

End-to-end Data Reliability in Optical Flow Switching

Rui Li

B.Eng., Electrical and Electronic Engineering
Nanyang Technological University (2008)

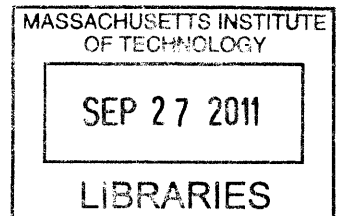
Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of
Master of Science
in
Electrical Engineering and Computer Science
at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2011

© 2011 Massachusetts Institute of Technology
All rights reserved

ARCHIVES



Author
Department of Electrical Engineering and Computer Science

August 18, 2011

Certified by
Vincent W. S. Chan
Joan and Irwin Jacobs Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Certified by
John M. Chapin
Visiting Scientist in Claude E. Shannon Communication and Network Group, RLE
Thesis Supervisor

Accepted by
Leslie A. Kolodziejski
Chair, Department Committee on Graduate Students

End-to-end Data Reliability in Optical Flow Switching

by

Rui Li

Submitted to the Department of Electrical Engineering and Computer Science on August 18, 2011 in partial fulfillment of the requirements for the degree of Master of Science in Electrical Engineering and Computer Science

Abstract

Ever since optical fiber was introduced in the 1970s as a communications medium, optical networking has revolutionized the telecommunications landscape. With sustained exponential increase in bandwidth demand, innovation in optical networking needs to continue to ensure cost-effective communications in the future.

Optical flow switching (OFS) has been proposed for future optical networks to serve large transactions in a cost-effective manner, by means of an all-optical data plane employing end-to-end lightpaths. Due to noise added in the transmission and detection processes, the channel has non-zero probability of bit errors that may corrupt the useful data or flows transmitted. In this thesis, we focus on the end-to-end reliable data delivery part of the Transport Layer protocol and propose effective and efficient algorithms to ensure error-free end-to-end communications for OFS. We will analyze the performance of each algorithm and suggest optimal algorithm(s) to minimize the total delay.

We propose four classes of OFS protocols, then compare them with the Transport Control Protocol (TCP) over Electronic Packet Switching (EPS) and indicate under what values of the parameters: file size, bit error rate (BER), propagation delay and loading factor is OFS better than EPS. This analysis can serve as important guidelines for practical protocol designs for end-to-end data transfer reliability of OFS.

Thesis Supervisor: Vincent W.S. Chan

Title: Joan and Irwin Jacobs Professor of Electrical Engineering and Computer Science

Thesis Supervisor: John M. Chapin

Title: Visiting Scientist in Claude E. Shannon Communication and Network Group, RLE

Acknowledgments

I can never forget when I first arrived at MIT two years ago. Filled in this totally unknown environment were uncertainties and cultural shock - I did not even know what to reply when people asked me "What's up?". It was with so many friends' kind support and company that I spent such a wonderful and meaningful time here.

I feel very grateful and fortunate to have Professor Vincent W.S. Chan as my research supervisor. Through daily interactions, research meetings and writing of this thesis, I have learnt tremendous much from him, not only about how to do research but also about how to become a better self. His patience, insightfulness, and kindness have all helped make my research experience so fulfilling and enriching.

My sincere gratitude also goes to Dr. John Chapin for his invaluable suggestions and guidance in research meetings and development of this thesis. His sharpness, feedback and experience helped me in many aspects. I am particularly impressed by his conciseness and clearness in presenting his ideas.

I hope to take this opportunity to thank Professor Perry Shum at Nanyang Technological University (NTU), Singapore. It was he who introduced me to the exciting world of academic research. I cannot express in word my gratefulness for his care, support and help all along the way. I would also like to thank Professor Robert Gallager for his insightful advice and suggestions. He is always so kind, approachable, inspiring and ready to answer any questions that I may have, and like a kind grandfather. My entry to the communications and networks fields benefited in a great deal from his books (3 out of 4 text books I used in the first year were written by him). I also appreciate so much the help provided by Professor Alan Oppenheim through the two GMS': Graduate Mentoring Seminar and Graduate Magic Seminar. It was such an enjoyable time for me in both seminars and made my life here much smoother and more colorful.

