, Distributed PEP implementation consists of two or more PEP components or applications running on multiple nodes at the edges (gateway and user terminal) of satellite link on the network as demonstrated in Fig. 3.6, this is the most frequently use PEP implementation. These two implementations normally split TCP (TCP-Splitting) connections as standard and modified PEP connections. For instance, integrated PEP comprises of two TCP connections (see Fig. 3.5) while distributed PEP consists of three TCP connections as in Fig. 3.6 [41–44].

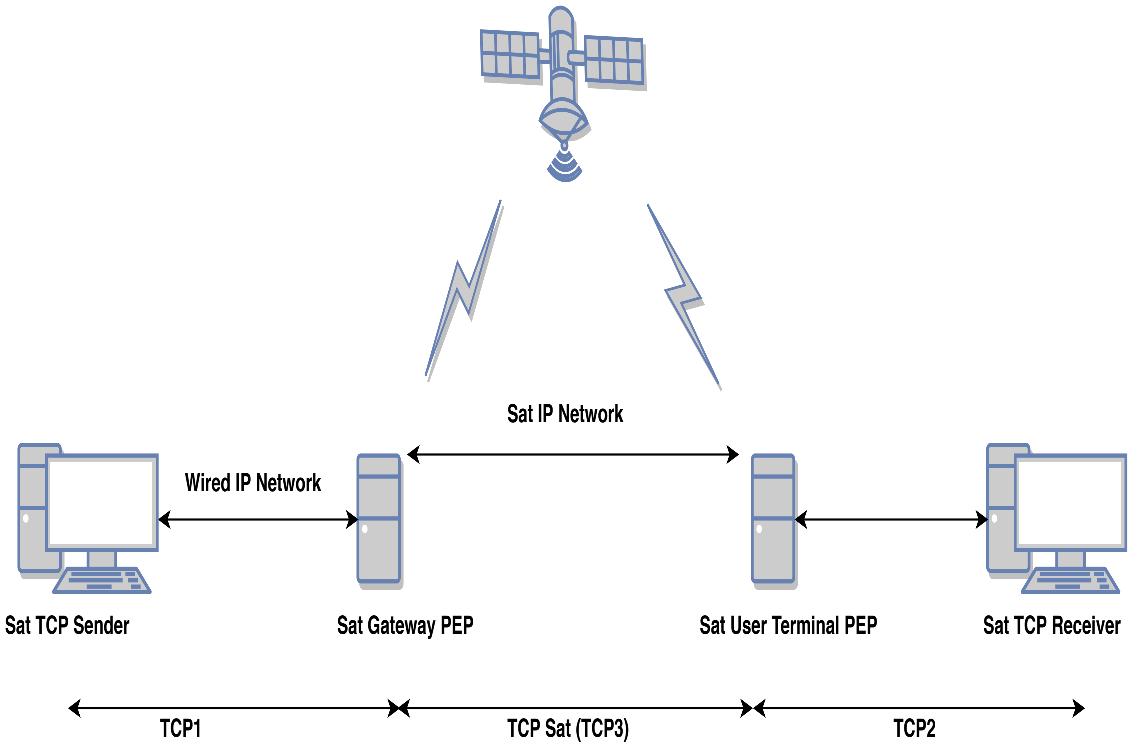


Figure 3.6: Distributed PEP Architecture in ISTN Environment

2. **Symmetric and Asymmetric Implementation:** Apart from being distributed or integrated implementation, PEP may be Symmetric or Asymmetric implementation. Symmetric PEPs employ similar behaviour in both directions in which the actions taken by the PEP is independent of the interface from which segment is received. While the Asymmetric PEP have different behaviour in each direction, its implementation is normally employed at the point where the attributes of the links on each side of the PEP is unique or with asymmetric protocol traffic. The direction can be described in terms of the link such as from central to remote location or in terms of protocol traffic like the direction of TCP data flow (TCP data channel), or the direction of TCP ACK flow (TCP ACK channel). An asymmetric PEP may be located at the intersection of satellite and terrestrial or wired and wireless networks. However, PEP implementation may be both symmetric and asymmetric simultaneously with respect to unique mechanisms it uses, and whether a PEP implementation is symmetric or asymmetric is independent of being integrated

or distributed implementation. For instant, a distributed PEP might function as symmetric at each end of a link with two PEPs identical operations, or asymmetric with different PEP implementations at each end of the link [41].

Moreover, PEP employs TCP splitting and TCP spoofing as the main mechanisms for its implementation. The TCP splitting is employed by PEP to administer each TCP connections for each host via correspondence and buffering, splitting TCP connection allow all kinds of modifications to be deployed on the satellite segment [41, 44]. On the other hand, TCP spoofing involves sending back ACKs for TCP segments, which gives the TCP sender the illusion of a shorter RTT that rapidly increases the amount of data in flight (CWND), this means data buffering at spoofing PEP level. However, TCP splitting is commonly used (see Figs. 3.5 and 3.6) since TCP spoofing is not always sufficient [41, 44]. The PEP implementation using a split TCP connection terminates the TCP connection received from an end system with different TCP connection (e.g standard TCP) and establishes a corresponding TCP connection (e.g satellite TCP like HYBLA or CUBIC) to the other end system. A distributed PEP implementation as shown in Fig. 3.6, allows a third TCP connection between two PEPs optimised for the link. Several integrated PEP implementation also employ TCP splitting mechanism implementation for addressing a mismatch in TCP capabilities between two end systems [41].

However, PEP implementation can effectively leverage TCP performance improvements that are optimum for a specific link like satellites, but cannot necessarily be used safely over the global Internet since the E2E semantics of the widely deployed TCP/IP connections is infringed upon, which may add to complexity of the protocol implementation and recommendation for general use over the Internet [41]. Additionally, breaking E2E semantics of the TCP/IP connections have detrimental negative security implications by disabling the E2E utilisation of IPsec, which a user or network administrator have to trade-off for using PEPs [41].

3.3 User Datagram Protocol

Another important data transport protocol used by the Internet (IP) nowadays is User Datagram Protocol (UDP), which is defined to make available a datagram mode of packet-switched communication in interconnected communication environments. UDP provides a procedure for applications to exchange messages with other programs using minimum protocol mechanisms. Unlike TCP, UDP is transaction oriented, therefore, reliable delivery and duplicate protection for congestion control and loss recovery are not guaranteed. Thus, applications and services needing ordered reliable delivery and congestion/loss control of data streams should not employ UDP [45].

3.3.1 Standard User Datagram Protocol

The Standard User Datagram Protocol (UDP) is the pioneer connectionless protocol designed to maintained message boundaries, with no connection setup or feature negotiation [45, 46]. This protocol employs independent messages referred to as datagrams and offers minimum transport service using non-guaranteed datagram delivery and allows direct access to the datagram service of the IP layer by the applications that do not need the level of reliable and guaranteed service of TCP and applications that require the use of communications service such as multicast or broadcast delivery that may not be offered by TCP [6, 45, 46]. Standard UDP is nearly a null protocol, which provides checksumming of data and multiplexing using port number over the IP. These are the services provided by standard UDP, therefore, an application program running over it must deal directly with E2E communications issues like retransmission for reliable delivery, flow control, congestion avoidance, packetisation and reassembly that a connection-oriented protocol such as TCP would have dealt with [6]. However, fairly complex coupling between IP and TCP will be reflected in the coupling between UDP and most applications utilising it. Well-known ports are used by UDP to exchange datagram and follow the same rule like TCP well-known ports, when datagram with UDP port address arrived and there is no pending LISTEN call, UDP sends an Internet Control Message Protocol

(ICMP) port unreachable message. UDP must pass any IP option received from the IP layer transparently to the application layer, and an application must be able to specify IP options to be sent in its UDP datagrams, which must be pass to the IP layer [6, 45]. The transmission of data in UDP is carried out by encapsulating each datagram into a single IP packet or several IP packet fragments, which allows a datagram to be larger than the effective path Maximum Transmission Unit (MTU), the fragments are reassembled before delivery to the UDP receiver to make it transparent to the user of the transport service [46]. Larger messages may be sent without fragmentation if jumbograms are supported. The standard UDP header format consists of fields of 16-bit each that include, source port, destination port, message length, checksum and data octets or payload as shown in Fig. 3.7 [45].

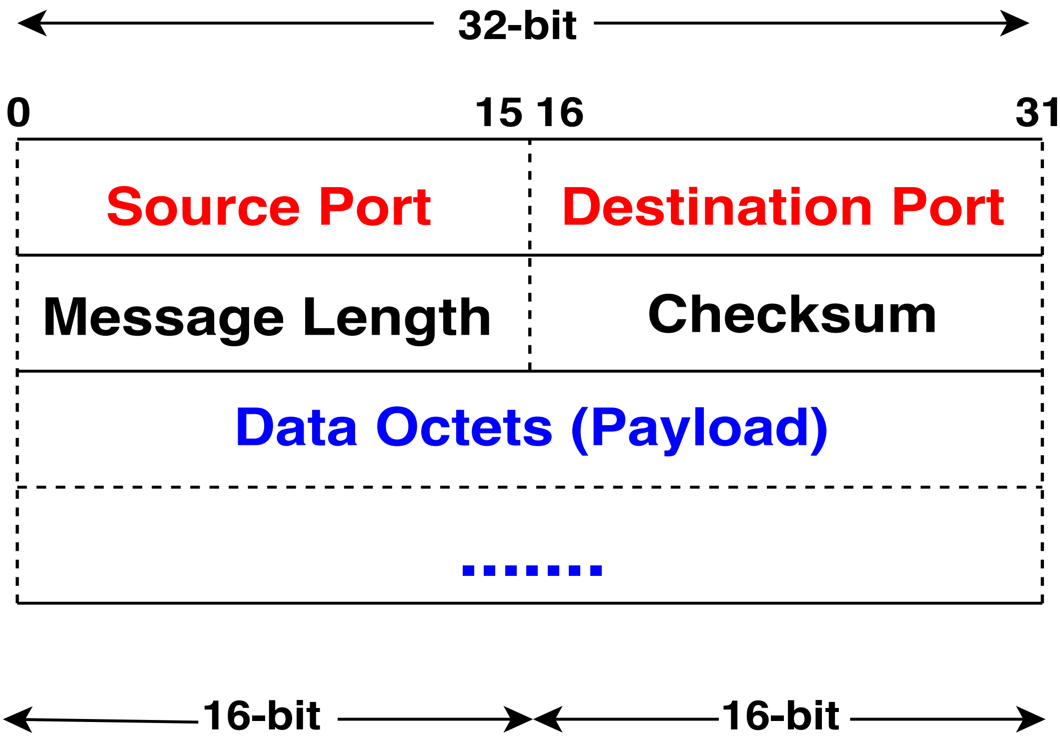


Figure 3.7: Standard UDP Header Format [45, 46]

Standard UDP does not have the capabilities for the providing congestion control, flow control or error correction, but has capabilities for datagrams broadcast, multicast, unicast and any-cast [46]. The protocol is widely utilised by some applications such as Domain Name Services (DNS) and streaming services that do not require guaranteed data delivery and retransmissions of lost data. UDP only detects datagrams/payload errors and delivered packets to an unintended destination, which leads to discard of received datagrams without notifying the end-user (sender) of the service [46]. Lack of flow control in UDP results in missing messages by a receiving application that is not able to run sufficiently fast, or frequently. While the lack of congestion control handling may causes UDP traffic to experience loss when utilising an overloaded path and may also result in loss messages from other protocols like TCP when utilising the same network path [46].

3.3.2 Lightweight User Datagram Protocol

The Lightweight User Datagram Protocol (UDP-Lite or UDPLite) is a new protocol that is identical to the standard UDP [45, 46], but useful for the application programs in error-prone network environments in which partially damaged data payloads are preferably been delivered than discarded by the network. When partially damaged data payloads is preferred to be delivered rather than discarded as in standard UDP, then UDP-Lite is semantically similar to UDP [49]. The UDP-Lite is based on three observations about the behaviour of standard UDP as follows [49]:

1. Certain classes of applications like audio and video *codecs* benefit from having damaged data payloads being delivered rather than discarded by the network. These codecs (voice and video) applications include speech codec, the Internet Low Bit Rate Codec (ILBRC), error resilient codecs (H.263+ and H.264) and MPEG-4 video codecs. These codecs application programs were designed to better handle errors in the data payload than loose the entire packets [47–49].

2. Data transportation links utilising IP employs a strong link layer integrity or error checking like Cyclic Redundancy Check 32 *(CRC-32)*, which must be used by default by any IP data traffic [49]. Traffic using UDP-Lite may benefit from a unique link behaviour that allows partially damaged IP packets to be forwarded when requested and the under-lying support through link layer error checking [49]. Many radio technologies support this unique behaviour when operating at a point where cost and delay are sufficiently low, and when error-

prone links are aware of error sensitive part of the packet, the physical link has a chance of providing high protection that reduce the likelihood of error sensitive bytes corruption by employing unequal Forward Error Correction (FEC) [49].

3. The intermediate layers like transport and IP layers should not impede error-tolerant applications from flowing well in the presence of error-prone links. IP layer header has no checksum that blankets the IP data payload, thus IP layer protocol is not the issue here, and the normally available transport layer protocol best fit for these kinds of applications is UDP, which require no overhead for retransmissions of erroneous/loss packets, in-order delivery, or error correction [49].

A more efficient transport protocol is required [48], that matches the properties of link layers and applications as mentioned, and the error-control mechanisms of the transport layer must provide protection to significant information like headers, but optionally ignore errors that are best handled by the applications. The key difference between standard UDP and UDP-Lite, is the *checksum* with optional partial coverage (Checksum Coverage filed) shown in Fig. 3.8. This replaces the UDP's message length field (see Fig. 3.7), and the sender divides the packet into a sensitive part covered by the checksum, and an insensitive part that is not covered by the checksum [48–50].

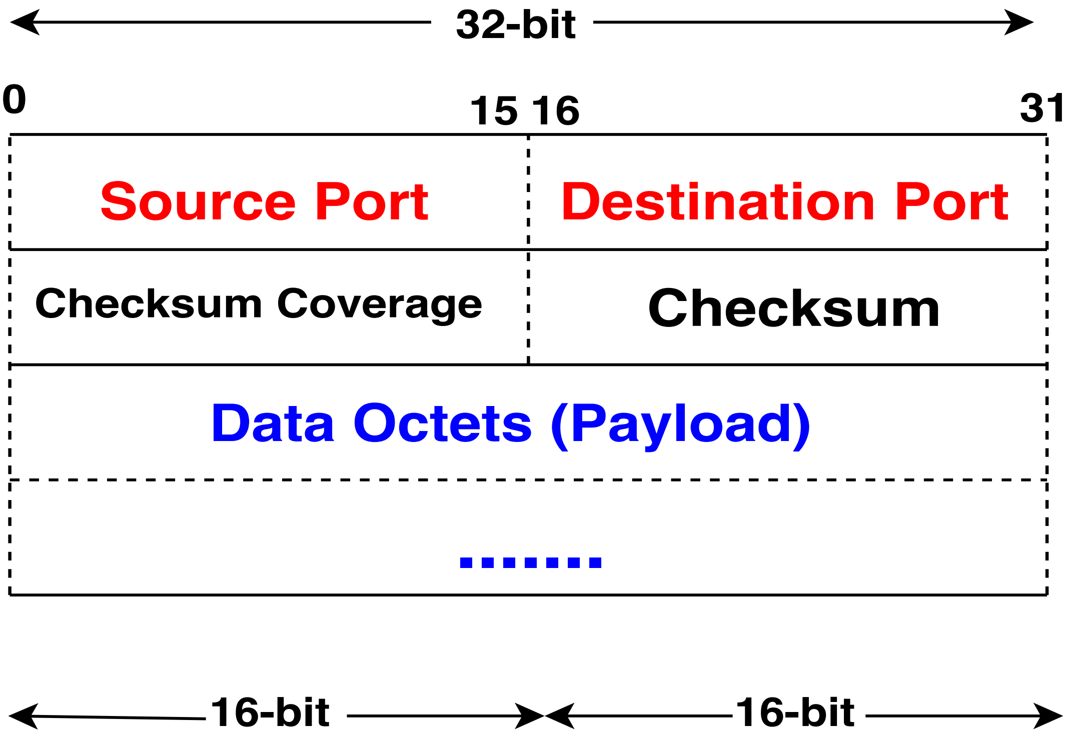


Figure 3.8: UDP-Lite Header Format [47, 49]

When UDP-Lite option is used, errors in the insensitive part of the packet will not result in the packet being discarded by the transport layer at the receiver side, while errors in the sensitive part (entire packet covered by the checksum) normally as default lead to UDP-Lite that is semantically similar to standard UDP [49]. Therefore, UDP-Lite sender's partial checksum provides additional flexibility compared to standard UDP [47–49], particularly for applications that want to classify the data payload as *partially insensitive* to bit errors [49].

However, both UDP-Lite and standard UDP are unreliable transport protocols, utilising the same set of port numbers assigned by the Internet Assigned Numbers Association (IANA) for use by the UDP, with no congestion and flow control mechanisms available. Thus, any application that require reliable E2E connection-oriented transport protocol still need to employ TCP.

The standard TCP schemes such as Tahoe, Reno and NewReno discussed in this chapter are heavily dependent on RTT and designed to assume packet losses are due to the congestion on the link. These dependence on the RTT and attributing all loss event are due the link becoming congested lead to performance degradation and disparity when using standard TCP over heterogeneous networks. Heterogeneous networks incorporate satellites and wireless links characterised with high latency (RTT), high capacity/bandwidth and high wireless link errors that can affect the performance of standard TCP in terms of capacity utilisation, fairness, scalability, and stability.

TCP schemes such as BIC, CUBIC, and HYBLA were proposed to mitigate the performance disparity due to the high RTT and/or high bandwidth network links such as satellites. However, each of these schemes considered the performance improvement of large BDP networks based on either high RTT or high capacity networks. Pure satellite and ISTN links have characteristics of high RTT, high capacity, and high link errors.

This thesis studied and investigated how the congestion control algorithms of the heterogeneous TCP schemes such as CUBIC and HYBLA can be improved through integration of their best features for better performance in terms of capacity utilisation, packet delivery rate, and fairness in a realistic heterogeneous network environment such as ISTN with both high RTT and high capacity leading to a larger BDP network path. The design, implementation, testing, and results obtained from the enhanced algorithm proposal is discussed in the subsequent chapters of this thesis. This improved algorithm is aimed at optimum and efficient utilisation of large BDP heterogeneous network environment incorporating at least a GEO satellite link and ISTN of the future networks such as Satellite and Terrestrial networks for 5G (SaT5G).

References

[1] J. Postel, "Transmission Control Protocol", *Defense Advanced Research Projects Agency (DARPA) Internet Program Protocol Specification , RFC 793*, DoD, 1981.

[2] J. Postel, "Internet Protocol", *Defence Advanced Research Projects Agency (DARPA) Internet Program Protocol Specification , RFC 791*, DoD, 1981.

[3] D. E. Comer, *Internetworking with TCP/IP: Principles, Protocol, and Architecture*, 5th Ed., Vol. 1, Pearson Prentice Hall, New Jersey, USA, 2006.

[4] V. Cerf, and R. Kahn, "A Protocol for Packet Network Intercommunication", *IEEE Transactions on Communications*, Vol. COM-22, No. 5, pp 637-648, 1974.

[5] V. Jacobson, "Congestion Avoidance and Control", *In Proc. of ACM SIGCOMM and Computer Communication Review*, Vol. 18, no. 4, pp. 314-329, Stanford, CA 1988.

[6] R. Braden (Editor), "Requirements for Internet Hosts–Communication Layers," *The Internet Engineering Task Force (IETF)*,RFC 1122, 1989.

[7] V. Paxson, and M. Allman, "Computing TCP's Retransmission Timer," *The Internet Society Network Working Group, Standards Track*,RFC 2988, 2000.

[8] V. Paxson, M. Allman, J. Chu, and M. Sargent, "Computing TCP's Retransmission Timer," *The Internet Engineering Task Force (IETF), Standards Track*,RFC 6298, 2011.

[9] W. Stevens "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms," *Network Working Group, Standard Track*,RFC 2001, 1997.

[10] M. Marchese, *QoS Over Heterogeneous Networks*, John Wiley & Sons, England, 2007.

[11] C-Y. Ho, Y. Chen, Y. Chan, and C. Ho, " Fast Retransmit and Fast Recovery Schemes of Transport Protocols: A Survey and Taxonomy", *Elsevier Computer Networks*, Vol. 52, pp. 1308-1327, 2008.

[12] K. Ramakrishnan, S. Floyd and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", *Network Working Group, Standard Track*, RFC 3168, 2001.

[13] M. Mathis, J. Mahdavi, S. Floyd and A. Romanow, "TCP Selective Acknowledgement Options", *Network Working Group, Standard Track*, RFC 2018, 1996.

[14] S. Floyd, T. Henderson, "The NewReno Modification to TCP's Fast Recovery Algorithm", *Network Working Group, Experimental*, RFC 2582, 1999.

[15] S. Floyd, T. Henderson, and A. Gurtov "The NewReno Modification to TCP's Fast Recovery Algorithm", *Network Working Group, Standards Track*, RFC 3782, 2004.

[16] T. Henderson, S. Floyd, A. Gurtov and Y. Nishida "The NewReno Modification to TCP's Fast Recovery Algorithm", *Internet Engineering Task Force (IETF), Standards Track*, RFC 6582, 2012.

[17] V. Jacobson, R. Braden, and D. Borman, "TCP Extension for High Performance," *Internet Society Network Working Group,*,RFC 1323, 1992.

[18] V. Jacobson, and R. Braden, "TCP Extensions for Long-Delay Paths", *Network Working Group*, RFC 1072, 1988.

[19] T. Braun, M. Diaz, J. Enriquez-Gabeiras, and T. Staub, *End-to-End Quality of Service Over Heterogeneous Networks*, Springer, Berlin, 2010.

[20] J. S. Stadler, and J. Gelman, ”Performance Enhancement for TCP/IP on a Satellite channel”, *In Proc. of IEEE Military Communications Conference (MILCOM)*, Vol. 1, pp. 270-276, 1999.

[21] N. Ghani, and S. Dixit ”TCP/IP Enhancement for Satellite Networks”, *IEEE Communications Magazine*, pp. 64-72, 1999.

[22] M. Allman, D. Glover, and L. Sanchez, ”Enhancing TCP Over Satellite Channels Using Standard Mechanisms”, *The Internet Society/IETF*, RFC 2488 , 1999.

[23] M. Allman, et al., ”Ongoing TCP Research Related to Satellites,” *Internet Society Network Working Group, RFC*, 2760, BCP. 28 2000.

[24] S. Kota, and M. Marchese, ”Quality of Service for Satellite IP Networks: A Survey”, *International Journal of Satellite Communications and Networking*, Vol. 21, pp. 303-349, John Wiley, 2003.

[25] D. J. Bem, T. W. Wieckowski and R. J. Zielinski, ”Broadband Satellite Systems”, *IEEE Communications Surveys & Tutorials*, Vol. 3, no. 1 pp. 2-15, 2000.

[26] J. Farserotu, and R. Prasad, ” A Survey of Future Broadband Multimedia Satellite Systems, Issues and Trends”, *IEEE Communications Magazine, Broadband Direct to Home/User Wireless Systems*, pp. 128-133, 2000.

[27] ITU, ” Transmission Control Protocol (TCP) Over Satellite Networks”, *International Telecommunication Union (ITU), Report*, S Series Rep,. ITU-R S.2148, pp. 1-24, 2009.

[28] ITU, ” Performance Enhancements of Transmission Control Protocol Over Satellite Networks”, *International Telecommunication Union (ITU), Recommendations*, S Series Rec., ITU-R S.1711-1, pp. 1-50, 2010.

[29] L. Xu, K. Harfoush, and I. Rhee, ”Binary Increase Congestion Control (BIC) for Fast Long-Distance Networks”, *IEEE INFOCOM*, Vol. 4, 2514-2524, 2004.

[30] A. Pirovano, and F. Garcia ”A New Survey on Improving TCP Performances over Geostationary Satellite Link”, *Network and Communication Technologies*, Vol. 2, No. 1, pp. 1-18, Canadian Center of Science and Education, 2013.

[31] I. Rhee, and L. Xu, ”CUBIC: A New TCP-Friendly High-Speed TCP Variant”, *In Proc. Protocols for Fast Long-Distance Networks (FLDN) Workshop*, Feb., 2005.

[32] S. Ha, I. Rhee, and L. Xu, ”CUBIC: A New TCP-Friendly High-Speed TCP Variant”, *ACM SIGOPS Operating Systems Review - Research and Developments in the Linux Kernel*, Volume 42 Issue 5, pp. 64-74, 2008.

[33] I. Rhee, *et al.*, ”CUBIC for Fast Long-Distance Networks”, *The IETF TCP Maintenance and Minor Extensions (TCPM) Working Group*, Internet-Draft , 2017.

[34] I. Rhee, *et al.*, ”CUBIC for Fast Long-Distance Networks”, *Internet Engineering Task Force*, RFC 8312, 2018.

[35] C. caini, and R. Firrincieli, ”TCP HYBLA: A TCP Enhancement for Heterogeneous Networks”, *International Journal of Satellite Communications and Networking (IJSCN)*, Vol. 22, pp. 547-566, John Wiley, 2004.

[36] C. caini, and R. Firrincieli, ”End-to-End TCP Enhancements Performance on Satellite Links”, *In Proc. 11th IEEE Symposium on Computers and Communications (ISCC’06)*, 1031-1036, 2006.

[37] C. caini, R. Firrincieli, D. Lacamera, T. de Cola, M. Marchese, C. Marcondes, M. Y. Sanadidi, M. Gerla, ”Analysis of TCP Live Experiments on a Real GEO Satellite Testbed”, *Performance Evaluation*, Vol. 66, Issue 6, pp. 287-300, Elsevier, 2009.

[38] S. Trivedi, S. Jaiswal, R. Kumar, and R. Rao, ”Comparative Performance Evaluation of TCP HYBLA and TCP Cubic for Satellite Communication Under Low Error Conditions”, *4th IEEE International Conference on Internet Multimedia*

Services Architecture and Application (IMSAA), Bangalore, India, 15-17 Dec. 2010.

[39] C. caini, T. Firrincieli, and D. Lacamera, "Comparative Performance Evaluation of TCP variants on Satellite Environments", *In Proc. of IEEE International Conference on Communications (ICC)*, 2009.

[40] M. Park, M. Shin, D. Oh, B. Kim, and J. Lee, "TCP HYBLA+: Making TCP More Robust Against Packet Loss in Satellite Networks', *Proceedings of International Conference on Computational Science and its Applications (ICCSA), Spain, Jun 20-23*pp. 424-435, Springer, 2011.

[41] J. Border, *et al.*, "Performance Enhancing Proxies Intended to Mitigate Link-Related Degradation", *The Internet Society, Network Working Group* RFC 3135 , 2001.

[42] E. Dubois, J. Fasson, C. Donny, and E. Chaput, "Enhancing TCP Based Communications in Mobile Satellite Scenarios: TCP PEPs Issues and Solutions", *In Proc. IEEE 5th Advanced Satellite Multimedia Systems Conference and 11th Signal Processing for Space Communications Workshop*, 2010.

[43] P. E. Finch, D. V. Sullivan, and W. D. Ivancic, "An Evaluation of Protocol Enhancing Proxies and Modern File Transport Protocols for Geostationary Satellite Communication", *In Proc. IEEE Aerospace Conference*, 2010.

[44] Y. Kim, H. Park, J. Kim and Y. Choi, "Fairness PEP Solution for Satellite TCP", *In Proc. IEEE 7th International Conference on Ubiquitous and Future Networks*, pp. 83-85, 2015.

[45] J. Postel, "User Datagram Protocol", *Defense Advanced Research Projects Agency (DARPA) Internet Program Protocol Specification , RFC 768*, DoD, 1980.

[46] G. Fairhurst, B. Trammell, and M. Kuehlewind, "Services Provided by IETF Transport Protocols and Congestion Control Mechanisms", *Internet Engineering Task Force (IETF)*, RFC 8095, 2017.

[47] L-A. Larzon, M. Degermark, and S. Pink, "UDP Lite for Real Time Multimedia Applications", *Hewlett Parkard*, 1999.

[48] L-A. Larzon, M. Degermark, and S. Pink, "Efficient Use of Wireless Bandwidth for Multimedia Applications", *In Proc. of IEEE International Workshop on Mobile Multimedia Communications (MoMuC)*, pp. 187-193, 1999.

[49] L-A. Larzon, M. Degermark, S. Pink, L. Jonsson, and G. Fairhurst,, "The Lightweight User Datagram Protocol (UDP-Lite)", *Internet Society Network Working Group*, RFC 3828, 2004.

[50] W. Stanislaus, G. Fairhurst, and J. Radzik, "Cross Layer Techniques for Flexible Transport Protocol Using UDP-Lite over Satellite Network", *Internet Society Network Working Group*, RFC 3828, 2004.

Chapter 4

Experiments: Satellite Channel Measurements and Modelling

This chapter presents the channel measurements, numerical simulations and emulation of latency in pure satellite and hybrid satellite-terrestrial channels, followed by performance evaluation of the impact of latency on communications. The chapter also discusses the real-world practical and emulation scenarios used to measure the performance of ISTN environments and the scenarios developed for the channel measurement experiments. Our developed framework for End-to-End (E2E) latency measurement in relation to performance of realistic heterogeneous network (ISTN) environments incorporating GEO satellite channels is presented and discussed.

4.1 Theoretical Model of Latency

One of the most significant performance parameters used for measuring the quality of communication networks is the average latency (delay) [1], for delivering messages from the transmitter (source) to the receiver (sink) on the network. This parameter strongly influences the choice and performance of communications network algorithms such as congestion control, flow control and routing [2–4]. Therefore, it is vital to measure, study, and understand the nature and mechanism of communications latency, particularly when using TCP over satellite channels to deliver internet data. Internet Protocol (IP) based communications networks employing TCP dominate all kinds of today's communication services and are envisaged to form the basis of all future communication services and applications such as data transfer, voice telephony, television, Virtual Reality (VR), Augmented Reality (AR), etc. Internet end users expect some level of quality of services and experience. Therefore, there is a need for Quality of Service (QoS) and Quality of Experience (QoE) support for IP networks, especially at the transport layer [2–4]. The measure of the ability of a communication network and computing systems to provide unique levels of services to specific applications and associated network flows is called the QoS while the quality experienced by the end user is the QoE [5, 6]. Although, communication network performance can be measured by several parameters or metrics called QoS parameters (metrics) like latency, jitter, bandwidth and packet loss rate, the key and most significant of them is latency (delay) as it has direct or indirect impacts on the other performance and QoS metrics (see Fig. 4.1), especially at the transport layer level of the TCP/IP internet model [2–4]. The following sections and subsections described the framework, scenarios, models, results of the latency measurement experiments and performance evaluation and analysis of the impacts of latency measured on the QoS and on the performance of data communication under realistic network environments.

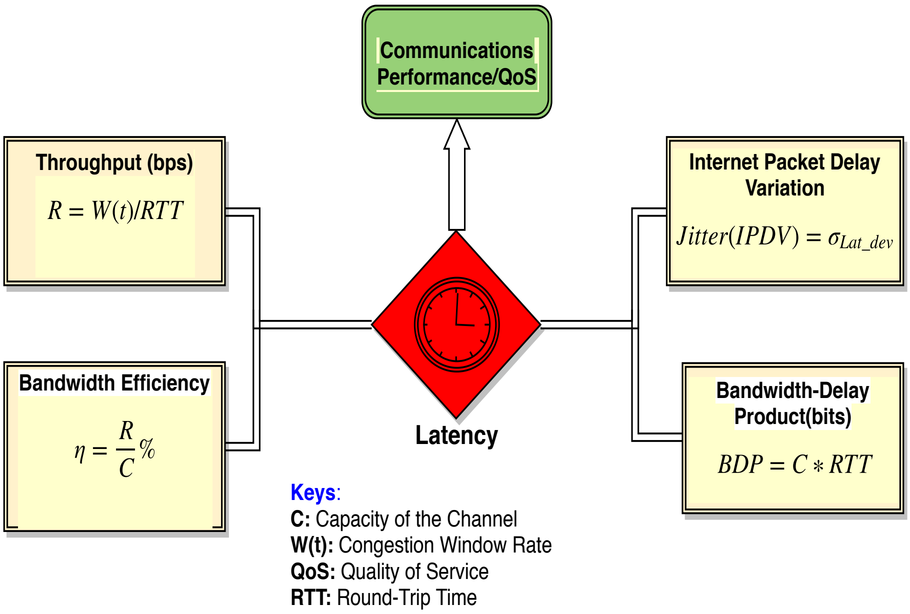

Communications Performance/QoS
Throughput (bps)
$R = W(t)/RTT$
Internet Packet Delay Variation
$Jitter(IPDV) = \sigma_{Lat\_dev}$
Bandwidth Efficiency
$\eta = \frac{R}{C}\%$
Latency
Bandwidth-Delay Product(bits)
$BDP = C * RTT$
Keys:
C: Capacity of the Channel
W(t): Congestion Window Rate
QoS: Quality of Service
RTT: Round-Trip Time

Figure 4.1: Latency Dependent Performance and QoS Parameters

4.1.1 Satellite and User Terminal Locations Modelling

The location of ground segments (Satellite User Terminal SUT, and Gateway Station GWS) with respect to the Satellite Space Segment (SSS) contributed to the changes in propagation and E2E latency. The sources of these variabilities were investigated analytically and discussed in this section. Changes in the distance from the user terminal to a satellite to gateway station might change the propagation latency components were evaluated analytically with a user terminal location in Europe (Durham, England) and Africa (Zamfara, Nigeria). The graphical representation of the space and ground segments of the satellite communication network is shown in Fig. 4.2. Analytical expressions were derived by careful analysis of the earth to satellite distance. The distance D_{PS} (in km), from the ground terminal or gateway, located at a particular point P on earth, to the satellite, located somewhere at a point S in space relative to the centre of the earth with radius R_E and sub-satellite point distance, the satellite altitude from the centre of the earth, h_S (in km) to the satellite is determined by Eq. (4.1.1) [7].

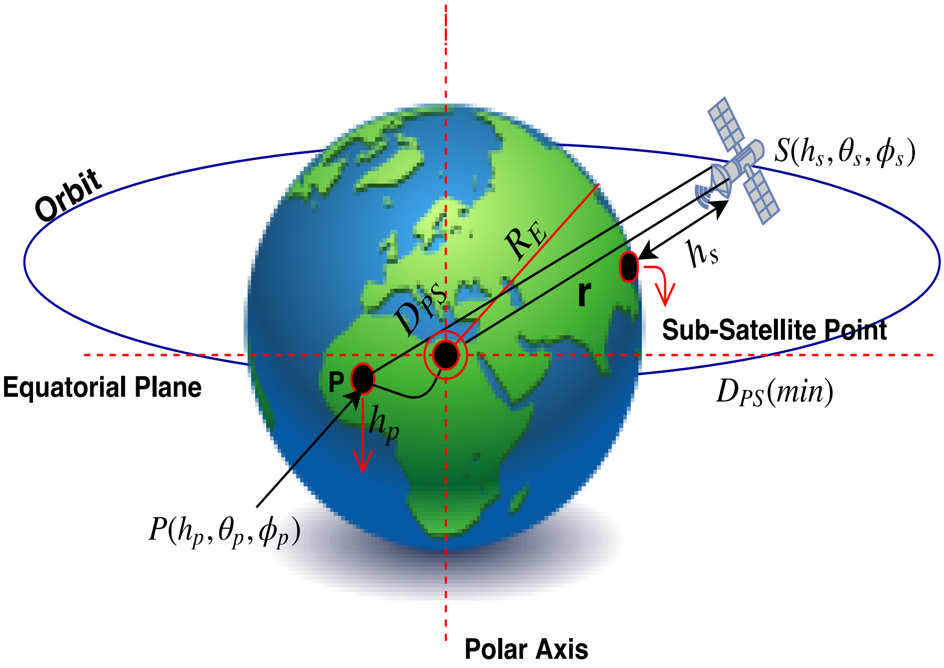


Figure 4.2: Earth to Satellite Distance Geometry

$$D_{TS} = \sqrt{R_E^2 + r^2 - 2R_E r \cos(\phi_P - \phi_S)\cos(\theta_S)\cos(\theta_P) + \sin(\theta_S)\sin(\theta_P)} \quad (4.1.1)$$

where D_{TS} is the link distance from ground segment (SUT or GWS) to SSS, $r = R_E + h_S$ (km) is the distance from the earth centre to the SSS, $R_E = 6,378.388$ km is the radius of the earth, $h_S = 35,805$ km is the GEO satellite altitude from the earth surface, while $\phi_S, \phi_P, \theta_S, \theta_P$ are longitudes (ϕ) and latitudes (θ) of satellite and terminal locations respectively. Equation (4.1.1) can be simplified to (4.1.2).

$$D_{TS} = \sqrt{R_E^2 + r^2 - 2R_E r \cos(\phi)} \quad (4.1.2)$$

The analytical expression of the distance (altitude) between the terminal location and the satellite, D_{TS} (km) could also be derived using a 3-Dimensional (3D) cartesian and spherical coordinate systems conversion, as shown by Eq. (4.1.3) and graphically represented in Fig. 4.2.

$$D_{TS} = \sqrt{(x_s - x_p)^2 + (y_s - y_p)^2 + (z_s - z_p)^2} \quad (4.1.3)$$

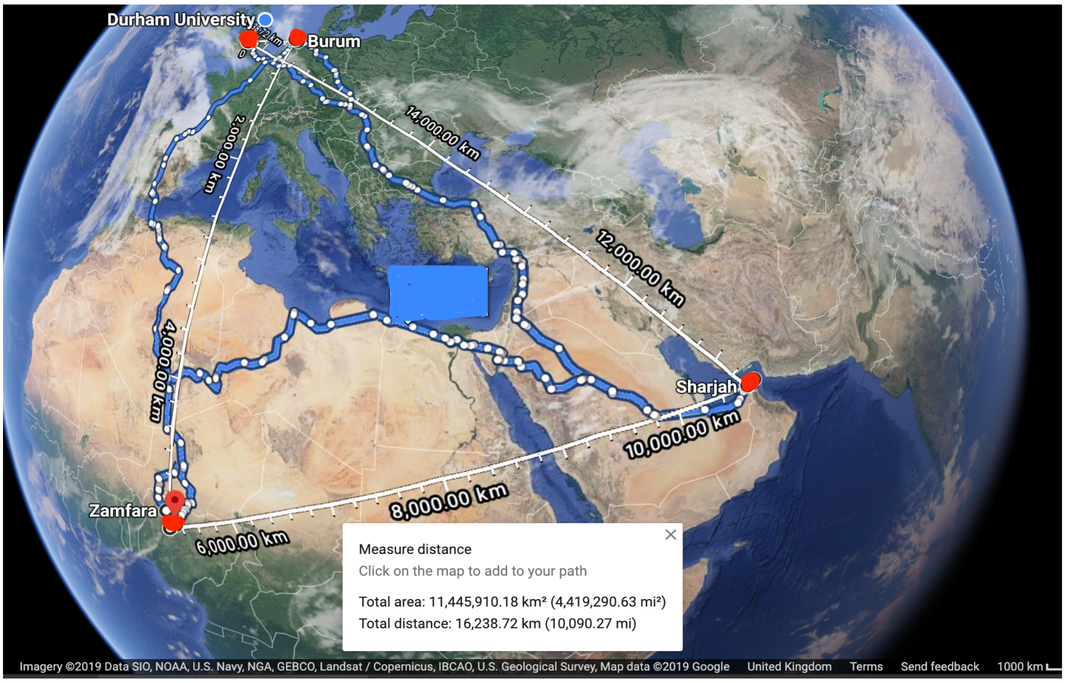
where *x, y* and z are cartesian coordinates of satellite $S(x_s, y_s, z_s)$ and terminal $P(x_p, y_p, z_p)$. The values are determined by coordinate transformation from cartesian to spherical coordinates $(x, y, z) \iff (r, \theta, \phi) = (Altitude, Latitude, Longitude)$ as: $x = r\cos\phi\cos\theta$, $y = r\sin\phi\cos\theta$,$z = r\sin\theta$, and $r = R_E + h$, where h is the altitude of the satellite or terminal location. Thus, the coordinate transformation from cartesian to spherical is given by: $(x, y, z) \iff (r\cos\phi\cos\theta, r\sin\phi\cos\theta, r\sin\theta)$, which yields Eq. (4.1.1) after trigonometric simplifications and substitutions.

Although, equations (4.1.1) and (4.1.3) give slightly different D_{TS} results, the resulting propagation latency differences are very negligible. Therefore, either of the equations could be used for analysis of latency changes due to the terminal location on earth. For GEO satellites without an Inter-Satellite Link (ISL), the minimum distance $D_{TS}(Min)$ is obtained when the terminal is located at the *sub-satellite* point (i.e directly under the satellite on earth), while the maximum distance $D_{TS}(Max)$ is when the terminal is located at the edge of coverage footprint with 0^o elevation angle and considering the radius of the earth, R_E from the centre of the earth [7].

$$D_{TS}(Min) = h = 35,786\ km \tag{4.1.4}$$

$$D_{TS}(Max) = (R_E + h) = r \tag{4.1.5}$$

Equations (4.1.4) and (4.1.5) were used to calculate minimum (sub-satellite point) and maximum (from the earth centre) satellite hop delay, while other SUT and GWS location distances were also computed using Eqn.(4.1.2) by transformation from spherical coordinates. Different propagation latencies such as One-Way-Delay (OWD_{prop}) from each of the ground segment locations to the satellite space segment, shown in Fig. 4.3, were determined using Eq. (4.1.8)) and summarised in Table 4.1. The same latency is determined for the terrestrial propagation delay from the GWS to the different locations where the experiments were conducted using the developed scenarios described in the following sections, the minimum and maximum values are given in the subsequent section.


Durham University
538.72 km
0
Burum
2,000.00 km
14,000.00 km
12,000.00 km
4,000.00 km
Sharjah
10,000.00 km
8,000.00 km
Zamfara
6,000.00 km
Imagery ©2019 Data SIO, NOAA, U.S. Navy, NGA, GEBCO, Landsat / Copernicus, IBCAO, U.S. Geological Survey, Map data ©2019 Google United Kingdom Terms Send feedback 1000 km

[redacted]

Measure distance

Click on the map to add to your path

Total area: 11,445,910.18 km² (4,419,290.63 mi²)

Total distance: 16,238.72 km (10,090.27 mi)

Figure 4.3: Ground Segment Sites Distances and Locations Mapping

Although the results from service providers were different, an analytical study of the distances to the satellite was conducted and the resulting changes as SUT and

Table 4.1: Ground Segment Location and Distance to Satellite

Location	$\theta(lat^0)$	$\phi(lon^0)$	$h(km)$	$D_{TS}(km)$	$OWD_{prop}(ms)$
Zamfara, Nigeria	12.2N	6.3E	0.450	35,805	119.35
Durham, England	54.8N	1.6W	0.097	35,805	119.35
Burum, Netherlands (PGWSI)	53.27N	6.23E	0	40,499	134.997
Fucino, Italy (SGWSI)	41.99N	13.55E	0.661	39,475	131.583
Sharjah, UAE (GWST)	25.36N	55.39E	0.024	36,529	121.763
Sub-Sat Point(θ_s, ϕ_s)	0.9N	64E	0	35,786	119.29
Earth-Centre P$(0, 0)$	0	0	6,378.388	42,164.39	140.55

GWS move from point to point on earth are given in Table 4.1. The testbed for the analysis is based on Inmarsat I-4 satellites located at $S(0.9^0N, 64^0E)$ at an altitude of 35,786 km above the Earth's surface with primary GWS at Burum Netherlands and Fucino, Italy as the secondary GWS, and its Broadband Global Area Network (BGAN) terminals located in Nigeria and England, as shown in Fig. 4.4.

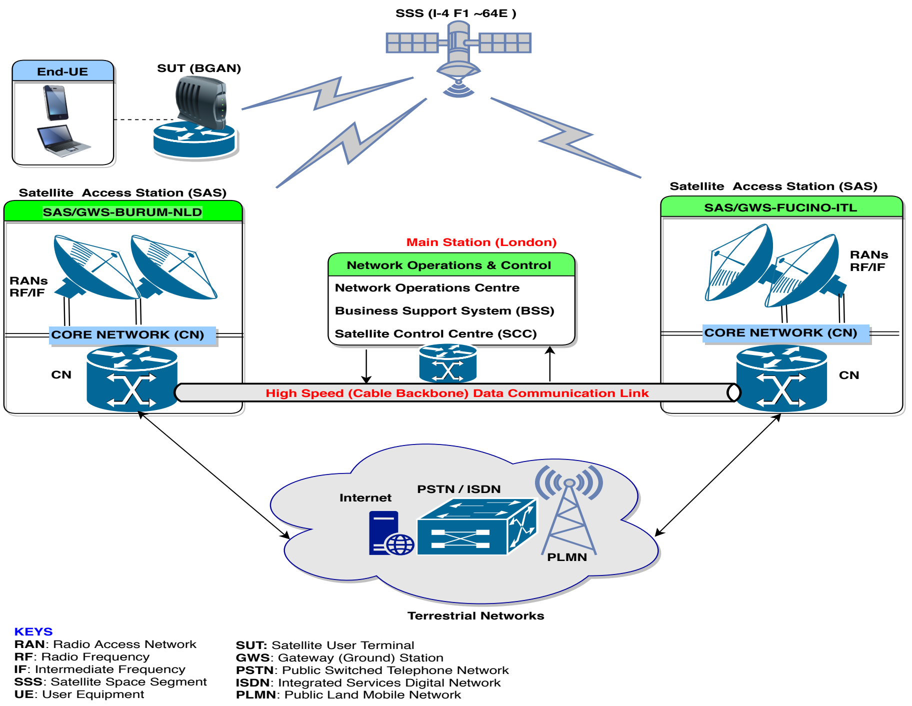

SSS (I-4 F1 ~64E)
End-UE
SUT (BGAN)
Satellite Access Station (SAS)
SAS/GWS-BURUM-NLD
RANs
RF/IF
CORE NETWORK (CN)
CN
Satellite Access Station (SAS)
SAS/GWS-FUCINO-ITL
RANs
RF/IF
CORE NETWORK (CN)
CN
Main Station (London)
Network Operations & Control
Network Operations Centre
Business Support System (BSS)
Satellite Control Centre (SCC)
High Speed (Cable Backbone) Data Communication Link
Internet
PSTN / ISDN
PLMN
Terrestrial Networks
KEYS
RAN: Radio Access Network
RF: Radio Frequency
IF: Intermediate Frequency
SSS: Satellite Space Segment
UE: User Equipment
SUT: Satellite User Terminal
GWS: Gateway (Ground) Station
PSTN: Public Switched Telephone Network
ISDN: Integrated Services Digital Network
PLMN: Public Land Mobile Network

Figure 4.4: Inmarsat Satellite BGAN Architecture [8]

The results summary in Table 4.1, indicate the minimum changes in the distances and latencies. The propagation latency component has quite a high contribution to the E2E latency as measured and is discussed in the following sections. The results presented in Table 4.1 did not include ground (terrestrial) network propagation delays such as signal propagation from the Satellite Access Station (SAS) or Radio Access Network (RAN) to the Network Operations Centre (NOC) and Satellite Control Centre (SCC) being part of the satellite network. The results (see Table 4.1) are for hops from SUT or GWS point to the SSS point.