I would also like to thank so many friends that are already part of my life. My office mates Matthew Carey and Shane Fink have provided much support, joy and laughter. Discussions with them can often be inspiring and fun. Special thanks go to Lei Zhang who is such a considerate friend and caring mentor and helped me in many aspects since the moment I landed here. All other members of the Claude E. Shannon Communication and Network Group, including Donna Beaudry (our kind responsible secretary), Andrew Puryear, Mia Qian, Katherine Lin, Henna Huang, and David Cole have all made experience in this group so wonderful.

Also thank Irwin Mark Jacobs and Joan Klein Jacobs for the financial support.

Last but not least, I would like to thank my parents, Quansheng Li and Shimei Zhang, for their selfless care, love and raising my curiosity in mathematics while I was little. Also great thanks go to my siblings, Junying Li, Jun Li and Junyan Li for their love. It is to my family that I dedicate this thesis.

Contents

List of Figures	11
List of Notation	17
1 Introduction	23
1.1 OFS Physical Layer	26
1.1.1 Bit Error Rate	26
1.1.2 Round Trip Time	27
1.2 OFS Higher Layers	27
1.3 Performance Metrics	29
1.3.1 Delay	30
1.3.2 Other Metrics	32
1.4 Key Contributions and Results	33
1.5 Thesis Organization	34

2	Transmission Control Protocol (TCP) with Electronic Packet Switching (EPS)	37
2.1	Standard TCP Description	38
2.2	Delay Analysis of Standard TCP over EPS	40
2.2.1	Delay of TCP with BER = 0	40
2.2.2	Delay of TCP with BER > 0	46
2.3	Summary of Chapter 2	59
3	Protocol with Error Detection by Backward Comparison (EDBC)	61
3.1	EDBC Algorithm Description and Flowchart	62
3.2	Delay Analysis of EDBC Algorithm	71
3.2.1	Delay of EDBC with BER = 0	71
3.2.2	Delay of EDBC with BER > 0	81
3.3	Summary of Chapter 3	89
4	Protocol with Error Detection Code and No Segmentation (EDC-NS)	91
4.1	EDC-NS Algorithm Description and Flowchart	92
4.2	Delay Analysis of EDC-NS Algorithm	98
4.3	Summary of Chapter 4	105
5	Protocol with Error Detection Code and Segmentation (EDC-S)	107
5.1	EDC-S Algorithm Description and Flowchart	108
5.2	Delay Analysis of EDC-S with Retransmissions via OFS (EDC-S (OFS))	114
5.3	Summary of Chapter 5	134

6 Protocol with Forward Error Correction and Segmentation (FEC-S)	135
6.1 FEC-S Algorithm Description	136
6.2 Delay Analysis of FEC-S Algorithm with Retransmissions via OFS (FEC-S (OFS))	138
6.2.1 Preliminaries on Coding	138
6.2.2 Segmentation or Not	146
6.3 Summary of Chapter 6	159
7 Comparison of Various Protocols	161
7.1 Delay Comparison of Various Protocols	162
7.1.1 Delay Comparison of Various Protocols for OFS	162
7.1.2 Delay Comparison of Best Protocol over OFS and TCP over EPS	170
7.2 Preference Maps	185
7.3 Summary of Chapter 7	199
8 Discussion of Results in the Larger Context of the Transport Layer Problems	203
9 Conclusion	207
9.1 Summary of Contributions	207
9.2 Future Work and Challenges	209
Appendix A	211
Bibliography	217

List of Figures

1.1	OFS Overview	25
2.1	Plot of TCP window size vs. number of RTTs $n_t = \left\lceil \frac{t}{RTT} \right\rceil$ since TCP connection establishment	39
2.2	Normalized total delay D_t vs. L_f for standard TCP over EPS when the probability of error is zero	45
2.3	Total delay D_t vs. L_f for standard TCP over EPS when the probability of error is zero. The RTT is assumed to be 91 ms	45
2.4	TCP linear increase (lower bound) Markov chain. State n represents a window size of n , and the maximum number of states in the Markov chain is $n_{max} = M$ [27]	47
2.5	TCP exponential increase (upper bound) Markov chain. State n represents a window size of 2^{n-1} and the maximum number of states is $n_{max} = \log_2 M + 1$ [27]	47
2.6a	Expected number of packets in the K^{th} RTT when $p_e = 0$	51
2.6b	Expected number of packets in K RTTs when $p_e = 0$	51
2.6c	Delay (number of RTTs) vs. file size when $p_e = 0$	52
2.7a	Expected number of packets in the K^{th} RTT when $p_e = 10^{-10}$	52
2.7b	Expected number of packets in K RTTs when $p_e = 10^{-10}$	53
2.7c	Delay (number of RTTs) vs. file size when $p_e = 10^{-10}$	53
2.8a	Expected number of packets in the K^{th} RTT when $p_e = 10^{-8}$	54