4.1.2 E2E Latency Framework in a Satellite Environment

Revisiting the mathematical framework for computing different components of E2E latency in communication networks is required in order to develop a model framework for the measurements of the actual E2E latency in a network environment incorporating at least one satellite hop. This helped to investigate and evaluate the impacts of the delay accumulated from transmitter to the receiver site on E2E connection-oriented data transport TCP protocol. The general latency model in data communications was used as a basis from which our new model framework was derived and proposed [9]. Latency model can be categorised in terms of Fixed or Variable components [10] in the form of the four key components contributing to the total latency in any communication network, these components are **Propagation (T_{PG}), Processing (T_P), Transmission (T_{TX}) and Queuing (T_Q)**, the generalised mathematical model of the total OWD latency Φ_{E2E} is given by Eqs. (4.1.6). These components and what gives rise to them, are discussed in the next section with a graphical model representation shown in Fig. 4.5.

$$\Phi_{E2E} = \sum_{i=1}^{n} T_{PG}^{i} + \sum_{j=1}^{n} T_{P}^{j} + \sum_{k=1}^{n} T_{TX}^{k} + \sum_{l=1}^{n} T_{Q}^{l} \tag{4.1.6}$$

Where, Φ_{E2E} is the total E2E OWD latency, and i,j,k,l = 1,2,...,n are numbers of nodes (vertices) and links (edges) along the E2E path. The E2E RTT is given by Eq. (4.1.7) below:

$$RTT_{E2E} = 2 * \Phi_{E2E} \tag{4.1.7}$$

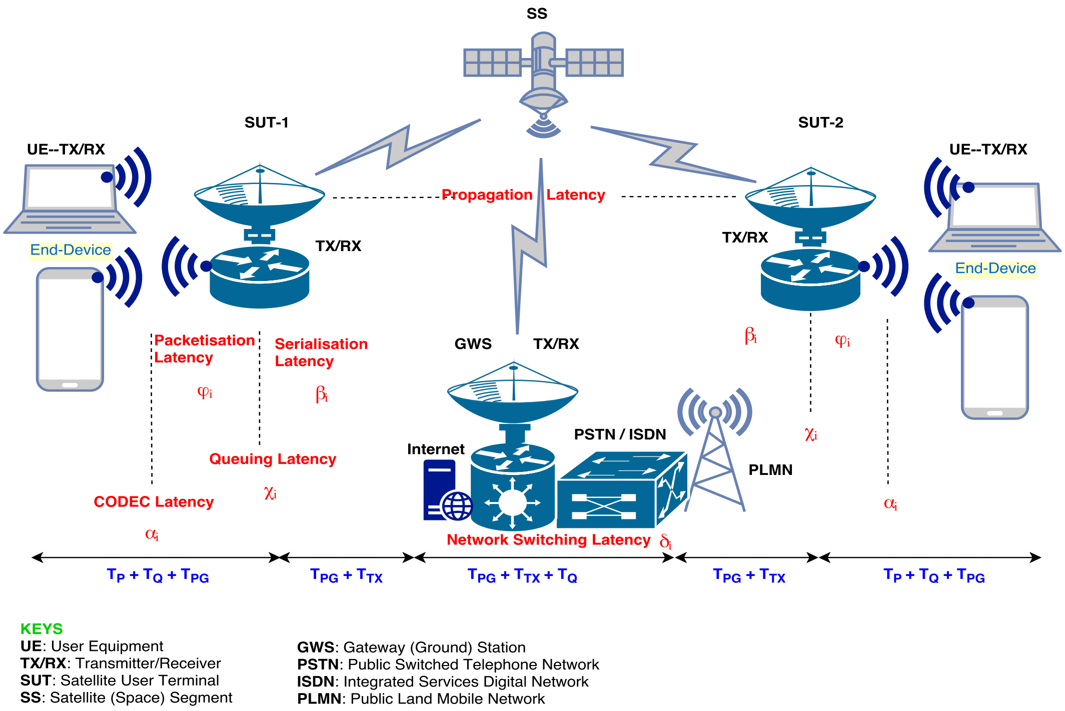


Figure 4.5: Generic End-to-End Latency Graphical Model

where β is serialisation/switching part of transmission delay, T_{Tx}, packetisation, φ and CODEC, α are parts of processing delay, T_P, and queuing delay, T_Q is represented by χ in Fig. 4.5.

Fixed and Variable Delays in Communication Networks

The fixed and variable latency components directly add to the overall E2E latency of the connection. Fixed latency usually depends on the physical link characteristics and the message or packet size, but is independent of the amount of traffic on the node and link, on which the variable latency normally depends. The fixed latency components in communication networks include; *Propagation, Processing,* and *Transmission* delays, while the variable component is contributed by *Queuing/Buffering* delay as described as follows.

1. **Propagation Delay**: This is fixed, and depends on the type of channel with speed v [kms^{-1}] and path length d [km]

$$T_{PG} = \frac{d}{v} \tag{4.1.8}$$

2. **Transmission Delay**: This is fixed comprising Serialisation and Network Switching delays, depending on the link speed C [Kbps], Frame/Packet/Message size M_s [bytes/bits].

$$T_{TX} = \frac{M_s}{C} \tag{4.1.9}$$

Network switching latency δ are the most difficult to quantify, but depend on the switching mechanism employed by network providers.

3. **Processing Delay**: This is fixed comprising packetisation and CODEC (Compression/Decompression) delays, depends on the type of Codec with speed (rate) S_c [Kbps], Packet size P_s [bytes/bits], Codec block sample and Compression algorithm. The CODEC α and Packetisation φ delays are given by (4.1.10) below.

$$T_P = \frac{P_s}{S_c} \tag{4.1.10}$$

$$\alpha = CT + DT.N + AD \tag{4.1.11}$$

Where; CT/DT: Compression/De-Compression Time per block, N: Number of blocks in frame and AD: Algorithm Delay.

4. **Queuing (Buffering) Delay**: Variable delay depends on the line speed (C) and the state of the queue (how much traffic, λ already in the buffer).This is variable component contribution from queuing in the egress (out-going) trunk buffers on the E2E communication path connecting the source and destination. This generates variable latencies giving rise to *Queuing* delay and *Internet Packet Delay Variation (IPDV) or Jitter*. This latency depends on the amount of traffic on the system as expressed mathematically by Eq. (4.1.12).

$$T_Q = \frac{\lambda}{C} \tag{4.1.12}$$

4.2 Practical End-to-End Latency Measurement

This section, describes the experimental activities conducted to measure practical E2E latency under heterogeneous network environment incorporating at least one satellite hop link. The measurement experiments were conducted with a real satellite testbed using *practical experiments* as discussed in this section, *simulation* and *emulation* as discussed in the next section. Performance of the TCP over satellite communication link was evaluated and analysed based on the achieved throughput from the measured latency in this experiment.

4.2.1 Measurement Procedure

The Explorer 510 SUT connected to BGAN from Inmarsat SatCom company and SatSleeve+ SUT connected to Thuraya SatCom Company, were used through out the measurement experiments. Both network service providers use networks of GEO satellites. Each of these terminals is portable with capabilities such as access point, routing, water and dust protection (IP66), while end UE connected to the SUT via WiFi link. The four ground segment equipment involved in the measurement testbed; were a mobile smart device with an application (SatSleeve+ Hotspot or Explorer Connect) for connecting a computer with the software (Audacity program) for capturing transmit/receive signals, and the SUT for transmitting (uplink) and receiving (downlink) to and from the the Inmarsat or Thuraya GEO Satellite network. The Inmarsat network testbed is shown in Fig. 4.6 while that of Thuraya is shown in Fig. 4.7.

The measurement procedure starts with the transmission of an audio signal from the source UE which is connected to either SUT or Public Land Mobile Network (PLMN) depending on the scenario employed as described in the following subsection. The audio signal is picked and recorded using Audacity- an audio signal processing software at the highest sample rate of 384 kHz for better data resolution.

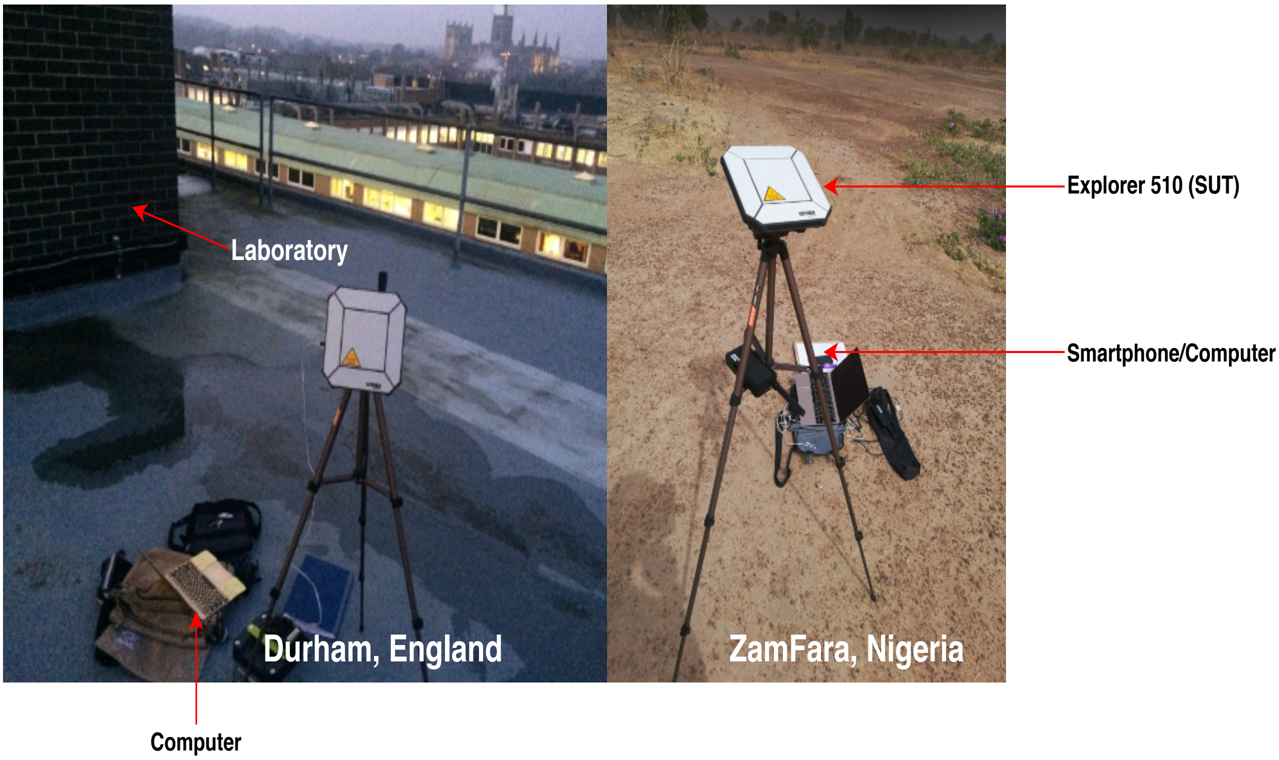


Figure 4.6: Inmarsat SatCom Network Testbed

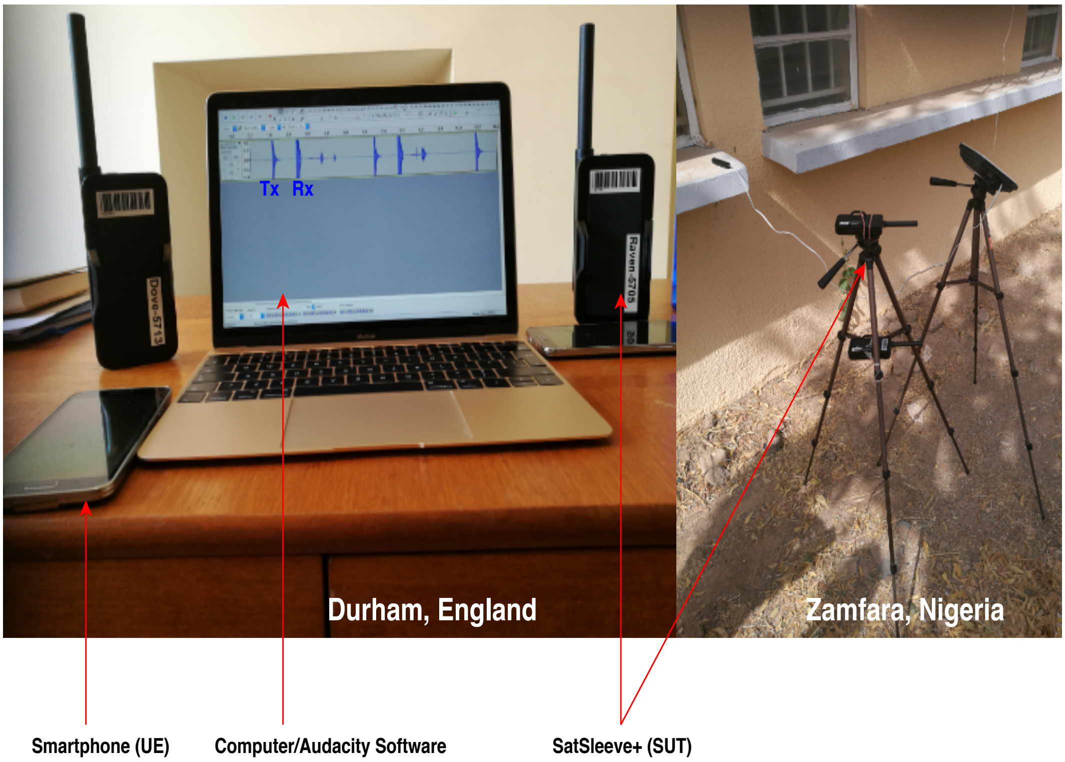


Figure 4.7: Thuraya SatCom Network Testbed

Table 4.2 summarised the technical specifications of the testbeds hardware used.

Table 4.2: Testbed Hardware Technical Specifications

Terminal	Dimensions(mm)	Weight	Mobility	Data Rate	Interfaces	Service
Explorer 510	202 x 202 x 51.8	1.4 kg	P/R	32-464 kbps	W/U/E	V/D
SatSleeve+	138 x 69 x 42	0.256 kg	P/R	15-60 kbps	W	V/D
Computer	281 x 13 x 197	0.92 kg	P/R	1.3 Gbps	W/U/E	V/D
Smartphone	72.5 x 142 x 8.1	0.145 kg	P/R	4G LTE	W/U	V/D

P: Portable, R: Rechargeable, W: WiFi, U: USB, E: Ethernet V: Voice (4-64kbps), D: Data

The transmitted signal traverses either a pure satellite or hybrid satellite-terrestrial network depending on the scenario employed as described in the subsequent sections, which was received at the destination UE and the same signal is picked and recorded by the audacity software. These signals (transmitted and received) recorded and stored as audio signals were then processed using MATLAB programming to determined OWD through correlation function of the signals.

A correlation function for discrete time was used to compute the delay between the pair of signals (Tx and Rx) to find the maximum similarity of the signals [11].The peak value can be achieved when the two signals are exactly the same [12]. This peak is then used to compute the time shift between two signals. Eq. (**??**) forms the basis for the computation of the actual E2E latency measured using the proposed scenarios in our framework [13, 14]. The flowchart of the practical E2E latency measurement procedure is given by Fig. 4.8.

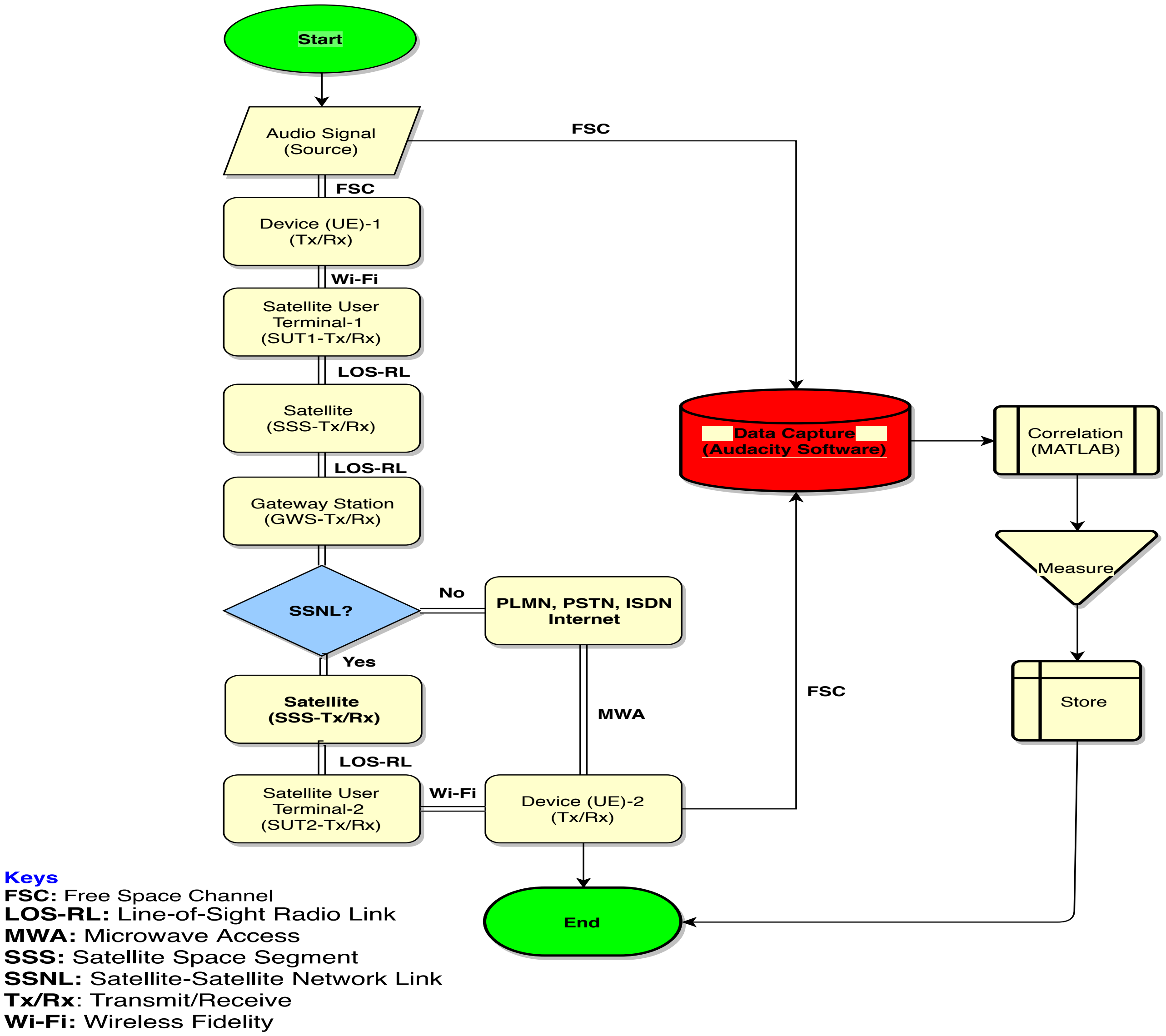


Figure 4.8: Latency Measurement and Data Acquisition Flow Diagram

4.2.2 Experiments Setup and Scenarios

The experimental setup for the measurements used the testbeds in Fig. 4.6 and 4.7 with two developed case study scenarios for pure satellite and hybrid satellite-terrestrial network links. These scenarios were referred to as Sateliite-Satellite Network Link (SSNL) as shown graphically by Fig. 4.9 and Satellite-Terrestrial Network Link (STNL) that represents a realistic hybrid ISTN communication network environments that are heterogeneous in nature as shown in Fig. 4.10.

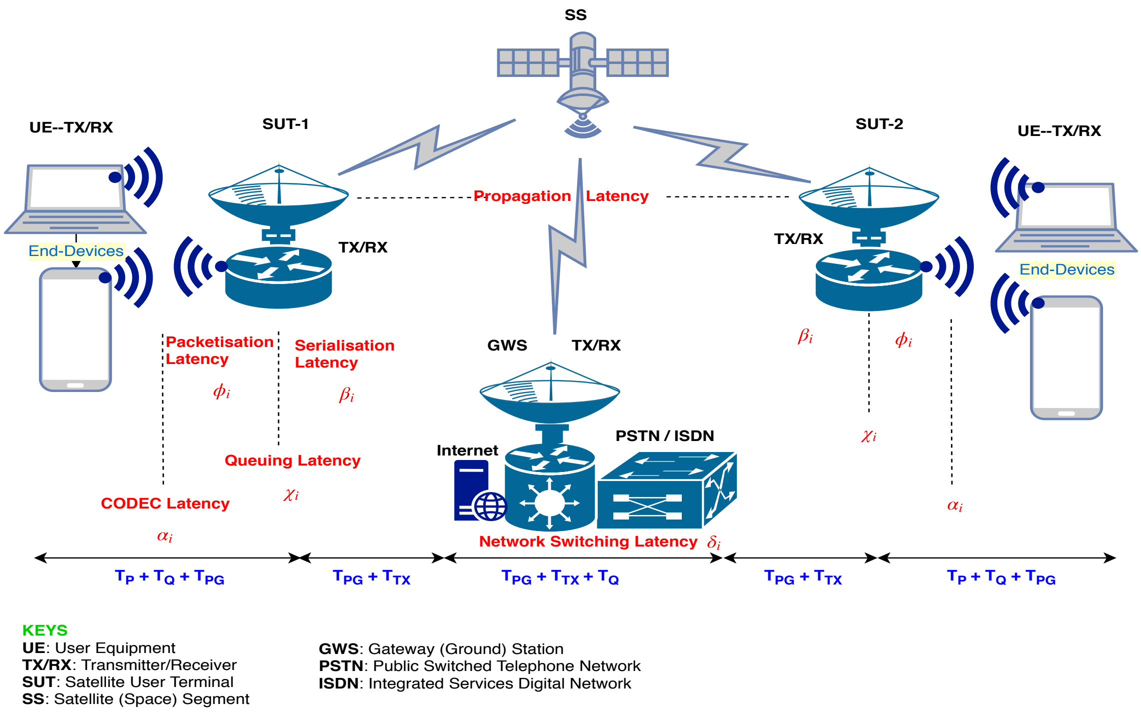


Figure 4.9: E2E Satellite-Satellite Link (SSL) Graphical Model

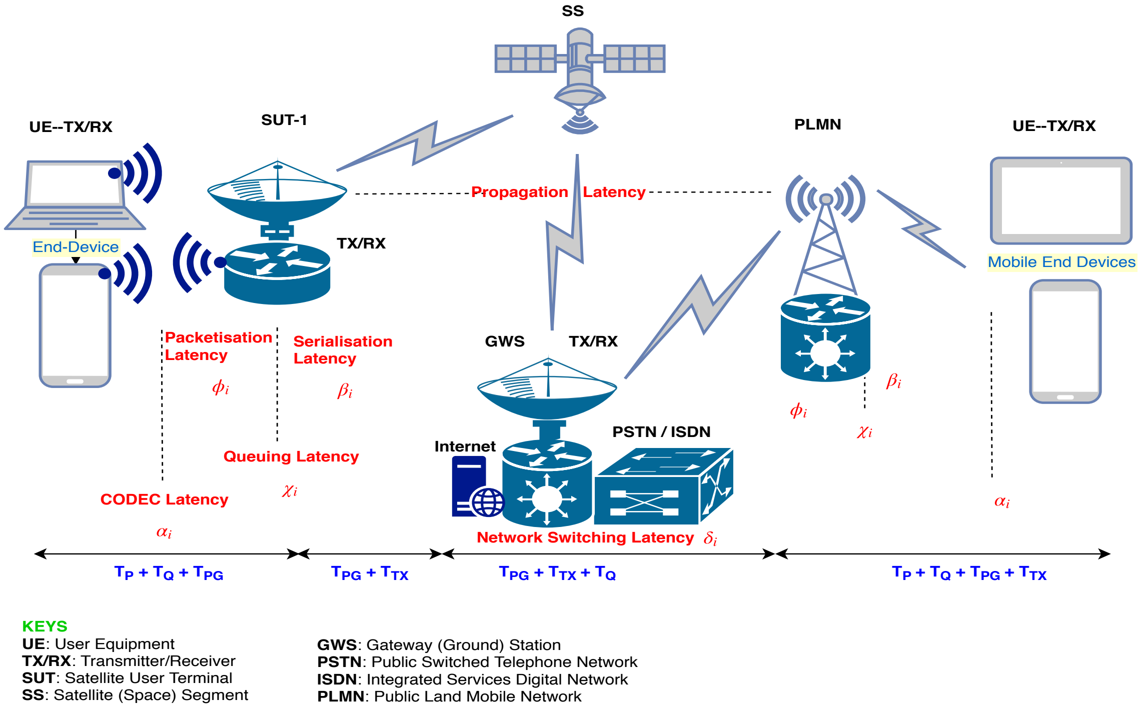


Figure 4.10: E2E Satellite-Terrestrial Link (STL) Graphical Model

These scenarios can also be interchangeably called Satellite-Satellite Link (SSL) and Satellite Terrestrial Link (STL) to indicate the existence of at least one leg that connects the end user to the satellite. Latency performance was measured and evaluated based on these unique study scenarios developed by this thesis.

Satellite-Satellite Link

This scenario, SSL was developed to establish connectivity between two remote end-users both linked to the satellite network via satellite ground user terminals (SUT) as the last legs as shown in graphical model Fig. 4.9 and single hop configuration model in Fig. 4.11. This case study scenario was used to measure and study the performance of a purely satellite network environment for connecting remotely isolated rural end users. This scenario depicts two remote rural locations without terrestrial communications infrastructure, not economically feasible, or not reliable where it exists such as remote isolated rural areas. The SSL established E2E connectivity between two semi-fixed Inmarsat or Thuraya SUTs shown in the testbeds of Fig. 4.6 and 4.7, the two SUTs were set to point to the satellite using a Line-of-Sight (LOS) link. Adjustments (see Appendix B) were made until satisfactory signal strength for communication was obtained within $70-90\%$ after the LOS pointing. The end user UE and SUTs were connected using Wi-Fi link 1-20 m apart as shown in Fig. 4.9 of the graphical model.

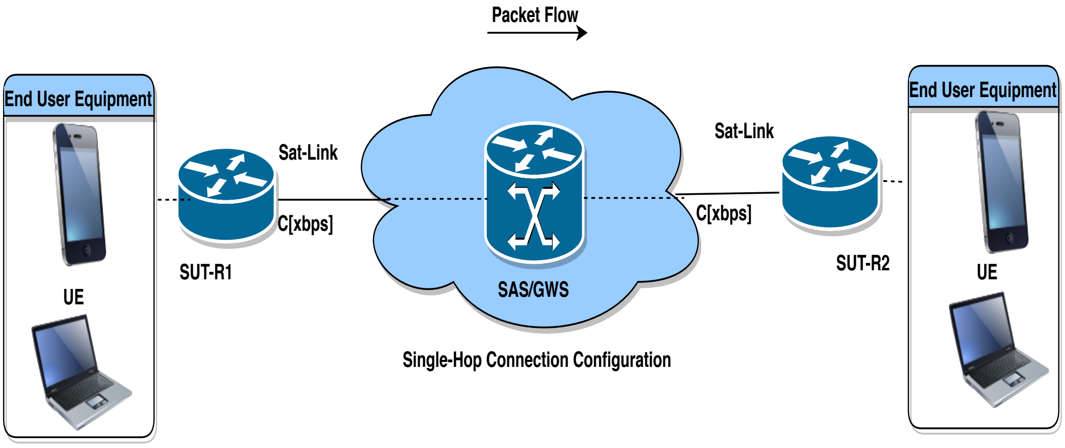


Figure 4.11: Single-Hop SSL Model Configuration of E2E Latency

Satellite-Terrestrial Link

The STL scenario represents hybrid of ISTN) links that connect end users on a heterogeneous (Satellite and Terrestrial) network infrastructure. This case study scenario setup is the most realistic for the hybrid channels found in today's communication networks and in future 5G New Radio (NR) networks. The scenario as shown in Fig. 4.10 depicts a network connecting users in an isolated remote rural location via satellite network infrastructure and another end user in developed urban area utilising available terrestrial network infrastructure such as PLMN. This was developed to evaluate the performance of heterogeneous ISTN environments. The STL E2E connection was achieved through a heterogeneous network including satellite, Wi-Fi and PLMN links. The UE on the satellite leg in the remotely isolated rural area connected to the satellite-pointed SUT via Wi-Fi, while the UE on the terrestrial leg connected to PLMN via the nearest 3/4 G Base Transceiver Station (BTS) in the urban area. This scenario was developed to measure E2E latency of heterogeneous network testbeds shown in Fig. 4.6 and 4.7 with graphical model in Fig. 4.10 and configured as dual-hop model as in Fig. 4.12.

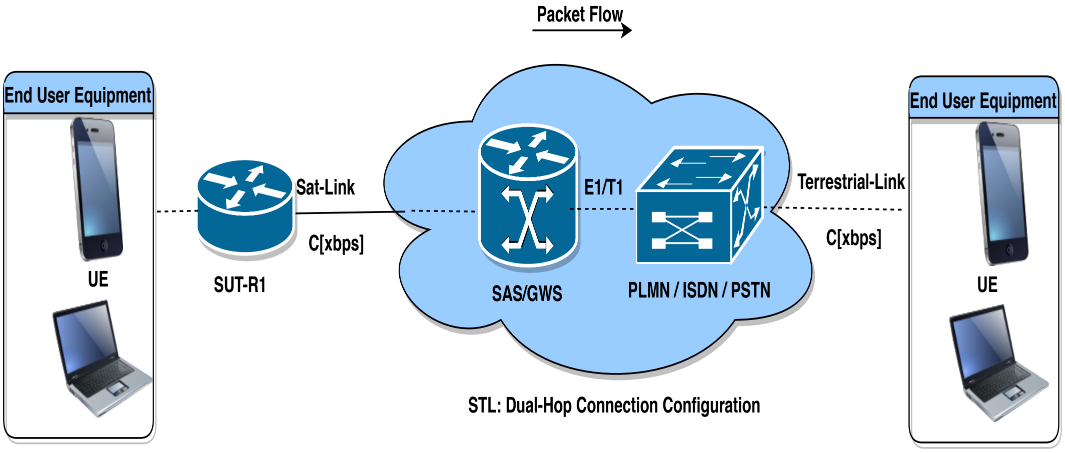


Figure 4.12: Dual-Hop STL Model Configuration of E2E Latency

4.2.3 E2E Latency Model for Satellite Environment

Following the successful design described in the previous sections, development and measurement using the SSL and STL scenarios, three mathematical models were established as a framework for the evaluation and analysis of the actual E2E latency in a heterogeneous ISTN environment using the scenarios and network topology developed by this thesis. A more general model is given by Eq. (4.2.13) as a linear summation of OWD consisting of propagation, processing, queuing and transmission components along the communication network path with at least a satellite leg [9].

$$\Phi_{E2E} = KT_{Prop} + \sum_{i=1}^{n} T_i \tag{4.2.13}$$

Where Φ_{E2E} is the total OWD (latency) in milliseconds (ms), K is the number of link(s) traversed by the traffic from source to the destination and sum of T_i represents other delay components (non-propagation latency components) within the satellite network such as queuing, processing and transmission delays in GWS NOC. The other two models as described by Eq. (4.2.14) and (4.2.15) were developed from the SSL and STL scenarios respectively considering their E2E signal path topologies.

$$\Phi_{SSL} = K_{SL}T_{PropS} + \sum_{i=1}^{n} T_i + K_{TL}T_{PropT} \tag{4.2.14}$$

$$\Phi_{STL} = K_{SL}T_{PropS} + \sum_{i=1}^{n} T_i + K_{TL}T_{PropT} + \sum_{j=1}^{m} T_j \tag{4.2.15}$$

Considering our two study scenarios, K_{SL} in (4.2.14) and (4.2.15) would assume a constant value of 4 and 2, respectively, while K_{TL} can take values from $1, 2, ...l$ in both Eq. (4.2.14) and (4.2.15), since there is at least one terrestrial link in the form of wireless (WiFi) or wireline and from/to ground network component such as GWS. T_j represents non-propagation delay components within the terrestrial network such as queuing, processing and transmission delays in BTS and MSC.

Moreover, careful observation of the STL scenario topology given in Fig. 4.10 led to identification of an additional propagation delay, KT_{PropT} contributed by the geographical distance separating the GWS and the PLMN connecting the UE on the terrestrial leg end. The geographical site locations of the testbed and network providers GWS were shown in Fig. 4.3. A summarised computation of the propagation latency using wireless link ($c = 3x10^8$ ms^{-1}), or wired link ($2c/3$ ms^{-1}) as minimum and maximum OWD_{prop} (Φ) respectively as shown in Table 4.3.

Table 4.3: Ground Segment (GWS) Propagation Delays to PLMN

Location-Link	**Distance(km)**	$\Phi_{min\_prop}(ms)$	$\Phi_{max\_prop}(ms)$	$\Phi_{avg\_prop}(ms)$
Burum ↔ Durham	1,113	5.57	3.72	4.65
Burum ↔ Zamfara	6,396	31.98	21.32	26.65
Sharjah ↔ Durham	7,556	37.78	25.19	31.49
Burum ↔ Zamfara	8,495	42.48	28.32	35.40

4.2.4 Experimental Results and Analysis

Latency of a realistic hybrid satellite terrestrial network environment with random traffic can be quite stochastic, due to the autonomous nature of the network elements enabling the data communication from source to destination. This led to wide variations in results at any point in time. Therefore, we computed the OWD latency performance three times a day over the period of seventeen days using the two scenarios (SSL and STL), these were developed using the two Satellite Network Providers (SNPs) ground terminals (SUTs) and satellites, namely Inmarsat (SNP1) and Thuraya (SNP2). The results for scenarios using SNP1 is given by Fig. 4.13 and scenarios using SNP2 is given by Fig. 4.14 shown that of STL scenario.

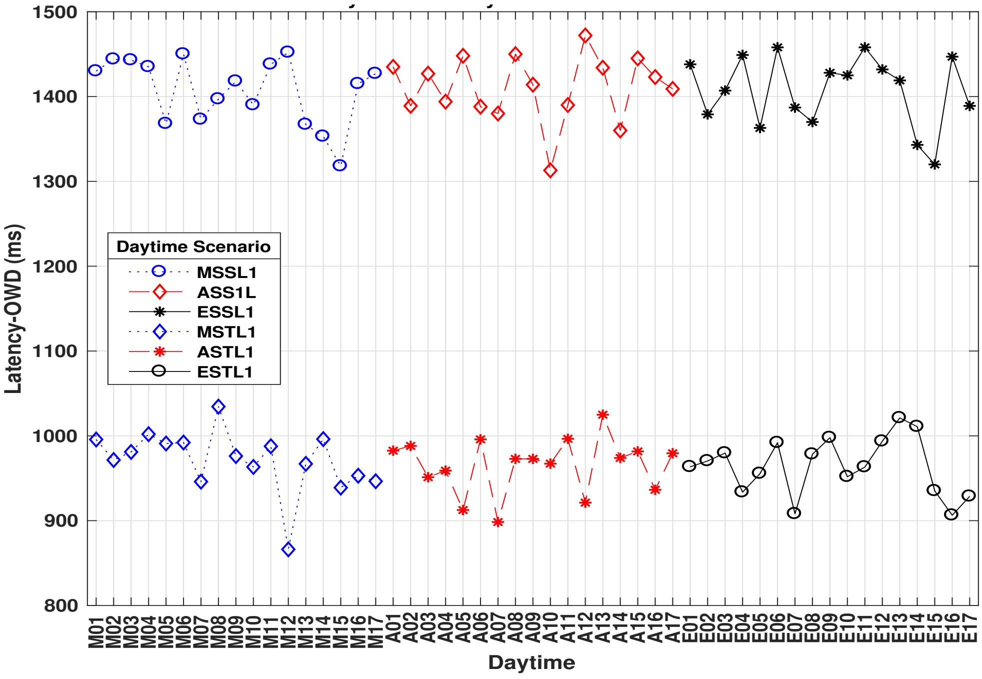


Figure 4.13: Daytime Latency Performance for Satellite Network Provider 1

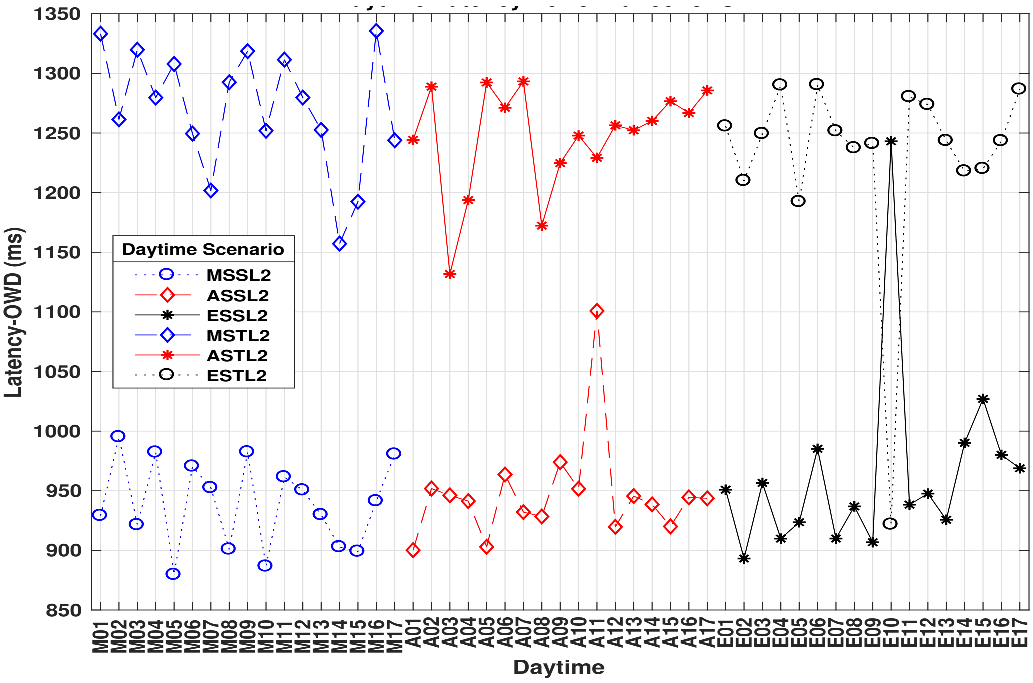


Figure 4.14: Daytime Latency Performance for Satellite Network Provider 2

The SSL scenario performance comparison between the two SNPs is given by Fig. 4.15 while STL Fig. 4.16. The statistical summary of the latency performance measured using both the SNPs are given in Tables 4.4 and 4.5 for SSL and STL scenarios respectively. Independent daytime measurements were conducted such that the sample standard deviation (σ_{std}) of each connection's Latency was within 5% of its sample mean (Φ_{avg}) , this generally required around 17 data points. These statistical summaries of the data points were then used to determine the overall performance in terms of maximum, minimum, and average latency of a given scenario topology.

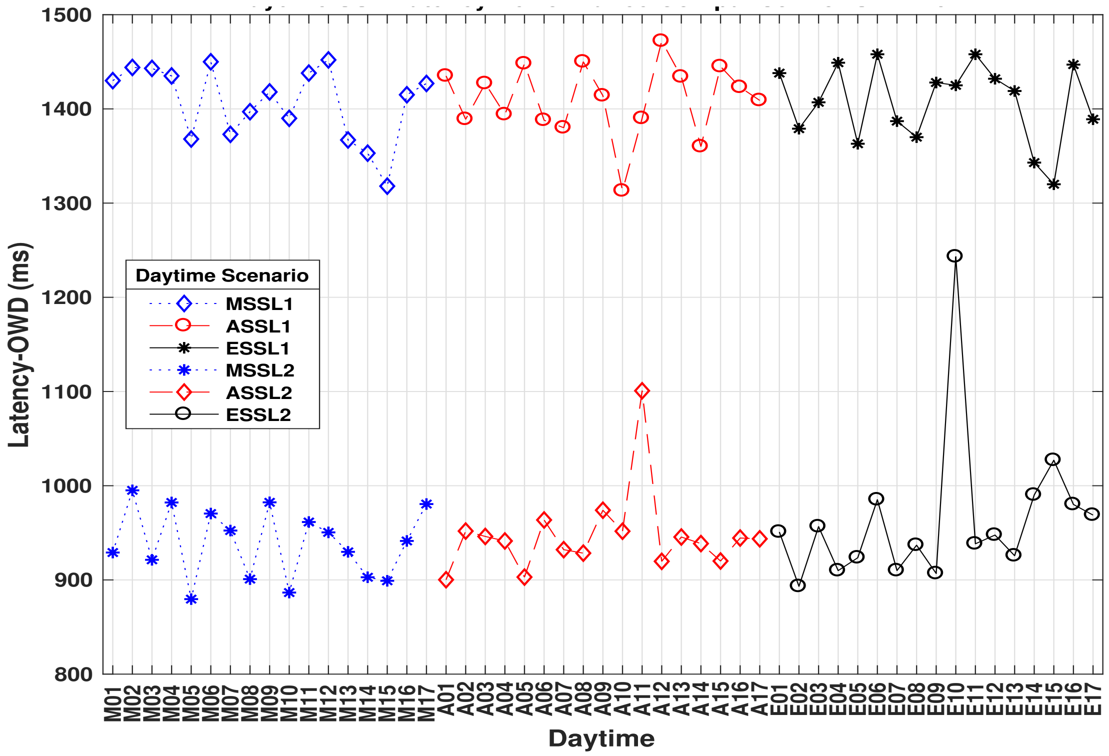


Figure 4.15: SSL Performance Comparison for Satellite Network Providers

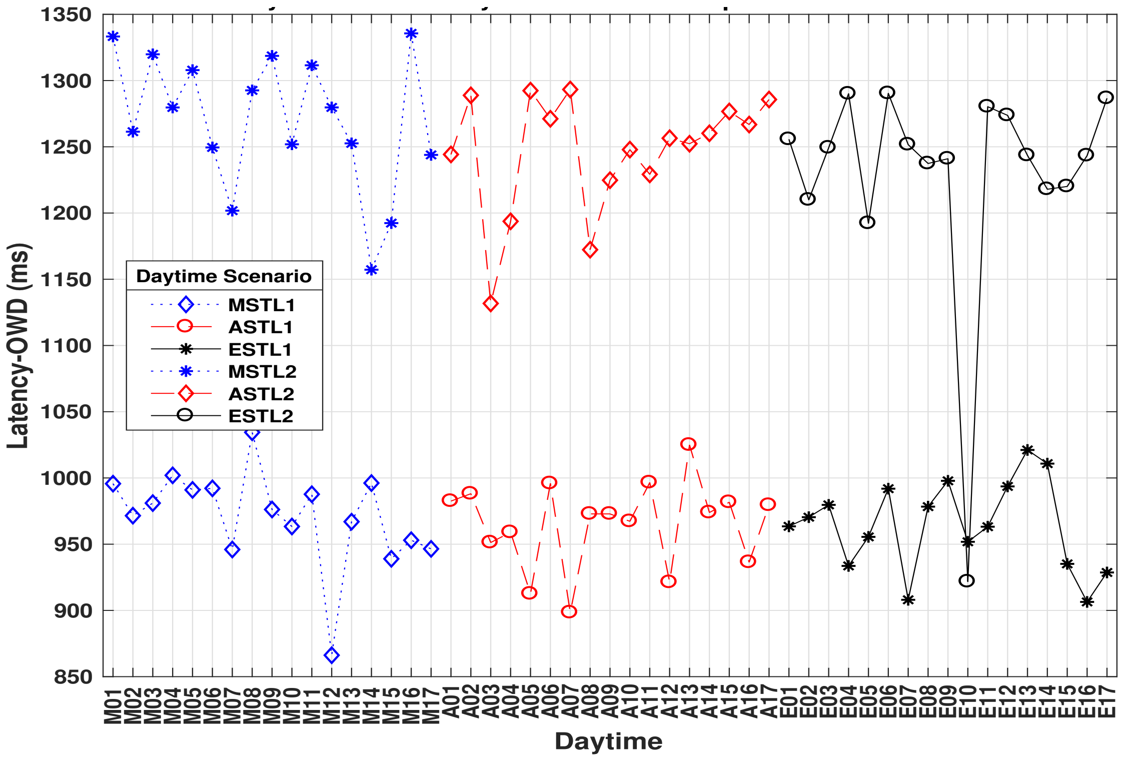


Figure 4.16: STL Performance Comparison for Satellite Network Providers

Table 4.4: Daytime SSL Latency Performance Summary

Scenario	$\Phi_{max}(ms)$	$\Phi_{min}(ms)$	$\Phi_{avg}(ms)$	$\sigma_{std}(ms)$
MSSL1	1452	1318	1407	39.45
ASSL1	1472	1313	1410	38.87
ESSL1	1458	1320	1407	41.49
MSSL2	995	880	939	36.71
ASSL2	1101	900	947	44.01
ESSL2	1243	893	964	80
Overall	1472	880	1179	

Table 4.5: Daytime STL Latency Performance Summary

Scenario	$\Phi_{max}(ms)$	$\Phi_{min}(ms)$	$\Phi_{avg}(ms)$	$\sigma_{std}(ms)$
MSTL1	1035	866	971	36.33
ASTL1	1025	898	966	33
ESTL1	1021	906	964	34
MSTL2	1336	1157	1270	51
ASTL2	1293	1132	1246	45
ESTL2	1290	922	1230	84
Overall	1336	866	1108	

Figure 4.13 shows the results obtained with both SSL and STL study scenarios using SNP1 for different times of the day that include morning (M), afternoon (A) and in the evening (E). Latency measurements corresponding to the three daytimes mentioned for a period of seventeen (17) days, were carried out using the testbed and method described in subsection 4.2.1. For instance, the daytimes of the first day of the experiment is represented by M01 (morning of day 1), A01 (afternoon of day 1), and E01 (evening of day 1), while day 2 is represented by M02, A02, and E02. The resulting latency by SSL scenario using SNP1 shown in the upper part of Fig. 4.13 appeared to be higher in each daytime compared to STL scenario shown in the lower part of Fig. 4.13. The statistical summary of the performance of SSL scenario using SNP1 given as SSL1 in Table 4.4 showed that the maximum latency value, Φ_{max} was 1472 ms and minimum value Φ_{min} of 1310 ms obtained in

the afternoon daytime while the average is between 1407 ms and 1410 ms, which is about 1408 ms.

Although the measured latencies were stochastic in nature, the standard deviation, σ_{std} (variation from the average value Φ_{avg}) is still small as shown in Table 4.4. The values of the standard deviation σ_{std} (see Table 4.4) for the daytime measurements were 39.45 ms (2.80% of 1407 ms), 38.87 ms (2.76% of 1410 ms), and 41.49 ms (2.95% of 1407 ms), in the morning, afternoon and evening respectively. These gave less than 3% deviation from the average values of the measured latencies. The STL scenario using SNP1 (STL1) given in Table 4.5 showed similar stochastic nature, but resulted to lower latency values with maximum (Φ_{max}) of 1035 ms, minimum of 866 ms in the morning daytime, and the average value between 964 ms and 971 ms, which is about 967 ms. The deviation σ_{std} values were about the same (¡ 5%) with that of SSL1 with values of 36.33 ms (3.74% of 971 ms), 33 ms (3.42% of 966 ms), and 34 ms (3.53% of 964 ms) as shown in Table 4.5.

Considering both SSL1 and STL1 scenarios, the overall latency performance measured had a maximum value of 1472 ms obtained in the afternoon using SSL1 scenario with an average of 1410 ms, minimum value of 866 ms in the morning using STL1 scenario with an average of 971 ms. The minimum RTT = 2*Φ that was obtained by using SNP1 testbed is 1942 ms while the maximum was 2944 ms with the implication of degrading the performance of standard TCP as discussed in chapter 3 and analysed using the achievable throughput in the following subsection. The smaller value of less than 4% of the average σ_{std}, throughout the measured values using the two scenarios, will reduce the impact on jitter in data communications over the heterogeneous internet environment.

Moreover, measurements were also carried out using SNP2 with both SSL2 and STL2 scenarios that resulted in Fig. 4.14. The statistical summary of the measurement results for SSL2 provided in Table 4.4 and STL2 given by Table 4.5. However, the STL2 scenario was found to have higher latency values compared to SSL2, which is the complete opposite to what was obtained using SNP1 results described ealier. The SSL2 resulted in highest maximum (Φ_{max}) of 1243 ms obtained in the evening daytime of the measurements, lowest minimum (Φ_{min}) of 880 ms in the morning, and an overall average (Φ_{avg}) between 939 ms and 964 ms is 949 ms as shown in Table 4.4. The STL2 maximum latency in the morning with value of 1336 ms, minimum of 922 ms, and average of 1249 ms that is between 1230 ms and 1270 ms averages.

The performance comparison of SNP1 and SNP2 using the developed scenarios, SSL and STL were given in Fig. 4.15 and Fig. 4.16 respectively. The high values obtained with STL2 could be due to the location of the SNP's primary GWS, PLMN and the location of SUT during the measurement experiment as discussed in section 4.1.1, and shown in Fig. 4.3. Additional terrestrial propagation (see Table 4.3) and processing delays contributed to higher values in the case of STL2 as compared to STL1, this is due to the proximity of the GWS to the SUTs in the United Kingdom and Nigeria. Another reason may be that, SNP1 SUTs were optimised for data services using BGAN while SNP2 is optimised mainly for voice services with less sophisticated network compared to BGAN.

Generally, the overall latency performance using both SNPs indicated that the highest latency (Φ_{max}) value of 1472 ms (see Table 4.4) was obtained from SSL scenario while the lowest (Φ_{min}) value of 866 ms (see Table 4.5) was obtained from STL scenario. The lowest overall average latency (Φ_{avg}) value of 1108 ms was also obtained using the STL scenario as given in Table 4.5. These indicate better latency (RTT) performance with the STL scenario as shown in Table 4.6, which represents the more realistic heterogeneous communication network environments of today and the future networks such as ISTNs that have a key role in the next generation 5G networks.

Table 4.6: Overall RTT Performance Summary

Scenario	$RTT_{max}(ms)$	$RTT_{min}(ms)$	$RTT_{avg}(ms)$
SSL	2944	1760	2358
STL	2672	1732	2216
Overall	2944	1760	2287

4.3 Emulation and Simulation Measurements

The emulation/profiling and simulation experiments were designed and conducted to measure the E2E latency of the heterogeneous network environment using the scenarios developed and discussed earlier. The results obtained from both the emulation and simulation experiments are then compared with the real practical results obtained. The emulation experiment testbed shown in Fig. 4.17 is described in the following subsection.

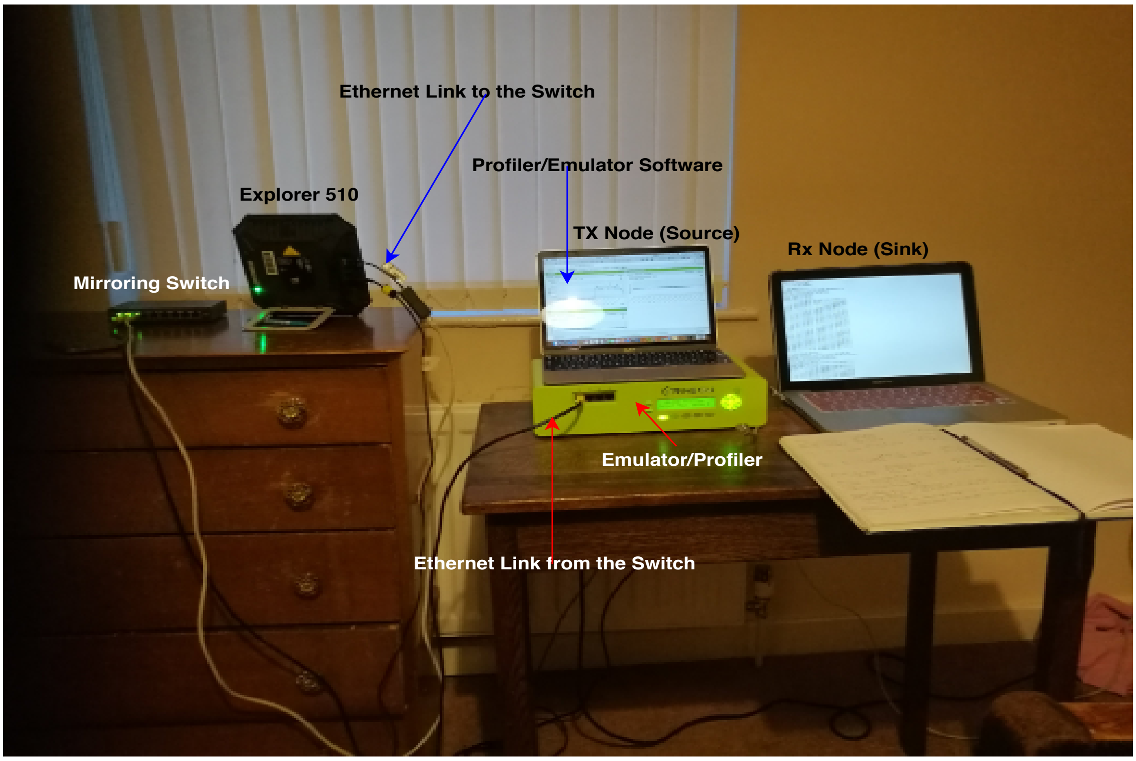

Ethernet Link to the Switch
Profiler/Emulator Software
Explorer 510
TX Node (Source)
Rx Node (Sink)
Mirroring Switch
Emulator/Profiler
Ethernet Link from the Switch

Figure 4.17: Emulation and Profiling Testbed Network

4.3.1 Materials and Method

The equipment used for the *emulation* and *profiling* experiment are an NE-ONE network emulator and *profiler* appliance, a data GS108T smart switch for port mirroring, explorer 510, an end user device for data *Tx/Rx*, and an ethernet cables for communication and networking as shown in Fig. 4.17. The NE-ONE appliance is unique in offering two powerful, complementary network and application performance capabilities like network emulation (Virtual Test Network) and network performance profiling [16, 17]. This appliance or instrument has four ethernet ports (0,1,2 and 3) [16, 17], on the front panel (see Fig. 4.18) of which only ports 0 and 1 are used for network performance profiling [17], while all the four can be used for network emulation [16].

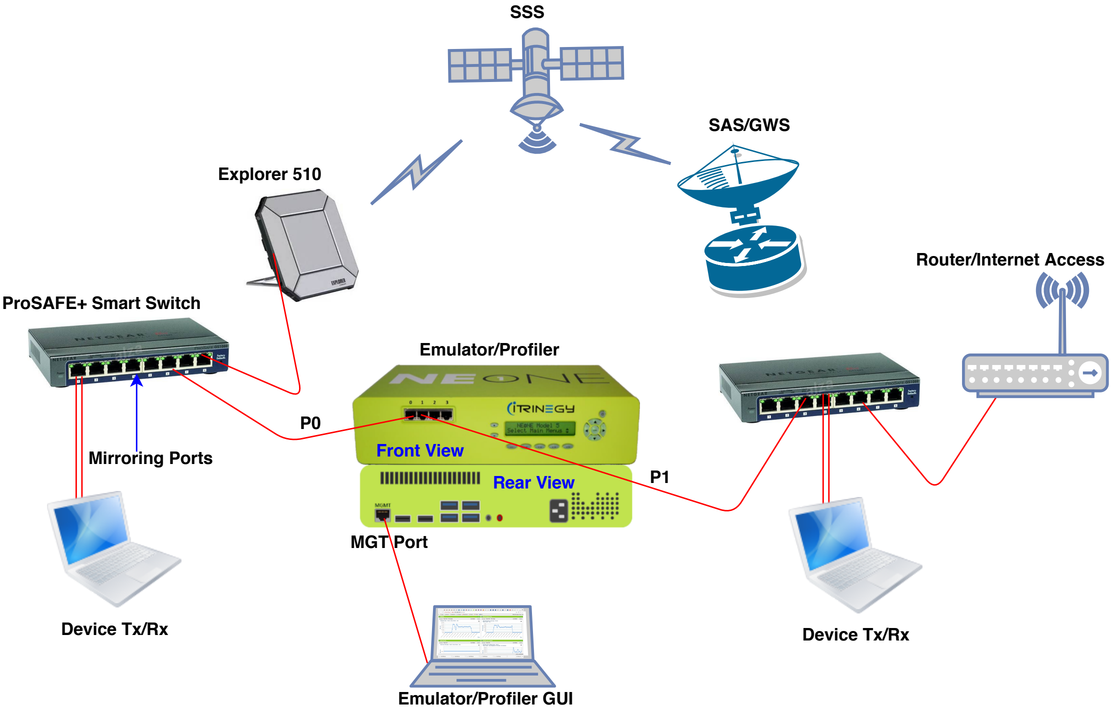


Figure 4.18: Network Emulation and Profiling Topology

An additional ethernet port for profiler management (MGT) at the rear panel of the appliance is used for device configuration and monitoring via the ethernet cable interface, Fig. 4.18 gives the network topology and setup configuration used for the performance profiling and emulation using NE-ONE and testbed equipment listed above with technical specifications of the key hardware given in Table 4.7.

Among the key device in the testbed is the *smart switch*, which enabled port mirroring of the data traffic. This switch functionality allowed mirroring of the incoming (ingress) and outgoing (egress) traffic of one or more ports (the source ports) to a single predefined and configured destination port [19]. This predefined port was then tapped and fed to port 0 of the NE-ONE for network performance profiling as shown in the testbed (Fig. 4.17) and the physical topology in Fig. 4.18.

Table 4.7: Testbed Hardware Technical Specifications [18, 19]

Feature	**NE-ONE Appliance**	**Smart Switch**	**Cable Link**
Model	Desktop M10	ProSAFE GS108Ev3	Category 5e (UTP)
Dimensions(WxDxH mm^3)	204 x 324 x 77	158 x 101 x 29	24-AWG
Weight(kg)	5 kg	0.508 kg	
Power (V/A/W)	100-230(AC)/4-2/180	12(DC)/0.5/4.45	30(DC)/0.577
Operating Temperature (^0C)	-5 to 35	0 to 40	-55 to 60
Operating Rel. Humidity (%)	8 to 90	10 to 90	N/A
Data Rate(Gbps)	1	1	1
Interfaces/Ports	5 x Gigabit Ethernet	8 x Gigabit Ethernet	Gigabit Ethernet

UTP: Unshielded Twisted Pair, WxDxH: Width x Depth x Height, V: Voltage, A: Ampere, W: Watts

The Explorer 510 SUT provides a link to the satellite network (see Fig. 4.18) from which the performance was measured by the testbed.

Moreover, simulations were carried out using Network Simulator 2 (NS-2) version 2.35 installed on Linux operating system (Ubuntu 14.04.5). Before the simulation experiments started, a network topology was designed and implemented using Tool Command Language (TCL) a programming language within the in NS-2 and Network Animator (Nam) tools for visualising network simulations and data trace for analysis. Nam is a Tcl/Tk based animation tool that supports topology layout, packet level animation and data inspection tools.

The simulation topology shown in Fig. 4.19 consisted of the designated nodes (in green) for transmitting and receiving TCP or UDP packets over the period of simulation. The simulation data was recorded in a trace file (contains information about nodes, links, packet traces, timestamps, events, and protocols), which was then processed using AWK and PERL programming languages to extract the E2E latency from the transmission node to the receiving node over the duration of the simulation of about two hours for three days. The terrestrial network part of the simulation topology consisted of a 3m ethernet cable link of 100 Mbps, 15 ns propagation latency to connect Tx/Rx nodes and 4G PLMN of 10 Mbps and 100 ms latency. File Transfer Protocol (ftp) traffic over TCP was used with 552 bytes (MSS) and DropTail queue management enabled.

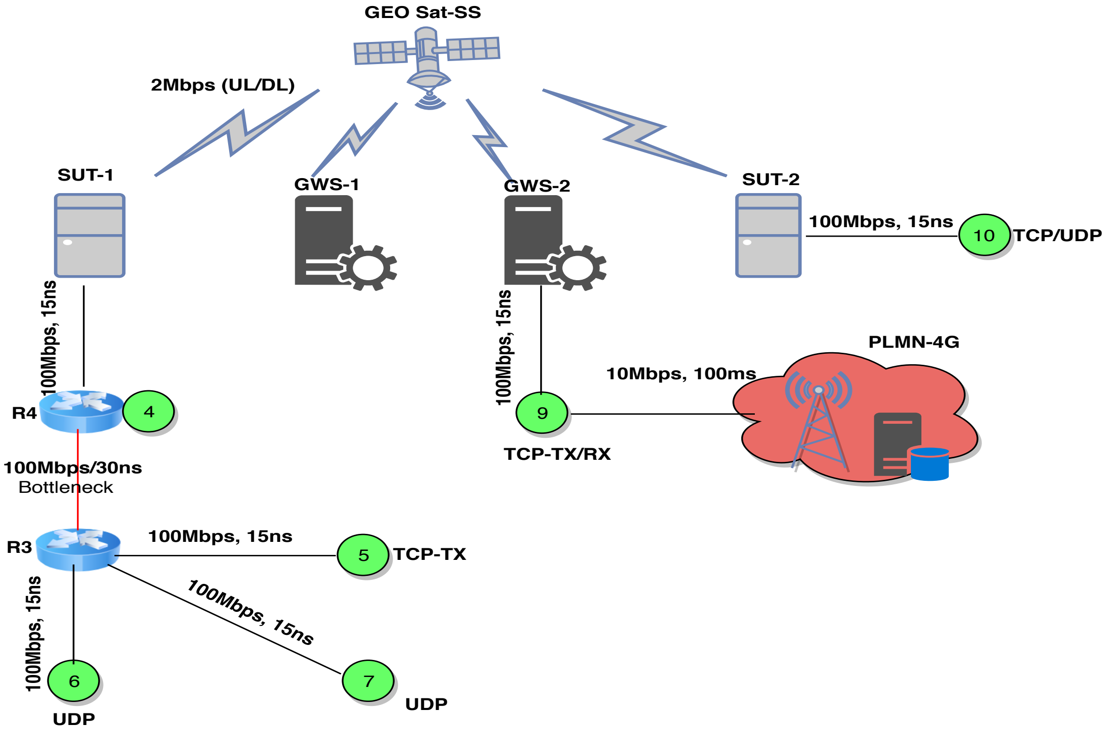


Figure 4.19: Benchmark Network Topology and Simulation Setup

Latency Measurements using both the network profiler and simulator employed a *passive method* of measurement as against the active correlation method used in the experimental measurement in section 4.2.1 above. The passive method monitored the current data traffic flow on the network to measure latency metric using network monitoring capabilities of the NE-ONE Profiler appliance and the NS2 tool.

4.3.2 Emulator Profiled and Simulation Results Discussions

The results obtained by network profiling within the period of two hours daily (2hrs) for three days are shown in Fig. 4.20, indicate the stochastic nature of latency as observed from the practical results discussed in the previous section. The profiled latency results showed correlation with the experimental results, but lower values were obtained statistically and summarised in Table 4.8.

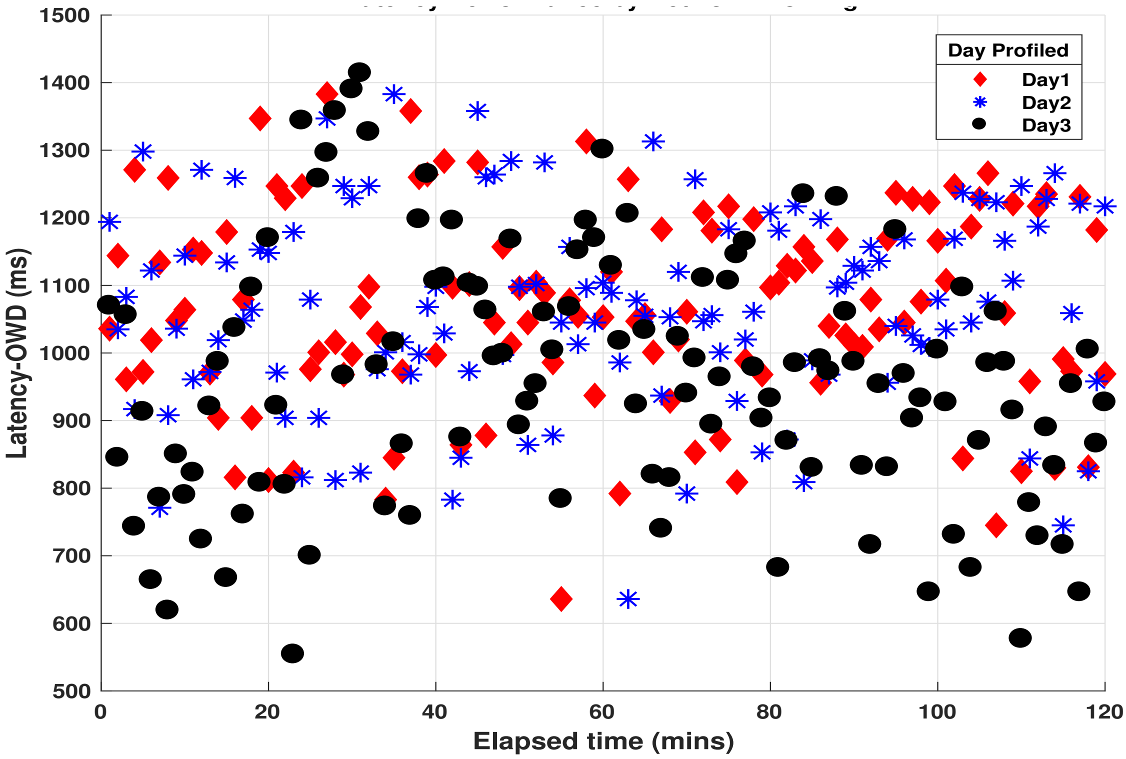


Figure 4.20: Profiled Network Latency Performance

Table 4.8: Latency Performance Profiled Summary

Day	$\Phi_{mx}(ms)$	$\Phi_{mn}(ms)$	$\Phi_{av}(ms)$	$\sigma_{sd}(ms)$	$RTT_{mx}(ms)$	$RTT_{mn}(ms)$	$RTT_{av}(ms)$
1	1383	636	1067	147.48	2766	1272	2134
2	1383	636	1068	147.62	2766	1272	2136
3	1414	554	963	186.23	2828	1108	1926
Overall	1414	963	1033		2828	1108	2066

The statistical analysis of latency measured in the first and second day (Day 1 and 2)) gave a maximum value of 1383 ms, minimum of 636 ms, the averages and deviation for these days were about the same as shown in Table 4.8. However, the deviations from the average values were large compared to the experimental results, these varied from 147.48 ms (13.82% of 1067 ms) obtained in the first day to 186.23 ms (19.34% of 963 ms) in the third day of the profiling.

The overall network performance profiled using the NE-ONE emulator were very close to the overall performance measured by practical testbed using active experiment method and within similar bounds with the overall minimum value of Φ_{min}

= 963 ms, maximum value of Φ_{max} = 1414 ms, and an average of Φ_{avg} = 1033 ms. The main disparity occurred in the the large deviation, which might be caused by the differences in the data sample resolution, equipments and the different method employed.

Figure 4.21 showed the latency performance results obtained from the simulation experiments described in the previous section. The results from the simulated scenarios (SSL and STL) are compared in Fig. 4.21 and statistically summarised in Table 4.9. The overall simulated values of the latency (OWD) obtained using these scenarios with SSL (red points) having the highest latency value of 767 ms as compared with the lowest of 266 ms STLS (blue points) as given by the overall values of Table 4.9. The average latencies were 536 ms, 271 ms and 404 ms for SSLS, STLS and overall average respectively. Compared to the results obtained by practical and emulator profiled measurements, simulation gave the lowest latencies and standard deviations of 16.78 ms (3.13% of 536 ms) and 6.01 ms (2.22% of 271 ms), which indicates better performance compared to the other measurement methods. However, practical and emulator profiled results are closer to a real communication environment than the simulated environment. The RTT of the simulation results were also determined by $2 * \Phi$ (2xOWD) and summarised in Table 4.9.

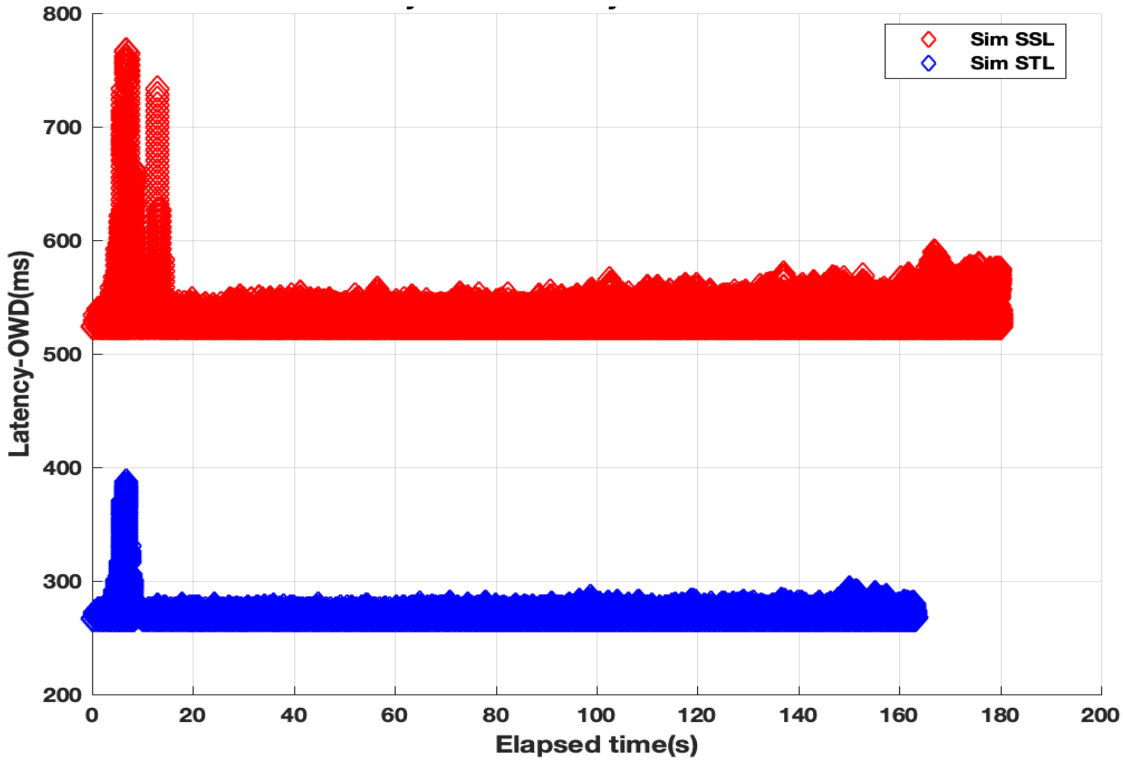


Figure 4.21: Network Simulation Latency Performance

Table 4.9: Simulated Latency Performance Summary

Scenario	$\Phi_{mx}(ms)$	$\Phi_{mn}(ms)$	$\Phi_{av}(ms)$	$\sigma_{sd}(ms)$	$RTT_{mx}(ms)$	$RTT_{mn}(ms)$	$RTT_{av}(ms)$
SSLS	767	524	536	16.78	1534	1048	1072
STLS	387	266	271	6.01	774	532	542
Overall	767	266	404		1534	532	807

4.4 Throughput Analysis

The results obtained from the measurements and discussed in the previous sections were used to analyse the achievable throughput *(R)* using the standard TCP algorithm scheme. The analysis is based on an ideal channel (i.e PER = 0) and the set values of the TCP parameters that include initial cwnd (IW) of 10 segs, ssthresh (γ) of 128 segs, and maximum segment size (MSS) of 1448 bytes.

The results in Fig. 4.22, 4.23, and 4.24 for practical, emulator profiled and simulation measurements showed how the RTT degraded the performance of communication by decreasing the instantaneous transmission rate (throughput) as the E2E latency increases. Three different overall RTT values namely; maximum, minimum and average were used to analyse the performance of TCP due to the increased latency. The summary of the performance were obtained for practical, profiled, and simulated E2E latencies and shown in Table 4.10, 4.11, and 4.12 respectively.

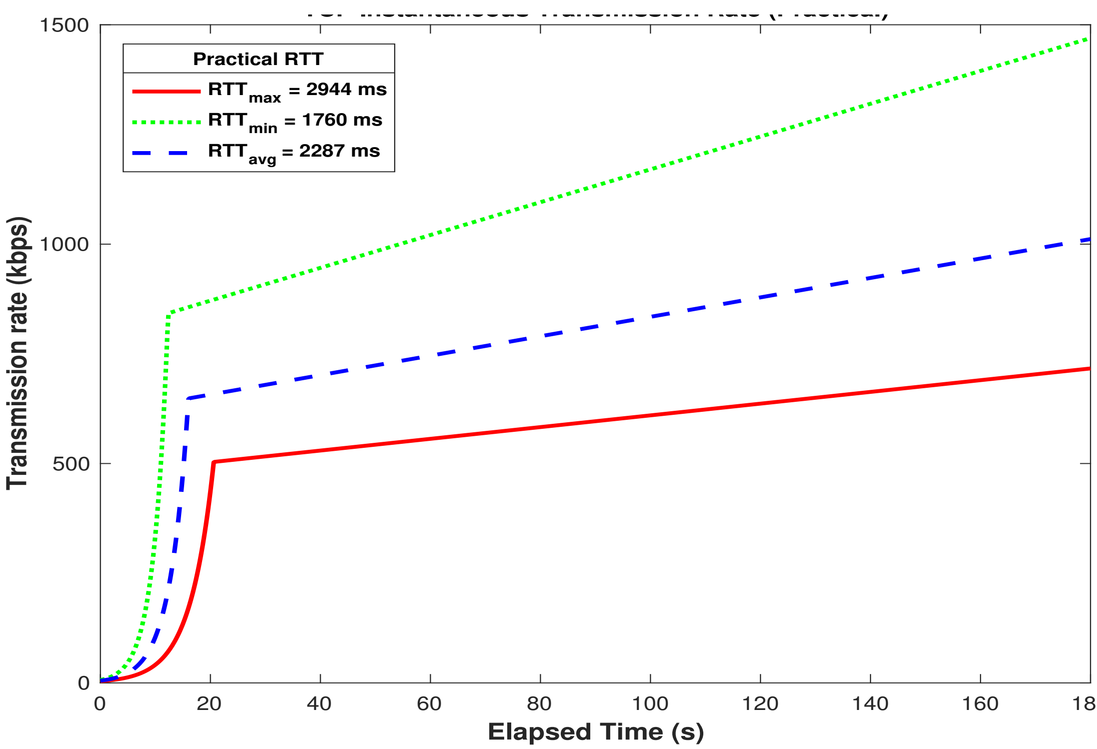


Figure 4.22: Instantaneous Transmission Rate (Throughput) of Practical RTT

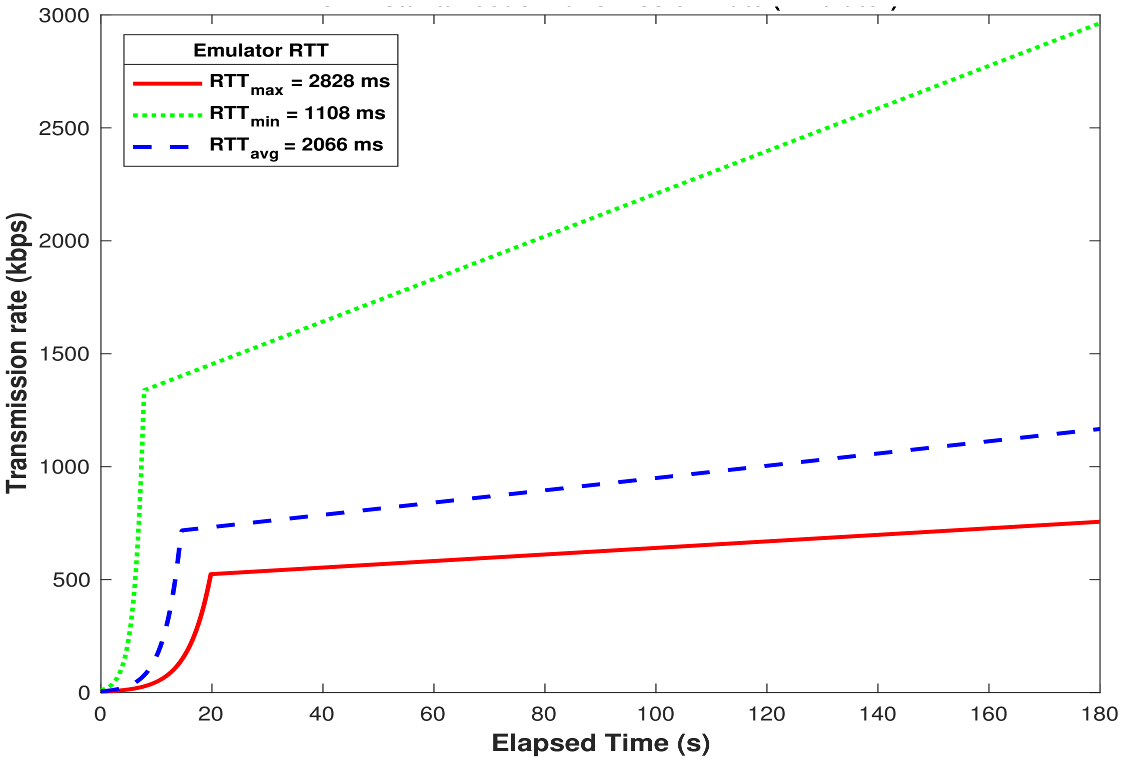


Figure 4.23: Instantaneous Transmission Rate (Throughput) of Profiled RTT

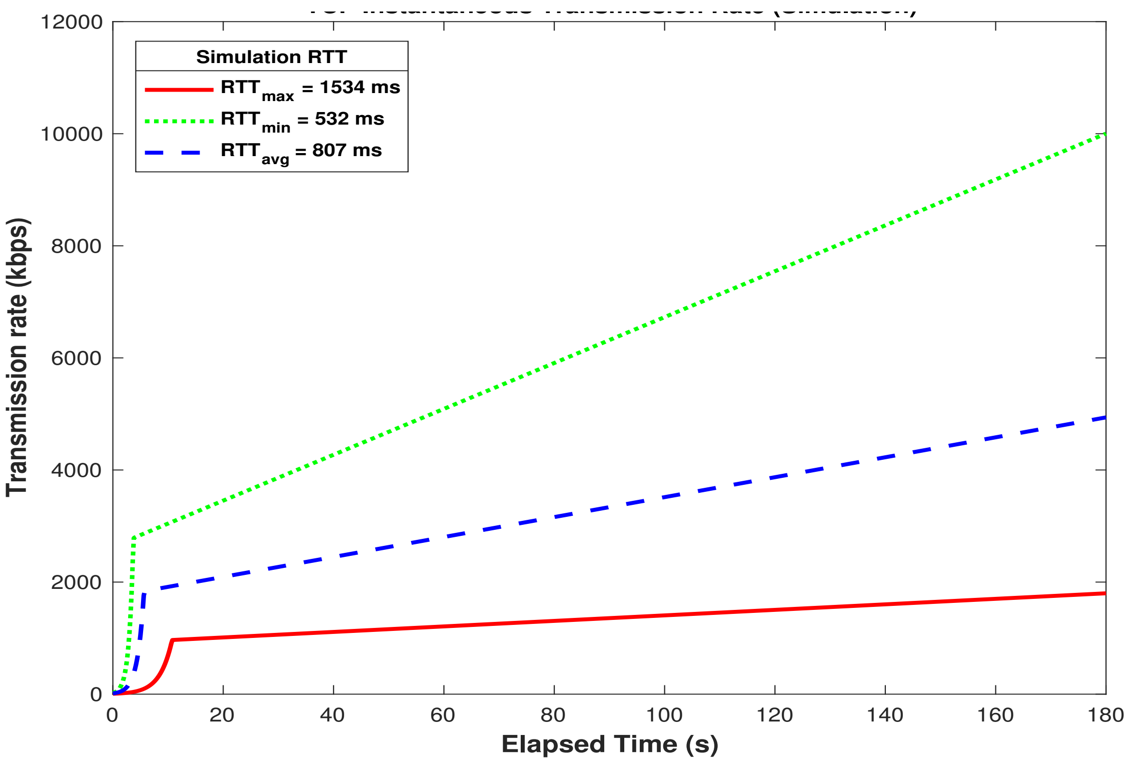


Figure 4.24: Instantaneous Transmission Rate (Throughput) of Simulation RTT

Table 4.10: Throughput Performance Summary by Overall Practical RTT

RTT	$R_{max}(kbps)$	$R_{min}(kbps)$	$R_{avg}(kbps)$
Maximum	717	4	552
Minimum	1470	7	1088
Average	1012	5	768
Overall	1470	4	803

Table 4.11: Throughput Performance Summary by Overall Profiled RTT

RTT	$R_{max}(kbps)$	$R_{min}(kbps)$	$R_{avg}(kbps)$
Maximum	756	4	582
Minimum	2964	11	2069
Average	1167	6	878
Overall	2964	4	1176

Table 4.12: Throughput Performance Summary of the Overall Simulation RTT

RTT	$R_{max}(kbps)$	$R_{min}(kbps)$	$R_{avg}(kbps)$
Maximum	1800	8	1312
Minimum	10002	22	6273
Average	4939	14	3293
Overall	10002	8	3626

Figure 4.22 and Table 4.10 showed the performance for practically measured latency with 1470 kbps the overall highest throughput achieved, an average of 803 kbps, and lowest value of 4 kbps using standard TCP like NewReno. The maximum throughput was achieved from the minimum delay RTT_{min} while the lowest was from the maximum delay, RTT_{max}. This clearly show (see Fig. 4.22) the negative impact of long latency in hybrid satellite network environments on the throughput and capacity efficient utilisation.

The same features were observed for profiled and simulated results in Fig. 4.23, and 4.24 respectively. Although better throughput performance was achieved with overall highest value of up to 10002 kbps (about 10 Mbps) with $RTT_{min} = 532$ ms from simulation as compared with the highest of 2964 kbps (about 3 Mbps) with $RTT_{min} = 1108$ ms from the profiling that is closer to the reality than simulation.

These results showed how much impact latency could have on the throughput performance and capacity utilisation of the next generation of High Throughput Satellites (HTS) and 5G communications networks with extremely high capacity.

The key similarities among all the results from practical to simulation are the overall lowest (R_{min}) values were between 4-8 kbps obtained from RTT_{max} values and the overall highest (R_{max}) between 1470 kbps (practical) and 10 Mbps (simulation) derived from RTT_{min}. The overall average throughput (R_{avg}) values were between 803 kbps (practical) and 3.6 Mbps (simulation). Compared with overall values obtained from the practical measurements (see Table 4.10), simulation results (see Table 4.12) were too high to be obtained in real heterogenous network environments involving at least a satellite leg.

Moreover, most satellite network environment latency values of 500-600 ms found in most literature were more closer to simulated values than the realistically practical values as measured and described in this thesis. In most cases, the E2E latency component contributed by communications between satellite space segment and gateway stations are among the most critical propagation latencies and operations like network operation, satellite control and satellite access carried out contributed an additional processing and switching delays.

References

[1] S. Kota, and M. Marchese, "Quality of Service for Satellite IP Networks: A Survey", *International Journal of Satellite Communications and Networking*, Vol. 21, pp. 303-349, John Wiley, 2003.

[2] J. Postel, "Transmission Control Protocol", *Defense Advanced Research Projects Agency (DARPA) Internet Program Protocol Specification , RFC 793*, DoD, 1981.

[3] V. Jacobson, and R. Braden, "TCP Extensions for Long-Delay Paths", *Network Working Group*, RFC 1072, 1988.

[4] V. Jacobson, R. Braden, and D. Borman, "TCP Extension for High Performance," *Internet Society Network Working Group,*,RFC 1323, 1992.

[5] T. Braun, M. Diaz, J. E. Gabeiras, and T. Staub, *End-to-End Quality of Service Over Heterogeneous Networks*, pp. 1-129, Springer, 2010.

[6] M. Marchese, *QoS Over Heterogeneous Networks*, John Wiley & Sons, England, 2007.

[7] G. Maral and M. Bousquet, *Satellite Communications Systems: Systems, Techniques and Technology*, 5th Ed. John Wiley, West Sussex, UK, 2009.

[8] Z. Mrak, and D. Ogrizovic, "Inmarsat Broadband Global Area Network", *Pomorstvo*, god. 19, pp. 265-274, 2005.

[9] A. A. Bisu, A. Purvis, K. Brigham, and H. Sun, " A Framework for End-to-End Latency Measurements in a Satellite Network Environment", *In Proc. IEEE International Conference on Communications (ICC)*, pp. 1-6, 2018.

[10] Cisco, "Understanding Dealy in Packet Voice Networks", White Papers,Cisco inc. 2008.

[11] X. Jin, Y. Hua, and Y. Cao, “Research on A New Method of Time Delay Measurement in Telephone Time Service,” in *Proc. IEEE International Frequency Control Symposium (IFCS)*, 19-22, 2014.

[12] K. Dudacek Jr., K. Dudacek, and V. Vavficka, “Comparison of Short Delay Measurement Methods,” in *Proc. International Conference on Applied Electronics (AE)*, Pilsen, Czech Republic, 8-9, 2015.

[13] P. Hinton, E. Baker, and C. Hill, “Latency Time for lawyers to get up to Speed?”, *Computer Law & Security Review*, vol. 28, no. 3, pp. 340- 346, 2012.

[14] J. W. Stahlhut, T. J. Browne, G. T. Heydt, and V. Vittal, "Latency Viewed as a Stochastic Process and its Impact on Wide Area Power System Control Signals,” *IEEE Transactions on Power Systems*, Vol. 23, No. 1, pp. 84-91, 2008.

[15] T. Eylen, and C. F. Bazlamacci, "One-Way Active Delay Measurement with Error Bounds,” *IEEE Transactions on Instrumentation and Measurement*, Vol. 64, No. 12, pp. 3476-3489, 2015.

[16] iTrinegy, "NE-ONE Network Emulator User and Administration Guide,” *iTrinegy Limited*,Version 3.0.0, 2017.

[17] iTrinegy, "NE-ONE Profiler User and Administration Guide,” *iTrinegy Limited*,Version 1.8.0, 2016.

[18] iTrinegy, "Technical Specification NE-ONE Desktop Appliance,” *iTrinegy Limited*,Ref. NE1-DESK-TECSPEC-04252018. Accessed Online on 18/04/2019 via https://itrinegy.com/itrinegy/wp-content/uploads/2011/05/NE-ONE-Desktop-Technical-Specifications-Sheet.pdf

[19] Netgear, "Gigabit Ethernet Smart Managed Plus Switches User Manual,” *NETGEAR, Inc.*,202-11700-05. Accessed Online on 18/04/2019 via http://www.downloads.netgear.com/files/GDC/GS105EV2/WebManagedSwitches_UM_EN

Chapter 5

TCP Optimisation: Modelling, Implementation and Testing

This chapter presents modelling, implementation and testing of the optimum TCP over heterogeneous (IST) networks involving GEO satellite links using numerical simulations and protocol implementation for the analysis and evaluation of the performance of our improved algorithm proposal. This was and compared with other optimised TCP algorithms designed to solve the performance degradation issue of TCP over pure satellite and heterogeneous network environments such as ISTN that incorporate at least a satellite leg.

5.1 Mathematical Model

The mathematical model described and proposed in this thesis is based on the standard, HYBLA and CUBIC TCP congestion algorithms. These were to improve the performance of TCP over long RTT and large BDP networks such as a hybrid SatCom network environment. The key aim of this model is to mitigate the impact of long RTT and large BDP environment involving at least one satellite link.

To achieve optimum performance with TCP over long RTT channels like satellite with the focus on improving the congestion window (evolution) growth rate $W(t)$, which has negative impact on the throughput performance with increase RTT. The TCP instantaneous transmission rate (throughput) $R(t)$ is given by Eq. (5.1.1).

$$R(t) = \frac{W(t)}{RTT} \tag{5.1.1}$$

where $W(t)$ is window evolution for SS and CA phases of the standard TCP algorithm is given by Eq. (5.1.2) as detailed in chapter 3.

$$W(t) = \begin{cases} 2^{\frac{t}{RTT}} & 0 \leq t < t_\gamma \quad SS\ (Exponent) \\ \frac{t-t_\gamma}{RTT} + \gamma & t \geq t_\gamma \quad CA\ (Linear) \end{cases} \tag{5.1.2}$$

Where t is the elapsed time since the data transmission started and t_γ is the time that the *ssthresh* value, γ is reached in a given RTT, computed from Eq. (5.1.3).

$$t_\gamma = RTT \log_2(\gamma) \tag{5.1.3}$$

The instantaneous *throughput* $R(t)$ given by Eq. (5.1.4) for standard TCP was derived by substituting $W(t)$ given by Eq. (5.1.2) in Eq. (5.1.1).

$$R(t) = \begin{cases} \frac{2^{t/RTT}}{RTT}; & 0 \leq t < t_\gamma \quad SS \\ \frac{1}{RTT}(\frac{t-t_\gamma}{RTT} + \gamma); & t \geq t_\gamma \quad CA \end{cases} \tag{5.1.4}$$

However, to achieve optimum TCP performance and efficient utilisation of the capacity (bandwidth) C over a long RTT and large BDP channel, $R(t)$ needs to be maximised by making $W(t)$ independent of the RTT, particularly in a heterogeneous network environment where longer RTT connection are penalised and disadvantaged by competing shorter RTT flows [1–4, 6–11, 13].

The constraints of achieving maximum throughput in TCP is the connection RTT and the channel available capacity C [2, 9–11, 13, 16, 17]. The optimum solution obtained by careful maximisation of $W(t)$ through minimisation of RTT impact, as shown by Eqs. 5.1.5, 5.1.6 and 5.1.7 [12, 14]. Figure 5.1 is an ITU reference star topology model adopted for the TCP performance optimisation by this thesis.

$$\overset{max}{R \geq 0} \sum_{i=0}^{N} W_i(R_i) \tag{5.1.5}$$

Subject to constraints given by (5.1.6) and (5.1.7)

$$W \leq cwnd \tag{5.1.6}$$

$$R \leq C \tag{5.1.7}$$

where R is the aggregate throughput achieved and R_i is the average throughput (instantaneous transmission rate) of the i^{th} flow of the TCP source, t is the latency, W is the current window growth rate obtained from the measured congestion window, $cwnd$ of the flow path and the receiver advertised window, $rawnd$, usually derived from $W = min(cwnd, rawnd)$ in algorithms such as TCP Reno, and C is the capacity/bandwidth of i^{th} link L_i, along the TCP connection path as shown in Fig. 5.1. L_i is the bottleneck link capacity in a heterogeneous network path for obtaining the optimum value of R_i subject to the constraints in Eq. (5.1.6), at the transport layer. Considering the physical layer, R is also is limited by the channel capacity C as expressed by (5.1.7). However, the optimum transmission rate of TCP source considered in this thesis considered the constraints in Eq. (5.1.6).The channel utilisation efficiency is computed using Eq. (5.1.8).

$$\eta(\%) = \frac{R_i}{C_i}100 \tag{5.1.8}$$

The original HYBLA and CUBIC algorithms discussed in chapter 4 were modified and integrated to achieve optimum TCP performance with minimised impacts of RTT on both the $W(t)$ and $R(t)$. A normalised RTT, ρ given by Eq. (5.1.9) was used as in original HYBLA with modified and realistic reference value of RTT_{ref} of at least 100 ms such as realistic 4G network. This was aimed at achieving the same $R(t)$ for both longer RTT_{sat} and short $RTT_{ref} = 100$ ms by making $W(t)$ independent of the RTT_{sat} through elapsed time scale, t modification with ρt in Eq. (5.1.2) and compensating the effect of the RTT_{sat} by multiplying by ρ the resulting $W(t)$ from time scaling just like the steps in HYBLA [10, 11].