2.8b	Expected number of packets in K RTTs when $p_e = 10^{-8}$	54
2.8c	Delay (number of RTTs) vs. file size when $p_e = 10^{-8}$	55
2.9a	Expected number of packets in the K^{th} RTT when $p_e = 10^{-6}$	55
2.9b	Expected number of packets in K RTTs when $p_e = 10^{-6}$	56
2.9c	Delay (number of RTTs) vs. file size when $p_e = 10^{-6}$	56
2.10a	Expected number of packets in the K^{th} RTT when $p_e = 10^{-5}$	57
2.10b	Expected number of packets in K RTTs when $p_e = 10^{-5}$	57
2.10c	Delay (number of RTTs) vs. file size when $p_e = 10^{-5}$	58
3.1	EDBC flowchart with phase I, II and III shown separately.	69
3.2	Flowchart for cases (a) when A stalls (b) when B stalls	70
3.3	Estimation of EPS RTT between MIT and Stanford University using the "ping" command in Windows OS	76
3.4a	EDBC total delay vs. loading factor with $L = 10s, p_e = 0$	78
3.4b	EDBC total delay vs. loading factor with $L = 1s, p_e = 0$	78
3.4c	EDBC total delay vs. loading factor with $L = 100ms, p_e = 0$	79
3.4d	EDBC total delay vs. loading factor with $L = 10ms, p_e = 0$	79
3.5	EDBC normalized delay (number of transaction lengths) vs. loading factor for different transaction lengths. $p_e = 0$	80
3.6	EDBC total delay vs. loading factor with $L = 10s, p_e = 10^{-14}$	83
3.7	EDBC normalized delay (number of transaction lengths) vs. loading factor for different transaction lengths. $p_e = 10^{-14}$	84
3.8	EDBC total delay vs. loading factor with $L = 10s, p_e = 10^{-12}$	85
3.9	EDBC normalized delay (number of transaction lengths) vs. loading factor for different transaction lengths. $p_e = 10^{-12}$	86
3.10	EDBC total delay vs. loading factor with $L = 10s, p_e = 10^{-10}$	87
3.11	EDBC normalized delay (number of transaction lengths) vs. loading factor for different transaction lengths. $p_e = 10^{-10}$	88
4.1	EDC-NS protocol flowchart. There are three different phases: Preparation,	97

Transmission and Conclusion. The "green path" shows the algorithm flow when there is no bit error

4.2	EDC-NS normalized delay (number of transaction lengths) vs. loading factor for different transaction lengths. $p_e = 0$	101
4.3	EDC-NS normalized delay (number of transaction lengths) vs. loading factor for different transaction lengths. $p_e = 10^{-14}$	102
4.4	EDC-NS normalized delay (number of transaction lengths) vs. loading factor for different transaction lengths. $p_e = 10^{-12}$	103
4.5	EDC-NS normalized delay (number of transaction lengths) vs. loading factor for different transaction lengths. $p_e = 10^{-10}$	104
5.1	EDC-S protocol flowchart. The "green path" is the default path if there is no error	113
5.2a	Plot of $E[N_t]$ vs. L_b for $L_f = 10^{10}$ bits, $L_h = 0$.	117
5.2b	Plot of $E[N_t]$ vs. L_b for $L_f = 10^{11}$ bits, $L_h = 0$.	118
5.2c	Plot of $E[N_t]$ vs. L_b for $L_f = 10^{12}$ bits, $L_h = 0$.	118
5.2d	Plot of $E[N_t]$ vs. L_b for $L_f = 10^{13}$ bits, $L_h = 0$.	119
5.3a	Plot of $E[N_t]$ vs. L_b for $L_f = 10^{10}$ bits, $L_h = 320$.	120
5.3b	Plot of $E[N_t]$ vs. L_b for $L_f = 10^{11}$ bits, $L_h = 320$.	121
5.3c	Plot of $E[N_t]$ vs. L_b for $L_f = 10^{12}$ bits, $L_h = 320$.	121
5.3d	Plot of $E[N_t]$ vs. L_b for $L_f = 10^{13}$ bits, $L_h = 320$.	122
5.4a	Plot of $E[D_t]$ vs. L_b for $L_h = 0$, $L_f = 10^{10}$ bits	125
5.4b	Plot of $E[D_t]$ vs. L_b for $L_h = 320$ bits, $L_f = 10^{10}$ bits	126
5.5a	Plot of $E[D_t]$ vs. L_b for $L_h = 0$, $L_f = 10^{11}$ bits	127
5.5b	Plot of $E[D_t]$ vs. L_b for $L_h = 320$ bits, $L_f = 10^{11}$ bits	128
5.6a	Plot of $E[D_t]$ vs. L_b for $L_h = 0$, $L_f = 10^{12}$ bits	129
5.6b	Plot of $E[D_t]$ vs. L_b for $L_h = 320$, $L_f = 10^{12}$ bits, and $p_e = 10^{-10}$, 10^{-8} and 10^{-6}	130
5.7a	Plot of $E[D_t]$ vs. L_b for $L_h = 0$, $L_f = 10^{13}$ bits, and $p_e = 10^{-10}$, 10^{-8} and 10^{-6}	131
5.7b	Plot of $E[D_t]$ vs. L_b for $L_h = 320$, $L_f = 10^{13}$ bits, and $p_e = 10^{-10}$, 10^{-8} and 10^{-6}	132
6.1	Illustration of flow transmission in the case of segmentations	137

6.2	A binary symmetric channel with bit error rate ϵ	140
6.3	Random coding exponent $E_r(R)$ vs. R with $\epsilon = p_e = 10^{-1}$. The maximum value of R is the capacity $\ln 2 + \epsilon \ln(\epsilon) + (1 - \epsilon) \ln(1 - \epsilon) \approx 0.368$	142
6.4	Random coding exponent $E_r(R)$ vs. R with $\epsilon = p_e = 10^{-2}$. The maximum value of R is the capacity $\ln 2 + \epsilon \ln(\epsilon) + (1 - \epsilon) \ln(1 - \epsilon) \approx 0.637$	143
6.5	Random coding exponent $E_r(R)$ vs. R with $\epsilon = p_e = 10^{-3}$. The maximum value of R is the capacity $\ln 2 + \epsilon \ln(\epsilon) + (1 - \epsilon) \ln(1 - \epsilon) \approx 0.685$	144
6.6	Random coding exponent $E_r(R)$ vs. R with $\epsilon = p_e = 10^{-8}$. The maximum value of R is the capacity $\ln 2 + \epsilon \ln(\epsilon) + (1 - \epsilon) \ln(1 - \epsilon) \approx 0.693$	145
6.7a	Probability mass function of total amount of transmitted data which is normalized to the number of encoded flow length L_f'	147
6.7b	Probability mass function of total amount of transmitted data which is normalized to the number of encoded flow length L_f'	148
6.7c	Probability mass function of total amount of transmitted data which is normalized to the number of encoded flow length L_f'	149
6.7d	Probability mass function of total amount of transmitted data which is normalized to the number of encoded flow length L_f'	150
6.7e	Probability mass function of total amount of transmitted data which is normalized to the number of encoded flow length L_f'	151
6.8	Plot of $E[D_t]$ vs. L_b when $\Delta = 0.1$, $L_f = 10^{12}$ bits, $L_h = 0$ and $p_e = 10^{-6}$	154
6.9	Plot of $E[D_t]$ vs. L_b when $\Delta = 1$, $L_f = 10^{12}$ bits, $L_h = 0$ and $p_e = 10^{-6}$	155
6.10	Plot of $E[D_t]$ vs. L_b when $\Delta = 0.1$, $L_f = 10^{10}$ bits, $L_h = 0$ and $p_e = 10^{-6}$	156
6.11	Plot of $E[D_t]$ vs. L_b when $\Delta = 1$, $L_f = 10^{10}$ bits, $L_h = 0$ and $p_e = 10^{-6}$	157
7.1	Comparison of EDC-S and FEC-S in terms of $E[N_t]$ vs. L_b for $L_f = 10^{10}$ bits, $L_h = 0$, and $p_e = 10^{-6}$	164
7.2	Comparison of EDC-S and FEC-S in terms of $E[N_t]$ vs. L_b for $L_f = 10^{12}$ bits, $L_h = 0$, and $p_e = 10^{-6}$	165
7.3	Comparison of EDC-S and FEC-S in terms of $E[N_t]$ vs. L_b for $L_f = 10^{10}$ bits, $L_h = 320$	165