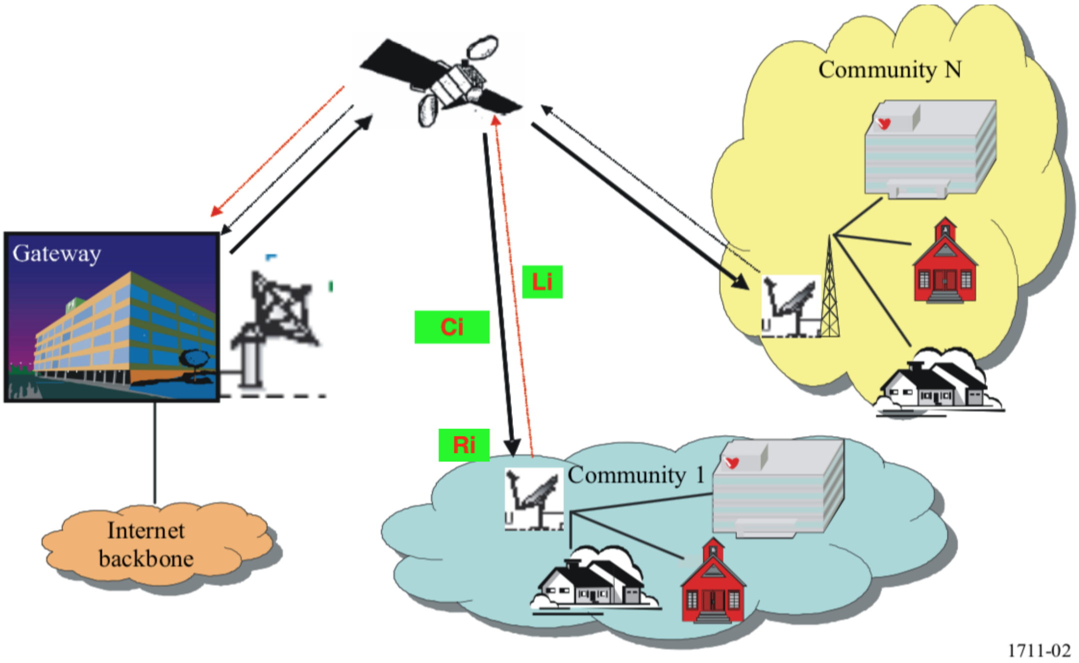


Figure 5.1: Reference Model Star topology (adopted from [16, 17])

Therefore, using the first step (elapsed time scaling) for the original HYBLA SS phase and modification of the CA phase using CUBIC function without time scaling, since CUBIC was originally independent of RTT and scaled by a factor of C. This yielded an improved congestion control algorithm with $W^m t$ given in Eq. (5.1.10) with modified SS and CA algorithms. Moreover, compensating the effect of RTT in the next step resulted to $R^m(t)$ independent of the RTT_{sat} as given in Eq. (5.1.11), which was derived by substituting resulting $W^m(t)$ in Eq.(5.1.1).

$$\rho = \frac{RTT_{sat}}{RTT_{ref}} \tag{5.1.9}$$

$$W^m(t) = \begin{cases} \rho * 2^{(\frac{t}{RTT_{ref}})}, & 0 \leq t < t_\gamma, \quad SS\ (HYBLA) \\ C(t - \sqrt[3]{\frac{\beta W_{max}}{C}})^3 + W_{max}, & t \geq t_\gamma, \quad CA\ (CUBIC) \end{cases} \tag{5.1.10}$$

$$R^m(t) = \begin{cases} \frac{2^{(t/RTT_{ref})}}{RTT_{ref}}, & 0 \leq t < t_{\gamma,ref}, \quad SS \\ \frac{1}{RTT_{ref}}(C(t - \sqrt[3]{\frac{\beta W_{max}}{C}})^3 + W_{max}), & t \geq t_{\gamma,ref}, \quad CA \end{cases} \tag{5.1.11}$$

where C is a constant (usually 0.4) scaling factor that determines the aggressiveness of window increase in large BDP network environments like hybrid satellite networks, t is elapsed time from the last window reduction at the event of packet loss (the beginning of the current CA) or since the data transmission start under ideal conditions without packet loss, W_{max} (origin point) is window size just before the last window reduction at the event of loss or the maximum at the end of the SS phase ($t \geq t_{\gamma,ref}$), β is a constant Multiplicative Decrease (MD) factor for window reduction at the time of loss, which replaced the halving window in the standard TCP algorithms [13, 20].

The factor $\sqrt[3]{\frac{\beta W_{max}}{C}}$ (K in CUBIC) is a constant that determines how *slow* or *fast* the *cwnd* size increases or decreases, i.e the time interval CUBIC window function takes to increase W_i to W_{max} when no further loss event occurred within that period as described in chapter 3 [13, 20]. However, the *cwnd* update rule, W_{i+1} remains the same as in original HYBLA for SS phase and CUBIC for CA phase as detailed in chapter 3 of this thesis.

As a result of the modifications shown in Eqs (5.1.10) and (5.1.11), the time at which *cwnd (W)* reaches the *ssthresh*, γ given by Eq. (5.1.3), which serves as switching time, $t_{\gamma,ref}$ between modified SS and CA algorithms is redefined as the time at which W reaches the value $\rho\gamma$ and rewritten as Eq. (5.1.12) to reflect the compensation of the effect of division by RTT and also include an error merging of $RTT_{ref} \log_2(\rho)$ compared to HYBLA and other algorithms.

$$t_{\gamma,ref} = RTT_{ref} \log_2(\rho\gamma) = RTT_{ref} \log_2(\rho) + RTT_{ref} \log_2(\gamma) \tag{5.1.12}$$

This switching time, $t_{\gamma,ref}$ is different for different RTT_{sat} contrary to the HYBLA algorithm where it is the same for every RTT value [10]. In this case, as the RTT_{sat} increases the switching time increases additively by a factor of $t_{\rho,ref} = RTT_{ref} \log_2(\rho)$ as shown in Eq. (5.1.12).

5.2 Numerical Simulation and Analysis

The numerical simulations and analysis of the modified mathematical model in Eqs. (5.1.10), (5.1.11) and (5.1.12)were conducted by implementing the modified SS and CA algorithms in MATLAB under ideal (error free) channel connection. This new implementation called HYBIC was compared with both HYBLA and CUBIC algorithms under the same channel conditions to evaluate throughput performance and capacity utilisation.

5.2.1 HYBIC Implementation and Analysis

HYBIC is the new modification derived from both TCP HYBLA and CUBIC algorithms to enhance the performance of TCP over long RTT channels such as satellite. The implementation was analysed using numerical simulation under ideal channel conditions, different values of long RTT_{sat} measured in chapter 4, realistic fast reference RTT_{ref} and varied TCP parameters such as *ssthresh* (γ), initial *cwnd (IW)*, maximum segment size (MSS) and elapsed time, t were used for the simulation.

The results shown in this section were obtained with realistic values of RTT_{ref} from 25-100 ms in step of 25 ms and three worst case scenario practical values of RTT_{sat} (maximum, minimum and average) measured experimentally and discussed in chapter 4. Moreover, IW is set to 10 seg, γ of 128 seg, MSS of 1448 bytes and elapsed time t starting at 0 s with step of 200 ms.

The window evolution, $W^m(t)$ shown in Fig. 5.2 was the result of the numerical implementation of the modified algorithm called HYBIC Eq. (5.1.10) of the previous section) using a fast reference RTT_{ref} =25 ms. The implementation was subjected to three high RTT_{sat} measured practically as maximum (RTT_{max} = 2944 ms), minimum (RTT_{min} = 1760 ms) and average (RTT_{avg} = 2287 ms) as shown in Fig. 5.2. The window rate, $W(t)$ under RTT_{min} reached the *ssthresh* (γ = 128 seg) faster compared to RTT_{max} and RTT_{avg} values, but high window evolution, $W(t)$ values were achieved at these high RTT values. These high values of $W(t)$ led to a higher instantaneous transmission rate, $R^m(t)$ as described by the algorithm of Eq. (5.1.11) and shown by the results of Fig. 5.3.

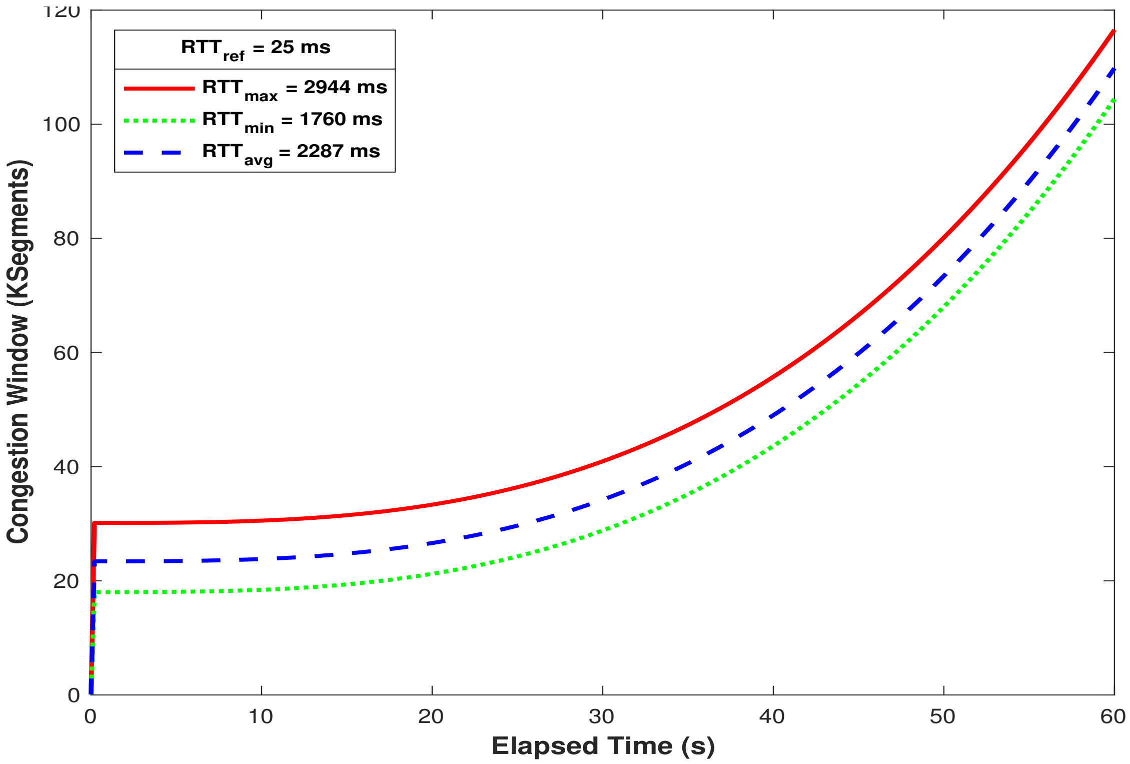


Figure 5.2: TCP HYBIC Window Evolution, $W^m(t)$ at $RTT_{ref} = 25\ ms$

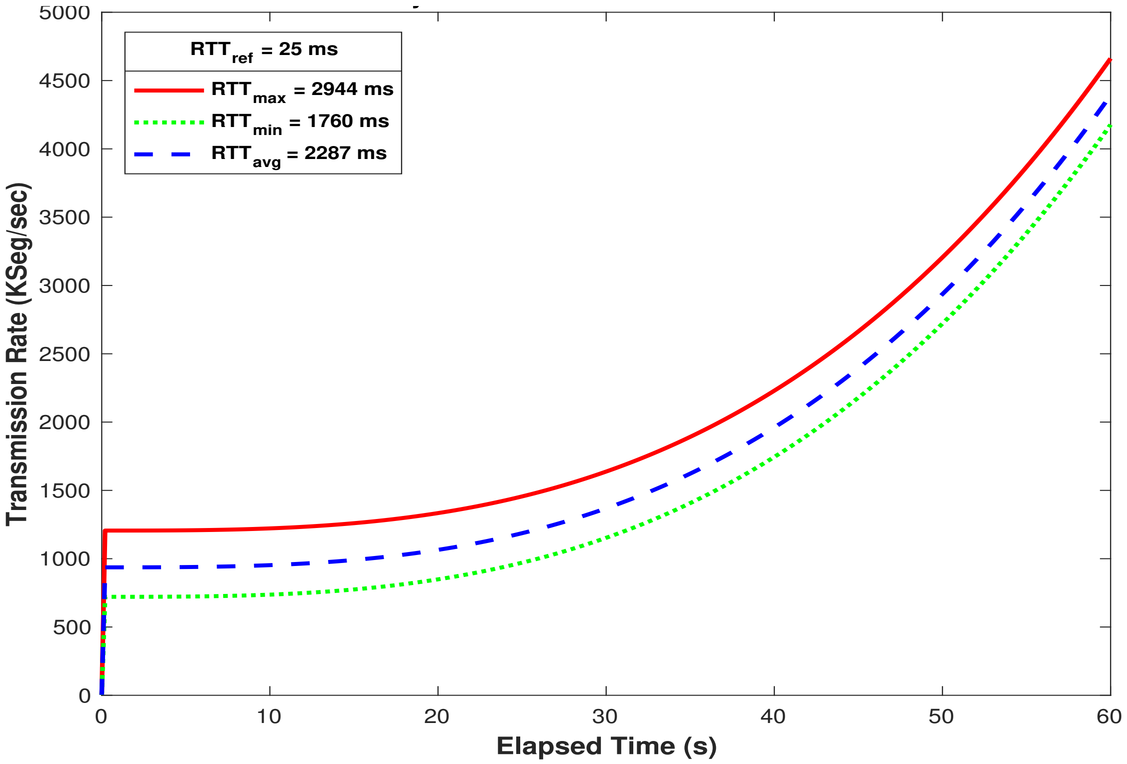


Figure 5.3: TCP HYBIC Transmission Rate, $R^m(t)$ at $RTT_{ref} = 25\ ms$

However, similar growth in both window and transmission rates were observed when the fast reference path (RTT_{ref}) is increased from 25 ms to 100 ms as shown in Fig. 5.4 and 5.5 in step of a 25 ms. These results also show that the negative impact of high satellite latency (RTT_{sat}) is removed as the algorithm is almost independent of the high RTT satellite path when the RTT_{ref} increases from 25 ms to 100 ms. Moreover, both the SS and CA algorithm phases of $W^m(t)$ and $R^m(t)$ tend to converge with the increase in RTT_{ref} as shown in Fig. 5.4 and 5.5.

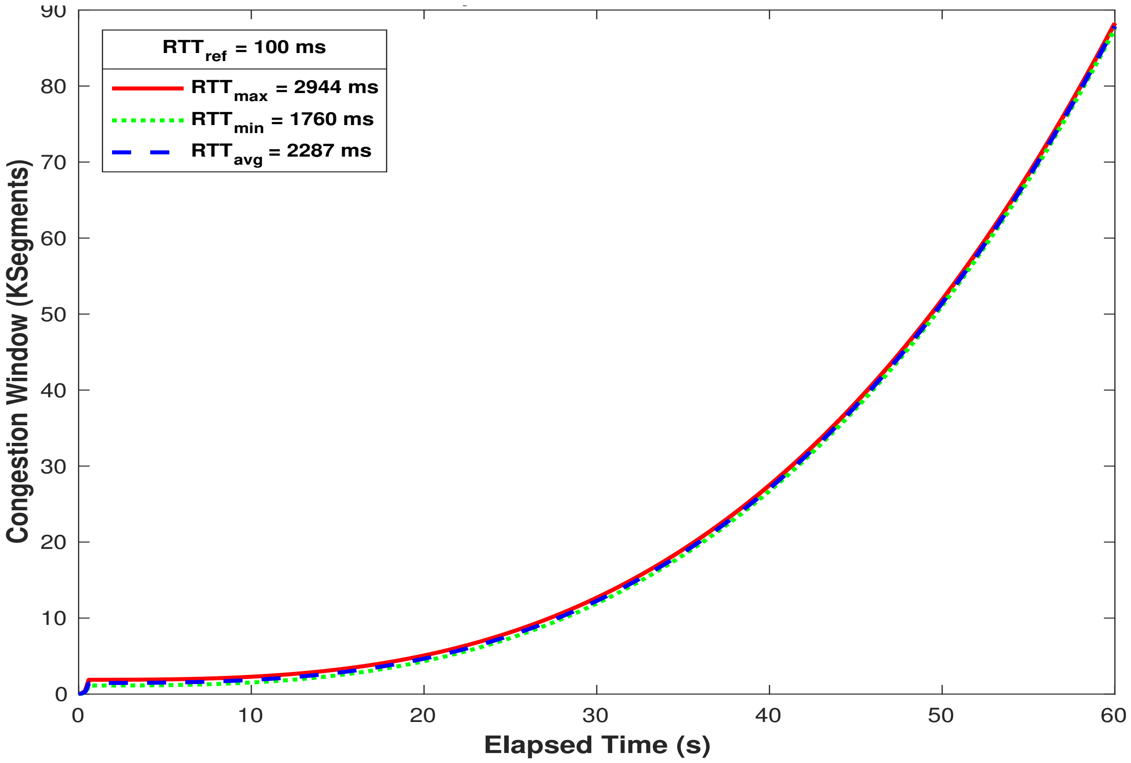


Figure 5.4: TCP HYBIC Window Evolution, $W^m(t)$ at $RTT_{ref} = 100\ ms$

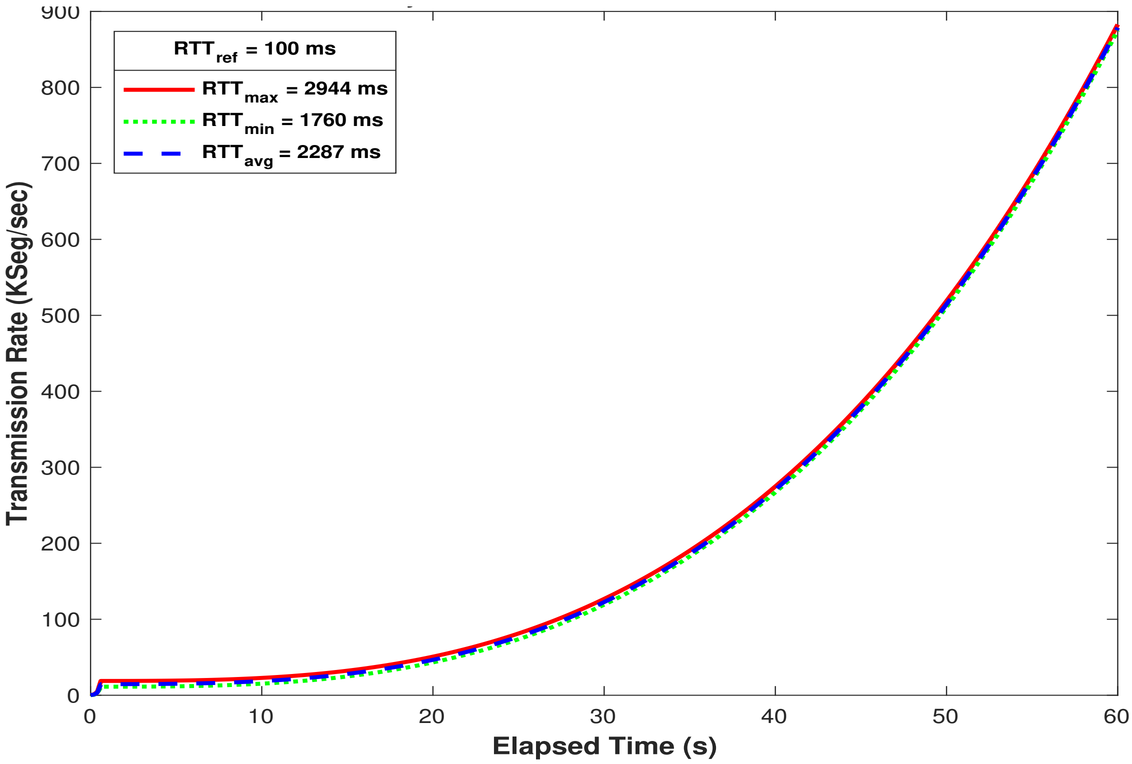


Figure 5.5: TCP HYBIC Transmission Rate, $R^m(t)$ at $RTT_{ref} = 100\ ms$

High performance was achieved with HYBIC by removing the dependence of $W^m(t)$ and $R^m(t)$ on long RTT_{sat} as shown in Fig. 5.4 and 5.5. A statistical summary of the HYBIC performance based on the lower (25 ms) and upper (100 ms) bounds considered is given in Table 5.1 and 5.2. Overall, HYBIC achieved up to $W^m_{max}(t) = 117$ kseg with $RTT_{ref} = 25$ ms and the lowest was of $W^m_{min} = 18$ seg with $RTT_{ref} = 100$ ms and an average between $W^m_{avg} = 23$-46 kseg for RTT_{ref} between 25-100 ms.

Table 5.1: HYBIC Window Evolution at RTT_{ref} of 25 ms and 100 ms

$RTT_{sat}(ms)$	$W^m_{max}(kseg)$	$W^m_{min}(seg)$	$W^m_{avg}(kseg)$	$W^m_{max}(kseg)$	$W^m_{min}(seg)$	$W^m_{avg}(kseg)$
2944	117	118	52	88.2	29	24
1760	104	70	40	87.5	18	23
2287	110	92	45	87.9	23	23
Overall	117	70	46	88.2	18	23

The improvement of *cwnd* rate replicated with the high achievable throughput or transmission rate $R^m(t)$ shown in Fig. 5.3 and 5.5 for RTT_{ref} values of 25 ms and

Table 5.2: HYBIC Transmission Rate at RTT_{ref} of 25 ms and 100 ms

$RTT_{sat}(ms)$	$R^m_{max}(Gbps)$	$R^m_{min}(Mbps)$	$R^m_{avg}(Gbps)$	$R^m_{max}(Gbps)$	$R^m_{min}(Mbps)$	$R^m_{avg}(Gbps)$
2944	54	55	24	10	3	3
1760	48	33	18	10	2	3
2287	51	42	21	10	3	3
Overall	54	33	21	10	2	3

100 ms respectively. The statistical summary of achieved throughput with HYBIC is given in Table 5.2, with highest achieved transmission rate of up to $R^m_{max}(t) =$ 54 Gbps at $RTT_{ref} =$ 25 ms and $RTT_{max} =$ 2944 ms, while the lowest achieved was $R^m_{min}(t) =$ 2 Mbps at $RTT_{ref} =$ 100 ms and $RTT_{min} =$ 1760 ms, the average transmission rate $R^m_{avg}(t)$ achieved was between 3 Gbps at $RTT_{ref} =$ 25 ms and 21 Gbps at $RTT_{ref} =$ 100 ms as shown in Table 5.2.

Figures 5.6 and 5.7 showed the changes and convergence of $W^m(t)$ and $R^m(t)$ by changing the RTT_{ref} from 25 ms (see Fig. 5.6 (a) and (b)) to 50 ms (see Fig. 5.6 (c) and (d)), while 75 ms and 100 ms is given by Fig. 5.7.

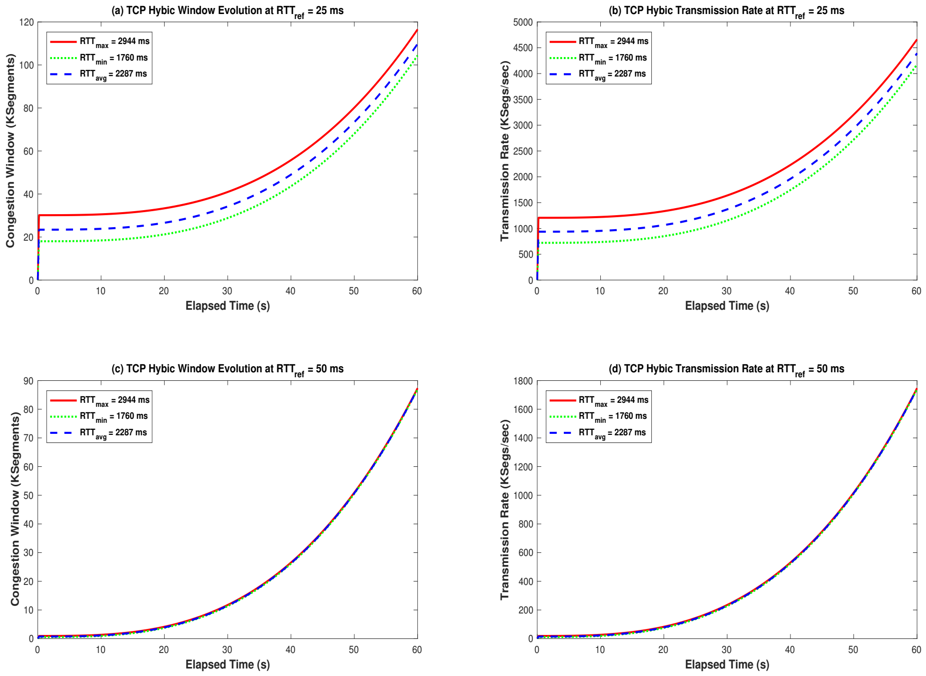


Figure 5.6: TCP HYBIC $W^m(t)$ and $R^m(t)$,, at RTT_{ref} of 25 ms and 50 ms

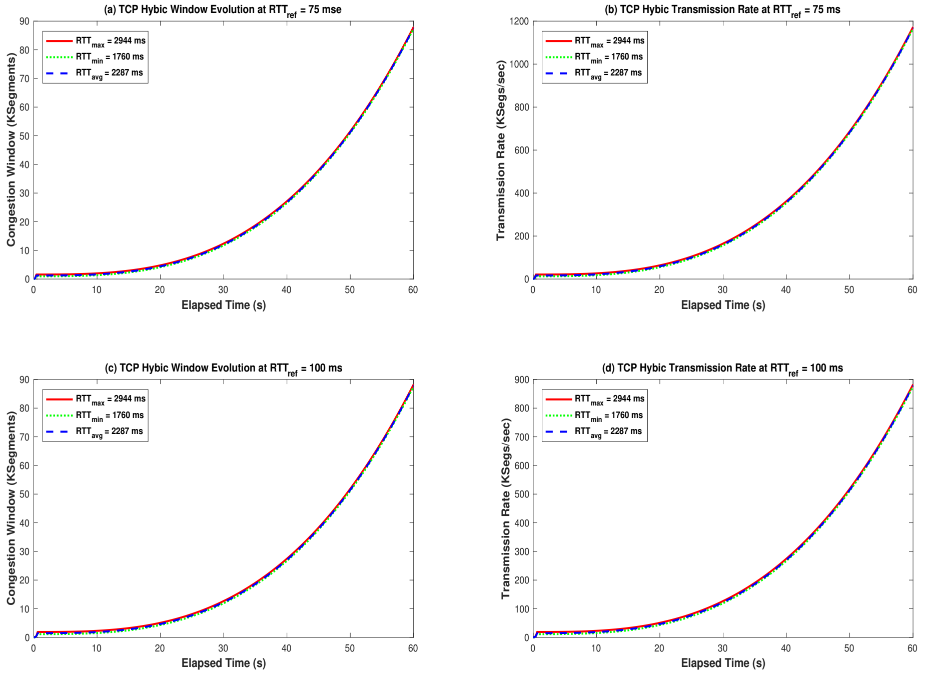


Figure 5.7: TCP HYBIC $W^m(t)$ and $R^m(t)$, at RTT_{ref} of 75 ms and 100 ms

When the RTT_{ref} exceeded 25 ms, the convergence of different RTT_{sat} curves become eminent and confirms the independence of both $W^m(t)$ and $R^m(t)$ on it. The overall values given in Tables 5.1 and 5.2 showed that increasing the value of RTT_{ref} from 25 ms to 100 ms has a negative impact on both the window growth and transmission rates. However, HYBIC still achieved by far better performance than the best performance achieved with standard TCP, HYBLA and CUBIC as discussed in the following sections.

5.2.2 HYBIC and HYBLA

The performance of *HYBIC* compared to *HYBLA* was observed by changing the values of RTT_{ref} with both algorithms almost independent of the RTT_{sat} as shown in Fig. 5.8 to 5.13. Careful observation of these figures of changing RTT_{ref}, window growth and transmission rates showed HYBIC has better performance and is more stable in both SS and CA phases with an increase in RTT_{ref} and RTT_{sat}, while HYBLA performance and stability reduced with increase in the values of these

$RTTs$ as shown by the green curves of the figures.

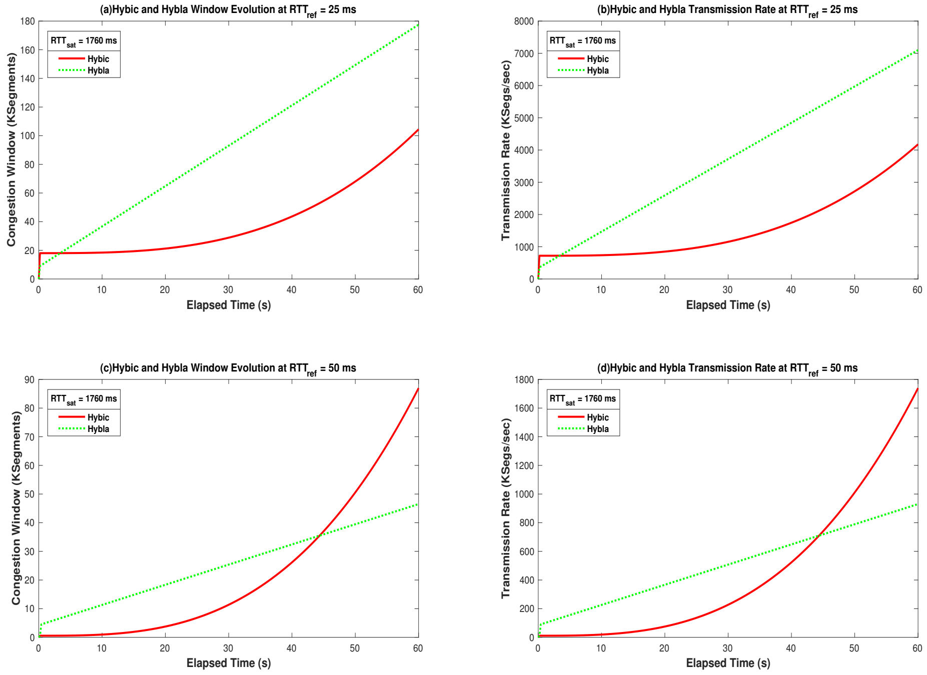


Figure 5.8: HYBIC and HYBLA $W(t)$ and $R(t)$ at RTT_{ref} of 25 ms and 50 ms

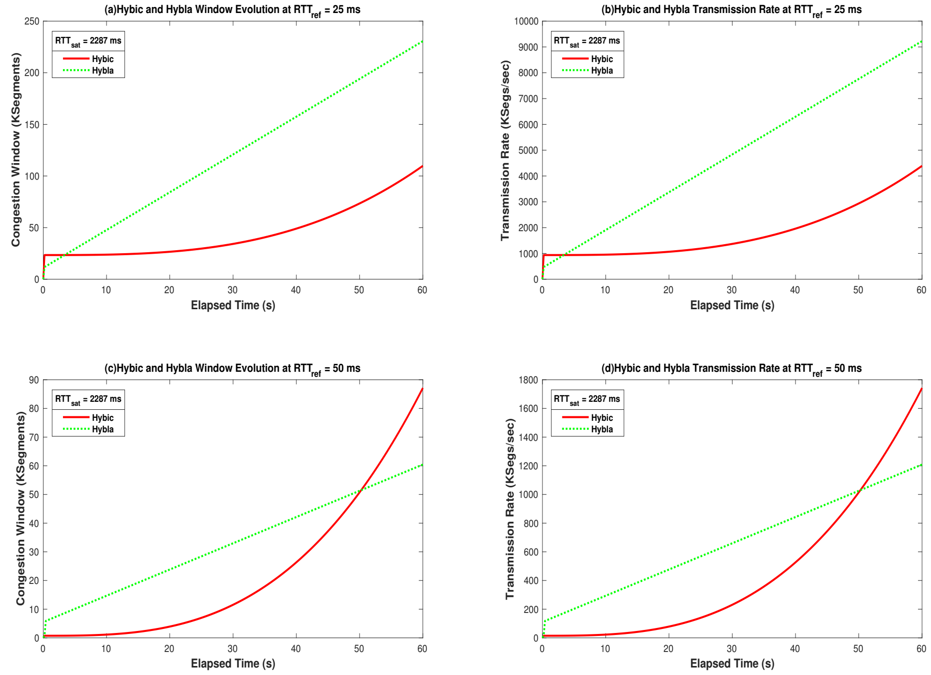


Figure 5.9: HYBIC and HYBLA $W(t)$ and $R(t)$, at RTT_{ref} of 25 ms and 50 ms

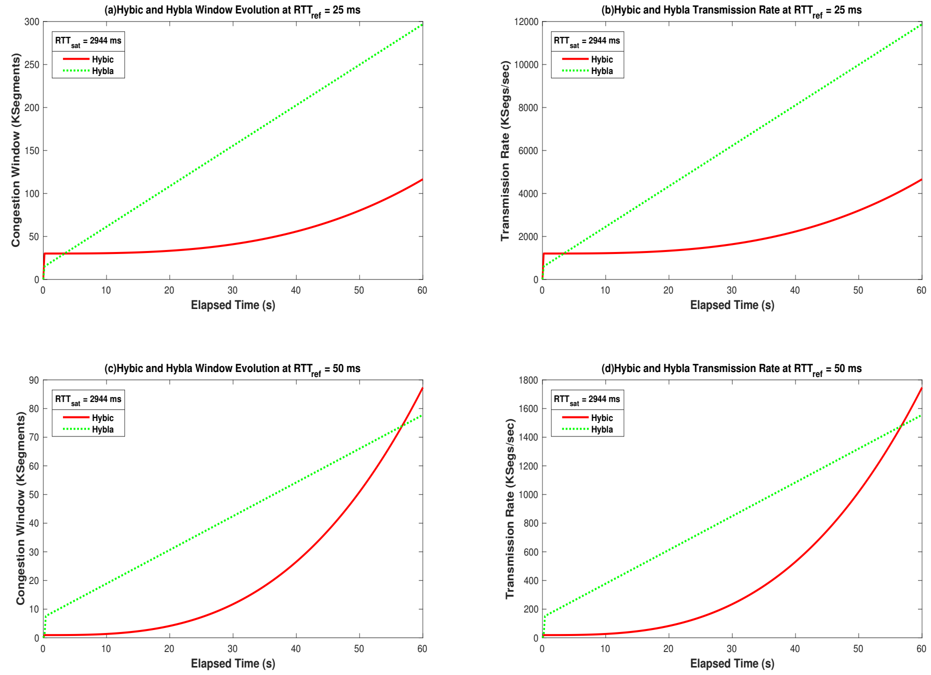


Figure 5.10: HYBIC and HYBLA $W(t)$ and $R(t)$,, at RTT_{ref} of 25 ms and 50 ms

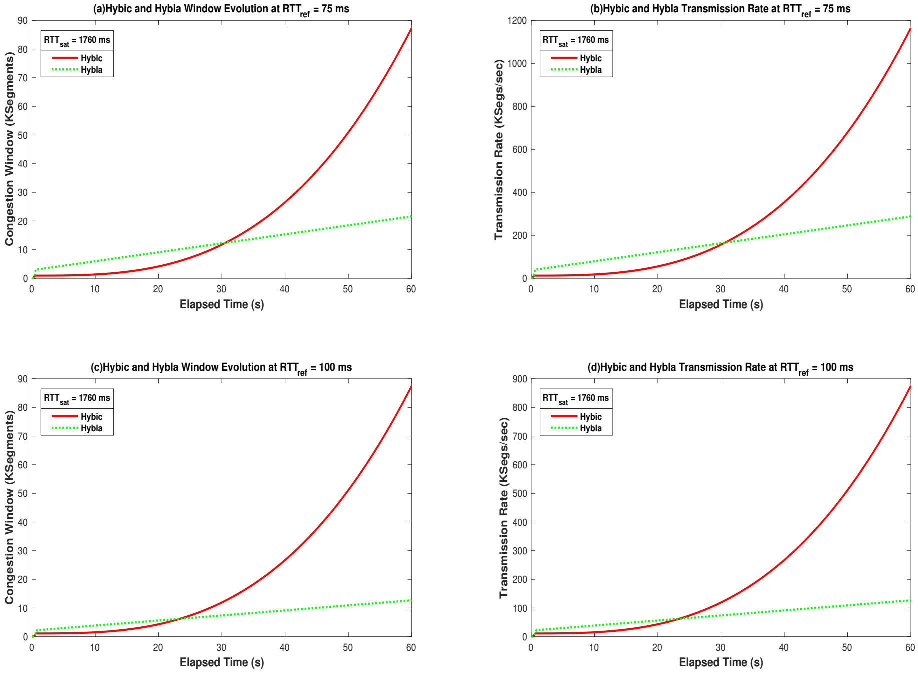


Figure 5.11: HYBIC and HYBLA $W(t)$ and $R(t)$, at RTT_{ref} of 75 ms and 100 ms

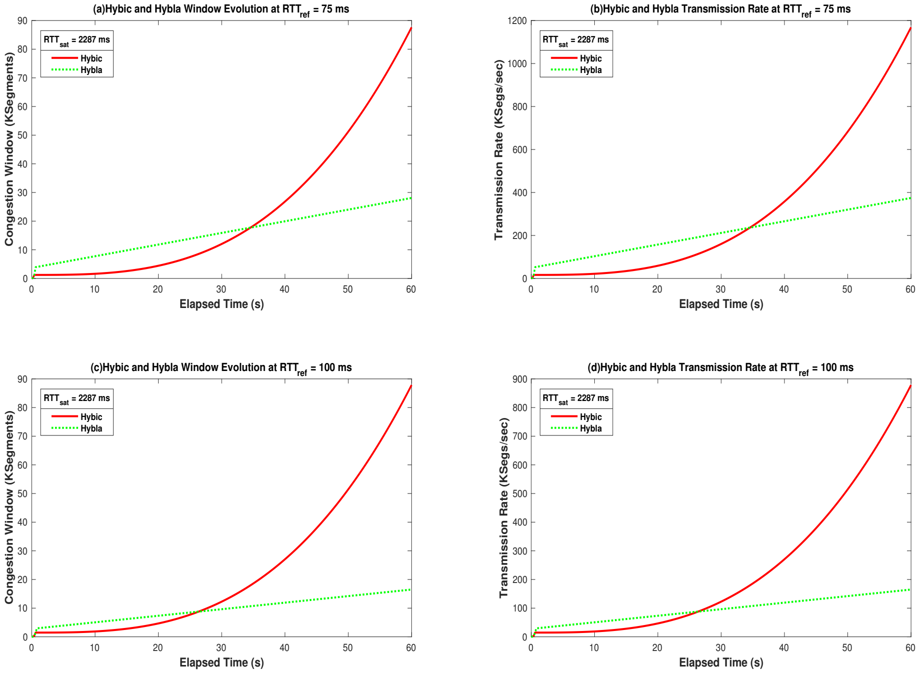


Figure 5.12: HYBIC and HYBLA $W(t)$ and $R(t)$, at RTT_{ref} of 75 ms and 100 ms

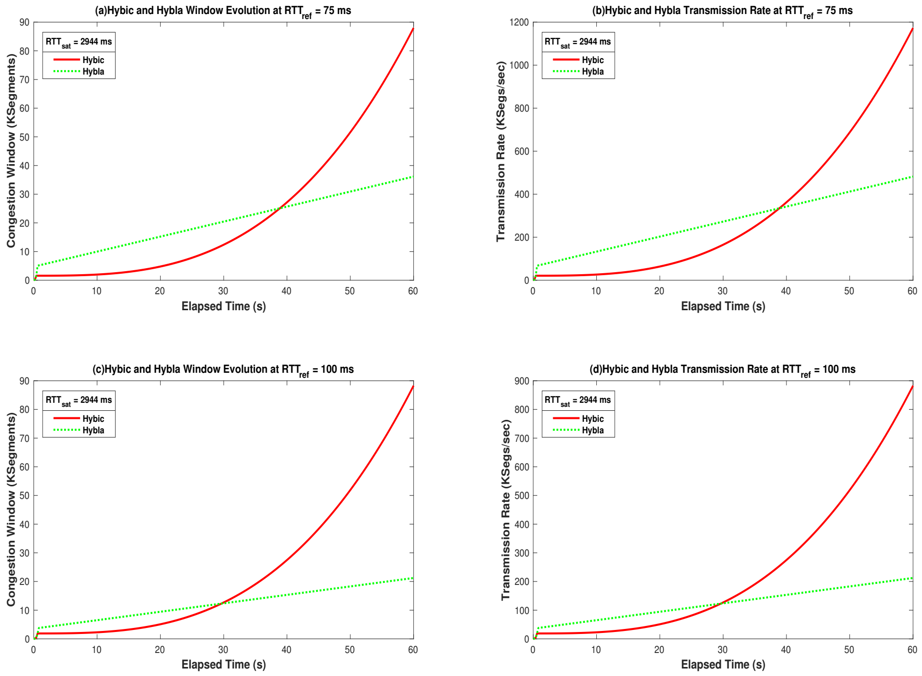


Figure 5.13: HYBIC and HYBLA $W(t)$ and $R(t)$, at RTT_{ref} of 75 ms and 100 ms

The performance of HYBLA degraded with the increase in the RTT values while HYBIC performance improved under the same conditions. These show that HYBIC performance had less dependency on all the RTT values. Moreover, HYBLA spent more time in SS phase than HYBIC (less than 1 s in all cases) and *cwnd* growth was dominated by the CA phase and HYBIC growth is not as aggressive as in HYBLA.

Tables 5.3 and 5.4 showed a statistical summary of the HYBLA performance for values of RTT_{sat} and RTT_{ref} of 25 ms (smallest) and 100 ms (highest) values. The overall performance obtained by numerical simulation and summarised in these tables showed W^h_{max}=297 kseg compared to HYBIC W^m_{max}=117 kseg (see Table 5.1) that correspond to R^h_{max} = 138 Gbps and R^m_{max} = 54 Gbps at 25 ms as shown in Table 5.2. The highest averages at this RTT_{ref} were W^h_{avg}=123 kseg compared to W^m_{avg}=46 kseg that correspond to R^h_{avg} = 57 Gbps and R^m_{avg} = 21 Gbps, these confirmed the better performance of HYBLA at RTT_{ref} = 25 ms.

However, increasing the value of RTT_{ref} = 100 ms showed decrease performance in HYBLA while HYBIC performance improved with W^h_{max}=21 kseg compared to

Table 5.3: HYBLA Window Evolution at RTT_{ref} of 25 ms and 100 ms

$RTT_{sat}(ms)$	$W^h_{max}(kseg)$	$W^h_{min}(seg)$	$W^h_{avg}(kseg)$	$W^h_{max}(kseg)$	$W^h_{min}(seg)$	$W^h_{avg}(kseg)$
2944	297	118	156	21	29	12
1760	178	70	93	13	18	7
2287	231	92	121	17	23	10
Overall	297	70	123	21	18	10

Table 5.4: HYBLA Transmission Rate at RTT_{ref} of 25 ms and 100 ms

$RTT_{sat}(ms)$	$R^h_{max}(Gbps)$	$R^h_{min}(Mbps)$	$R^h_{avg}(Gbps)$	$R^h_{max}(Gbps)$	$R^h_{min}(Mbps)$	$R^h_{avg}(Gbps)$
2944	138	55	72	3	3	1.4
1760	82	33	43	2	2	1
2287	107	42	56	2	3	1.1
Overall	138	33	57	3	2	1.2

W^m_{max}=88 kseg that correspond to R^h_{max} = 3 Gbps and R^m_{max} = 10 Gbps.The highest averages at this RTT_{ref} were W^h_{avg}=10 kseg compared to HYBIC W^m_{avg}=23 kseg that correspond to R^h_{avg} = 1.2 Gbps and R^m_{avg} = 3 Gbps. Table 5.5 gave the overall performance comparison and percentage improvement (when RTT_{ref}=100 ms) summary of HYBLA and HYBIC.

Table 5.5: HYBLA and HYBIC Overall $R(t)$ Comparison at RTT_{ref} of 25 ms and 100 ms

Algorithm	$R_{max}(Gbps)$	$R_{min}(Mbps)$	$R_{avg}(Gbps)$	$R_{max}(Gbps)$	$R_{min}(Mbps)$	$R_{avg}($
HYBLA	138	33	57	3	2	1.
HYBIC	54	33	21	10	2	3
Diff	84	0	36	7	0	1.
Improvement(%)	−61	0	−63	233	0	15

5.2.3 HYBIC and CUBIC

The performance of newly proposed *HYBIC* and the existing TCP *CUBIC* were also compared in Fig. 5.14 to 5.19 and summarised in Tables 5.6 through 5.8. The *cwnd* growth rate of HYBIC and CUBIC converged more quickly especially at RTT_{ref}

above 25 ms with fast exit of SS phase by $HYBIC$ as shown in Fig. 5.14 to 5.16. Increasing the RTT_{ref} (from 50 ms) and RTT_{sat} made the two curves, $W(t)$ of HYBIC (red) and CUBIC (blue) to converged more, particularly in the CA phase, while the instantaneous transmission rate of HYBIC ($R^m(t)$) remained highest compared to that of CUBIC ($R^c(t)$). The CUBIC window growth rate, $W^C(t)$ was independent of RTT and depends largely on the elapsed time, t, maximum window at the end of the SS phase and other constants ($C,\ K, and\ \beta$) values described in chapter 4. The performance of HYBIC in terms of window growth and transmission rates were better than that of CUBIC as summarised in Tables 5.6 and 5.7 respectively.

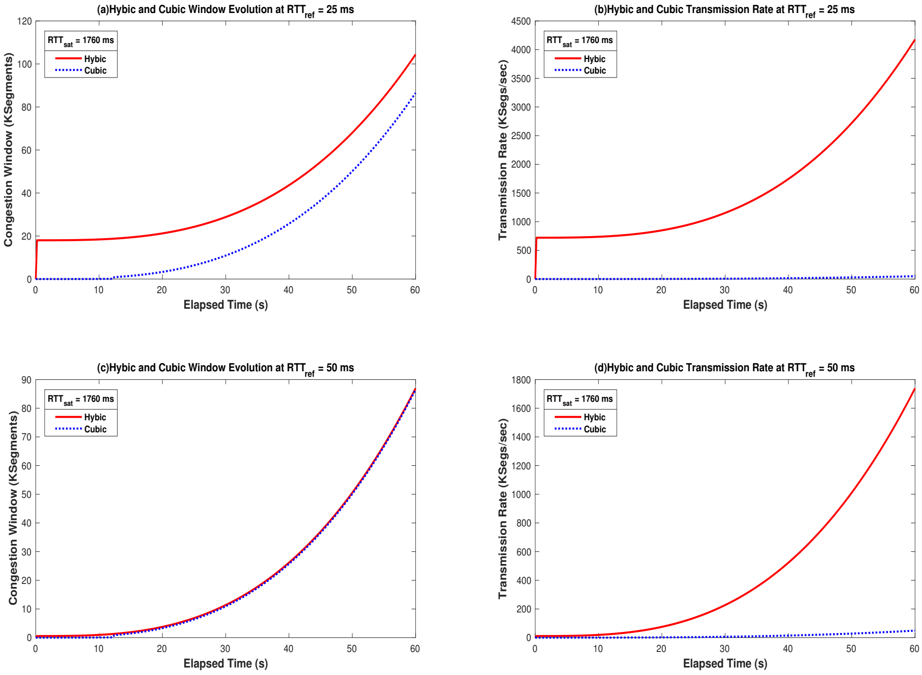


Figure 5.14: HYBIC and CUBIC $W(t)$ and $R(t)$,, at RTT_{ref} of 25 ms and 50 ms

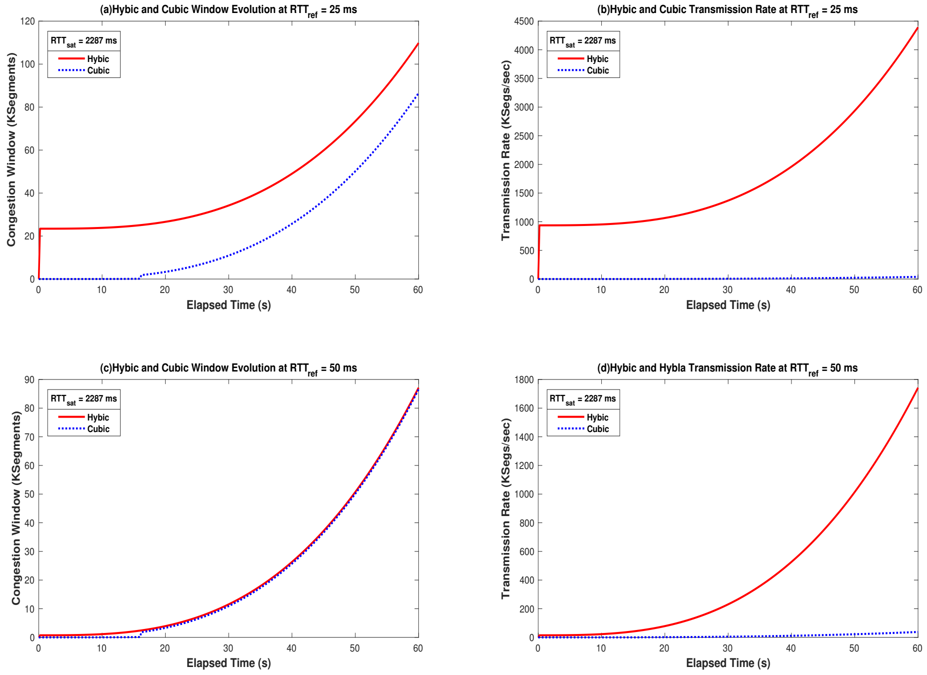


Figure 5.15: HYBIC and CUBIC $W(t)$ and $R(t)$, at RTT_{ref} of 25 ms and 50 ms

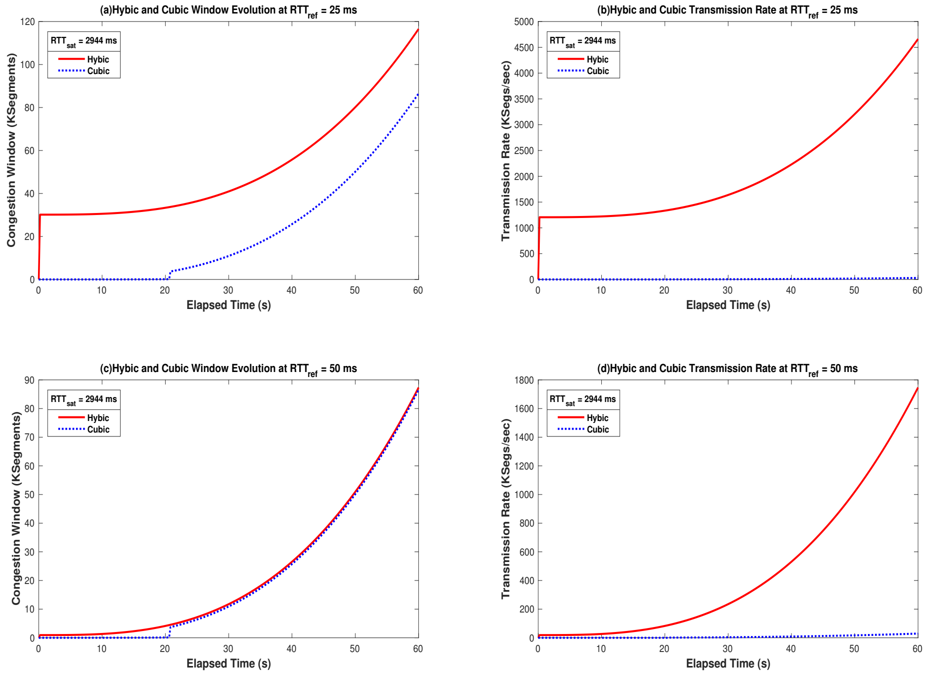


Figure 5.16: HYBIC and CUBIC $W(t)$ and $R(t)$,, at RTT_{ref} of 25 ms and 50 ms

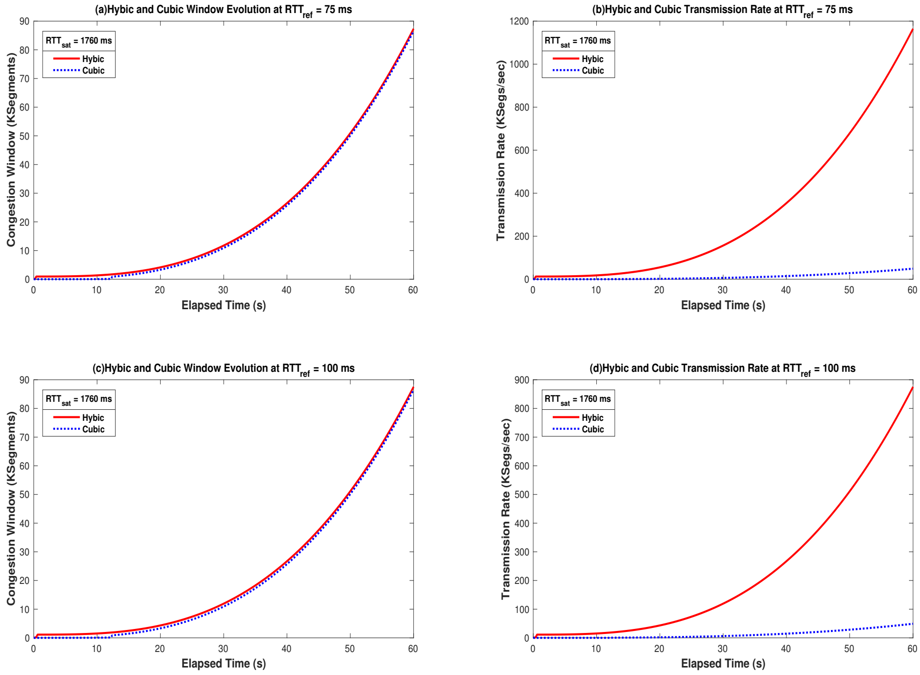


Figure 5.17: HYBIC and CUBIC $W(t)$ and $R(t)$, at RTT_{ref} of 75 ms and 100 ms

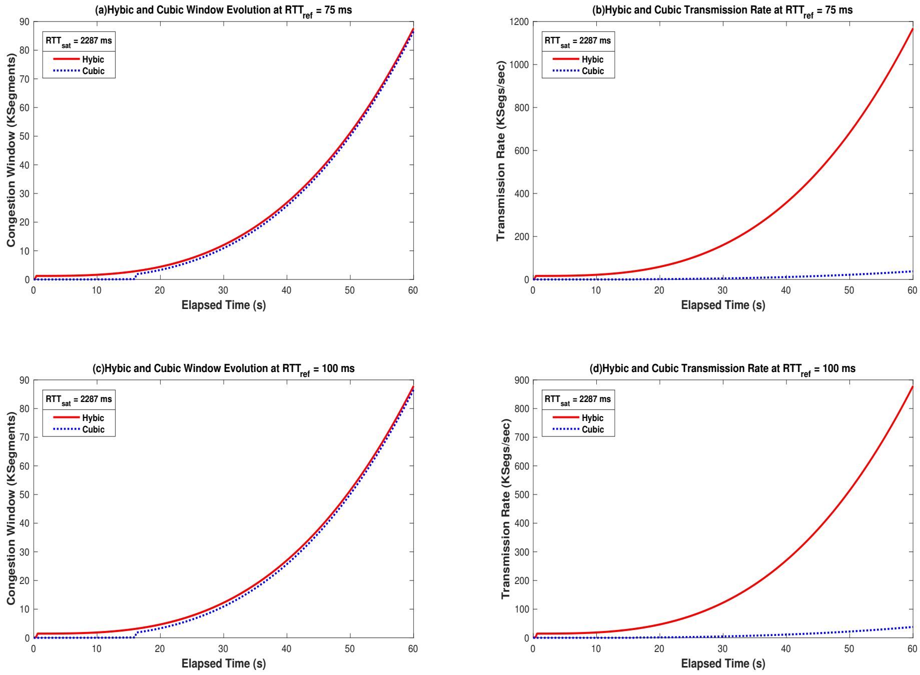


Figure 5.18: HYBIC and CUBIC $W(t)$ and $R(t)$,, at RTT_{ref} of 75 ms and 100 ms

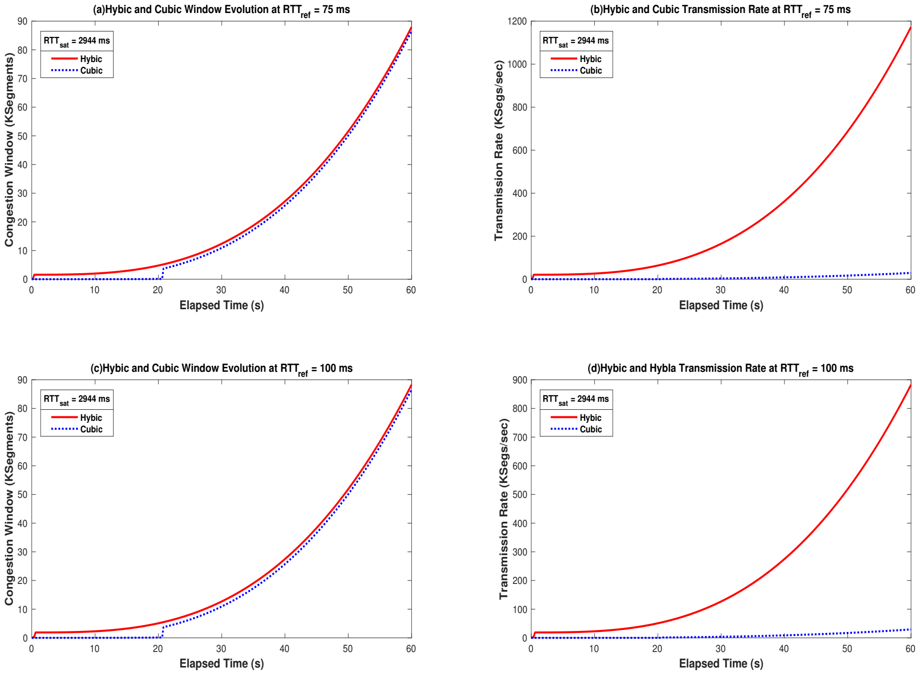


Figure 5.19: HYBIC and CUBIC $W(t)$ and $R(t)$, at RTT_{ref} of 75 ms and 100 ms

Table 5.6 summarises the *cwnd* evolution, $W^c(t)$ performance for all RTT_{sat} values considered and RTT_{ref} of 25 ms and 100 ms, while Table 5.7 gives the performance summary of the instantaneous transmission rate, $R^c(t)$ under the same parameter values. These summaries showed that CUBIC is independent of the RTT_{ref} changes in window growth and transmission rates algorithms under an error free or ideal channel conditions.

Table 5.6: CUBIC Window Evolution at RTT_{ref} of 25 ms and 100 ms

$RTT_{sat}(ms)$	$W^c_{max}(kseg)$	$W^c_{min}(seg)$	$W^c_{avg}(kseg)$	$W^c_{max}(kseg)$	$W^c_{min}(seg)$	$W^c_{avg}(kseg)$
2944	87	1000	22	87	1000	22
1760	87	1000	22	87	1000	22
2287	87	1000	221	87	1000	22
Overall	87	1000	22	87	1000	22

Table 5.8 summarises the performance improvements from CUBIC to HYBIC using the difference that represents the increase in transmission rate, $R(t)$ by HYBIC. The transmission rate performance of TCP CUBIC was lower compared to HYBIC

Table 5.7: CUBIC Transmission Rate at RTT_{ref} of 25 ms and 100 ms

$RTT_{sat}(ms)$	$R^c_{max}(Mbps)$	$R^c_{min}(bps)$	$R^c_{avg}(Mbps)$	$R^c_{max}(Mbps)$	$R^c_{min}(bps)$	$R^c_{avg}(Mbps)$
2944	341	3935	84	341	3935	84
1760	570	6582	143	570	6582	143
2287	438	5065	110	438	5065	110
Overall	570	3935	112	570	3935	112

with overall R^c_{max} = 341 Mbps and R^c_{avg} = 84 Mbps for all RTT values compared to HYBIC overall R^m_{max} = 54 Gbps and R^m_{avg} = 21 Gbps at RTT_{ref} =25 ms. Moreover, the performance obtained at high RTT values with HYBIC were much better compared to the best performance obtained with CUBIC as shown in Table 5.8. Although, CUBIC is RTT independent and stable under ideal channel conditions, it was out-performed by the new *HYBIC* proposal in terms of achievable throughput that improves available capacity utilisation.

Table 5.8: CUBIC and HYBIC Overall $R(t)$ Comparison at RTT_{ref} of 25 ms and 100 ms

Algorithm	$R_{max}(Gbps)$	$R_{min}(Mbps)$	$R_{avg}(Gbps)$	$R_{max}(Gbps)$	$R_{min}(Mbps)$	$R_{avg}(Gbps)$
CUBIC	0.341	3.935	0.084	0.341	0.3935	0.084
HYBIC	54	33	21	10	2	3
Diff	53.659	32.996	20.916	9.659	1.996	2.916

5.3 HYBIC Implementation and Simulation

The proposed HYBIC algorithm was implemented using the C language and and tested as a new protocol module with other high BDP TCP schemes implementations in Linux such as CUBIC and HYBLA. This also enabled simulations at the transport layer level to be carried out using Network Simulator version 2.35 (NS 2.35) installed on Ubuntu (Linux) operating system. Network simulations are required or become necessary in the process of designing and developing new and modified communication protocols like TCP.

NS (or *ns*) is a discrete event object oriented simulator written in C++ with an

Object oriented Tool Command Language (OTcl) frontend interpreter as a command and configuration interface [21]. This is one of the most effective communications network research tools that provides vital support for protocol simulation over wired and wireless (Terrestrial and Satellite) networks and has become a popular simulator in the scientific research environment. NS is useful for protocol design, traffic studies, protocol comparisons, performance analysis and evaluation and also help researchers increase results confidence. Generally, simulations are required to test, analyse, evaluate and optimise communications protocol and algorithms before being accepted as standards and added in the communication systems. The directory structure of NS-2.35 used for the HYBIC protocol module is shown in Fig. 5.20. The source code (*hybic.c*) and header (hybic.h) files most follow this directory structure for successful addition of the HYBIC protocol module among other TCP modules in Linux.

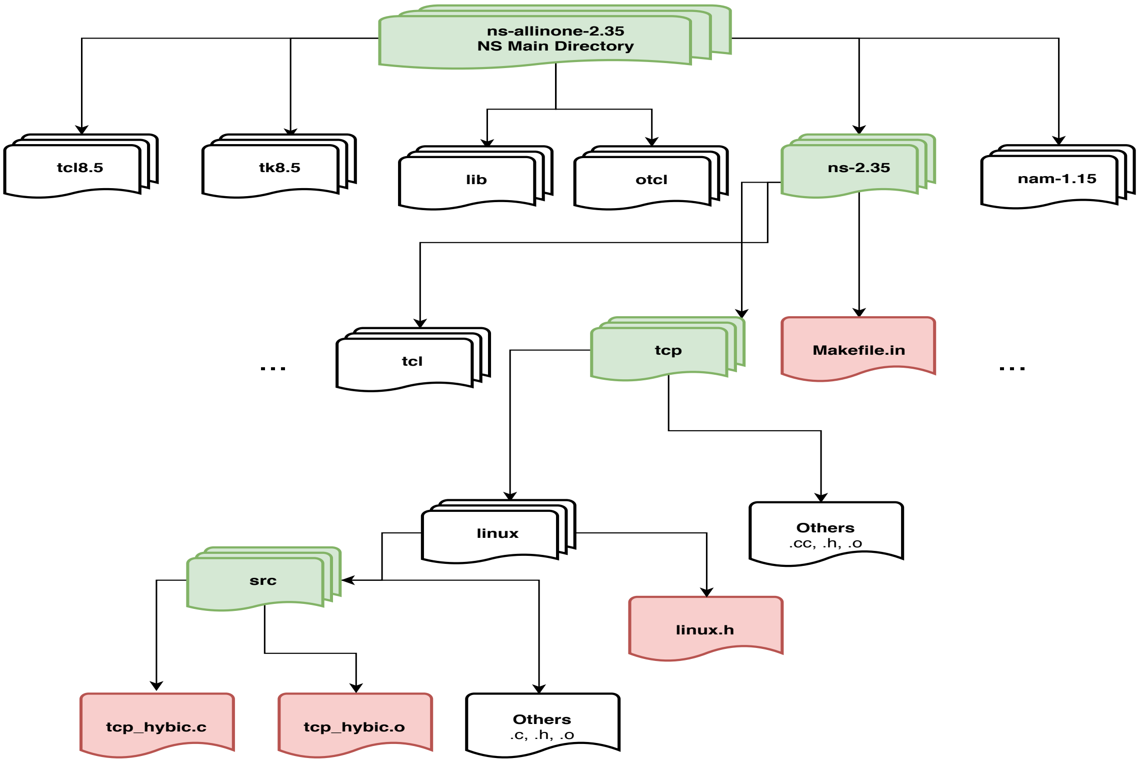


Figure 5.20: NS-2.35 and HYBIC Directory Structure

After the addition of all relevant files in the appropriate directory shown in Fig. 5.20, the simulation, testing and data (pre/post) processing are carried out in the hierarchy shown in Fig. 5.21. The simulation returns errors if the appropriate code and header files are not found in the correct directory.

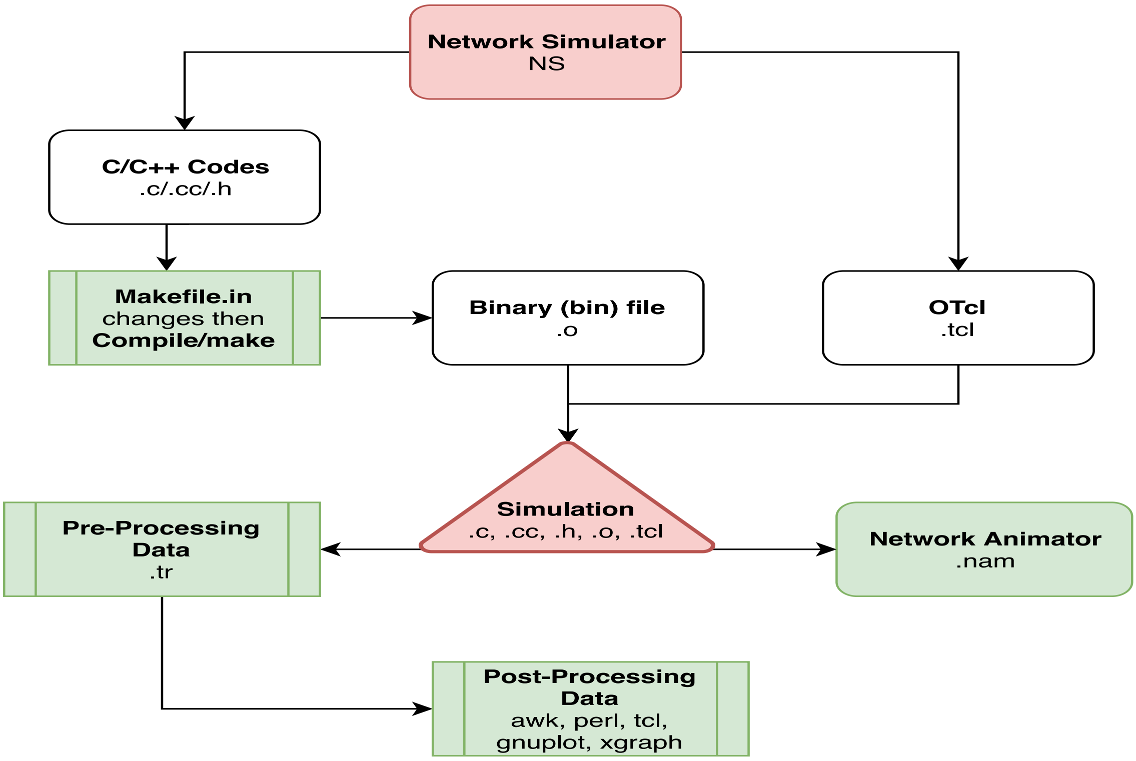


Figure 5.21: Simulation and Data Processing Hierarchy Flowchart

A number of simulation runs were carried out in order to analyse and evaluate the performance of HYBIC as a single TCP flow over a heterogeneous network (ISTN) environment and as a competing flow among other TCP flows such as NewReno, HYBLA, and CUBIC. These results were compared, and analysed in the subsequent sections of this chapter for evaluating the performance of HYBIC against the two high BDP schemes, CUBIC and HYBLA, already implemented as modules in NS with CUBIC being the default TCP in Linux. The ISTN topology used to test and simulate the new HYBIC proposal against other TCP schemes was designed using Tool Command Language *(TCL)*, which was then linked to the C language implementation by the NS. The Network Animator (NAM or nam) integrated in NS was used for topology and traffic flow visualisation as discuss and shown in the following simulation and testing section.

5.3.1 HYBIC Protocol Implementation

Before the simulations of the TCP HYBIC proposal, a typical software design and development cycle was carried out. These steps include the HYBIC *1) Software Coding (programming) using C language 2) Debugging errors using Gnu Debugger (GDB) in Ubuntu Linux 3) Implementation and addition as TCP Linux protocol module in NS-2.35 4) Testing using the designed and TCL coded network topology.* These processes shown in Fig. 5.22 are part of the well known Software Development Life Cycle (SDLC) with some of the phases already carried out before the numerical implementation and analysis.

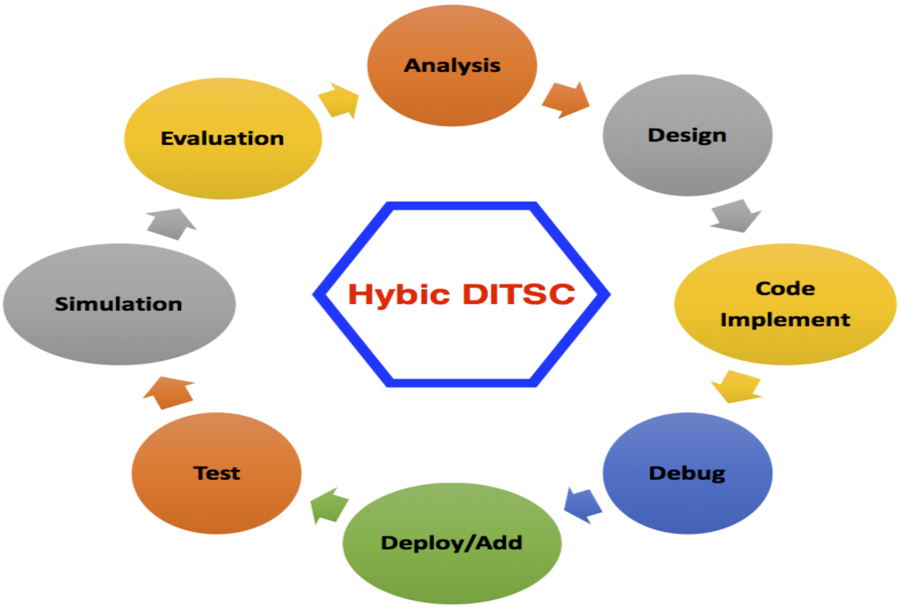


Figure 5.22: HYBIC Development, Implementation, Test and Simulation Cycle

The step by step execution of HYBIC algorithm is shown by the flowchart in Fig. 5.23. These processes execute multiple functions written and compiled in C Language, which formed part of the NS kernel to implement the kernel architecture of the HYBIC protocol. The simulator (NS-2.35) supports a *class hierarchy in C/C++* called the *compiled hierarchy* and similar class hierarchy within the *OTcl interpreter* called the *interpreted hierarchy.* These two hierarchies are closely related to each other with a one-to-one correspondence between a class in the interpreted hierarchy and one in the compiled hierarchy from the users perspective [21]. The complete C codes are given in the Appendix A.3.

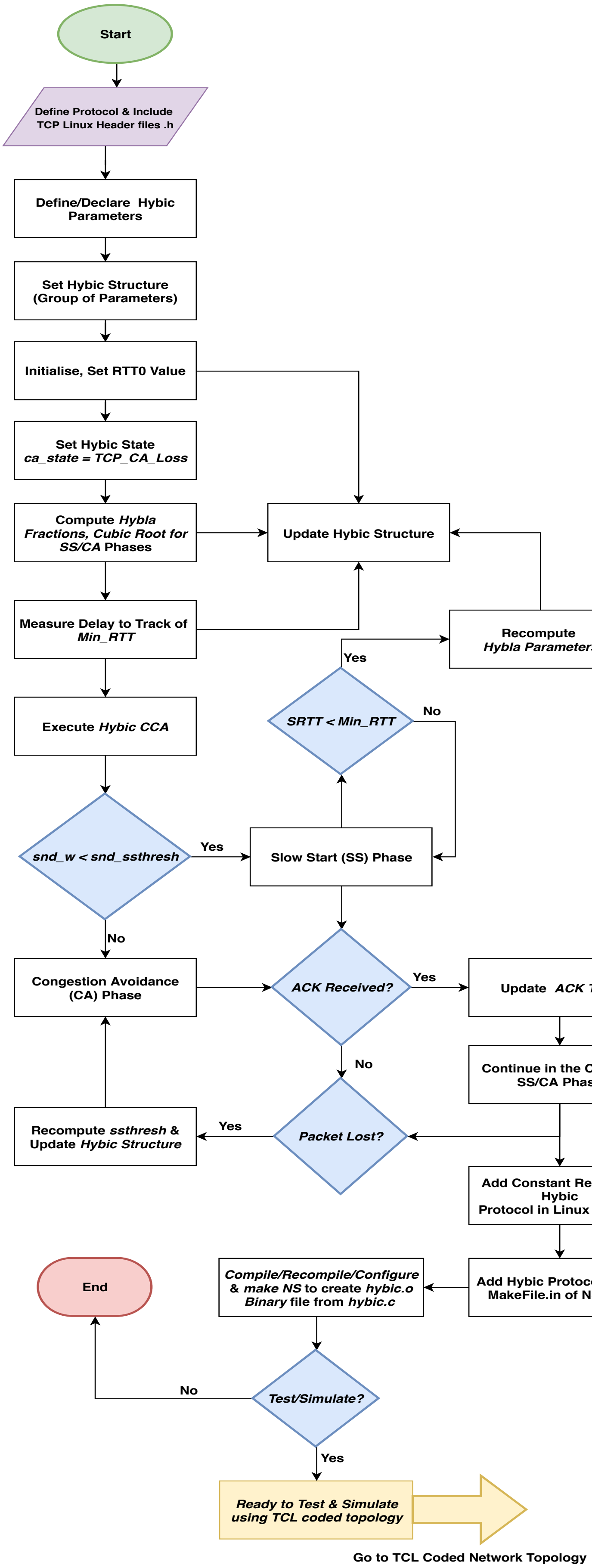

Start
Define Protocol & Include TCP Linux Header files .h
Define/Declare Hybic Parameters
Set Hybic Structure (Group of Parameters)
Initialise, Set RTT0 Value
Set Hybic State *ca_state = TCP_CA_Loss*
Compute *Hybla Fractions, Cubic Root for SS/CA* Phases
Update Hybic Structure
Measure Delay to Track of *Min_RTT*
Recompute *Hybla Parameters*
Yes
SRTT < Min_RTT
No
Execute *Hybic CCA*
snd_w < snd_ssthresh
Yes
Slow Start (SS) Phase
No
Congestion Avoidance (CA) Phase
ACK Received?
Yes
Update *ACK Time*
No
Continue in the Current SS/CA Phase
Recompute *ssthresh* & Update *Hybic Structure*
Yes
Packet Lost?
Add Constant Record of Hybic Protocol in Linux Module
End
Compile/Recompile/Configure & make NS to create *hybic.o Binary* file from *hybic.c*
Add Hybic Protocol in the MakeFile.in of NS-2.35
No
Test/Simulate?
Yes
Ready to Test & Simulate using TCL coded topology
Go to TCL Coded Network Topology

Figure 5.23: HYBIC Algorithm Execution Flowchart

Finally, implementation of the key Congestion Control Algorithm (CCA) functions defined in *struct tcp_congestion_ops* added a constant record for HYBIC CCA as static record of *struct tcp_congestion_ops tcp_HYBIC* that stored the key function calls and the new algorithm's name as shown and explained below.

```
static struct tcp_congestion_ops tcp_HYBIC = {
.init = HYBIC_init,
.ssthresh = bictcp_recalc_ssthresh,
.cong_avoid = HYBIC_cong_avoid,
.set_state = HYBIC_state,
.pkts_acked = bictcp_acked,
.owner = THIS_MODULE,
.name = "HYBIC"
};
```

1. ***HYBIC_init()***: Function called after the first ACK received and before the CCA is called for the first time. The private data (parameters and variables initialisation) of HYBIC CCA is also initialised by this function. Function implementation for CCA interface ***void (*init)(struct tcp_sock *sk);***.

2. ***bictcp_recalc_ssthresh()***: Function that returns the slow-start threshold (ssthresh or γ) after a loss event. implementation for CCA interface ***u32 (*ssthresh)(struct sock *sk);***.

3. ***HYBIC_cong_avoid***: This function executes the main routine of the TCP HYBIC CCA, which increases the congestion window (*cwnd*) for each ACK received, performs new *cwnd* calculation and selects whether to be in SS or CA phase based on the comparison of current *cwnd* and *ssthresh* value. Function implementation for CCA interface ***void (*cong_avoid)(struct sock *sk, u32 ack, u32 rtt, u32 in_flight, int good_ack);***.

4. ***HYBIC_state()***: This function is called before or when the congestion state (ca_state) of the HYBIC is changing/changed using *new_state or ca_state* variable, used as the state code to notify the HYBIC CCA the state it is going

to be in. This is also used by some algorithms to turn off special control during loss recovery. The possible states are *TCP_CA_Open* for *normal state* or *TCP_CA_Loss* for *Loss Recovery after a Timeout* other states can be found in the data structure interface of TCP. Function implementation for CCA interface ***void (*set_state)(struct sock *sk, u8 new_state);***.

5. ***bictcp_acked***: This function is called when there is an ACK that acknowledges some new packets using *num_acked*; as the number of packets that are ACKed by this acknowledgments. This serves as a hook for packet ACK accounting and for CCA interface ***void (*pkts_acked)(struct sock *sk, u32 num_acked, ktime_t last);***.

6. ***.name = "HYBIC"*** : Assigned name of the TCP CCA, *HYBIC* in this case. This will be the name for *"select_ca"* command in the tcl script for selecting the protocol to simulate. implementation for CCA interface ***char name[16];***.

The above implemented functions identified HYBIC as CCA in Linux and interfaced according to the Linux TCP *struct tcp_congestion_ops.* The HYBIC algorithm execution flowchart is shown in Fig. 5.23 and the reference structure of function call pointers in Linux TCP are defined below:

```
struct tcp_congestion_ops {
void (*init)(struct sock *sk);
u32 (*ssthresh)(struct sock *sk);
void (*cong_avoid)(struct tcp_sock *sk,u32 ack,u32 rtt,u32 in_flght,int gd_ack)
u32 (*ssthresh)(struct tcp_sock *sk);
u32 (*undo_cwnd)(struct tcp_sock *sk);
void (*set_state)(struct tcp_sock *sk, unsigned int newstate);
void (*pkts_acked)(struct tcp_sock *sk, unsigned int num_acked, ktime_t last);
char  name[TCP_CA_NAME_MAX];
struct module  *owner;
    };
```

5.3.2 HYBIC Simulation and Testing

Simulation experiments and testing were conducted using the designed and OTCL coded ISTN topology and visualised using NAM as shown in Fig. 5.24. Object TCL (OTcl) is an extension to TCL for object oriented programming used for building network structure and topology, which serves as the surface of network simulations. Unlike C/C++, OTcl is slow to run/execute, but easy to code/change that is why it is not used for protocol architecture and implementation. As mentioned earlier, class hierarchy within the OTcl interpreter is the interpreted hierarchy by which new simulator objects are created through the interpreter. These objects are instantiated within the interpreter, and are closely mirrored by a corresponding object in the compiled hierarchy [21].

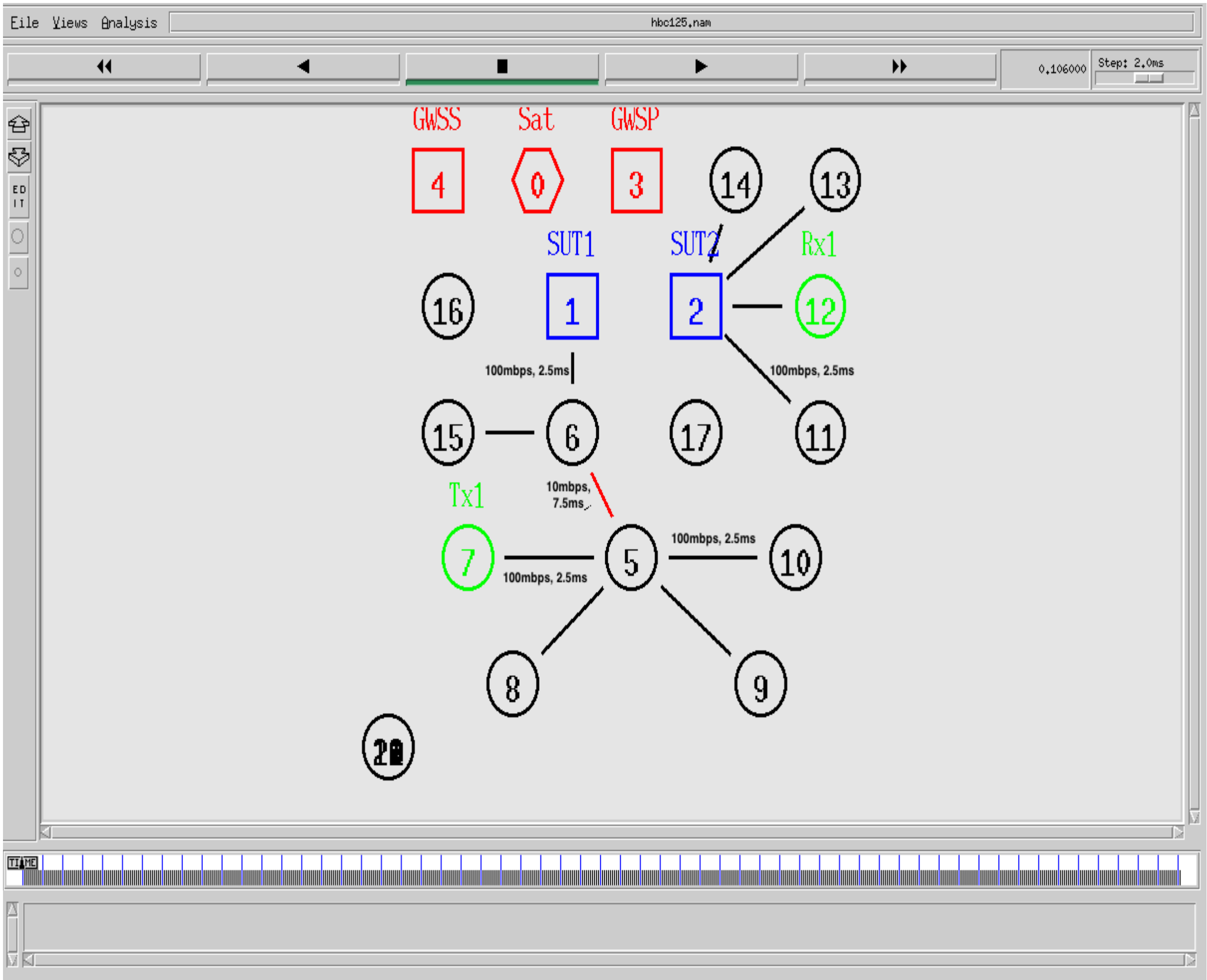


Figure 5.24: Simulator Network Animator ISTN Topology

Moreover, the two hierarchies using two languages (C/C++ and OTcl) are required because the simulator has two goals needed to be achieved. Firstly, is the detailed simulations of protocols that requires a systems level programming language which can efficiently manipulate bytes, packet headers, and implement algorithms that run over large data sets. For the algorithm implementation tasks, run-time speed is important and turn-around time (running simulation, finding/fixing bugs, recompile, re-run) is less important.

Second, a large part of network research involves varying parameters or configurations, or fast exploring a number of scenarios. In these cases, iteration time (i.e. reconfigure the model and re-run) is more important. Since configuration runs once (at the beginning of the simulation), run-time of the task is less important. The simulator (*NS*) satisfied and achieved these goals with the two languages, C/C++ and OTcl. C/C++ is fast to run but slower to change, making it suitable for detailed protocol implementation while OTcl runs much slower but can be changed very quickly (and interactively), making it ideal for simulation configuration. The simulator via *tclcl* provides linkage to make objects and variables appear on both languages [21].

The network topology consists of wired/wireless terrestrial and satellite nodes and links with two routers creating the bottleneck link of capacity, C_{BNL} as shown by Fig. 5.24. Wired nodes, serving as transmit (Tx) and receive (Rx) nodes have the same capacity C. TCP traffic was generated using a File Transfer Protocol (FTP) agent/applications and UDP traffic was generated by a Constant Bit Rate (CBR) application. These FTP and CBR applications generate packets of certain sizes by the application layer of the sender throughout the simulation period.

Moreover, the topology consisted of five satellite nodes (two ground SUT for Tx/Rx, one bent-pipe GEO SSS and two GWS for access to NOC/NCC), eight wired nodes (four each on the Tx and Rx ends) and three wireless nodes (for WiFi Tx/Rx) the link parameters are shown in Fig. 5.24. Table 5.9 provides the simulation parameters set in the oTcl code for the simulation and testing the HYBIC and other TCP schemes such as CUBIC and HYBLA using the same topology (see Fig. 5.24) that represent heterogeneous ISTN environment.

Table 5.9: Simulation Parameters

Parameter	*Satellite*	*Terrestrial*	*Bottleneck*
OWD (ms)	120	2.5	7.5
Queue	*DropTail*	*DropTail*	*DropTail*
Bandwidth (Mbps)	10	100	10
BDP (pkt)	289	60	18
MSS/Pkt_Sz (bytes)	1040/1000	1040/1000	1040/1000
PTR (pps)	1201/1250	12019/12500	1201/1250
Traffic/Application	*ftp/cbr*	*ftp/cbr*	*ftp/cbr*

The Bandwidth Delay Product (BDP in *packets*) and Packet Transmission Rate (PTR in *packet per seconds*) were computed using Eqn. (5.3.1) and (5.3.2) respectively.

$$BDP = \frac{BW RTT}{8PSZ} \tag{5.3.1}$$

where; BW is the bandwidth (capacity) of the link (in *bps*), RTT is the link round-trip-time (in *sec*) and PSZ is the packet size (in *bytes*)

$$PTR = \frac{BW}{8PSZ} \tag{5.3.2}$$

BDP is the maximum amount of data packets (in bytes or bits) that can be on the network circuit (pipe) at any given time before receiving an ACK, as mentioned in the previous chapter, BDP is a rule of thumb for sizing router buffers in conjunction with congestion avoidance algorithm, noticed that the satellite leg has higher value of BDP making it an *LFN*. The PTR on the other hand is the packet that can be sent over the link in each second based on the link capacity. BDP and PTR can be calculated in bytes by multiplying each by an MSS or set PSZ in the OTcl codes. The complete OTcl codes is given in the Appendix A.2.

The simulation and testing were carried out by enabling HYBIC, CUBIC, and HYBLA individually as single TCP connections with FTP traffic generated and sent over by the TCP sender while the CBR traffic was sent over UDP. Simulations were also run by enabling multiple TCP schemes such as HYBIC/HYBLA and

HYBIC/CUBIC with different sources sending FTP traffic over these TCP connections. The simulations were run multiple times (at least three times) with duration of 349 s simulation, which required about 2 hrs of computing time on a MacBook Pro Computer with Intel Core i5 (2.3 GHz) processor, 4GB RAM, and running Ubuntu 14.04.5 (64-bit) on Oracle VM VirtualBox. The simulation data is recorded in a trace file that was processed after the simulation. Results were extracted from the data, usually using *AWK, PERL* and *OTcl* languagesfor analysis and evaluation.

5.4 Results and Performance Evaluation

This section presents and discusses the results of different simulation scenarios from which the performance of HYBIC is being analysed and evaluated. The performance comparison with HYBLA and CUBIC is also presented, CUBIC is considered as the standard for high BDP (LFN) and the default CCA implemented in Linux. The key parameters of interest for the analysis and evaluation of the CCA are the *cwnd* (W), transmission rate (R), packet delivery ratio (PDR), E2E latency (OWD/RTT), and Internet packet delay variation ($IPDV$) also called *jitter*.

5.4.1 HYBIC Analysis and Evaluation

The TCP HYBIC algorithm implementation was tested by simulation using the parameters in Table 5.9 and the simulation run for 349 s. The simulation scenario was created as single E2E TCP HYBIC connection was established between a single sender (Tx, labelled node 7) connected on the Terrestrial part of the topology (see Fig. 5.24) sending through a router (node 5) that create a bottleneck Link (BNL) with another router (node 6). The single TCP HYBIC receiver (Rx, labelled node 11) is connected to the satellite via SUT (node 2), which creates a heterogeneous ISTN topology with one GEO satellite leg on the receiver site. This scenario and topology allowed us to analyse and evaluate the performance of HYBIC algorithm as a single TCP flow over a heterogeneous network environment with wired/wireless and long/short RTT. The simulations were run at least three times for each reference RTT_0 and the results in the trace file were inspected to make sure they were consistent when using the same simulation setup and scenario.

The *cwnd* (in *packets*) evolution for different RTT_0 shown in Fig. 5.25 were measured by tracing the traffic for each RTT_0 value for the same duration of the simulation given as *Elapsed Time in seconds (s)*. The traced packet size, PSZ or MSS was 1040 bytes which can be multiplied with *cwnd (pkts)* to get its value in bytes units. From the inspection of Fig. 5.25, it was observed that the *cwnd* growth is steady, stable and follow the mathematical model of the algorithm proposed, which exhibit an exponential SS phase and CUBIC function CA phase as shown by Fig. 5.25. For all the values of RTT_0 the SS phase of the HYBIC algorithm did lasted for less than 5 s of the simulation time and then entered the CA phase with CUBIC increase until packet loss was detected at about 100 s of the transmission.

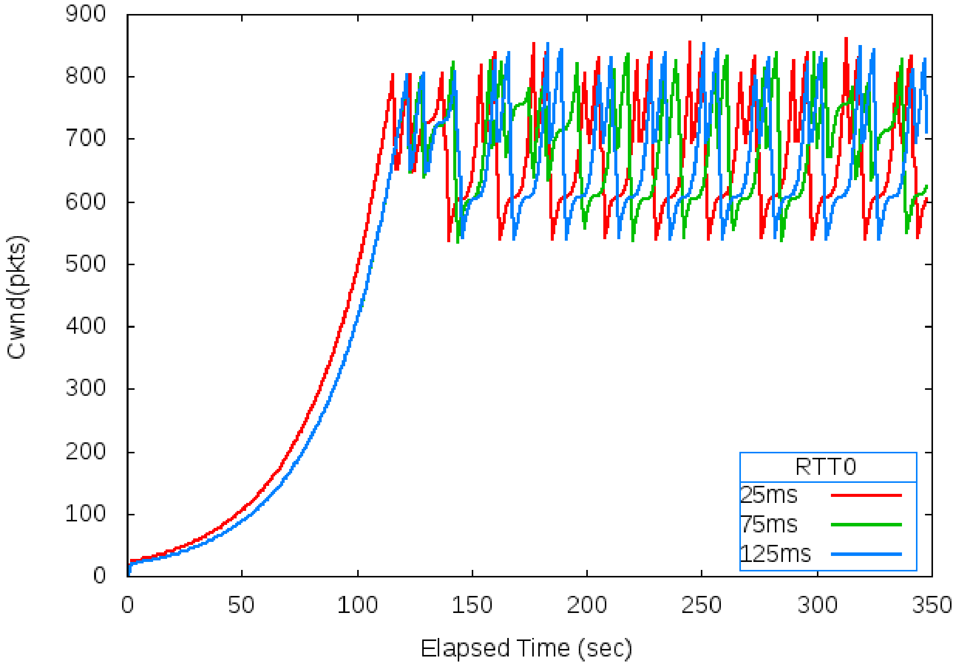


Figure 5.25: HYBIC Window Growth Rate, $W(t)$ at Varied RTT and Reference RTT_0

After the packet loss detection, HYBIC employs the CUBIC function by executing the TCP friendly, concave and convex regions instead of halving the *cwnd* and returning to the SS phase, this continued until the end of the transmission as shown by Fig. 5.25. This stable *cwnd* growth rate of HYBIC allowed it to grow to about 900 pkts (900 kbytes) after 100 s of data transmission, which then led to throughput performance improvement as shown in Fig. 5.26.

The Instantaneous Transmission Rate (ITR), or the throughput performance within the duration of the simulation is shown by Fig. 5.26. The maximum throughput achieved by HYBIC as a single flow, is about the capacity of the BNL (10 Mbps) as shown in Fig. 5.26. This indicates efficient capacity utilisation of the HYBIC implementation. The ITR performance of HYBIC was also consistent with the *cwnd* evolution that showed how the rates vary with respect to the change in RTT_0 from 25 ms to 125 ms. Although, the performance was not influenced so much with the different RTT_0, especially after the SS phase at about 100 s, it was observed that increasing RTT_0 led to a small decrease in both *cwnd* and the achievable throughput, R as shown in Fig. 5.25 and 5.26.