	bits, and $p_e = 10^{-6}$	
7.4	Comparison of EDC-S and FEC-S in terms of $E[N_t]$ vs. L_b for $L_f = 10^{12}$ bits, $L_h = 320$ bits, and $p_e = 10^{-6}$	166
7.5	Comparison of EDC-S and FEC-S in terms of $E[D_t]$ vs. L_b for $L_f = 10^{10}$ bits, $L_h = 0$, and $p_e = 10^{-6}$.	166
7.6	Comparison of EDC-S and FEC-S in terms of $E[D_t]$ vs. L_b for $L_f = 10^{12}$ bits, $L_h = 0$, and $p_e = 10^{-6}$.	167
7.7	Comparison of EDC-S and FEC-S in terms of $E[D_t]$ vs. L_b for $L_f = 10^{10}$ bits, $L_h = 320$ bits, and $p_e = 10^{-6}$.	167
7.8	Comparison of EDC-S and FEC-S in terms of $E[D_t]$ vs. L_b for $L_f = 10^{12}$ bits, $L_h = 320$ bits, and $p_e = 10^{-6}$	168
7.9	Delay comparison of TCP with FEC-S (OFS) when $p_e = 0$, $L_h = 320$ bits, $T_{pg} = 100$ ms	171
7.10	Delay comparison of TCP with FEC-S (OFS) when $p_e = 10^{-10}$, $L_h = 320$ bits, $T_{pg} = 100$ ms	172
7.11	Delay comparison of TCP with FEC-S (OFS) when $p_e = 10^{-8}$, $L_h = 320$ bits, $T_{pg} = 100$ ms	173
7.12	Delay comparison of TCP with FEC-S (OFS) when $p_e = 10^{-6}$, $L_h = 320$ bits, $T_{pg} = 100$ ms	174
7.13	Delay comparison of TCP with FEC-S (OFS) when $p_e = 0$, $L_h = 320$ bits, $T_{pg} = 10$ ms	176
7.14	Delay comparison of TCP with FEC-S (OFS) when $p_e = 10^{-10}$, $L_h = 320$ bits, $T_{pg} = 10$ ms	177
7.15	Delay comparison of TCP with FEC-S (OFS) when $p_e = 10^{-8}$, $L_h = 320$ bits, $T_{pg} = 10$ ms	178
7.16	Delay comparison of TCP with FEC-S (OFS) when $p_e = 10^{-6}$, $L_h = 320$ bits, $T_{pg} = 10$ ms, $R_{OFS} = R_{EPS} = 10$ Gbps	179
7.17	Delay comparison of TCP with FEC-S (OFS) when $p_e = 0$, $L_h = 320$ bits, $T_{pg} = 10$ ms	181
7.18	Delay comparison of TCP with FEC-S (OFS) when $p_e = 10^{-10}$, $L_h = 320$ bits	182
7.19	Delay comparison of TCP with FEC-S (OFS) when $p_e = 10^{-10}$, $L_h = 320$ bits	183