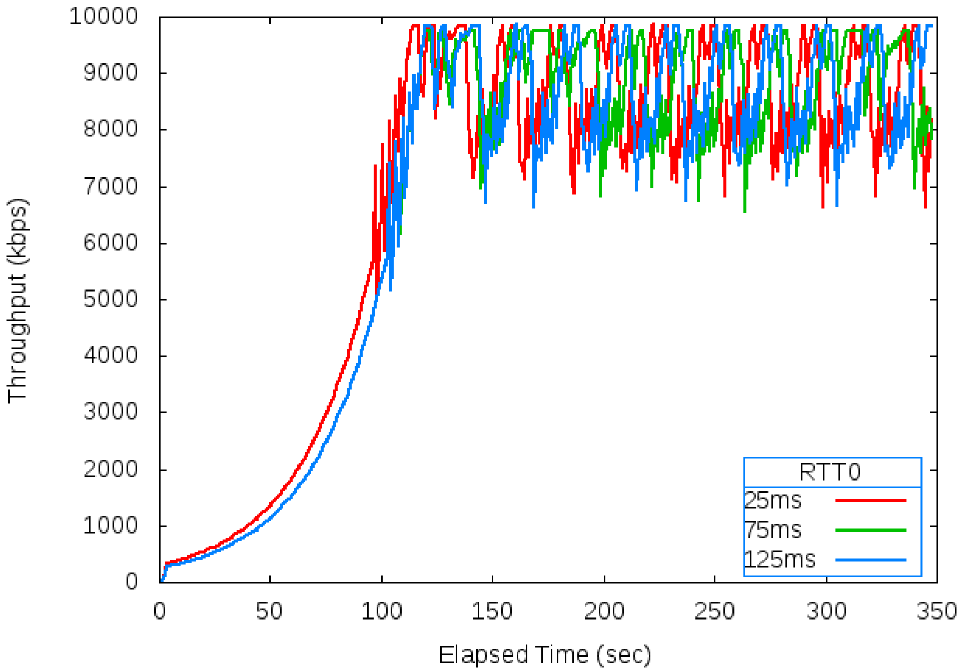


Figure 5.26: HYBIC Transmission Rate, $R(t)$ at Varied RTT and Reference RTT_0

Other parameters measured in real time during the single flow scenario of TCP HYBIC simulations are E2E OWD/RTT and jitter as shown in Fig. 5.27 and 5.28. Two different network topology configurations were considered on the Satellite leg of the ISTN; 1) Point-to-Point (P2P) topology configuration between the two SUT's and 2) Star topology between the SUT's and the GWS. The second topology configuration considered is more realistic and practical, because most of the satellites

(SS) are signal reflectors (bent-pipe) without routing and other data processing functions such as signalling, billing and user location functions. Therefore, such functions are expected to be carried out at the ground segment of the satellite network, usually the Network Operation/Control Centre (NOC/NCC). This Star topology configuration considered the additional latency (OWD/RTT) component contributions to and from the GWS, which was not considered by the P2P topology.

Fig. 5.27 shows the latency results of P2P OWD/RTT (in red/green) and Star OWDGW/RTTGW (in blue/purple) topologies. The realistic star topology results were higher than the P2P configuration. This was expected since latency to/from the GWS was added to the E2E OWDGW (blue) and RTTGW (purple) as shown in 5.28.

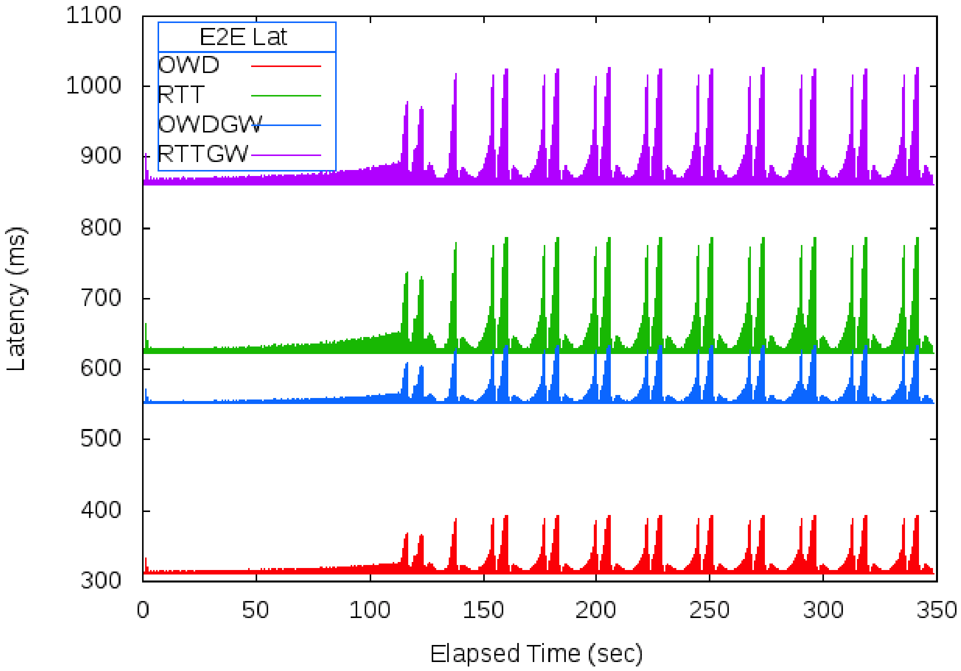


Figure 5.27: Measured E2E Latency During HYBIC Traffic Flows

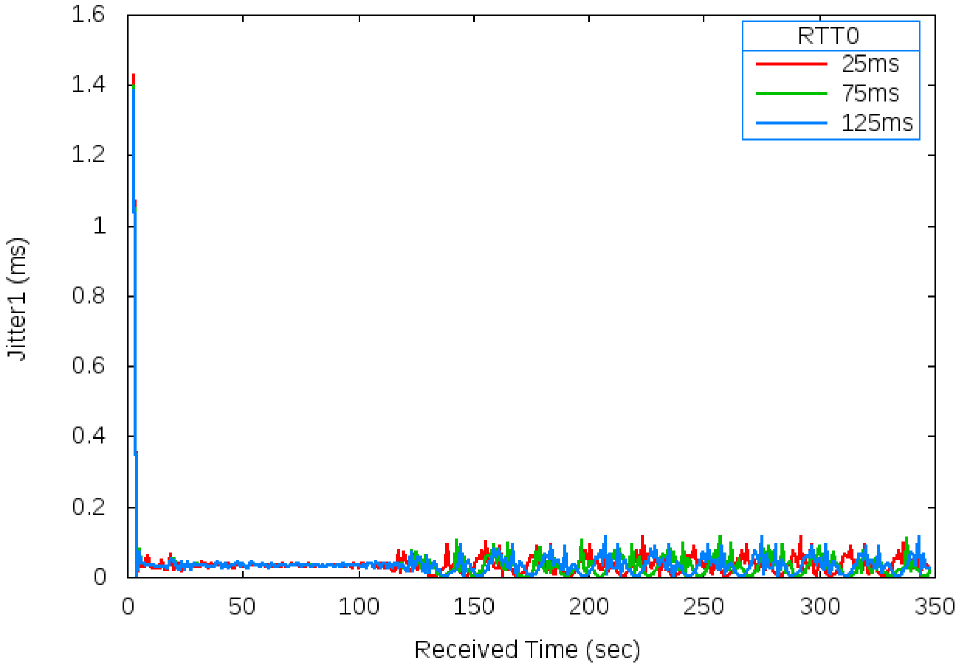


Figure 5.28: Measured jitter During HYBIC Traffic Flows at Varied Reference RTT_0

The key observation was that, most E2E latencies for a GEO satellite network being reported [4, 9, 10] do not include these additional components due to GWS. Moreover, the packet delay variation (jitter) was also measured in real time during the simulation of HYBIC as a single flow scenario. The results shown in Fig. 5.28 indicate a very low jitter achieved by the HYBIC algorithm compared to the HYBLA and standard TCP algorithms, which is good for jitter sensitive applications such as real-time applications. The highest jitter in HYBIC was less than 1.5 ms (see Fig. 5.28) and within less than 10 s of transmission time for all RTT_0 values considered. Even at this highest jitter value, the real-time applications over TCP HYBIC would be stable and provide meaningful data communications without congestion or breakdown.

The summary of the overall results obtained with a single TCP HYBIC connection at variable RTT_0 given in Table 5.10. The summary indicates an average performance from each parameter measured during the simulation against different reference RTT_0. HYBIC was observed to have an average capacity utilisation of 68% of the 10 Mbps capacity of the BNL for all the implemented values of RTT_0 considering the average throughput (R_avg) values achieved, this shows little or no influence of RTT on the performance of HYBIC. Another key performance indicator of HYBIC is the low average jitter of 0.035 ms (35 μs) and high average *cwnd* *(W_avg)* of at least 513 kbytes in all the RTT_0 implementations.

Table 5.10: HYBIC Overall Simulated Performance at Varied Reference RTT_0

$RTT_0(ms)$	$W_{avg}(pkts)$	$R_{avg}(kbps)$	$OWD_{avg}(ms)$	$RTT_{avg}(ms)$	$Jitt_{avg}(ms)$
25	522	6800	328	656	0.035
75	520	6776	329	657	0.033
125	513	6663	328	657	0.035
Overall Avg	518	6746	328	657	0.034

The performance of HYBIC was also analysed and evaluated based on packet delivery, which includes packets transmitted (Tx), Received (Rx), Dropped (Dropd) and packet delivery ratio (PDR = Rx/Tx). Table 5.11 summarised the packet delivery performance statistics in terms of packets sent (Tx), received (Rx) and dropped. These indicated an impressive performance of HYBIC with at least 99% of the packets sent were received (DR) over the network using HYBIC flow and $\leq$ 1% of the packets dropped.

Table 5.11: HYBIC Simulated Packet Delivery Performance Summary

$RTT_0(ms)$	$TX(pkts)$	$RX(pkts)$	$DP(pkts)$	$DR(\%)$	$\eta(\%)$
25	1420789	1420243	546	99.96	68
75	1415602	1415141	461	99.97	67.76
125	1394000	1393456	544	99.96	66.63
Overall Avg	1410130	1409613	517	99.96	67.46

Moreover, the performance of HYBIC in terms of capacity utilisation, $\eta(\%) = R_{avg}/C_{BNL}$ was derived from average throughput (R_{avg}) achieved by each RTT_0 implementation in Table 5.10 and the capacity of the BNL (C_{BNL}). This showed that HYBIC was able to achieve an efficient capacity utilisation of 68%, as given in Table 5.11.

Considering the heterogeneous and LFN nature of the topology used for the simulation and testing of the new TCP HYBIC proposal, as described in the previous sections, the bandwidth (capacity) of the network path was set to 10 Mbps (BNL), the measured average RTT = 657 ms and packet size of 1040 bytes for the HYBIC flow network path (see Table 5.9 and 5.10), which gave the network path BDP of about 790 pkts computed using Eq. (5.3.1). HYBIC grows its *cwnd* to this BDP value at about 100 s (see Fig. 5.25), this would have taken the standard TCP about 519 s (790 RTTs = $790{*}657x10^{-3}$ s) to grow its *cwnd* to the same value by which the flow has finished (simulation time was 350 s), severely under-utilising the capacity of the path. The fast *cwnd* growth of HYBIC to reach the network path BDP (the amount of packets needed in flight while keeping the bandwidth fully utilised) makes it a faster data transport protocol with efficiently high network utilisation as shown in Table 5.11.

Therefore, the new proposal of TCP HYBIC achieved an improved performance in terms of fast stable *cwnd* growth, packet losses (PDR), capacity utilisation (η), and stability in the execution of both SS and CA phases. In the next sections we will see how the HYBIC performance was compared to the other TCP schemes implemented for high BDP (LFN) such as HYBLA and CUBIC in terms of fairness, friendliness and packet losses.

5.4.2 HYBIC and HYBLA

Performance of TCP HYBIC was analysed and evaluated when subjected to being part of the three flows from three sources (Tx) using different schemes of data transport protocols. In this section, a scenario of a HYBIC connection sharing the BNL with HYBLA, and CBR traffic of 10 kbps over UDP connection of PSZ=1000 byte was setup and simulated. These multiple connections of high BDP protocols (HYBIC and HYBLA) sharing a 10 Mbps BNL allowed investigating, analysis and evaluation of how fair and friendly TCP HYBIC would be in terms of bandwidth sharing and congestion avoidance due to packet losses and link congestions in the presence of multiple flows/connections over single BNL.

Figure 5.29 showed the *cwnd* growth rate, $W(t)$ for HYBIC and HYBLA over the time of packet data transmission. The same simulation parameters (see Table 5.9) at RTT_0 = 25 ms and 125 ms as described in the previous simulation section were used for this simulation.

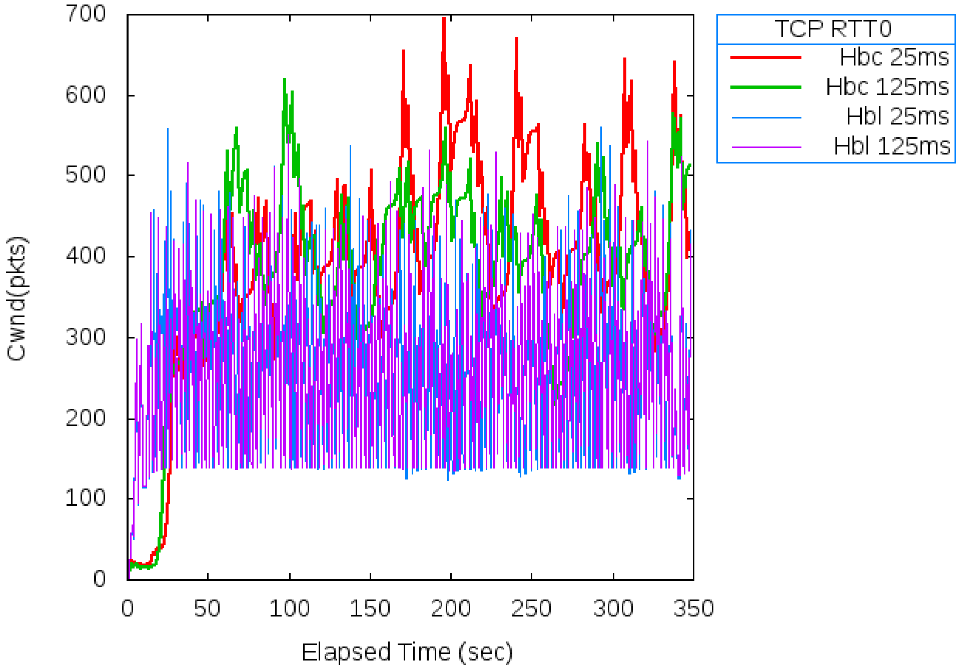


Figure 5.29: HYBIC Vs HYBLA Window Growth Rate, $W(t)$ at Varied $RTTs$

Various RTT_0 implementations were simulated with this scenario and it was observed that, the rate at which *cwnd* of HYBIC (*Hbc*) and HYBLA (*Hbl*) grow vary according to the RTT_0 used for the implementation. These variations were more noticeable in HYBIC as HYBLA maintained almost the same growth in all cases. HYBIC *cwnd* was observed to have similar steady and stable growth for all RTT_0, but grows faster when RTT_0 =25 ms (in red colour) after the CA phase. However, HYBIC *cwnd* growth increases with the elapsed time unlike the HYBLA which tends to decrease with the elapsed time and did not reach the BDP value before the flow (transmission) finishes. Severe under-utilisation may arise when the flow finishes before the time it takes to grow the *cwnd* to al least midpoint of the path BDP, which is about 395 pkts in this case.

Although HYBLA (blue and purple colour) exited the SS phase faster than HYBIC (red and green colour), it was observed that HYBLA experienced more frequent losses and reductions of *cwnd* to a specific *ssthresh* value below 150 pkts whenever loss events occurred (see Fig. 5.29) unlike HYBIC which has less packet losses and handled the event without too much reduction of its current *cwnd*, as shown in Fig.

5.29. This consistent growth and stability contributed to more achievable throughput (see Fig. 5.30) and efficient capacity utilisation, even when it competes for bandwidth with other traffic flows such as HYBLA and UDP as shown in Fig. 5.30.

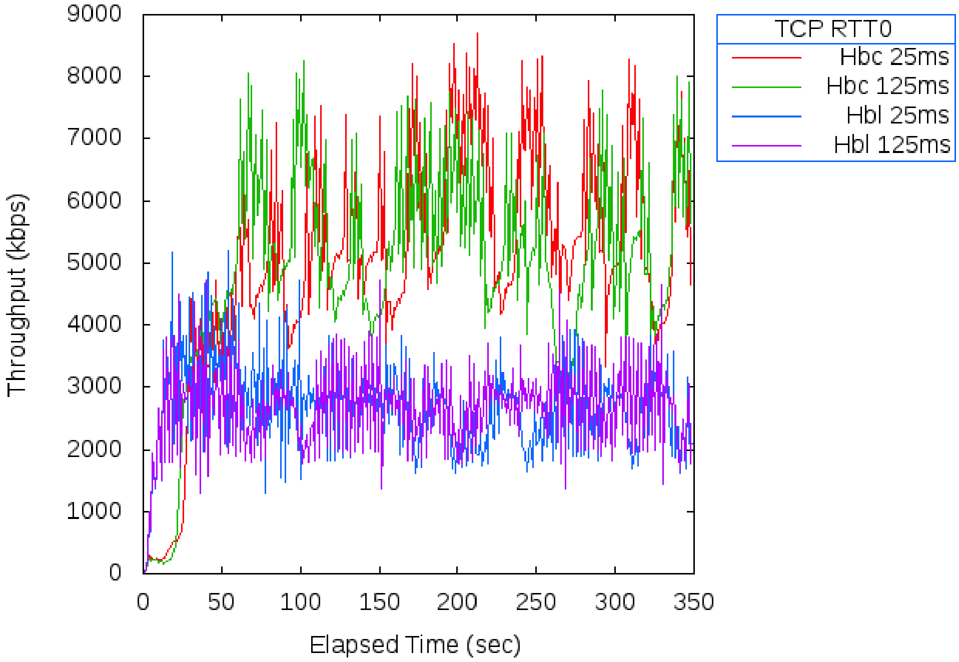


Figure 5.30: HYBIC Vs HYBLA Transmission Rate, $R(t)$ at Varied $RTTs$

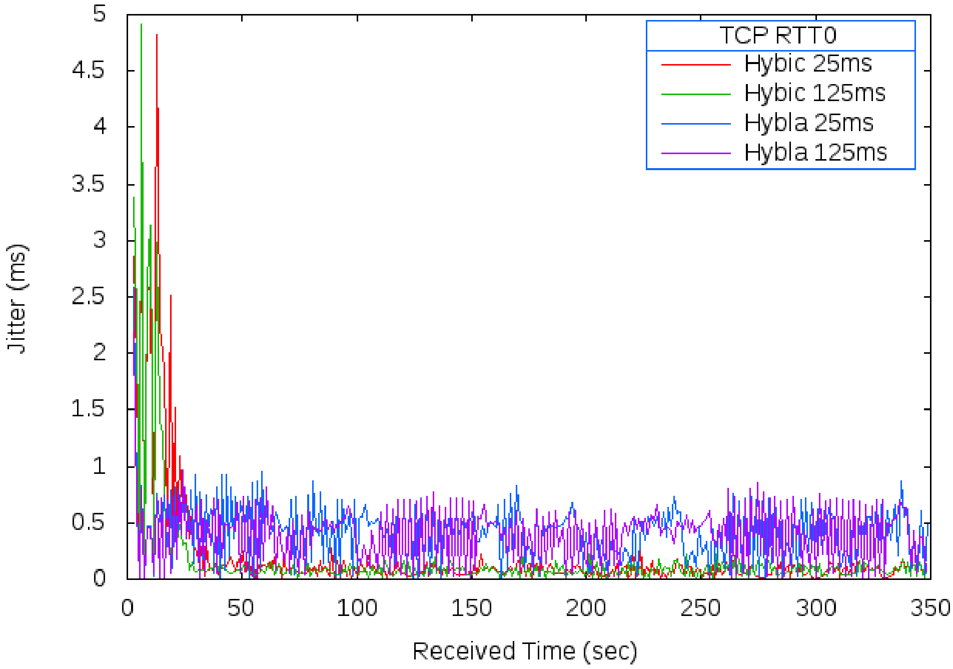


Figure 5.31: Measured Jitter During HYBIC and HYBLA Flows at Varied RTT_0

At RTT_0 values set to 25 ms and 125 ms, the instantaneous throughput of HYBIC (green and red colour) increases as the transmission (simulation) time increases, unlike HYBLA (blue and purple colour), which began to fall with the transmission time as shown in Fig 5.30. Moreover, jitter performance was observed to be less than 5 ms and decreases to less than 0.5 ms in HYBIC within the first 30 s of data transmission as shown in Fig. 5.31. Although, both HYBIC and HYBLA have excellent jitter performance, HYBIC achieved an overall better performance as summarised in Table 5.12.

Table 5.12: HYBIC Vs HYBLA Overall Simulated Performance at Varied RTT_0

$RTT_0(ms)$	$W_{avg}^{HC}(pkts)$	$W_{avg}^{H}(pkts)$	$R_{avg}^{HC}(kbps)$	$R_{avg}^{H}(kbps)$	$Jitt_{avg}^{HC}(ms)$	$Jitt_{avg}^{H}(ms)$
25	386	259	5025	2698	0.0886	0.4681
75	358	263	4672	2783	0.0895	0.4712
125	376	260	4920	2678	0.0828	0.4523
Overall Avg	373	261	4872	2720	0.0870	0.4639

The overall average performance (see Table 5.12) of HYBIC as compared to HYBLA sharing the same BNL in the multiple flows scenario showed that, HYBIC achieved better performance in all the metrics measured.

Performance of HYBIC and HYBLA was also compared under multiple flow scenario in terms of the packet delivery performance (PDP) and capacity utilisation as given by Fig. 5.32 and Table 5.13.

HYBIC transferred successfully over a million packets (see Fig. 5.32) with extremely low packet drops/losses of an overall average of 371 pkts as compared to HYBLA with more packet drops/losses of 27128 pkts as shown in Table 5.13. This showed that, HYBIC achieved better overall performance including the *cwnd* evolution, transmission rate (*throughput*) packet delivery ratio (PDR) and better capacity utilisation of 21.52% more than HYBLA, as given in Table 5.13.

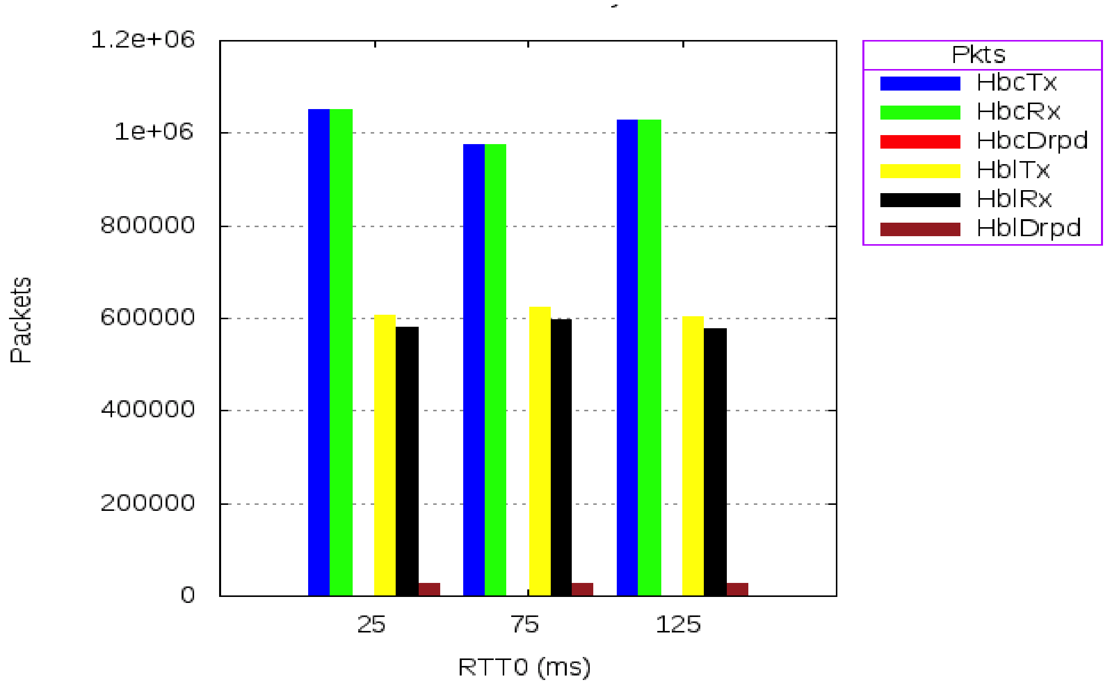


Figure 5.32: HYBIC Vs HYBLA Packet Delivery Performance at Varied RTT_0

Table 5.13: HYBIC Vs HYBLA Overall PDP at Varied RTT_0

$RTT_0(ms)$	$Drpd^{HC}(pkts)$	$Drpd^{H}(pkts)$	$PDR^{HC}(\%)$	$PDR^{H}(\%)$	$\eta^{HC}(\%)$	$\eta^{H}(\%)$
25	471	26898	99.96	95.57	50.25	26.98
75	273	26721	99.97	95.71	46.72	27.83
125	370	27764	99.96	95.40	49.20	26.78
Overall Avg	371	27128	99.96	95.56	48.72	27.20

5.4.3 HYBIC and CUBIC

HYBIC performance was evaluated and compared with CUBIC under multiple flows scenarios with CBR traffic over UDP connection similar to the setup discussed in the previous section. CUBIC is an accepted large BDP (LFN) protocol, set as default TCP on Linux and recently published by IETF as an RFC 8312 [20].

Figure 5.33 shows the *cwnd* evolution function of HYBIC compared to CUBIC under a scenario of multiple competing flows. As observed from the *cwnd* growth functions, both HYBIC and CUBIC used the cubic function of the elapsed time from the last congestion or loss event and CA phase, but each of the protocols used different algorithms for the SS phase. Although, the CUBIC algorithm exits the SS phase faster (in ≤ 5 s) compared to HYBIC (about 50 s in SS), CUBIC *cwnd* started decreasing with both increasing RTT_0 and elapsed time as shown in Fig 5.33. This was not the case for HYBIC in which the *cwnd* increases steadily with the transmission (elapsed) time except when packet loss occurred as shown by Fig 5.33.

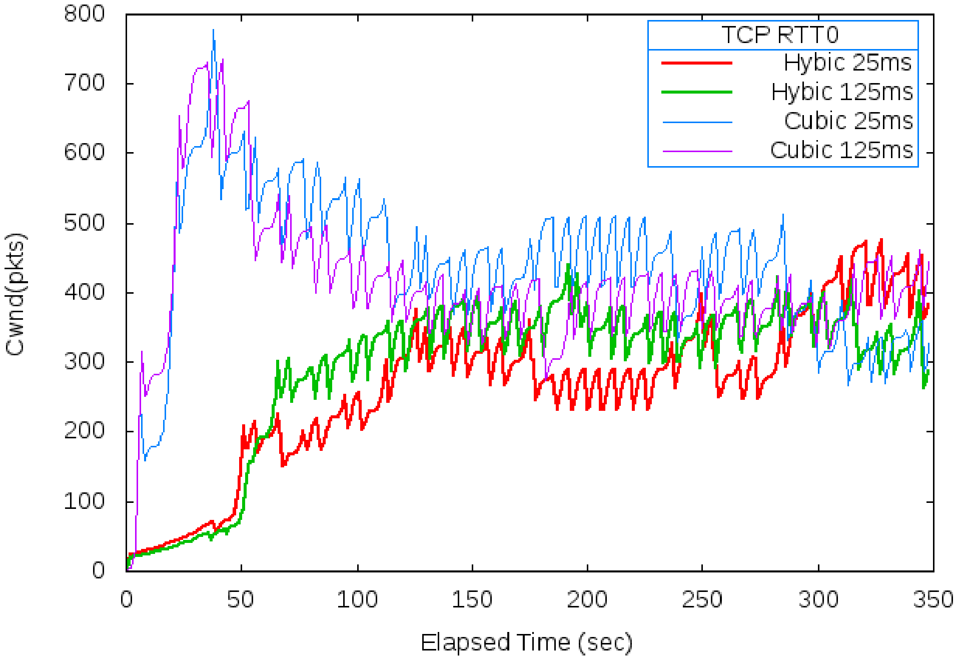


Figure 5.33: HYBIC Vs CUBIC Window Growth Rate, $W(t)$ at Varied $RTTs$

Under the competing flow scenarios, the two algorithms were observed to started converging after 100 s of data transmission, when HYBIC *cwnd* growth increases while CUBIC *cwnd* growth decreases. The effect of changing RTT_0 in the implementation was noticeable without much negative impact on the *cwd* growth of both algorithms, since the difference was less than 30 pkts between all the RTT_0 implementations used in this scenario.

The instantaneous throughput performance of HYBIC and CUBIC given by Fig. 5.34 showed a steady increase in performance of HYBIC as the elapsed time increased. This is an indication of efficient utilisation of HYBIC while competing with well established protocols such as CUBIC and UDP flows. Similar observations made about the *cwnd* growth of CUBIC and HYBIC decreasing and increasing respectively (see Fig. 5.34) with the increase in the elapsed time. This indicated that CUBIC was aggressive at the start of the transmission of competing flows, but falls as more packets are transmitted over the period of time to a point of convergence where HYBIC achieved more throughput.

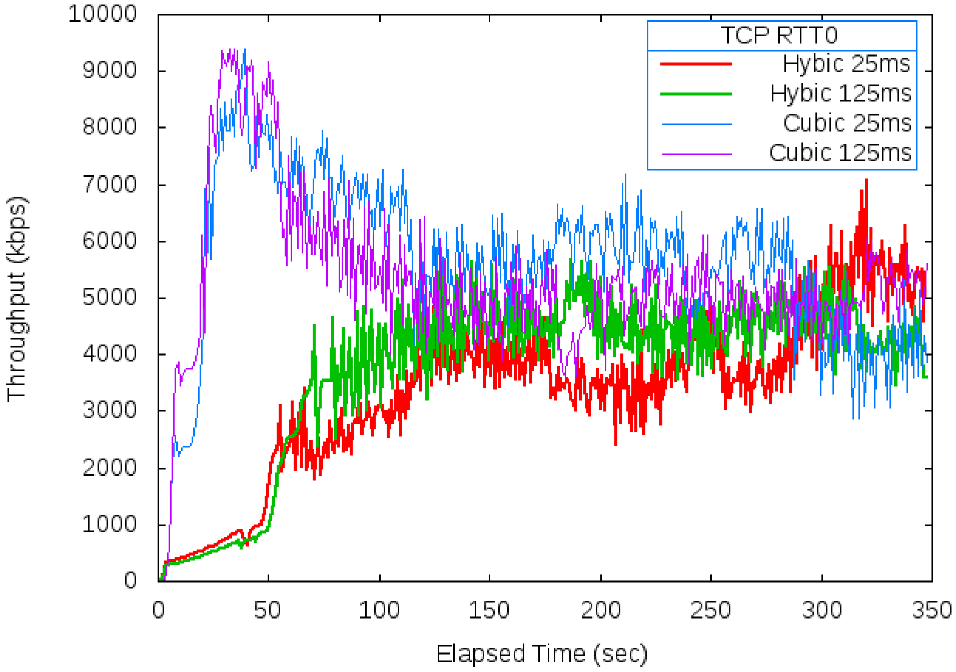


Figure 5.34: HYBIC Vs CUBIC Transmission Rate, $R(t)$ at Varied $RTTs$

Another performance metric measured under multiple competing flows scenario is the realtime jitter (IPDV) of the competing TCP flows shown in Fig. 5.35. The jitter of both algorithms were below 5 ms with CUBIC experiencing the highest jitter between 4-5 ms for both RTT_0 implemented during the first 5 s and decreased to about 0 ms afterward. The highest jitter recorded for HYBIC was less than 2 ms in all the RTT_0 implementations as given in Fig. 5.35.

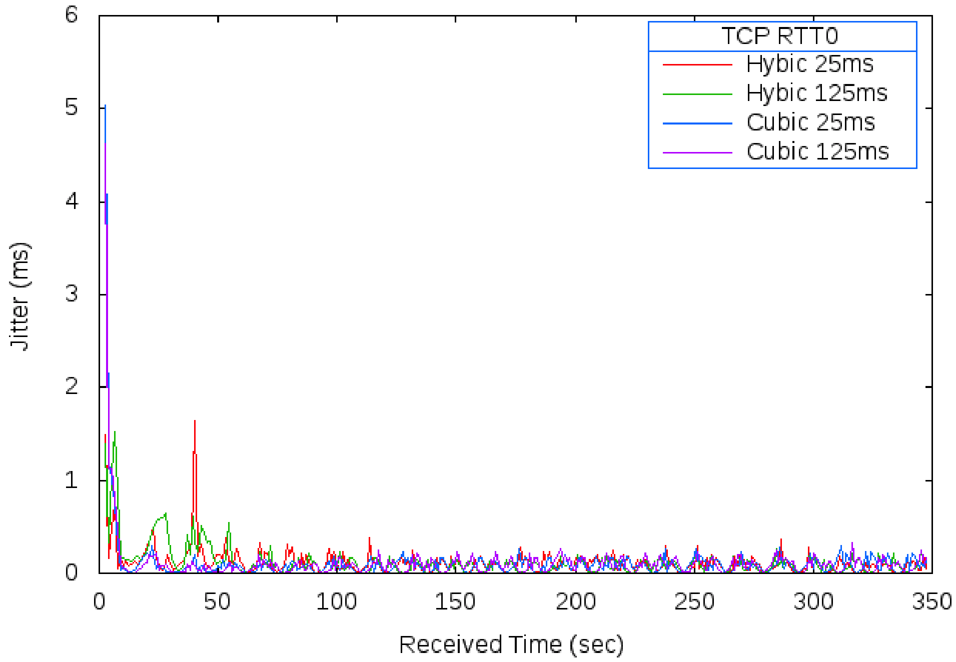


Figure 5.35: Measured jitter During HYBIC and CUBIC Flows at Varied RTT_0

The overall summary of the performance and its metrics are given in Table 5.14 for comparison between HYBIC and CUBIC schemes. Although, the overall summary indicates high performance metric values for CUBIC, it was observed that HYBIC is more consistent and stable with increased performance throughout the duration of the data transmission, as observed form Figs. 5.33, 5.34, and 5.35. These indicated improved fairness, friendliness and stability of HYBIC among the competing flows of high-speed TCP schemes such as CUBIC.

Table 5.14: HYBIC Vs CUBIC Overall Simulated Performance at Varied RTT_0

$RTT_0(ms)$	$W_{avg}^{HC}(pkts)$	$W_{avg}^{C}(pkts)$	$R_{avg}^{HC}(kbps)$	$R_{avg}^{C}(kbps)$	$Jitt_{avg}^{HC}(ms)$	$Jitt_{avg}^{C}(ms)$
25	262	436	3379	5643	0.0868	0.0799
75	289	416	3713	5366	0.0797	0.0790
125	291	415	3743	5357	0.0805	0.0811
Overall Avg	281	422	3612	5455	0.0823	0.0800

Keys: HC: HYBIC, C: CUBIC

Finally, the packet delivery performance (PDP) of these two high-speed schemes were also measured and compared. The dropped packets were extremely low compared to the sent (Tx) and received (Rx) packets as given in Table 5.15. These were for comparison between HYBIC and CUBIC performance in terms of packet drops and delivery ratio. Although, HYBIC is not standardised protocol like CUBIC, it showed an impressive level of performance under competing flows with accepted standards such as CUBIC.

Table 5.15: HYBIC Vs CUBIC Overall Packet Delivery Performance at Varied RTT_0

$RTT_0(ms)$	$Drpd^{HC}(pkts)$	$Drpd^{C}(pkts)$	$PDR^{HC}(\%)$	$PDR^{C}(\%)$	$\eta^{HC}(\%)$	$\eta^{C}(\%)$
25	449	351	99.94	99.97	33.79	56.43
75	412	466	99.95	99.96	37.13	53.66
125	326	480	99.96	99.96	37.43	53.57
Overall Avg	396	432	99.95	99.96	36.12	54.55

By careful inspection of Table 5.15, HYBIC was observed to have fewer packet drops during the data transmission period, while CUBIC was able to deliver more packets within that time. This may be due to priority given to the CUBIC as the default algorithm on Linux or the lack of fairness and friendliness with other competing flows considering the capacity utilisation. The PDR of HYBIC was about the same as that of CUBIC, achieving an excellent overall average value of up to 99.95% (see Table 5.15) from all RTT_0 implementations.

References

[1] V. Jacobson, and R. Braden, "TCP Extensions for Long-Delay Paths", *Network Working Group*, RFC 1072, 1988.

[2] T. R. Henderson, and R. H. Katz, "TCP Performance over Satellite channels", *University of California Berkeley, Report No.*, UCB/CSD-99-1083, California, 1999.

[3] J. S. Stadler, and J. Gelman, "Performance Enhancement for TCP/IP on a Satellite channel", *In Proc. of IEEE Military Communications Conference (MILCOM)*, Vol. 1, pp. 270-276, 1999.

[4] C. Metz, "TCP over Satellite...The Final Frontier", *IEEE Internet Computing, On the Wire*, pp. 76-80, 1999.

[5] N. Ghani, and S. Dixit "TCP/IP Enhancement for Satellite Networks", *IEEE Communications Magazine*, pp. 64-72, 1999.

[6] M. Allman, D. Glover, and L. Sanchez, "Enhancing TCP Over Satellite Channels Using Standard Mechanisms", *The Internet Society/IETF*, RFC 2488 , 1999.

[7] M. Allman, et al., "Ongoing TCP Research Related to Satellites," *Internet Society Network Working Group, RFC*, 2760, BCP. 28 2000.

[8] B. Barakat, E. Altman, and W. Dabbous, " On TCP Performance in a Heterogeneous Network: A Survey", *IEEE Communications Magazine, Telecommunications at the Start of the New Millennium*, pp. 40-46, 2000.

[9] S. Kota, and M. Marchese, ”Quality of Service for Satellite IP Networks: A Survey”, *International Journal of Satellite Communications and Networking*, Vol. 21, pp. 303-349, John Wiley, 2003.

[10] C. Caini, and R. Firrincieli, ”TCP HYBLA: A TCP Enhancement for Heterogeneous Networks”, *International Journal of Satellite Communications and Networking (IJSCN)*, Vol. 22, pp. 547-566, John Wiley, 2004.

[11] C. Caini, and R. Firrincieli, ”End-to-End TCP Enhancements Performance on Satellite Links”, *In Proc. 11th IEEE Symposium on Computers and Communications (ISCC'06)*, 1031-1036, 2006.

[12] A. Tang, J. Wang, S. H. Low and M. Chiang, ”Equilibrium of Heterogeneous Congestion Control: Optimality and Stability”, *IEEE/ACM Transaction on Networking*,Vol. 18, No. 3, 2010.

[13] S. Ha, I. Rhee, and L. Xu, ”CUBIC: A New TCP-Friendly High-Speed TCP Variant”, *ACM SIGOPS Operating Systems Review - Research and Developments in the Linux Kernel*, Volume 42 Issue 5, pp. 64-74, 2008.

[14] A. Tang, J. Wang, S. H. Low and M. Chiang, ”Equilibrium of Heterogeneous Congestion Control: Existence and Uniqueness”, *IEEE/ACM Transaction on Networking*,Vol. 15, No. 4, 2007.

[15] C. Caini, R. Firrincieli, D. Lacamera, T. de Cola, M. Marchese, C. Marcondes, M. Y. Sanadidi, M. Gerla, ”Analysis of TCP Live Experiments on a Real GEO Satellite Testbed”, *Performance Evaluation*, Vol. 66, Issue 6, pp. 287-300, Elsevier, 2009.

[16] ITU, ” Transmission Control Protocol (TCP) Over Satellite Networks”, *International Telecommunication Union (ITU), Report*, S Series Rep,. ITU-R S.2148, pp. 1-24, 2009.

[17] ITU, ” Performance Enhancements of Transmission Control Protocol Over Satellite Networks”, *International Telecommunication Union (ITU), Recommendations*, S Series Rec., ITU-R S.1711-1, pp. 1-50, 2010.

[18] S. Trivedi, S. Jaiswal, R. Kumar, and R. Rao, "Comparative Performance Evaluation of TCP HYBLA and TCP CUBIC for Satellite Communication Under Low Error Conditions", *4th IEEE International Conference on Internet Multimedia Services Architecture and Application (IMSAA)*, Bangalore, India, 15-17 Dec. 2010.

[19] A. Pirovano, and F. Garcia "A New Survey on Improving TCP Performances over Geostationary Satellite Link", *Network and Communication Technologies*, Vol. 2, No. 1, pp. 1-18, Canadian Center of Science and Education, 2013.

[20] I. Rhee, *et al.*, "CUBIC for Fast Long-Distance Networks", *Internet Engineering Task Force*, RFC 8312, 2018.

[21] K. Fall, and K. Varadhan, "The *ns* Manual", *The VINT Project*, 2011. Accessed via www.isi.edu/nsnam/ns/ns-documentation

Chapter 6

Conclusions and Recommendation for Future Work

6.1 Conclusions

The high demand for digital connectivity is increasing exponentially due to the emergence of various and potential applications such as improved healthcare, agriculture, education, commerce and industry for the wellbeing and socio-economic development of today's modern society. No doubt, digital connectedness has become indispensable for achieving the SDGs and making the world a global village regardless of where you live as human. However, those who live in urban areas enjoy more digital connectivity than those in isolated remote rural areas, especially in Africa where about 66% [1], of the population are digitally unconnected/disconnected due to economic or terrain difficulties with about 60% rural dwellers [2, 3]. Moreover, recent studies showed that about 4 billion of the 7.593 billion world's population are disconnected digitally without internet access [1, 4].

Optimised data transmission over SatComms and heterogeneous networks characterised by *high bandwidth/capacity* and *global coverage* is an excellent candidate for most if not all of these applications and help in bridging the digital divide between the connected and unconnected world. Additionally, these characteristics (high capacity and global coverage) of SatComms makes it one of the key requirements that enable and support the use cases for the next 5th generation new radio

(NR) communications network in proving enhanced mobile broadband (eMBB) and massive machine type communications (mMTC) services. Therefore, optimised data transmission over SatComms networks such as GEO and non-GEO HTS designed to provide both high capacity, efficient bandwidth utilisation and global coverage using smart beam forming/steering to priority areas become vital in achieving these goals.

However, the major challenge of using satellite channels for data transportation especially the famous and most widely used transport protocol, TCP over GEO satellite links are capacity under utilisation and other performance degradation such as packet losses/drops due the inherent large BDP and higher PER of SatComms and wireless network environment compared to short RTT and low PER terrestrial counterparts. This performance degradation resulted due to the in ability of the standard TCP to distinguish and resolved congestions problems due to link errors or congested link since it was designed based on RTT/ACK to perform effectively for short RTT and low PER networks like wired terrestrial networks.

There are different enhanced TCP algorithm schemes proposed to solve the drawbacks of the standard TCP, but these proposals mainly focused on either solving the negative impact of long RTT or high capacity or PER of the wireless links. Since satellite links exhibit all these characteristics, this thesis focused on the combined effects of long RTT and high capacity that led to large BDP of purely satellite or a more realistic hybrid satellite-terrestrial network (ISTN) environments that described practical communications network nowadays. The practically realistic communication networks consisted of wireless, wired, satellite, terrestrial, different RTTs (short and long), and different (high and low) capacity/bandwidth. These led to heterogeneous network environment with combination of different BDP and PER, which in turn lead to performance degradation of standard TCP such as severe capacity under utilisation, congestion problems, packet losses/drops, unwanted packets retransmissions, and unfair share of available bandwidth among competing flows of different network characteristics.

Methods proposed that attempted to solve the problems of large BDP due to either long RTT or high capacity, such as HYBLA (reduced the impacts of long

RTT) and CUBIC (reduced the impacts of high capacity), often targeted one of the parameters (RTT or high capacity) and sometimes resulted to aggressive transmissions starts, drastic decrease with increasing transmission time and unfair to other competing connections.

In order to mitigate and overcome these performance degradation problems of TCP over satellite or ISTN environment and provide a sustainable and resilient SatComms that can help in bridging the digital gap, particularly in isolated remote rural communities in Africa, the research work presented in this thesis investigated:

1. The actual (practical), emulated and simulated E2E latency of pure Satellite and hybrid ISTN environment, detailed in Chapter 4.

2. An effective Framework for E2E latency measurements in a pure Satellite and IST network environments. Developed for effective measurements of the practical E2E latency in a satellite IP network environment as described in Chapter 4.

3. The impacts of the practical E2E latency on performance of HYBLA, CUBIC and standard TCP over ISTN environment. Actual E2E latency was used as an input parameter in the numerical analysis and evaluation of different TCP schemes in Chapter 4 and 5.

4. Design and development of an improved transport protocol, called HYBIC for better performance of data transmission using TCP/IP over a pure satellite or heterogeneous IST network environment. Simulations and numerical evaluation of HYBIC performance and comparison with other high-speed TCP schemes were also conducted as discussed in Chapter 5 of this thesis.

The TCP HYBIC scheme developed by this thesis and proposed for high-speed and long RTT networks achieved improved performance as single flow and among multiple competing flows over simulated heterogeneous IST network scenario. Simulated as a single flow, HYBIC achieved up to 99.96% PDR, 30μs IPDV (jitter) and efficient capacity utilisation of 67.46%. While subjected to competing multiple flows of other high-speed and long RTT TCP schemes [5, 6] and CUBIC [7, 8], HYBIC showed a more stable and steady *cwnd* and transmission rates with increase elapsed time. Competing flows of HYBIC and CUBIC achieved up to 99.95% PDR, 82μ s and capacity utilisation of 36.12%, while competing HYBIC with HYBLA achieved 99.96% PDR, 87μs and capacity utilisation of 48.72%, which is about 22% more than HYBLA capacity utilisation. The PDR and IPDV performance of HYBIC are almost the same with that of CUBIC, which is the standard and default TCP in Linux OS, but out performed HYBLA in both cases. Therefore, the developed TCP HYBIC was able to achieved better and effective performance in terms of PDR, jitter, and capacity utilisation under long RTT, high bandwidth and competing flows network environments.

There are many areas of applications for this work as mentioned ealier, include but not limited to *Tele-medicine, Tele-agriculture, emergency* and *disaster management* communications as described in Chapter 6 of this thesis.

6.2 Recommendations for Future Work

Based on the experience acquired in the cause of this research work and discussed in this thesis, due to time and funding constraints, the following recommendations can provide ways for improvements in the future;

The E2E practical latency measured with two satellite network providers (SNPs), namely Inmarsat and Thuraya could be extended to include more satellite providers such as Eutelsat and Avanti using GEO Satellites. This process could take longer experimental time and very expensive considering the financial commitments, but could potentially further validate and improved the confidence in results of the practical E2E latency incurred by satellite IP networks environment. Moreover, more high data resolution could be obtained when a software is developed and integrated with the SUTs to transmit (Tx) and receive (Rx) latency measurement signals in almost real time as obtained by emulation and simulation. This will involve development and interfacing Hardware and software on the SUTs from different SNPs used in the testbed, which could also help to further investigate the impact of geographical location of both SUT and GWS.

More WiFi nodes as TCP HYBIC connections could be added on the ISTN topology and simulate single and multiple flows scenarios to further investigate the impacts of terrestrial wireless link errors as part of heterogeneous ISTN environment. This topology modifications can be tested with more high-speed and wireless TCP schemes such as HTCP, Vegas and Westwood with variable high BDP, BNL capacity, different types of Queue management and error models.