7.20	Delay comparison of TCP with FEC-S (OFS) when $p_e = 10^{-6}$, $L_h = 320$ bits	184
7.21a	Preference maps for EPS and OFS for different flow lengths L_f and propagation delays when $p_e = 10^{-8}$	186
7.21b	Preference maps for EPS and OFS for different flow lengths L_f and propagation delays when $p_e = 10^{-8}$	187
7.21c	Preference maps for EPS and OFS for different flow lengths L_f and propagation delays when $p_e = 10^{-8}$	188
7.22a	Preference maps for EPS and OFS for different flow lengths L_f and propagation delays when $p_e = 10^{-8}$	189
7.22b	Preference maps for EPS and OFS for different flow lengths L_f and propagation delays when $p_e = 10^{-8}$	190
7.22c	Preference maps for EPS and OFS for different flow lengths L_f and propagation delays when $p_e = 10^{-8}$	191
7.23a	Preference maps for EPS and OFS for different flow lengths L_f and loading factors S when $p_e = 10^{-8}$	193
7.23b	Preference maps for EPS and OFS for different flow lengths L_f and loading factors S when $p_e = 10^{-8}$	194
7.23c	Preference maps for EPS and OFS for different flow lengths L_f and loading factors S when $p_e = 10^{-8}$	195
7.24a	Preference maps for EPS and OFS for different flow lengths L_f and loading factors S when $p_e = 10^{-8}$	196
7.24b	Preference maps for EPS and OFS for different flow lengths L_f and loading factors S when $p_e = 10^{-8}$	197
7.24c	Preference maps for EPS and OFS for different flow lengths L_f and loading factors S when $p_e = 10^{-8}$	198
7.25	Preference map as in Fig. 7.24(c) except for the addition of a possible boundary line when the buffers have finite memory.	200

List of Notation

Acronyms

ACK	positive acknowledgement
BER	bit error rate
CRC	cyclic redundancy check
DLC	Data Link Control
EDC-NS	error detection code and no segmentation
EDC-S	error detection code with segmentation
EDFA	erbium doped fiber amplifier
EPS	electronic packet switching
ERU	EPS resource usage
FEC-S	forward error correction and segmentation

FEC-S (OFS) FEC-S with retransmissions over OFS

MAN metropolitan-area network

NACK negative acknowledgement

OCU OFS channel usage

OFS optical flow switching

OXC optical cross connect

TCP transmission control protocol

WAN wide-area network

WDM wavelength division multiplexing

Notations

D_f Delay caused by the first transmission

D_{pg}^{sch} Total propagation delay in the process of scheduling

D_q Queuing delay at the scheduler for channel reservation

D_r Delay caused by retransmission(s) until the flow is error-free

D_t Total delay defined as the time from the moment the user first requests transmission via OFS for a particular flow, until the last bit of the flow is successfully passed to the application layer at the destination, with possible retransmission(s)

D_{TCP} Time taken to establish all three TCP connections between the sender, receiver and scheduler before scheduling processes can begin

D_{TCP}^{AB}	Time taken to establish TCP connection between A and B
D_{TCP}^{As}	Time taken to establish TCP connection between A and scheduler
D_{TCP}^{Bs}	Time taken to establish TCP connection between B and scheduler
D_{TCP}^{inf}	Time for A to send the informative message to and receives confirmation from B
L	Flow transmission time
l_b	Block length of a segmented flow in bytes
L_b	Block length of a segmented flow in bits
l_f	Flow length in bytes
L_f	Flow length in bits before encoding
L'_f	Flow length in bits after encoding
L_f^r	Retransmitted flow length in bits
l_p	Average length (in bytes) of transmitted units other than the flow, e.g. EPS packets
n_b	Number of blocks in a segmented flow
n_p	Number of transmitted EPS packets
p_e	Probability of error for a bit (i.e. bit error rate)
$p_{e,b}$	Probability of error for a block (i.e. block error rate)
$p_{e,f}$	Probability of error for a flow (i.e. flow error rate)
$p_{e,p}$	Probability of error for an EPS packet (i.e. packet error rate)
R	(Code) Rate defined in the context of coding (See eqn. (6.1))