More performance analysis and evaluation using simulations of HYBIC in comparison with HYBLA and CUBIC with more realistic/practical values of large BDP based on longer RTT and higher bandwidth. Performance analysis, evaluation and comparison can be extended to include more TCP schemes designed for high wireless link error networks. This will take long experimental time and required more computing resources in terms of memory, processing and storage.

The performance of HYBIC in comparison to CUBIC can be further improved by simulation with longer duration of the data transmission (increase elapsed time above 350 s), this can be conducted using the recommended changes in both topology and implementation. This may require more computing resources as it will provide large traced data, the simulation will take longer time and need much more computing power.

As part of future work, we intend to explore the effect of background traffic on fairness and the effect of more complex network topologies could also be interesting. Experiments with more queue management schemes such as Active Queue Management (AQM) where packets are dropped from the queue before the queue overflows could also be an interesting work for the future. Many of the AQM schemes incorporate fairness enforcement schemes where they drop packets belonging to the larger flows. Behaviour of the high-speed TCPs in such an environment should be fairer. It would be interesting to see how much effect this has on their performance. Other things include more number of flows over the BNL at the same time and also inter-protocol fairness with many and different RTT flows. Potential areas of applications such as Tele-medicine and Tele-agriculture could be tested with network topologies designed for resilient and sustainable Satellite and ISTN communications. These would be useful for Telecommunications services and application in remote rural areas, and for emergency/disaster management when terrestrial communications infrastructure failed in the event of disaster.

References

[1] Hootsuit and We are Social, Digital in 2018: Essential Insights Into Internet, Social Media, Mobile and E-Commerce Use Around the World, accessed 17/04/2018 via https://wearesocial.com/uk/blog/2018/01/global-digital-report-2018. Jan 2018.

[2] A. Hall, and M. Rao "Future-Sat Africa... To Connect All ", *in Worldwide Satellite Magazine (SatMagazine) Nov. 2016*, pp. 44-47. Available: www.satnews.com/magazines.php.

[3] W. B. Group, Rural Population (% of Total Population), Available: www.data.worldbank.org/indicator/SP.RUR.TOTL.ZS end=2015&locations=ZG&start=1960 accessed on 31/03/2017.

[4] G. Giambene, S. Kota, and P. Pillai, "Satellite-5G Integration: A Network Perspective", *IEEE Network*, pp. 25-31, 2018

[5] C. Caini, and R. Firrincieli, "TCP HYBLA: A TCP Enhancement for Heterogeneous Networks", *International Journal of Satellite Communications and Networking (IJSCN)*, Vol. 22, pp. 547-566, John Wiley, 2004.

[6] C. Caini, R. Firrincieli, D. Lacamera, T. de Cola, M. Marchese, C. Marcondes, M. Y. Sanadidi, M. Gerla, "Analysis of TCP Live Experiments on a Real GEO Satellite Testbed", *Performance Evaluation*, Vol. 66, Issue 6, pp. 287-300, Elsevier, 2009.

[7] S. Ha, I. Rhee, and L. Xu, "CUBIC: A New TCP-Friendly High-Speed TCP Variant", *ACM SIGOPS Operating Systems Review - Research and Developments in the Linux Kernel*, Volume 42 Issue 5, pp. 64-74, 2008.

[8] I. Rhee, *et al.*, "CUBIC for Fast Long-Distance Networks", *Internet Engineering Task Force*, RFC 8312, 2018.

Appendix A

Software: Codes

A.1 MATLAB Codes

```
%The following codes were written as a  program that numerically implement
%and compare various TCP schemes for high BDP/LFN environments.
%the start of the main function called tcpnum19()
function tcpnum19()
%Note: K=10^3 and k=2^10 =1024-bytees
% menu to select functions to run
a=menu('Run','Algorithms','Data Processes');
if a==1
%input the different RTT values
x= inputdlg({'RTT_{max}','RTT_{min}','RTT_{avg}','RTT_{ref}'},
                               'Round-Trip-Time (ms)',1);

%input the initial cwnd (IW) = 10segs, slow start threshold (ssth = 128segs),
%& TCP max segment size(MSS) 1024
y = inputdlg({'Initial CWND, IW (segs)','SS Threshold (Segs)','MSS (bytes)'},
                                                             'TCP Parameters',1);

%Initial window size in segments
```

```
IW =str2double(y{1});

% Return slow start threshold (ssth) for the TCP algorithm
ssth =str2double(y{2});

%Maximum Segment Size (MSS)in bytes, 1448 or 1024 in hybla
MSS =str2double(y{3});

%Start &Elapsed times since transmission started, t = t_min:dt:t_max;
z = inputdlg({'Initial (Start) Time (s)','Time Step(s)','Elapsed Time (s)'},
                                                'Elapsed Time (secs)',1);

%start time for data transmission
t_min =str2double(z{1});

%Time step
dt =str2double(z{2});

% elapse time
t_max =str2double(z{3});
else

end
%create empty 0x0 table to store data
SHT19 =table;
ST19 =table;

%%%---------------select option (function) to execute----------%%%
exit_loop = 1;
while (exit_loop)
    A = menu('Choose Parameter to Compute','Standard TCP', 'Hybla','Mod-Hybla',
                'Cubic','Hybic','Mod-Hybic','Plot','Stats', 'Done');
    switch(A)
        case{1}
            %tcp
            [SHT19] = stcp(x,t_min, dt, t_max,ssth, MSS, SHT19);
            %break; it terminates the execution of next case/while
        case{2}
            %Hybla
            [SHT19] = hybla(x,t_min, dt, t_max,ssth, MSS, SHT19);
        case{3}
            [SHT19] = mhybla(x,t_min, dt, t_max,ssth, MSS, SHT19);
        case{4}
            %Cubic
            [SHT19] = cubic(x,t_min, dt, t_max,ssth, MSS, SHT19);
        case{5}
            %Hybic
            [SHT19] = hybic(x,t_min, dt, t_max,ssth, MSS, SHT19);
        case{6}
            %modified hybic
            [SHT19] = mhybic(x,t_min, dt, t_max,ssth, MSS, SHT19);
        case{7}
            %data plots
            [SHT19] = plotprm(SHT19);
        case{8}
            % statisitcs
            [ST19] = statsd();
        case{9}
            exit_loop = 0;
        otherwise
            errordlg('Invalid Parameter input','Check');
    end
```

```
end


%%%---------------standard tcp function to execute----------%%%
    function [SHT19] = stcp(x,t_min,dt,t_max,ssth,MSS,SHT19)
        for i=1:3
            % Time when ssth is reached
            t_sst = (str2double(x{i})* 10^-3) * log2(ssth);
            j=1;
            for t = t_min:dt:t_max
                if (t < t_sst)
                    %slow start phase
                    % parameters in segments unit
                    % W (cwnd) in segs
                    ws = 2^(t/(str2double(x{i})* 10^-3));
                    %Instantaneous segment transmissio rate (STR/B) segs/sec
                    str_s = ws/(str2double(x{i})* 10^-3);
                    %Total segments/data transmitted (segs) since TX starts
                    tst_s = ((2^(t/(str2double(x{i})* 10^-3)) - 1)/log(2));
                    % parameters in bytes unit
                    ws_B = ws * MSS;
                    str_sB = str_s * MSS;
                    tst_sB =tst_s * MSS;
                    % parameters in bits unit
                    ws_b = ws * MSS * 8;
                    str_sb = str_s * MSS * 8;
                    tst_sb =tst_s * MSS * 8;
                elseif (t >= t_sst)
                    % Congestion Avoidance phase
                    % parameters in segments unit
                    ws = (((t-t_sst)/(str2double(x{i})* 10^-3)) +ssth);
                    str_s = ws/(str2double(x{i})* 10^-3);
                    tst_s = (((ssth-1)/log(2)) +
                    ((t-t_sst)^2/(2*(str2double(x{i})*10^-3)^2)) +
                    (ssth*(t-t_sst)/(str2double(x{i})* 10^-3)));
                    % parameters in bytes unit
                    ws_B = ws * MSS;
                    str_sB = str_s * MSS;
                    tst_sB =tst_s * MSS;
                    % parameters in bits unit
                    ws_b = ws * MSS * 8;
                    str_sb = str_s * MSS * 8;
                    tst_sb =tst_s * MSS * 8;
                else

                end
                WS(j)= ws;
                WS_B(j) = ws_B;
                WS_b(j) = ws_b;
                STR_s(j)= str_s;
                STR_sB(j) = str_sB;
                STR_sb(j) = str_sb;
                TST_s(j) = tst_s;
                TST_sB(j) = tst_sB;
                TST_sb(j) = tst_sb;
                tet(j) = t;
                j = j + 1;
            end
            b = menu('RTT Used', 'RTT_(Max)', 'RTT_(Min)', 'RTT_(Avg)');
            if b == 1
                SHT19.TTE = tet';
                SHT19.WS_mxS = WS';
                SHT19.WS_mxB = WS_B';
                SHT19.WS_mxb = WS_b';


                SHT19.ITR_mxS = STR_s';

                SHT19.ITR_mxB = STR_sB';

                SHT19.ITR_mxb = STR_sb';


                SHT19.TST_mxS = TST_s';

                SHT19.TST_mxB = TST_sB';

                SHT19.TST_mxb = TST_sb';

            elseif b==2

                SHT19.WS_mnS = WS';

                SHT19.WS_mnB = WS_B';

                SHT19.WS_mnb = WS_b';


                SHT19.ITR_mnS = STR_s';

                SHT19.ITR_mnB = STR_sB';

                SHT19.ITR_mnb = STR_sb';


                SHT19.TST_mnS = TST_s';

                SHT19.TST_mnB = TST_sB';

                SHT19.TST_mnb = TST_sb';

            else

                SHT19.WS_avS = WS';

                SHT19.WS_avB = WS_B';

                SHT19.WS_avb = WS_b';


                SHT19.ITR_avS = STR_s';

                SHT19.ITR_avB = STR_sB';

                SHT19.ITR_avb = STR_sb';


                SHT19.TST_avS = TST_s';
```

```
                SHT19.TST_avB = TST_sB’;
                SHT19.TST_avb = TST_sb’;
            end
            i = i + 1;
            writetable(SHT19,’iwcmc19SHTCP25.xlsx’);
        end
%%%---------------Hybla Algorithm function to execute----------%%%
    function [SHT19] = hybla(x,t_min, dt, t_max,ssth, MSS, SHT19)
        % Time when ssth is reached in hybla
        t_ssto = (str2double(x{4})* 10^-3) * log2(ssth);
        %l=1;
        for l=1:3
            a = menu(’RTT/Rho for Hybla’, ’RTT_(Max)’, ’RTT_(Min)’,
                                   ’RTT_(Avg)’);
            %a= 1,2 & 3
            if a==1
                rho = (str2double(x{1})* 10^-3) / (str2double(x{4})* 10^-3);
            elseif a==2
                rho = (str2double(x{2})* 10^-3) / (str2double(x{4})* 10^-3);
            else
                rho = (str2double(x{3})* 10^-3) / (str2double(x{4})* 10^-3);
            end
            k=1;
            %wh(k)=0;
            for t = t_min:dt:t_max
                if (t < t_ssto)
                    %slow start phase
                    %Segments unit
                    % W (cwnd)Ksegs
                    wh = rho*(2^(t/(str2double(x{4})* 10^-3)));
                    str_h = wh/(str2double(x{4})* 10^-3);
                    tst_h = ((2^((str2double(x{4})* 10^-3)) - 1)/log(2));

                    %Bytes unit
                    wh_B = wh * MSS; %Kbytes
                    str_hB = str_h * MSS; %Kbytes/sec
                    tst_hB = tst_h * MSS;
                    %Bits unit
                    wh_b = wh * MSS * 8; %Kbytes
                    str_hb = str_h * MSS * 8; %Kbytes/sec
                    tst_hb = tst_h * MSS * 8;
                elseif (t >= t_ssto)
                    % Congestion Avoidance phase
                    %Segments unit
                    wh = rho*(((t-t_ssto)/(str2double(x{4})* 10^-3)) +ssth);
                    str_h = wh/(str2double(x{4})* 10^-3);
                    tst_h = ((ssth-1)/log(2) +
                    ((t-t_ssto)^2)/2*(str2double(x{4})* 10^-3)^2 +
                    ssth*(t-t_ssto)/(str2double(x{4})* 10^-3));

                    %in bytes unit
                    wh_B = wh * MSS;
                    str_hB = str_h * MSS;
                    tst_hB = tst_h * MSS;
                    %Bits unit
                    wh_b = wh * MSS * 8;
                    str_hb = str_h * MSS * 8;
                    tst_hb = tst_h * MSS * 8;
                else

                end
                %kilos (10^-3)
```

```
        WH(k)= wh * 10^-3;

        WH_B(k) = wh_B * 10^-3;

        WH_b(k) = wh_b * 10^-3;

        STR_hs(k)= str_h * 10^-3;

        TST_hs(k) = tst_h * 10^-3;

        STR_hB(k) = str_hB * 10^-3;

        TST_hB(k) = tst_hB * 10^-3;

        STR_hb(k) = str_hb * 10^-3;

        TST_hb(k) = tst_hb * 10^-3;

        tet(k) =t;

        k = k+1;

    end


    if a==1

        SHT19.TTE = tet';

        SHT19.WH_mxS = WH';

        SHT19.WH_mxB = WH_B';

        SHT19.WH_mxb = WH_b';

        SHT19.ITR_mxHS = STR_hs';

        SHT19.ITR_mxHB = STR_hB';

        SHT19.ITR_mxHb = STR_hb';

        SHT19.TST_mxHS = TST_hs';

        SHT19.TST_mxHB = TST_hB';

        SHT19.TST_mxHb = TST_hb';

    elseif a==2

        SHT19.WH_mnS = WH';

        SHT19.WH_mnB = WH_B';

        SHT19.WH_mnb = WH_b';

        SHT19.ITR_mnHS = STR_hs';

        SHT19.ITR_mnHB = STR_hB';

        SHT19.ITR_mnHb = STR_hb';
```

```
                SHT19.TST_mnHS = TST_hs';
                SHT19.TST_mnHB = TST_hB';
                SHT19.TST_mnHb = TST_hb';
            else
                SHT19.WH_avS = WH';
                SHT19.WH_avB = WH_B';
                SHT19.WH_avb = WH_b';
                SHT19.ITR_avHS = STR_hs';
                SHT19.ITR_avHB = STR_hB';
                SHT19.ITR_avHb = STR_hb';
                SHT19.TST_avHS = TST_hs';
                SHT19.TST_avHB = TST_hB';
                SHT19.TST_avHb = TST_hb';
            end
            l = l+1;
           writetable(SHT19,'iwcmc19SHTCP25.xlsx');
        end

%%%--------------- modified (rho) Hybla Algorithm function to execute---------------------
    function [SHT19] = mhybla(x,t_min, dt, t_max,ssth, MSS, SHT19)
        for l=1:3
            a = menu('RTT/Rho for MHybla', 'RTT_(Max)', 'RTT_(Min)',
                                             'RTT_(Avg)');
            %a= 1,2 & 3
            if a==1
                rho = (str2double(x{1})* 10^-3) / (str2double(x{4})* 10^-3);
                t_sstor = (str2double(x{4})* 10^-3) * log2(rho*ssth);
            elseif a==2
                rho = (str2double(x{2})* 10^-3) / (str2double(x{4})* 10^-3);
                t_sstor = (str2double(x{4})* 10^-3) * log2(rho*ssth);
            else
                rho = (str2double(x{3})* 10^-3) / (str2double(x{4})* 10^-3);
                t_sstor = (str2double(x{4})* 10^-3) * log2(rho*ssth);
            end
            k=1;
            for t = t_min:dt:t_max
                if (t < t_sstor)
                    %slow start phase
                    %Segments unit
                    % W (cwnd)Ksegs
                    whm = rho*(2^(t/(str2double(x{4})* 10^-3)));
                    % Tx rate (Ksegs/sec)
                    str_hm = ((2^(t/(str2double(x{4})* 10^-3)))/(str2double(x{4
                     % Transmitted data since Tx started
                    tst_hm = ((2^((str2double(x{4})* 10^-3)) - 1)/log(2));

                    %Bytes unit
                    whm_B = whm * MSS;
                    str_hmB = str_hm * MSS;
                    tst_hmB = tst_hm * MSS;
                    %Bits unit
                    whm_b = whm * MSS * 8;
                    str_hmb = str_hm * MSS * 8;
                    tst_hmb = tst_hm * MSS * 8;
                elseif (t >= t_sstor)
                    % Congestion Avoidance phase
                    %Segments unit
                    whm = rho*(((t-t_sstor)/(str2double(x{4})* 10^-3)) +ssth);
                    str_hm = (1/(str2double(x{4})* 10^-3))*(((t-t_sstor)/(str2d
                    tst_hm = ((ssth-1)/log(2) +
                     ((t-t_sstor)^2)/2*(str2double(x{4})* 10^-3)^2 +
                     ssth*(t-t_sstor)/(str2double(x{4})* 10^-3));
```

```
            %in bytes unit
            whm_B = whm * MSS;  %Kbytes
            str_hmB = str_hm * MSS;
            tst_hmB = tst_hm * MSS;
            %Bits unit
            whm_b = whm * MSS * 8;
            str_hmb = str_hm * MSS * 8;
            tst_hmb = tst_hm * MSS * 8;
        else


        end
        WHM(k)= whm * 10^-3;
        WHM_B(k) = whm_B * 10^-3;
        WHM_b(k) = whm_b * 10^-3;
        STR_hms(k)= str_hm * 10^-3;
        TST_hms(k) = tst_hm * 10^-3;
        STR_hmB(k) = str_hmB * 10^-3;
        TST_hmB(k) = tst_hmB * 10^-3;
        STR_hmb(k) = str_hmb * 10^-3;
        TST_hmb(k) = tst_hmb * 10^-3;
        tet(k) =t;
        k = k+1;
    end
    if a==1
        SHT19.TTE = tet';
        SHT19.WHM_mxS = WHM';
        SHT19.WHM_mxB = WHM_B';
        SHT19.WHM_mxb = WHM_b';
        SHT19.ITR_mxHMS = STR_hms';
        SHT19.ITR_mxHMB = STR_hmB';
```

```
                SHT19.ITR_mxHMb = STR_hmb';
                SHT19.TST_mxHMS = TST_hms';
                SHT19.TST_mxHMB = TST_hmB';
                SHT19.TST_mxHMb = TST_hmb';
            elseif a==2
                SHT19.WHM_mnS = WHM';
                SHT19.WHM_mnB = WHM_B';
                SHT19.WHM_mnb = WHM_b';
                SHT19.ITR_mnHMS = STR_hms';
                SHT19.ITR_mnHMB = STR_hmB';
                SHT19.ITR_mnHMb = STR_hmb';
                SHT19.TST_mnHMS = TST_hms';
                SHT19.TST_mnHMB = TST_hmB';
                SHT19.TST_mnHMb = TST_hmb';
            else
                SHT19.WHM_avS = WHM';
                SHT19.WHM_avB = WHM_B';
                SHT19.WHM_avb = WHM_b';
                SHT19.ITR_avHMS = STR_hms';
                SHT19.ITR_avHMB = STR_hmB';
                SHT19.ITR_avHMb = STR_hmb';
                SHT19.TST_avHMS = TST_hms';
                SHT19.TST_avHMB = TST_hmB';
                SHT19.TST_avHMb = TST_hmb';
            end
            l = l+1;
        end
%%%---------------cubic algorithm function to execute----------%%%
    function [SHT19] = cubic(x,t_min, dt, t_max,ssth, MSS, SHT19)
            C=0.4;
            beta = 0.2;
```

```
   K=0;
for i=1:3
   % Time when ssth is reached
   t_sst = (str2double(x{i})* 10^-3) * log2(ssth);
   j=1;
   for t = t_min:dt:t_max

       if (t < t_sst)
           %slow start phase
           % parameters in segments unit
           % W (cwnd) in segs
           wc = 2^(t/(str2double(x{i})* 10^-3));
           %Instantaneous segment transmissio rate (STR/B) segs/sec
           str_c = wc/(str2double(x{i})* 10^-3);
           %Total segments/data transmitted (segs) since TX starts
           tst_c = ((2^(t/(str2double(x{i})* 10^-3)) - 1)/log(2));

           % parameters in bytes unit
           wc_B = wc * MSS;
           str_cB = str_c * MSS;
           tst_cB =tst_c * MSS;
           % parameters in bits unit
           wc_b = wc * MSS * 8;
           str_cb = str_c * MSS * 8;
           tst_cb =tst_c * MSS * 8;
           W_max_c =wc;
       elseif (t >= t_sst)
           % Congestion Avoidance phase
           % parameters in segments unit
           wc =  C*(t-K)^3 + W_max_c;
           str_c = wc/(str2double(x{i})* 10^-3);
```

```
        tst_c = (((ssth-1)/log(2)) +
         ((t-t_sst)^2/(2*(str2double(x{i})* 10^-3)^2)) +
         (ssth*(t-t_sst)/(str2double(x{i})* 10^-3)));

        % parameters in bytes unit
        wc_B = wc * MSS;
        str_cB = str_c * MSS;
        tst_cB =tst_c * MSS;
        % parameters in bits unit
        wc_b = wc * MSS * 8;
        str_cb = str_c * MSS * 8;
        tst_cb =tst_c * MSS * 8;
    else


    end
    WC(j)= wc * 10^(-3); %Ksegs
    WC_B(j) = wc_B * 10^(-3);
    WC_b(j) = wc_b * 10^(-3);
    STR_c(j)= str_c* 10^(-3);
    STR_cB(j) = str_cB * 10^(-3);
    STR_cb(j) = str_cb * 10^(-3);
    TST_c(j) = tst_c * 10^(-3);
    TST_cB(j) = tst_cB * 10^(-3);
    TST_cb(j) = tst_cb * 10^(-3);
    tet(j) =t;
    j = j + 1;
end
b = menu(’RTT Used’, ’RTT_(Max)’, ’RTT_(Min)’, ’RTT_(Avg)’);
if b == 1
    SHT19.TTE = tet’;
    SHT19.WC_mxS = WC’;
```

```
        SHT19.WC_mxB = WC_B';
        SHT19.WC_mxb = WC_b';

        SHT19.ITR_mxC = STR_c';
        SHT19.ITR_mxCB = STR_cB';
        SHT19.ITR_mxCb = STR_cb';

        SHT19.TST_mxCS = TST_c';
        SHT19.TST_mxCB = TST_cB';
        SHT19.TST_mxCb = TST_cb';
    elseif b==2
        SHT19.WC_mnS = WC';
        SHT19.WC_mnB = WC_B';
        SHT19.WC_mnb = WC_b';

        SHT19.ITR_mnCS = STR_c';
        SHT19.ITR_mnCB = STR_cB';
        SHT19.ITR_mnCb = STR_cb';

        SHT19.TST_mnCS = TST_c';
        SHT19.TST_mnCB = TST_cB';
        SHT19.TST_mnCb = TST_cb';
    else
        SHT19.WC_avS = WC';
        SHT19.WC_avB = WC_B';
        SHT19.WC_avb = WC_b';

        SHT19.ITR_avCS = STR_c';
        SHT19.ITR_avCB = STR_cB';
        SHT19.ITR_avCb = STR_cb';
```

```
                SHT19.TST_avCS = TST_c’;
                SHT19.TST_avCB = TST_cB’;
                SHT19.TST_avCb = TST_cb’;
            end
            i = i + 1;
        end

%%%---------------hybic algorithm function to execute----------%%%
    function [SHT19] = hybic(x,t_min, dt, t_max,ssth, MSS, SHT19)
        % Time when ssth is reached in hybla
        t_sst = (str2double(x{4})* 10^-3) * log2(ssth);
        C=0.4;
        beta = 0.2;
        K=0;
    for l=1:3
      a = menu(’RTT/Rho for Hybla’, ’RTT_(Max)’, ’RTT_(Min)’, ’RTT_(Avg)’);
            %a= 1,2 & 3
            if a==1
                rho = (str2double(x{1})* 10^-3) / (str2double(x{4})* 10^-3);
            elseif a==2
                rho = (str2double(x{2})* 10^-3) / (str2double(x{4})* 10^-3);
            else
                rho = (str2double(x{3})* 10^-3) / (str2double(x{4})* 10^-3);
            end
            k=1;
            whc=0;
            for t = t_min:dt:t_max
                if (t < t_sst || whc <= ssth)
                    %slow start phase
                    %Segments unit
                    whc = rho*(2^(t/(str2double(x{4})* 10^-3)));
                    str_hc = whc/(str2double(x{4})* 10^-3);
                 tst_hc = ((2^((str2double(x{4})* 10^-3)) - 1)/log(2)* 10^(-3))

                    %Bytes unit
                    whc_B = whc * MSS;
                    str_hcB = str_hc * MSS;
                    tst_hcB = tst_hc * MSS;
                    %Bits unit
                    whc_b = whc * MSS * 8;
                    str_hcb = str_hc * MSS * 8;
                    tst_hcb = tst_hc * MSS* 8;
                    W_max_hc =whc;
                elseif (t >= t_sst || whc > ssth)
                    % Congestion Avoidance phase
                    % parameters in segments unit
                    whc =  C *(t-K)^3 + W_max_hc;
                    str_hc = whc/(str2double(x{4})* 10^-3);
                    tst_hc = (((ssth-1)/log(2)) +
                    ((t-t_sst)^2/(2*(str2double(x{4})* 10^-3)^2)) +
                    (ssth*(t-t_sst)/(str2double(x{4})* 10^-3)));

                    % parameters in bytes unit
                    whc_B = whc * MSS;
                    str_hcB = str_hc * MSS;
                    tst_hcB =tst_hc * MSS;

                    % parameters in bits unit
                    whc_b = whc * MSS * 8;
                    str_hcb = str_hc * MSS * 8;
                    tst_hcb =tst_hc * MSS * 8;
                else
```

```
        end
        WHC(k)= whc * 10^(-3); %Kseg
        WHC_B(k) = whc_B * 10^(-3);
        WHC_b(k) = whc_b * 10^(-3);
        STR_hcs(k)= str_hc * 10^-3;
        TST_hcs(k) = tst_hc * 10^-3;
        STR_hcB(k) = str_hcB * 10^-3;
        TST_hcB(k) = tst_hcB * 10^-3;
        STR_hcb(k) = str_hcb * 10^-3;
        TST_hcb(k) = tst_hcb * 10^-3;
        tet(k) =t;
        k = k+1;
    end
    if a==1
        SHT19.TTE = tet';
        SHT19.WHC_mxS = WHC';
        SHT19.WHC_mxB = WHC_B';
        SHT19.WHC_mxb = WHC_b';
        SHT19.ITR_mxHCS = STR_hcs';
        SHT19.ITR_mxHCB = STR_hcB';
        SHT19.ITR_mxHCb = STR_hcb';
        SHT19.TST_mxHCS = TST_hcs';
        SHT19.TST_mxHCB = TST_hcB';
        SHT19.TST_mxHCb = TST_hcb';
    elseif a==2
        SHT19.WHC_mnS = WHC';
        SHT19.WHC_mnB = WHC_B';
        SHT19.WHC_mnb = WHC_b';
        SHT19.ITR_mnHCS = STR_hcs';
        SHT19.ITR_mnHCB = STR_hcB';
```

```
                SHT19.ITR_mnHCb = STR_hcb';

                SHT19.TST_mnHCS = TST_hcs';

                SHT19.TST_mnHCB = TST_hcB';

                SHT19.TST_mnHCb = TST_hcb';

            else

                SHT19.WHC_avS = WHC';

                SHT19.WHC_avB = WHC_B';

                SHT19.WHC_avb = WHC_b';

                SHT19.ITR_avHCS = STR_hcs';

                SHT19.ITR_avHCB = STR_hcB';

                SHT19.ITR_avHCb = STR_hcb';

                SHT19.TST_avHCS = TST_hcs';

                SHT19.TST_avHCB = TST_hcB';

                SHT19.TST_avHCb = TST_hcb';

            end

            l = l+1;

            %Raw data from algorithms implemented table sth19 to xlsx

            %writetable(SHT19,'TCPData19.xlsx','Sheet',5);

            %writetable(SHT19,'TCPHCData19all50e.xlsx');

            %writetable(SHT19,'HCTCPData19H25fe.xlsx');

        end


%%%%------------------Modified Hybic function--------------------%%%%

     function [SHT19] = mhybic(x,t_min, dt, t_max,ssth, MSS, SHT19)

        t_sst = (str2double(x{4})* 10^-3) * log2(ssth);

        C=0.4;

        beta = 0.2;

        K=0;

        for l=1:3

            a = menu('Rho for RTTsat', 'RTT_(Max)', 'RTT_(Min)', 'RTT_(Avg)');

            %a= 1,2 & 3
```

```
   if a==1
       rho = (str2double(x{1})* 10^-3) / (str2double(x{4})* 10^-3);
   elseif a==2
       rho = (str2double(x{2})* 10^-3) / (str2double(x{4})* 10^-3);
   else
       rho = (str2double(x{3})* 10^-3) / (str2double(x{4})* 10^-3);
   end
   k=1;
   whcm=0;
   for t = t_min:dt:t_max
       if (t < t_sst || whcm <= ssth)
           %slow start phase
           %Segments unit
           %W (cwnd)Ksegs
           whcm = rho*(2^(t/(str2double(x{4})* 10^-3)));
           str_hcm = whcm/(str2double(x{4})* 10^-3);
          %Transmitted data since Tx started
tst_hcm = ((2^((str2double(x{4})* 10^-3)) - 1)/log(2)* 10^(-3));

           %Bytes unit
           %Kbytes (kB), Kbytes/sec
           whcm_B = whcm * MSS;
           str_hcmB = str_hcm * MSS;
           tst_hcmB = tst_hcm * MSS;
           %Bits unit
           whcm_b = whcm * MSS * 8; %Kbits (kb)
           str_hcmb = str_hcm * MSS * 8; %Kbps
           tst_hcmb = tst_hcm * MSS* 8;
           W_max_hcm =whcm;
       elseif (t >= t_sst || whcm > ssth)
           % Congestion Avoidance phase
```

```
        % parameters in segments unit
        whcm = rho * (C * (t-K)^3 + W_max_hcm);
        str_hcm = whcm/(str2double(x{4})* 10^-3);
        tst_hc = (((ssth-1)/log(2)) +
         ((t-t_sst)^2/(2*(str2double(x{4})* 10^-3)^2)) +
          (ssth*(t-t_sst)/(str2double(x{4})* 10^-3)));

        % parameters in bytes unit
        whcm_B = whcm * MSS;
        str_hcmB = str_hcm * MSS;
        tst_hcmB =tst_hcm * MSS;
        % parameters in bits unit
        whcm_b = whcm * MSS * 8;
        str_hcmb = str_hcm * MSS * 8;
        tst_hcmb =tst_hcm * MSS * 8;
    else


    end
    WHCM(k)= whcm * 10^(-3); %Kseg
    WHCM_B(k) = whcm_B * 10^(-3);
    WHCM_b(k) = whcm_b * 10^(-3);
    STR_hcms(k)= str_hcm * 10^-3;
    TST_hcms(k) = tst_hcm * 10^-3;
    STR_hcmB(k) = str_hcmB * 10^-3;
    TST_hcmB(k) = tst_hcmB * 10^-3;
    STR_hcmb(k) = str_hcmb * 10^-3;
    TST_hcmb(k) = tst_hcmb * 10^-3;
    tet(k) =t;
    k = k+1;
end
```

```
if a==1
    SHT19.TTE = tet';
    SHT19.WHCM_mxS = WHCM';
    SHT19.WHCM_mxB = WHCM_B';
    SHT19.WHCM_mxb = WHCM_b';
    SHT19.ITR_mxHCMS = STR_hcms';
    SHT19.ITR_mxHCMB = STR_hcmB';
    SHT19.ITR_mxHCMb = STR_hcmb';
    SHT19.TST_mxHCMS = TST_hcms';
    SHT19.TST_mxHCMB = TST_hcmB';
    SHT19.TST_mxHCMb = TST_hcmb';
elseif a==2
    SHT19.WHCM_mnS = WHCM';
    SHT19.WHCM_mnB = WHCM_B';
    SHT19.WHCM_mnb = WHCM_b';
    SHT19.ITR_mnHCMS = STR_hcms';
    SHT19.ITR_mnHCMB = STR_hcmB';
    SHT19.ITR_mnHCMb = STR_hcmb';
    SHT19.TST_mnHCMS = TST_hcms';
    SHT19.TST_mnHCMB = TST_hcmB';
    SHT19.TST_mnHCMb = TST_hcmb';
else
    SHT19.WHCM_avS = WHCM';
    SHT19.WHCM_avB = WHCM_B';
    SHT19.WHCM_avb = WHCM_b';
    SHT19.ITR_avHCMS = STR_hcms';
    SHT19.ITR_avHCMB = STR_hcmB';
    SHT19.ITR_avHCMb = STR_hcmb';
    SHT19.TST_avHCMS = TST_hcms';
    SHT19.TST_avHCMB = TST_hcmB';
    SHT19.TST_avHCMb = TST_hcmb';
```

```
            end
            l = l+1;
        end

%%%%----------Data Plots and statistic functions------------%%%%

%comparison plots
    function [SHT19] = plotprm(SHT19)


    end
%statistics table (ST)
   function [ST19] = statsd()


    end
```

A.2 TCL Codes

```
#A codes for ISTN topology designed to run TCP over ISTN environment
global ns
set ns [new Simulator]
#Data packets flow color definition for NAM
$ns color 1 Blue
$ns color 2 Blue
$ns color 3 Blue
$ns color 4 Blue
$ns color 5 Red
$ns color 6 black
$ns color 7 black

# Global configuration parameters
# set these global options for the satellite terminals

global opt
set opt(chan) Channel/Sat
set opt(bw_up) 10Mb
set opt(bw_down) 10Mb
set opt(phy) Phy/Sat
set opt(mac) Mac/Sat
set opt(ifq) Queue/DropTail
set opt(qlim) 50
set opt(ll) LL/Sat
set opt(wiredRouting) ON

# This tracing enabling must precede link and node creation
set outfile [open hbcub25100mb.tr w]
set winfile [open Cwnd25 w]
$ns trace-all $outfile

#Create NAM trace file
set ntf1 [open hbc125.nam w]
$ns namtrace-all $ntf1

#Create thirty six nodes
#for {set i 0} {$i < 13} {incr i} {
 #        set n($i) [$ns node]
#}

# Set up satellite and terrestrial nodes

# Configure the node generator for bent-pipe satellite
# geo-repeater uses type Phy/Repeater
```

```
$ns node-config -satNodeType geo-repeater \
-phyType Phy/Repeater \
-channelType $opt(chan) \
-downlinkBW $opt(bw_down)  \
-wiredRouting $opt(wiredRouting)

# GEO satellite at 100 degrees longitude West
set n(0) [$ns node]
$n(0) set-position -100
$n(0) color red
$n(0) shape hexagon

# Configure the node generator for satellite terminals
$ns node-config -satNodeType terminal \
                -llType $opt(ll) \
                -ifqType $opt(ifq) \
                -ifqLen $opt(qlim) \
                -macType $opt(mac) \
                -phyType $opt(phy) \
                -channelType $opt(chan) \
                -downlinkBW $opt(bw_down) \
                -wiredRouting $opt(wiredRouting)

# SU Terminals in Nigeria
#create two SUTs and add locations
set n(1) [$ns node]; set n(2) [$ns node]
$n(1) color blue
$n(1) shape box
$n(2) color blue
$n(2) shape box
# Kanoma Village, Maru (lat, Lon, alt)
```

```
$n(1) set-position 11.4 6.2;
#Gusau
$n(2) set-position 12.1 6.4;

#Add SUT GSLs to GEO satellite n(0)
$n(1) add-gsl geo $opt(ll) $opt(ifq) $opt(qlim) $opt(mac) $opt(bw_up) \
    $opt(phy) [$n(0) set downlink_] [$n(0) set uplink_]
$n(2) add-gsl geo $opt(ll) $opt(ifq) $opt(qlim) $opt(mac) $opt(bw_up) \
    $opt(phy) [$n(0) set downlink_] [$n(0) set uplink_]

#Topology script creation with nodes & links with config
#GWS Terminals in EU
$ns node-config -satNodeType terminal
set n(3) [$ns node]; set n(4) [$ns node]
$n(3) set-position 53.2 6.14; #Burum, NLD- Prim SAS
$n(4) set-position 41.59 13.33; # Fucino, ITL-Sec SAS
$n(3) color red
$n(3) shape box
$n(4) color red
$n(4) shape box

#Create wired terrestrial nodes
$ns unset satNodeType_
set n(5) [$ns node]
set n(6) [$ns node]
set n(7) [$ns node]
set n(8) [$ns node]
set n(9) [$ns node]
set n(10) [$ns node]
set n(11) [$ns node]
set n(12) [$ns node]
set n(13) [$ns node]
set n(14) [$ns node]
set n(15) [$ns node]
set n(16) [$ns node]
set n(17) [$ns node]
set n(18) [$ns node]
set n(19) [$ns node]
set n(20) [$ns node]
set n(21) [$ns node]
set n(22) [$ns node]

#TCP Tx/Rx node color code
$n(7) color green
$n(12) color green

#Nodes Label
$ns at 0.0 "$n(0) label \"Sat\""
$ns at 0.0 "$n(1) label \"SUT1\""
$ns at 0.0 "$n(2) label \"SUT2\""
$ns at 0.0 "$n(7) label \"Tx1\""
$ns at 0.0 "$n(12) label \"Rx1\""
$ns at 0.0 "$n(3) label \"GWSP\""
$ns at 0.0 "$n(4) label \"GWSS\""

#Create links between nodes
$ns duplex-link $n(7) $n(5) 100mb 2.5ms DropTail
$ns duplex-link $n(8) $n(5) 100mb 2.5ms DropTail
$ns duplex-link $n(9) $n(5) 100mb 2.5ms DropTail
$ns duplex-link $n(5) $n(6) 10Mb 7.5ms DropTail #bottleneck link
$ns duplex-link $n(6) $n(1) 100Mb 2.5ms DropTail
$ns duplex-link $n(5) $n(10) 100Mb 2.5ms DropTail
```

```
$ns duplex-link $n(6) $n(15) 100mb 2.5ms DropTail
#$ns duplex-link $n(6) $n(16) 10mb 2.5ms DropTail
#$ns duplex-link $n(6) $n(17) 10mb 2.5ms DropTail
#$ns duplex-link $n(6) $n(18) 10mb 2.5ms DropTail
$ns duplex-link $n(2) $n(12) 100mb 2.5ms DropTail
$ns duplex-link $n(2) $n(13) 100mb 2.5ms DropTail
$ns duplex-link $n(2) $n(14) 100mb 2.5ms DropTail
$ns duplex-link $n(2) $n(11) 100mb 2.5ms DropTail

# Add GSLs to geo satellite and Gateway stations
$n(3) add-gsl geo $opt(ll) $opt(ifq) $opt(qlim) $opt(mac) $opt(bw_up) \
    $opt(phy) [$n(0) set downlink_] [$n(0) set uplink_]
$n(4) add-gsl geo $opt(ll) $opt(ifq) $opt(qlim) $opt(mac) $opt(bw_up) \
    $opt(phy) [$n(0) set downlink_] [$n(0) set uplink_]

# Bottleneck Link color
$ns duplex-link-op $n(5) $n(6) color "red"

#Describe the indices of the nodes and locations.
set indexLayout {
        {8 9}
{7 5 10}
{15 6 17 11}
        {16 1 2 12}
{4 0 3 14 13}
}

#Describe the space and layout
set originX 0
set originY 0
set width   100
set height  100

#Do the layout
set nRows [llength $indexLayout]
set rowsize [expr {$height / $nRows}]
set rowY [expr {$originY + $rowsize / 2}]
foreach row $indexLayout {
    set nCols [llength $row]
    set colsize [expr {$width / $nCols}]
    set rowX [expr {$originX + $colsize / 2}]
    foreach index $row {
        $n($index) set X_ $rowX
        $n($index) set Y_ $rowY
        set rowX [expr {$rowX + $colsize}]
    }
    set rowY [expr {$rowY + $rowsize}]
}

$ns trace-all-satlinks $outfile

#We use centralized routing
set satrouteobject_ [new SatRouteObject]
$satrouteobject_ compute_routes

proc finish {} {
global ns outfile ntf1 winfile
$ns flush-trace
close $outfile
        close $ntf1
close $winfile
#graphs with gnuplot
set plo [open plotta.out w]
puts $plo "set xlabel \"Time(s)\""
puts $plo "set ylabel \"Packets\""
puts $plo "set title \"Title \""
puts $plo "set output \"hbcwnd25100.eps\""
puts $plo "set terminal postscript eps"
puts $plo "plot \"Cwnd25\" with lines"
close $plo
exec gnuplot plotta.out

#exec awk -f avgStats.awk src=7 dst flow=1 pkt=1040 hbc19.tr > hbc19avgd
#exec awk -f instantThroughput.awk tic=1.0 src dst flow pkt cub19.tr > cub19Thp2
#exec awk -f e2eowdrttgw19.awk hbc19.tr > hbc19eedgws2
#exec awk -f e2elat19bisu.awk hbc19.tr > hbc19eedgws
#exec awk -f instantJitter.awk tic=1.0 src dst flow pkt hbc19.tr > hbc19ij
#exec awk -f pdrd19.awk hbc19.tr > hbc19pdrd
#puts "running nam..."

        exec nam hbc125.nam &

exit 0
}

#Insert a 'recv' function for the class 'Agent/Ping'
Agent/Ping instproc recv {from rtt} {
        $self instvar node_
        puts "node [$node_ id] received ping answer from \
              $from with roun-trip-time $rtt ms."
}
```

```
#create 2 ping agents and attach them to n0 and n3
set p0 [new Agent/Ping]
$ns attach-agent $n(9) $p0

set p1 [new Agent/Ping]
$ns attach-agent $n(14) $p1

#connect 2 agents
$ns connect $p0 $p1
$p0 set fid_ 6
$p1 set fid_ 7

$ns at 0.1 "$p0 send"
$ns at 0.5 "$p1 send"
$ns at 1.0 "$p0 send"
$ns at 1.5 "$p1 send"

#---TCP agents (hybic, hybla, cubic and newreno) added below.....
#Set protocols (agents) and application (ftp/cbr) traffic flows
#setup TCP schemes as agents

#WARNING: set "window_" option in tcp to be large enough to see the performance
#"window_" is the upper-bound of cwnd in a TCP, 20 by default.

#hybic
set tcp [new Agent/TCP/Linux]
#$tcp set timestamps_ true
$ns at 0 "$tcp select_ca hybic"
$ns attach-agent $n(7) $tcp
set sink [new Agent/TCPSink/Sack1]
$sink set ts_echo_rfc1323_ true
```

```
$ns attach-agent $n(11) $sink
$ns connect $tcp $sink
#tcp traffic color 1 (blue as above)
$tcp set fid_ 1
#disabled pktSz
#$tcp set packetSize_ 1024

#setup FTP traffic over TCP connection
set ftp [new Application/FTP]
$ftp attach-agent $tcp
#traffic type is ftp
$ftp set type_ FTP

#hybla
#set tcph [new Agent/TCP/Linux]
#$tcp set timestamps_ true
#$ns at 0 "$tcph select_ca hybla"
#$ns attach-agent $n(8) $tcph
#set sinkh [new Agent/TCPSink/Sack1]
#$sinkh set ts_echo_rfc1323_ true
#$ns attach-agent $n(12) $sinkh
#$ns connect $tcph $sinkh

#tcp traffic color 3 (as above)
#$tcph set fid_ 2

#disabled pktSz
#$tcph set packetSize_ 1024



#setup FTP traffic over TCP connection
```

```
#set ftph [new Application/FTP]
#$ftph attach-agent $tcph
#traffic type is ftp
#$ftph set type_ FTP



#cubic
set tcpc [new Agent/TCP/Linux]
#$tcp set timestamps_ true
$tcpc set window_ 100000
$ns at 0 "$tcpc select_ca cubic"
$ns attach-agent $n(9) $tcpc
set sinkc [new Agent/TCPSink/Sack1]
$sinkc set ts_echo_rfc1323_ true
$ns attach-agent $n(13) $sinkc
$ns connect $tcpc $sinkc

#tcp traffic color as above
$tcpc set fid_ 3

#disabled pktSz
#$tcpc set packetSize_ 1024

#setup FTP traffic over TCP connection
set ftpc [new Application/FTP]
$ftpc attach-agent $tcpc
#traffic type is ftp
$ftpc set type_ FTP
```

```
#Newreno

#set tcpn [new Agent/TCP/Newreno]

#$tcpn set timestamps_ true

#$tcpn set window_ 100000

#$ns attach-agent $n(9) $tcpn

#set sinkn [new Agent/TCPSink/DelAck]

#$ns attach-agent $n(13) $sinkn

#$ns connect $tcpn $sinkn


#tcp traffic color 1 (red as above)

#$tcpn set fid_ 4

#$tcpn set window_ 100000

#$tcpn set packetSize_ 1024

#$tcpn set ssthres_ 1


#set FTP traffic over TCP connection

#set ftpn [new Application/FTP]

#$ftpn attach-agent $tcpn

#traffic type is ftp

#$ftpn set type_ FTP


##..........end of tcp schemes and start of udp agent..........


#Set UDP connection

set udp [new Agent/UDP]

$ns attach-agent $n(10) $udp

#sink for udp source traffic

set null [new Agent/Null]

$ns attach-agent $n(14) $null

$ns connect $udp $null

$udp set fid_ 5
```

```
#Set CBR traffic over UDP connection
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR     #cbr traffic type
$cbr set packet_size_ 1000
$cbr set rate_ 0.01mb
$cbr set random_ false  #traffic generation parameter

##.................end of the protocol declarations.............



#protocols start
$ns at 1.0 "$ftp start"
#$ns at 1.0 "$ftph start"
$ns at 1.0 "$ftpc start"
#$ns at 1.0 "$ftpn start"
$ns at 1.1 "$cbr start"
#protocols stop
$ns at 348.0 "$ftp stop"
#$ns at 348.0 "$ftph stop"
$ns at 348.0 "$ftpc stop"
#$ns at 348.0 "$ftpn stop"
$ns at 348.5 "$cbr stop"


#Window plot procedure
proc plotwindow {tcpSource file} {
global ns
set time 1.0
set now [$ns now]
set cwnd [$tcpSource set cwnd_]
set wnd [$tcpSource set window_]
puts $file "$now $cwnd"
$ns at [expr $now + $time] "plotwindow $tcpSource $file" }
$ns at 1.0 "plotwindow $tcp $winfile"

#Stop Simulation at Time 349 sec
$ns at 349.0 "finish"
#--------------simulation ready to run.........
$ns run
```