R_2	Normalized code rate defined as the ratio of the message length to the codeword length
R_{EPS}	EPS link rate
R_{OFS}	OFS link rate
S	Loading factor (for EPS) or WAN wavelength channel utilization (for OFS)
T_{pg}	(Forward) Propagation delay from the sender to the receiver (for EPS)
T_{pc}^{EPS}	EPS processing delay at the sender and/or receiver added to the overall delay
T_{pc}^{OFS}	OFS processing delay at the sender and/or receiver added to the overall delay
T_{pg}^{EPS}	EPS one-way delay between the sender and receiver
T_{pg}^{OFS}	OFS one-way delay between the sender and receiver
T_q	Queuing delay (for EPS)
T_{tr}	Transmission delay (for EPS)
$\widehat{W}_{M,k}$	Average queuing delay of M/M/k queuing system as a function of offered load
X	Service time of a primary request
\bar{Y}	Average time spent at the head of the primary queue reserving resources in both the source and destination distribution networks (DNs)

Greek Symbols

α_a	Processing cost of a byte for algorithm a
β_a	Processing cost of a transmitted unit for algorithm a

- Δ Coefficient that captures the linear relationship between the decoding time and code word length
- Δt Guard time of the channel reservation
- ϵ A number within $[0, 1]$
- λ_c Arrival rate of OFS flows for a source-destination MAN pair, normalized by the number of provisioned wavelength channels
- λ_m Arrival rate of OFS flows for a source-destination MAN pair
- ρ Dummy variable in the range of $[0,1]$
- ρ_o Offered load
- τ Time taken for the receiver to pass the decoded data to the application layer

Chapter 1

Introduction

Optical networks have gone through several major technological advances since optical fibers were first employed in the 1970s. The first-generation optical networks were used in replacement of copper links for telephony communications. The intention behind this replacement was to utilize the large bandwidth of optical fibers - roughly 30 THz - to meet increasing telephony traffic demands. The replacement was only a partial success in the sense that the traditional architectures that used electronic networking components were maintained, which constrained the processing speed at network nodes due to the speed of electronics.

The second-generation optical networks in the 1990s started to employ optical networking devices in addition to fiber to reduce the limitations in electronics due to large increase in data traffic [1, 15]. The traffic not only increased in volume by orders of magnitude but also was characterized by a heavy-tailed distribution [2-6] that was quite different from the telephony traffic in the first-generation networks. Driven by this traffic change, architectural advances occurred in the wide-area network (WAN) with the introduction of wavelength division multiplexing (WDM) together with optical amplification (e.g. with erbium doped fiber amplifier (EDFA)) but using only electronics. However, there has been no imperative to have direct user access to the core network [1].

With the continuous rapid increase of bandwidth demand, optical flow switching (OFS) has been proposed [7] for future optical networks to serve large transactions in a cost-effective manner by means of an all-optical data plane employing end-to-end lightpaths [8]. OFS directly connects the source and destination end users via the access network, MAN and WAN. Moreover, it is intended for users with large transactions that can fully utilize a wavelength channel for at least hundreds of milliseconds or longer. OFS can achieve lower delay, higher throughput and lower cost than current electronic packet switched (EPS) networks [8-17, 19, 20], by bypassing intermediate routers, a computationally intensive and expensive part of the network. Fig. 1.1 shows the conceptual construct of the proposed OFS architecture.

In this new architecture, however, the transmission control protocol (TCP) that has been successful in current EPS networks may have to be revised, with its basic functions implemented in possibly different ways. These functions include performing congestion control, matching rates between the sender and receiver, and ensuring end-to-end data transfer reliability. For OFS, congestion control is taken care of by flow scheduling, and rate matching is done with an agreed constant rate between the end users over the entire flow duration (this is possible because in OFS a wavelength channel, once reserved, is dedicated to

a particular flow). However, end-to-end reliable data transfer remains a problem to be solved.