A.3 C Codes

```
//Parameters define for cubic implementation
//Scale factor beta calculation max_cwnd = snd_cwnd * beta
#define BICTCP_BETA_SCALE 1024
//In binary search,* go to point (max+min)/N
#define BICTCP_B 4
//BIC HZ 2^10 = 1024
#define BICTCP_HZ 10
#define ACK_RATIO_SHIFT 4

/*Define parameters for the algorithm; parameters have to be defined static
different modules might have different parameters with the same variable names *
static int fast_convergence __read_mostly = 1;
static int max_increment __read_mostly = 16;
static int beta __read_mostly = 819; /* = 819/1024 (BICTCP_BETA_SCALE) */
static int initial_ssthresh __read_mostly;
static int bic_scale __read_mostly = 41;
static int tcp_friendliness __read_mostly = 1;

//Parameters defined for precompute scale factors,not declared like previous
static u32 cube_rtt_scale __read_mostly;
static u32 beta_scale __read_mostly;
static u64 cube_factor __read_mostly;

/* Declare the following as parameters using module_param() */
/* Declare the explanation for parameters using MODULE_PARM_DESC()*/
//Note parameters that are used for precomputing scale factors are read-only
module_param(fast_convergence, int, 0644);
MODULE_PARM_DESC(fast_convergence, "turn on/off fast convergence");
module_param(max_increment, int, 0644);
MODULE_PARM_DESC(max_increment, "Lim increment allowed during binary search");
module_param(beta, int, 0444);
MODULE_PARM_DESC(beta, "beta for multiplicative increase");
module_param(initial_ssthresh, int, 0644);
MODULE_PARM_DESC(initial_ssthresh, "initial value of slow start threshold");
module_param(bic_scale, int, 0444);
MODULE_PARM_DESC(bic_scale,"scale(by1024) value for bic func(bic_scale/1024)");
module_param(tcp_friendliness, int, 0644);
MODULE_PARM_DESC(tcp_friendliness, "turn on/off tcp friendliness");



/* Hybla reference round trip time (default= 1/40 sec = 25 ms),
   expressed in jiffies, changed: 25ms to 125 */
static int rtt0 = 25;
module_param(rtt0, int, 0644);
MODULE_PARM_DESC(rtt0, "reference rout trip time (ms)");

//global parameters defined
```

```
/* time when updated last_cwnd */
static u32 last_time;
//static u8 hybla_en;

/* Tcp Hybic structure/parameters*/
/* HYBIC Parameters as a struct hybic used *ca pointers */
struct hybic {
u32   snd_cwnd_cents; /* Hybla Keeps increment values when it is <1, <<7 */
u32   rho;       /* Rho parameter, integer part  */
u32   rho2;       /* Rho * Rho, integer part */
u32   rho_3ls;       /* Rho parameter, <<3 */
u32   rho2_7ls;       /* Rho^2, <<7 */
u32   minrtt;       /* Minimum smoothed round trip time value seen */
u32 cnt; /*  Cubic increase cwnd by 1 after ACKs */
u32  last_max_cwnd; /* last maximum snd_cwnd */
u32 loss_cwnd; /* congestion window at last loss */
u32 last_cwnd; /* the last snd_cwnd */
u32 bic_origin_point;/* origin point of bic function */
u32 bic_K; /* time to origin point from the beginning of the current epoch */
u32 delay_min; /* min delay */
u32 epoch_start; /* beginning of an epoch */
u32 ack_cnt; /* number of acks */
u32 tcp_cwnd; /* estimated tcp cwnd */
u32 delayed_ack; /* estimate the ratio of Packets/ACKs << 4 */
};

static inline void hybla_recalc_param (struct sock *sk)
{
struct hybic *ca = inet_csk_ca(sk); //change from hybla *ca to hubic *ca
ca->rho_3ls = max_t(u32, tcp_sk(sk)->srtt / msecs_to_jiffies(rtt0), 8);
ca->rho = ca->rho_3ls >> 3;
```

```
ca->rho2_7ls = (ca->rho_3ls * ca->rho_3ls) << 1;
ca->rho2 = ca->rho2_7ls >>7;
}
static inline void bictcp_reset(struct hybic *ca)
{
ca->cnt = 0;
ca->last_max_cwnd = 0;
ca->loss_cwnd = 0;
ca->last_cwnd = 0;
last_time = 0;
ca->bic_origin_point = 0;
ca->bic_K = 0;
ca->delay_min = 0;
ca->epoch_start = 0;
ca->delayed_ack = 2 << ACK_RATIO_SHIFT;
ca->ack_cnt = 0;
ca->tcp_cwnd = 0;
}
/*CA has private data, should initialises its private date here.*/
/*hybic parameters/variables initialisation */
static void hybic_init(struct sock *sk)
{
struct tcp_sock *tp = tcp_sk(sk);
struct hybic *ca = inet_csk_ca(sk);
/* hybla param reset*/
ca->rho = 0;
ca->rho2 = 0;
ca->rho_3ls = 0;
ca->rho2_7ls = 0;
ca->snd_cwnd_cents = 0;
//hybla_en = 1;
```

```
tp->snd_cwnd = 2;
tp->snd_cwnd_clamp = 65535;

/* 1st rho measurement based on initial srtt */
hybla_recalc_param(sk);

/* set minimum rtt as this is the 1st ever seen */
ca->minrtt = tp->srtt;
tp->snd_cwnd = ca->rho;

/*merger with static void bictcp_init(struct sock *sk) */
bictcp_reset(inet_csk_ca(sk));
if (initial_ssthresh)
tcp_sk(sk)->snd_ssthresh = initial_ssthresh;
}
static void hybic_state(struct sock *sk, u8 ca_state)
{
if (ca_state == TCP_CA_Loss)
bictcp_reset(inet_csk_ca(sk));
}

static inline u32 hybla_fraction(u32 odds)
{
static const u32 fractions[] = {
128, 139, 152, 165, 181, 197, 215, 234,
};

return (odds < ARRAY_SIZE(fractions)) ? fractions[odds] : 128;
}

static u32 cubic_root(u64 a)
```

```
{
u32 x, b, shift;
/*
 * cbrt(x) MSB values for x MSB values in [0..63].
 * Precomputed then refined by hand - Willy Tarreau
 *
 * For x in [0..63],
 *   v = cbrt(x << 18) - 1
 *   cbrt(x) = (v[x] + 10) >> 6
 */
static const u8 v[] = {
/* 0x00 */    0,   54,   54,   54,  118,  118,  118,  118,
/* 0x08 */  123,  129,  134,  138,  143,  147,  151,  156,
/* 0x10 */  157,  161,  164,  168,  170,  173,  176,  179,
/* 0x18 */  181,  185,  187,  190,  192,  194,  197,  199,
/* 0x20 */  200,  202,  204,  206,  209,  211,  213,  215,
/* 0x28 */  217,  219,  221,  222,  224,  225,  227,  229,
/* 0x30 */  231,  232,  234,  236,  237,  239,  240,  242,
/* 0x38 */  244,  245,  246,  248,  250,  251,  252,  254,
};

b = fls64(a);
if (b < 7) {
/* a in [0..63] */
return ((u32)v[(u32)a] + 35) >> 6;
}

b = ((b * 84) >> 8) - 1;
shift = (a >> (b * 3));

x = ((u32)(((u32)v[shift] + 10) << b)) >> 6;
```

```
 * Newton-Raphson iteration
 *                         2
 * x    = ( 2 * x  +  a / x  ) / 3
 *   k+1            k         k
 */
x = (2 * x + (u32)div64_64(a, (u64)x * (u64)(x - 1)));
x = ((x * 341) >> 10);
return x;
}

//...........Cubic parameters calculations routine..............
 //Compute congestion window to use.

static inline void bictcp_update(struct hybic *ca, u32 cwnd)
{
u64 offs;
u32 delta, t, bic_target, min_cnt, max_cnt;

ca->ack_cnt++;   /* count the number of ACKs */

if (ca->last_cwnd == cwnd &&
    (s32)(tcp_time_stamp - last_time) <= HZ / 32)
return;

ca->last_cwnd = cwnd;
last_time = tcp_time_stamp;

if (ca->epoch_start == 0) {
ca->epoch_start = tcp_time_stamp; /* record beginning of an epoch */
ca->ack_cnt = 1; /* start counting */
ca->tcp_cwnd = cwnd;        /* syn with cubic */
if (ca->last_max_cwnd <= cwnd) {
ca->bic_K = 0;
ca->bic_origin_point = cwnd;
} else {
// Compute new K based on (wmax-cwnd) * (srtt>>3 / HZ) / c * 2^(3*bictcp_HZ)
ca->bic_K = cubic_root(cube_factor * (ca->last_max_cwnd - cwnd));
ca->bic_origin_point = ca->last_max_cwnd;
}
}

/* cubic function - calc*/
/* calculate c * time^3 / rtt,
 *  while considering overflow in calculation of time^3
 * (so time^3 is done by using 64 bit)
 * and without the support of division of 64bit numbers
 * (so all divisions are done by using 32 bit)
 *  also NOTE the unit of those veriables
 *   time  = (t - K) / 2^bictcp_HZ
 *   c = bic_scale >> 10
 * rtt  = (srtt >> 3) / HZ
 * !!! The following code does not have overflow problems,
 * if the cwnd < 1 million packets !!!*/

/* change the unit from HZ to bictcp_HZ */
t = ((tcp_time_stamp + (ca->delay_min>>3) - ca->epoch_start)
     << BICTCP_HZ) / HZ;

if (t < ca->bic_K) /* t - K */
offs = ca->bic_K - t;
else
```

```
offs = t - ca->bic_K;

/* c/rtt * (t-K)^3 */
delta = (cube_rtt_scale * offs * offs * offs) >> (10+3*BICTCP_HZ);
if (t < ca->bic_K)                                /* below origin*/
bic_target = ca->bic_origin_point - delta;
else                                                /* above origin*/
bic_target = ca->bic_origin_point + delta;

/* cubic function - calc bictcp_cnt*/
if (bic_target > cwnd) {
ca->cnt = cwnd / (bic_target - cwnd);
} else {
ca->cnt = 100 * cwnd;             /* very small increment*/
}

if (ca->delay_min > 0) {
/* max increment = Smax * rtt / 0.1  */
min_cnt = (cwnd * HZ * 8)/(10 * max_increment * ca->delay_min);

/* use concave growth when the target is above the origin */
if (ca->cnt < min_cnt && t >= ca->bic_K)
ca->cnt = min_cnt;
}

/* slow start and low utilization  */
if (ca->loss_cwnd == 0) /* could be aggressive in slow start */
ca->cnt = 50;

/* TCP Friendly */
if (tcp_friendliness) {
```

```
u32 scale = beta_scale;
delta = (cwnd * scale) >> 3;
while (ca->ack_cnt > delta) {/* update tcp cwnd */
ca->ack_cnt -= delta;
ca->tcp_cwnd++;
}

if (ca->tcp_cwnd > cwnd){ /* if bic is slower than tcp */
delta = ca->tcp_cwnd - cwnd;
max_cnt = cwnd / delta;
if (ca->cnt > max_cnt)
ca->cnt = max_cnt;
}
}
ca->cnt = (ca->cnt << ACK_RATIO_SHIFT) / ca->delayed_ack;
if (ca->cnt == 0) /* cannot be zero */
ca->cnt = 1;
}



/*Keep track of minimum rtt */
static inline void measure_delay(struct sock *sk)
{
const struct tcp_sock *tp = tcp_sk(sk);
struct hybic *ca = inet_csk_ca(sk);
u32 delay;

/* No time stamp */
if (!(tp->rx_opt.saw_tstamp && tp->rx_opt.rcv_tsecr) ||
     /* Discard delay samples right after fast recovery */
    (s32)(tcp_time_stamp - ca->epoch_start) < HZ)
```

```
return;

delay = (tcp_time_stamp - tp->rx_opt.rcv_tsecr)<<3;
if (delay == 0)
delay = 1;

/* first time call or link delay decreases */
if (ca->delay_min == 0 || ca->delay_min > delay)
ca->delay_min = delay;
}
/*.....................Hybic CCA........................*/


/* TCP Hybic main routine.
 * This is the algorithm behavior:
 *     o Recalc Hybla parameters if min_rtt has changed
 *     o Give cwnd a new value
the function is the main CCA that increase cwnd for each ack*/

static void hybic_cong_avoid(struct sock *sk, u32 ack, u32 rtt,
    u32 in_flight, int flag)
{
struct tcp_sock *tp = tcp_sk(sk);
struct hybic *ca = inet_csk_ca(sk);
u32 increment, odd, rho_fractions;
int is_slowstart = 0;

if (!tcp_is_cwnd_limited(sk, in_flight))
return;

if (tp->snd_cwnd < tp->snd_ssthresh) {
```

```
/*  Recalculate rho only if this srtt is the lowest */
if (tp->srtt < ca->minrtt){
hybla_recalc_param(sk);
ca->minrtt = tp->srtt;
}

if (ca->rho == 0)
hybla_recalc_param(sk);
rho_fractions = ca->rho_3ls - (ca->rho << 3);

 /* slow start
 *      INC = 2^RHO - 1
 * This is done by splitting the rho parameter
 * into 2 parts: an integer part and a fraction part.
 * Inrement<<7 is estimated by doing:
 *        [2^(int+fract)]<<7
 * that is equal to:
 *        (2^int) *  [(2^fract) <<7]
 * 2^int is straightly computed as 1<<int,
 * while we will use hybla_slowstart_fraction_increment() to
 * calculate 2^fract in a <<7 value.*/

is_slowstart = 1;
increment = ((1 << ca->rho) * hybla_fraction(rho_fractions))
- 128;

odd = increment % 128;
tp->snd_cwnd += increment >> 7;
ca->snd_cwnd_cents += odd;
```

```
/* check when fractions goes >=128 and increase cwnd by 1. */
while (ca->snd_cwnd_cents >= 128) {
tp->snd_cwnd++;
ca->snd_cwnd_cents -= 128;
tp->snd_cwnd_cnt = 0;
}

/* clamp down slowstart cwnd to ssthresh value. */
if (is_slowstart)
tp->snd_cwnd = min(tp->snd_cwnd, tp->snd_ssthresh);

tp->snd_cwnd = min_t(u32, tp->snd_cwnd, tp->snd_cwnd_clamp);

} else {

/*cubic congestion avoidance*/
ack = ack;
if (flag)
measure_delay(sk);
        bictcp_update(ca, tp->snd_cwnd);

/* In dangerous area, increase slowly, In theory tp->snd_cwnd += 1 / tp->snd_cw
if (tp->snd_cwnd_cnt >= ca->cnt) {
if (tp->snd_cwnd < tp->snd_cwnd_clamp)
tp->snd_cwnd++;
tp->snd_cwnd_cnt = 0;
} else
tp->snd_cwnd_cnt++;
}
}
```

```
/*.................Other cubic routines...........*/



/* .ssthresh function returns the slow-start threshold after a loss.*/
static u32 bictcp_recalc_ssthresh(struct sock *sk)
{

const struct tcp_sock *tp = tcp_sk(sk);
struct hybic *ca = inet_csk_ca(sk);

ca->epoch_start = 0; /* end of epoch */

/* Wmax and fast convergence */
if (tp->snd_cwnd < ca->last_max_cwnd && fast_convergence)
ca->last_max_cwnd = (tp->snd_cwnd * (BICTCP_BETA_SCALE + beta))
/ (2 * BICTCP_BETA_SCALE);
else
ca->last_max_cwnd = tp->snd_cwnd;

ca->loss_cwnd = tp->snd_cwnd;

return max((tp->snd_cwnd * beta) / BICTCP_BETA_SCALE, 2U);
}

static u32 bictcp_undo_cwnd(struct sock *sk)
{
struct hybic *ca = inet_csk_ca(sk);

return max(tcp_sk(sk)->snd_cwnd, ca->last_max_cwnd);
}
```

```
/* Track delayed ack ratio using sliding window
 * ratio = (15*ratio + sample) / 16
 */

/*function is called when there is an ack that acknowledges new packets.
num_acked is the number of packets that are acked by this acks.*/
static void bictcp_acked(struct sock *sk, u32 cnt, ktime_t last)
{
last = last;

const struct inet_connection_sock *icsk = inet_csk(sk);

if (cnt > 0 && icsk->icsk_ca_state == TCP_CA_Open) {
struct hybic *ca = inet_csk_ca(sk);
cnt -= ca->delayed_ack >> ACK_RATIO_SHIFT;
ca->delayed_ack += cnt;

}
}
/*................End of hybic CCA.........................*/



/*constant record for the hybic CCA
as static record of struct tcp_congestion_ops
to store the function calls and algorithm's name
Implement at least cong_avoid, ssthresh & min_cwnd;*/

static struct tcp_congestion_ops tcp_hybic = {
.init = hybic_init,
.ssthresh = bictcp_recalc_ssthresh,
.cong_avoid = hybic_cong_avoid,
.set_state = hybic_state,
.undo_cwnd = bictcp_undo_cwnd,
.pkts_acked     = bictcp_acked,
.owner = THIS_MODULE,
.name = "hybic"
};

static int __init hybic_register(void)
{
BUILD_BUG_ON(sizeof(struct hybic) > ICSK_CA_PRIV_SIZE);
/* Precompute a bunch of the scaling factors that are used
 per-packet based on SRTT of 100ms*/

beta_scale = 8*(BICTCP_BETA_SCALE+beta)/ 3 / (BICTCP_BETA_SCALE - beta);

cube_rtt_scale = (bic_scale * 10); /* 1024*c/rtt */

/* calculate the "K" for (wmax-cwnd) = c/rtt * K^3
*  so K = cubic_root( (wmax-cwnd)*rtt/c )
* the unit of K is bictcp_HZ=2^10, not HZ
*  c = bic_scale >> 10
*  rtt = 100ms
* the following code has been designed and tested for
* cwnd < 1 million packets
* RTT < 100 seconds
* HZ < 1,000,00  (corresponding to 10 nano-second) */
/* 1/c * 2^2*bictcp_HZ * srtt */
cube_factor = 1ull << (10+3*BICTCP_HZ); /* 2^40 */
/* divide by bic_scale and by constant Srtt (100ms) */

do_div(cube_factor, bic_scale * 10);
```

```
return tcp_register_congestion_control(&tcp_hybic);
}
static void __exit hybic_unregister(void)
{
tcp_unregister_congestion_control(&tcp_hybic);
}
module_init(hybic_register);
module_exit(hybic_unregister);

MODULE_AUTHOR("Anas A. Bisu");
MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("TCP Hybic");
MODULE_VERSION("7.2");
#undef NS_PROTOCOL
```

Appendix B

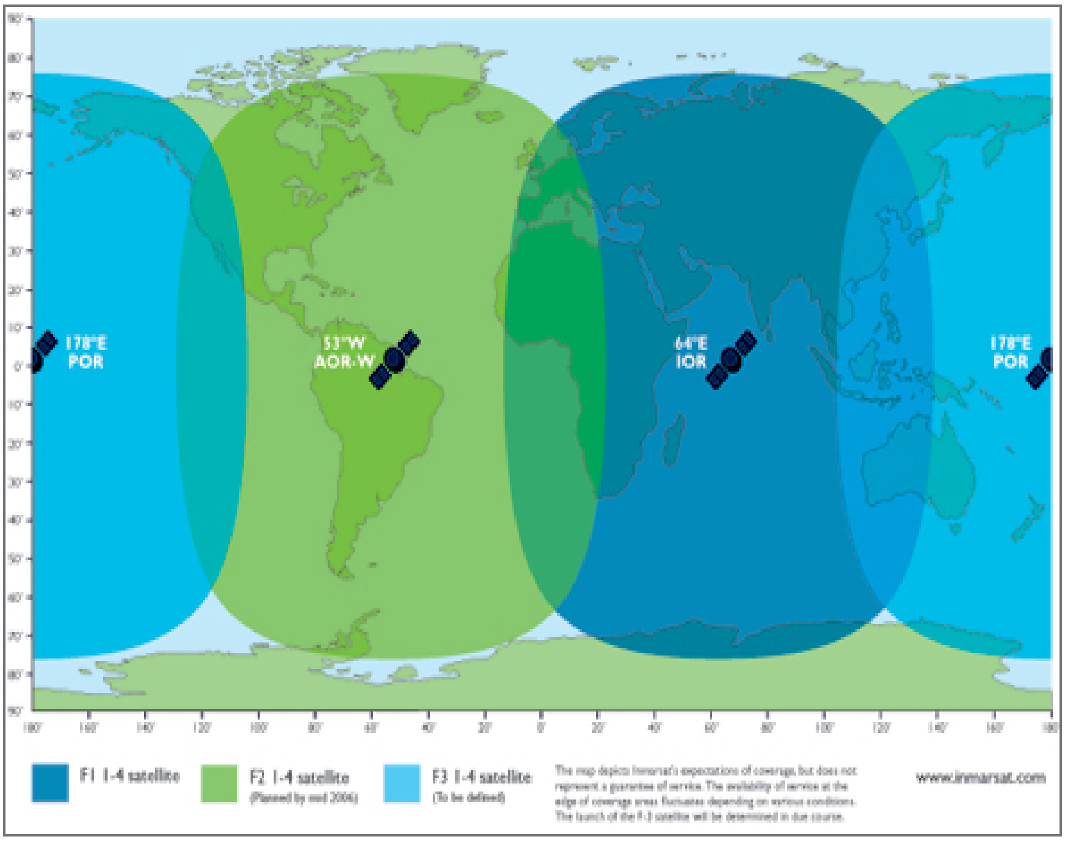
Hardware: Configurations and Data Sheets

B.1 Satellite Terminals

Two satellite network and service providers were used in the testbed designed for the practical measurements in this thesis. The two providers are Inmarsat and Thuraya both operating GEO Satellite position at different coordinates in space and maintained different ground segments (GWS) in different continents with Inmarsat in Europe and Thuraya in the middle. However, both network providers have footprint covering Africa and Europe, which are the locations of our practical measurement experiments detailed in this thesis.

B.1.1 Inmarsat Terminal: BGAN Explorer 510

The Explorer 510 is a compact satellite user terminal (SUT) with transceiver and antenna as unit that connect to any of the Inmarsat BGAN satellites constellations as shown in Fig. B.1. Explorer 510 is the smallest of the EXPLORER BGAN terminals that integrate performance and portability with simultaneous high quality voice and broadband access services.



The map depicts Inmarsat's expectations of coverage, but does not represent a guarantee of service. The availability of service at the edge of coverage areas fluctuates depending on various conditions. The launch of the F-3 satellite will be determined in due course.

Figure B.1: BGAN Satellite Coverage Map

This is perfect choice for ruggedness, durability, and reliable connection anywhere at anytime (see Fig. B.2) with durable magnesium casing, dust and water resistance design.

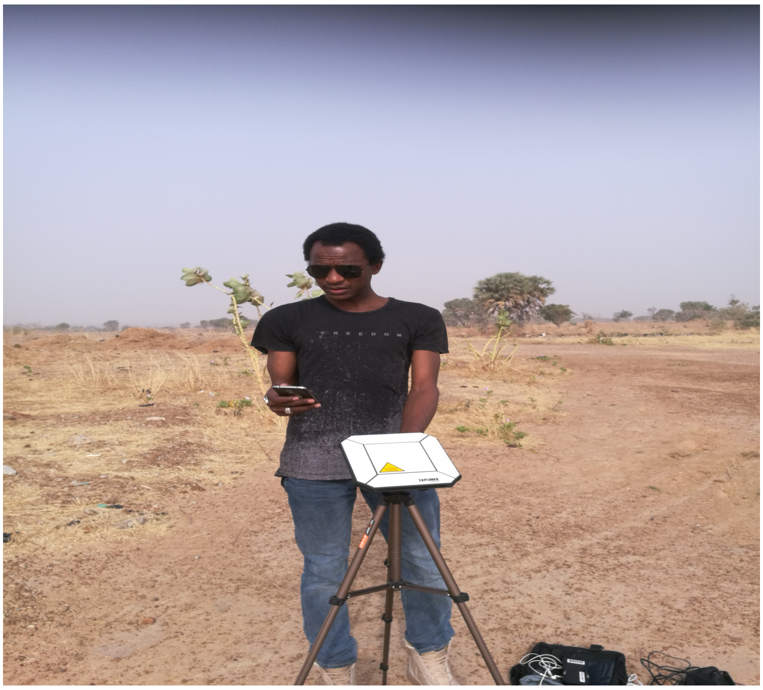


Figure B.2: Explorer 510 Pointing to BGAN Satellite at a Remote Location in Nigeria

The end user equipment (UE) such as Tablet, Smartphone, or Computer can connect to the EXPLORER 510 SUT using wired or wireless LAN interface through the *EXPLORER connect App* (available for iOS and Android devices) or *Web Interface* (for Laptops and PCs) using the terminal IP address. Features and technical specifications of the EXPLORER 510 are summarised in Table B.1, details on configuration and BGAN satellite pointing can be found in BGAN User Manual [1].

Figure B.3 showed the rear and side views of the EXPLORER 510 indicating the location of available buttons, antennas and communications interfaces.

Table B.1: EXPLORER 510 Features and Technical Specifications.

Feature	Specification
Model Class	EXPLORER 510, BGAN Class 2 Terminal
Standard IP Data Rate Tx/Rx	464 kbps both Up and Down Links
Streaming Data Rates (kbps)	32, 64, and 128
Voice Rate	4 kbps (Standard), 3.1kHz/64 kbps (Premium)
Dimensions (L x H x B mm)	197x197x40
Weight (kg)	1.4 Incl. Battery
Operating Temp. (^oC)	-25 to 55
Water and Dust Resistance	IP66
Interfaces	WLAN, USB, DC Input, USB to LAN, and SIM card
User Connectivity/Interface	App (iOS and Android) and Web Browser
AC Power (VAC)	100-240 V Mains via AC/DC Adapter
DC Power (VDC)	10-32 V
Solar Power (Panel)	Min. 65 W, 10-32 VDC
Antenna	GNSS (GPS, GLONASS, BelDou), WLAN, BGAN
Wireless Router	DHCP and NAT
Battery	Rechargeable Lith. Ion, 300-500 CC, 2-24hrs usage,
Power Consumption	2.8W (standby), 19W(Tx), 38W (charging)
Communications	Full duplex, Single/Multi-user, PPPoE, PBX/SIP Voice Server

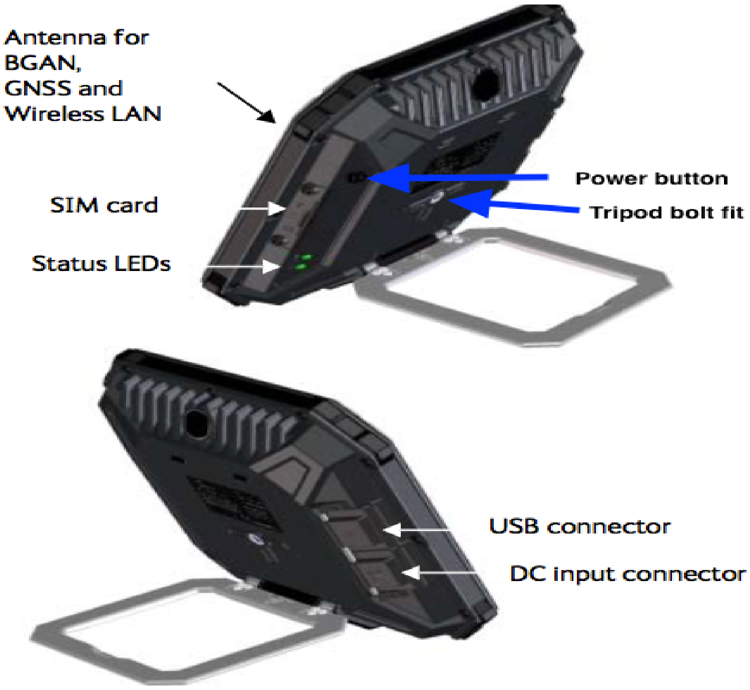


Figure B.3: Explorer 510 Rear and Side View

EXPLORER 510 Terminal User Interfaces

The web interface is a built-in web interface for easy configuration and daily use. The web interface is accessed from a computer, smartphone or tablet connected to the EXPLORER 510, using an Internet browser. No installation of software is needed. For further information on the web interface, see The web interface [1]. Moreover, a smartphone app, *EXPLORER Connect*, is also available for iOS and Android devices. This includes a satellite phone function that enables making and receiving calls with a smartphone over the satellite network using the EXPLORER 510 terminal. It also includes the complete feature set from the built-in web interface of the terminal, allowing you to set up and use the terminal with your smartphone.

B.1.2 Thuraya SatSleeve+ Terminal

The Thuraya SatSleeve+ SUT transforms a smartphone into a satellite phone through connectivity to the Thuraya 2 and 3 GEO satellites with footprint covering Europe, Africa, Asia, and Australia as shown in Fig. B.4. This is an excellent choice that provides portable and fastest way for mobile communications via satellites anywhere at anytime with immediate need for voice and data communications via smartphone transformed into a satellite phone as shown in Fig. B.5.

The terminal configurations is easy and fast using a user Interface Hotspot App for iOS and Android devices through the WLAN interfacing the UE and the SUT as summarised below, more details found in [2].

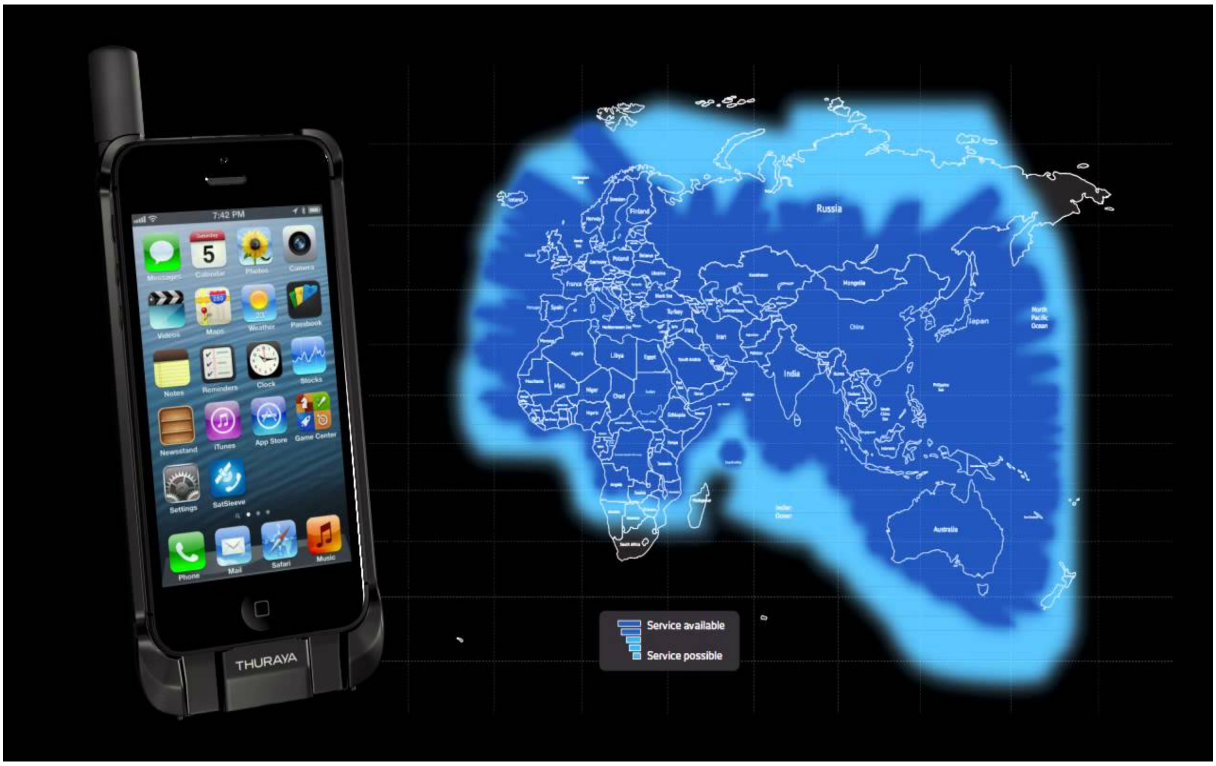


Figure B.4: Thuraya SatSleeve+ Satellite 2 & 3 Coverage Map

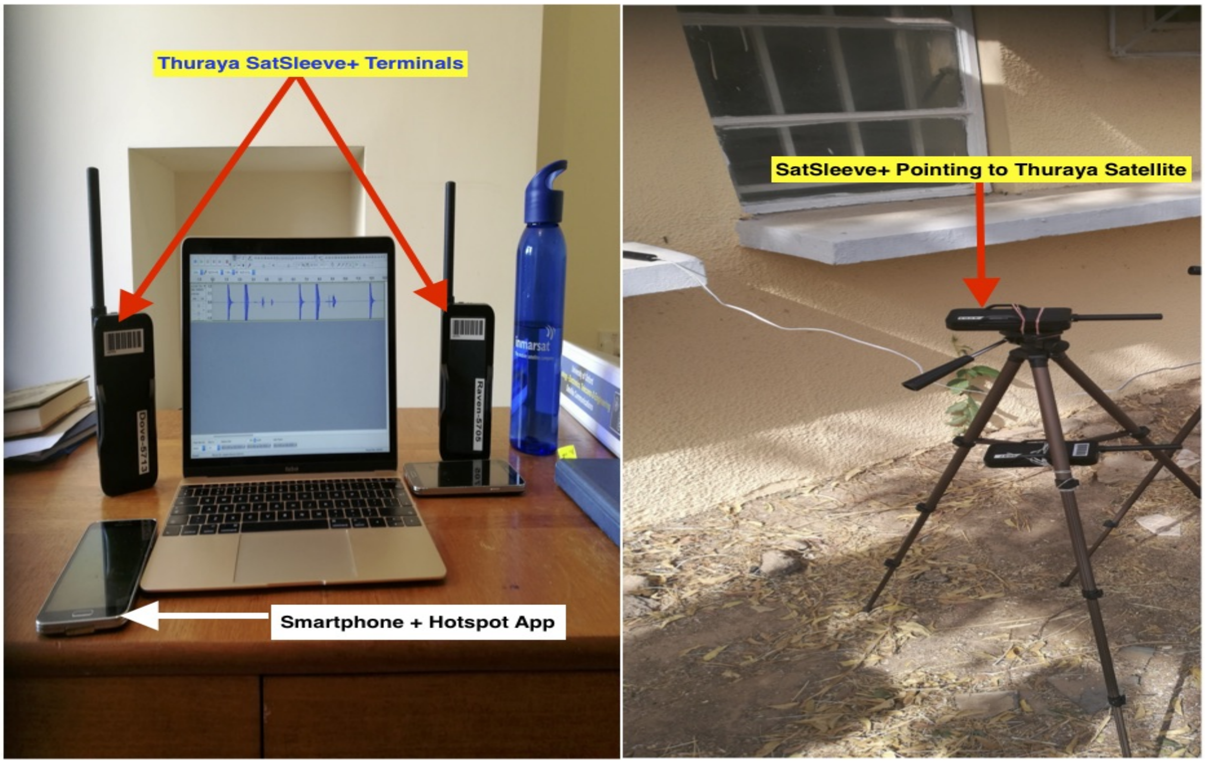

Thuraya SatSleeve+ Terminals
Dove-5713
Raven-5705
Smartphone + Hotspot App
SatSleeve+ Pointing to Thuraya Satellite

Figure B.5: Thuraya SatSleeve+ Terminals and SmartPhones

Table B.2: Thuraya SatSleeve+ Features and Technical Specifications.

Feature	**Specification**
Model Class	SatSleeve+
Data Rate Tx/Rx	60 kbps Download (DL), 15 kbps Upload (UL) GmPRS
Weight (g)	256 Incl. Battery
Dimensions (mm)	138x69x42
Operating Temp. (^oC)	-10 to 55
Interfaces	WLAN, Micro USB (charge/upgrade), DC Input, 3.5mm Jack (headset) and SIM card
User Connectivity/Interface	Hotspot App for iOS and Android
AC Power (VAC)	100-240 V Mains via AC/DC Adapter
DC Power	5VDC, 2.0A
Battery	Rechargeable Li-ion, 3.7V, 2440mAh, 3-9hrs usage
Communications	Full duplex

SatSleeve+ Configuration and Setup

Configurations of SatSleeve+ SUT and Connecting the terminal to the Smartphone (as Satellite phone) and Thuraya Satellite can be achieved quickly and easily by the following steps;

1. Go to the App Store (iOS) or Google Play (Android) on your Smartphone and download the Thuraya SatSleeve+ Hotspot app.
2. Fully Charge the Thuraya SatSleeve+ SUT
3. Insert the SIM card into the SatSleeve+ terminal and turn it on.
4. Go to Wi-Fi settings on your smartphone and connect to the SatSleeve+ unit named SATxxxxxxx.
5. Enter a default password of 12345678 on prompt, this can be changed in the SatSleeve+ Wi-Fi app settings.
6. Go to an area with an unobstructed view of the sky and direct line of sight to the satellite

7. Extend the SatSleeve+ unit antenna fully.

8. Open the SatSleeve+ Hotspot app on your smartphone.

9. The name ”Thuraya” appears on the app home screen.

10. Smartphone now Satphone is connected to the Thuraya satellite network and ready for communications.

B.2 Emulator and Profiler Appliance

Emulation and Profiling experiments in this thesis were conducted using the NE-ONE Appliance, which is a unique piece of equipment offering two powerful and complementary network and application performance capabilities. The NE-ONE appliance combines the network emulation (virtualisation) and Profiling functions on the same appliance unit. These functions can be carried out by the emulator unit or connected with the other equipment such as SUTs and Switches for profiling as highlighted in chapter 4 of this thesis, more details below.

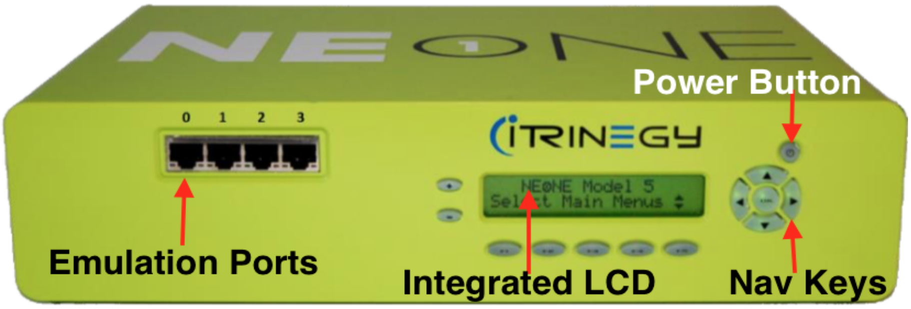


Figure B.6: NE-ONE Emulator and Profiler Appliance Unit

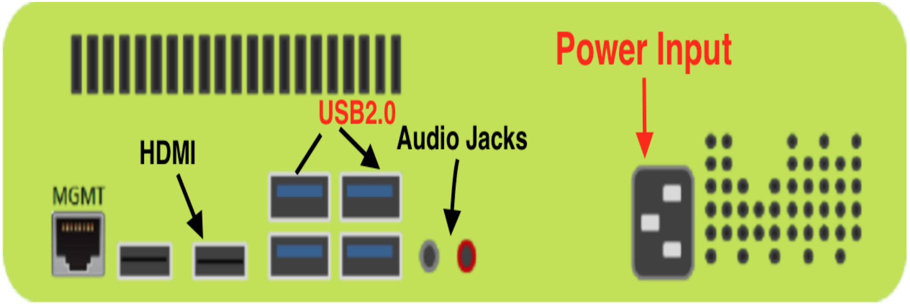


Figure B.7: Rear View NE-ONE Emulator and Profiler Appliance Unit

B.2.1 Emulation

The Emulator supports three types of emulation namely 1) Point to point with Single or multiple link configurations 2) Dual-Hop with Dual or multiple link configurations 3) Profiled Emulation created with the Profiler. These emulations are achieved using the two pairs of the four emulation ports on the front of the NE-ONE appliance, more details on emulation and scenario building found in [3].

B.2.2 Profiling

The Profiling experiment were carried out using the Profiler component of the NE-ONE Profiler appliance, which performs sophisticated network traffic analysis. Profiler performs full-stream reassembly and full content analysis of network traffic to extract, analyse and store valuable application and network performance metrics such as throughput and latency. Two of the front ports (0 & 1) are used and the connection for profiling are were made using a network tap (from SUT or LAN) and smart mirror switch that access network traffic data as copies of the traffic traversing the monitored network link(s) as shown in Fig B.8 [4].

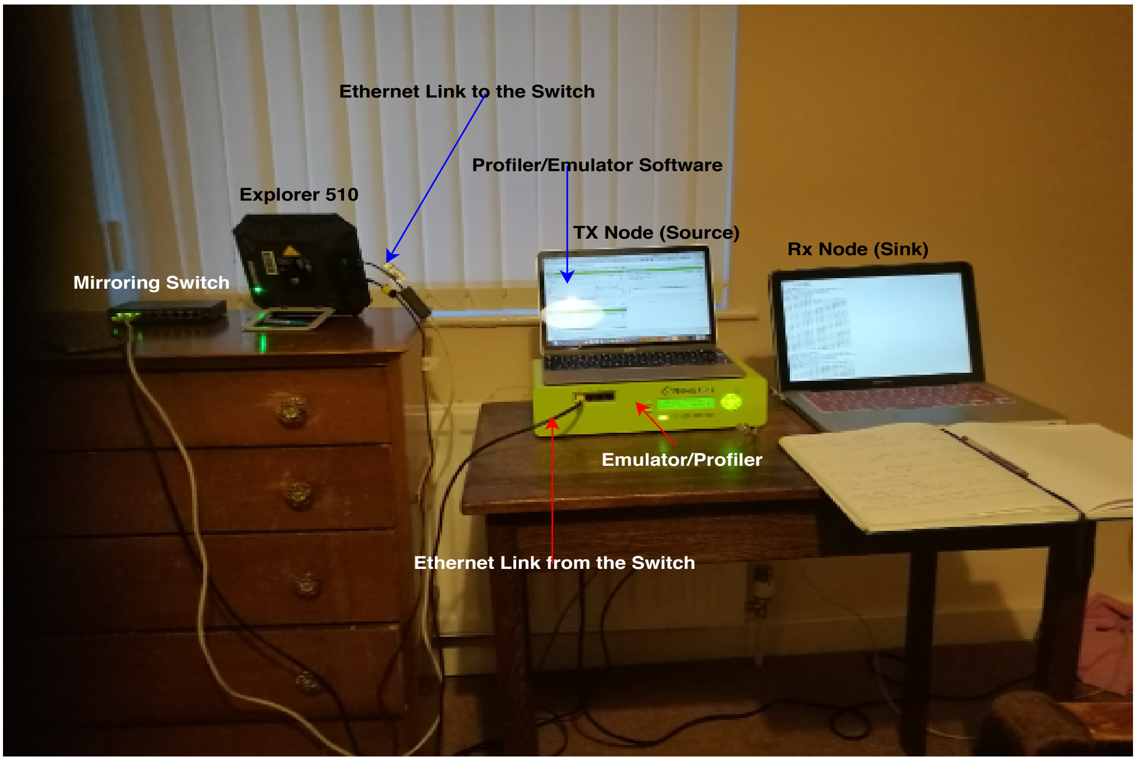

Ethernet Link to the Switch
Profiler/Emulator Software
Explorer 510
TX Node (Source)
Rx Node (Sink)
Mirroring Switch
Emulator/Profiler
Ethernet Link from the Switch

Figure B.8: Testbed for Emulation and Profiling Experiments

Appliance ports 2 & 3 are NOT used when running Profiler mode, these are for Network Emulation mode use. Although the appliance combines the two functions of emulation and profiling in one equipment unit, the Emulator and Profiler cannot be run simultaneously as they utilise the same Appliance resources. Therefore, you must switch between Emulation and Profiling modes based on the experiment requirements [3].

The NE-ONE Appliance has an LCD panel on the front of the unit which provides all the day to day configuration and management of the Appliance in both Emulator and Profiler modes and the LCD panel allows you to switch between Emulator and Profiler modes. More details found in [3, 4], and summary of features and technical specification is given in Table below.

Table B.3: NE-ONE Emulator Features and Technical Specifications [5, 6]

Feature	**Specification**
Platform Model Class	NE-ONE Model 10, Desktop Hardware Appliance
Data Rate Tx/Rx	200 Mbps, 1Gbps (MGMT) Ports
Licensed Emulation Ports	2
Weight (kg)	5 Excl. case
Network Impairment Creation	Latency, Jitter, Loss, BER, Dup/Out-of-Order Pkts, Fragment
Dimensions (HxWxD)	77x204x324 mm
Operating Temp. (^oC)	-5-35, -40-60 (no operation)
Built-in Database	≥100 predefined networks, properties, types & conditions
Interfaces	USB, HDMI, Display, AC Input, Ethernet, Audio Jack
User Connectivity/Interface	Multi-User Web Browser GUI, Control Panel & LCD Setup
AC Power Input	100-230 V, 50/60 Hz, 2-4A, 180W
Link Type Support	P2P, Dual-Hop (Single or multiple link), Multiple Link
Communication Link Support	LAN, WLAN, ADSL, 2G, 3G, 4G/LTE, 5G, Sat, Custom
Maximum Link Support	10
Network Scenario Builder	Parameters change; Gradual, Variable, Outage (increase loss & outage)
Operating Rel. Humidity (%)	8-90

References

[1] Thrane & Thrane, ”COBHAM EXPLORER 510 User MAnual”, Doc. No. 98-143082-A, Thrane & Thrane A/S , 2014 accessed on 25/09/2019 via https://www.groundcontrol.com/bgan/Explorer_510_BGAN_User_Manual.pdf

[2] Thuraya, ”Thuraya SatSleeve+ Brochure (Eng)” accessed on 25/09/2018 via https://www.thuraya.com/sites/all/modules/ckeditor/ckfinder/userfiles/files/SatSleeve_Plu

[3] iTrinegy, ”NE-ONE Network Emulator User and Administration Guide,” *iTrinegy Limited*,Version 3.0.0, 2017.

[4] iTrinegy, ”NE-ONE Profiler User and Administration Guide,” *iTrinegy Limited*,Version 1.8.0, 2016.

[5] iTrinegy, ”Product Comparison Sheet: NE-ONE Network Emulator Range,” *iTrinegy Limited*,Ref. NE1-ComparisonSheet-25012019. Accessed Online on 26/09/2019 via https://itrinegy.com/itrinegy/wp-content/uploads/2019/02/NE-ONE-Emulator-Comparison-Sheet.pdf

[6] iTrinegy, ”Technical Specification NE-ONE Desktop Appliance,” *iTrinegy Limited*,Ref. NE1-DESK-TECSPEC-04252018. Accessed Online on 18/04/2019 via https://itrinegy.com/itrinegy/wp-content/uploads/2011/05/NE-ONE-Desktop-Technical-Specifications-Sheet.pdf