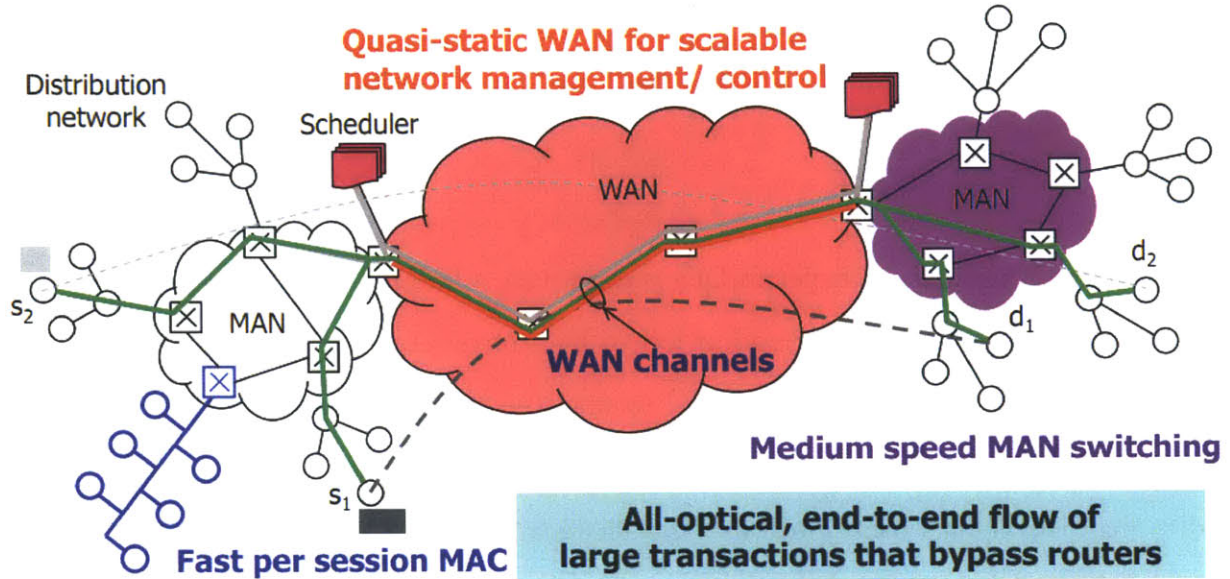


Figure 1.1: OFS Overview [21].

The focus of our research is to propose effective and efficient algorithms to ensure error-free end-to-end communications for OFS. In this context, an algorithm is effective if it works correctly so that the data can be transferred error-free by a user prescribed time deadline. An algorithm is delay efficient if the delay advantage of OFS over EPS is preserved¹. We will compare delay performances of the optimal OFS algorithm with TCP over EPS, and provide guidance for protocol choice between OFS and EPS.

Like EPS networks, OFS networks can also be viewed to have a layered structure. From low to high, the layers are respectively: the Physical Layer, the Data Link Control (DLC) Layer, the Network Layer, the Transport Layer, and the Application Layer [23]. End-to-end reliability can be implemented at multiple layers in OFS, ranging from the Physical Layer to

¹ Though OFS has other advantages over EPS, such as in terms of throughput and cost, we focus our attention to the delay in this work due to time limit.

the Transport Layer. The optimal choice of protocols at higher layers depends on the Physical Layer characteristics. We therefore briefly discuss the OFS Physical Layer before going into detailed discussions of the protocols for end-to-end reliability.

1.1 OFS Physical Layer

The function of the Physical Layer in OFS is to provide a link for transmitting a sequence of bits between the source and destination joined by a physical communication channel. The modulator at the transmitting side is used to map the incoming bits from the next higher layer into signals appropriate for the channel, and the demodulator at the receiving end is used to map the signals back into bits. The Physical Layer often suffers from noise in the channel and at the receiver which may translate to bit errors at the receiver.

1.1.1 Bit Error Rate

Optical signals transmitted over the optical fiber suffer from attenuation, have noise added to them from optical amplifiers, and experience a variety of other impairments, such as dispersion and nonlinearity. At the receiver, the transmitted optical data is converted back to electrical signals, and recovered with an acceptable bit error rate (BER), which is defined as the ratio of the number of detected bits that are incorrect to the total number of transferred bits. In general optical channel effects contain two types of errors: independent and identically distributed (IID) errors where each bit has the same probability of being erroneous and errors occur independently, and burst errors where a contiguous sequence of bits may be in error. IID errors can be caused by thermal noise, shot noise, and amplified spontaneous emission noise, etc [22]. Burst errors can be caused by sudden, irregular events that last for a short period and cause contiguous errors in the networks, such as power

undershoot or overshoot during EDFA transients when wavelength channels are added or dropped while others are being used for transmissions [22].

In this work, we limit our attention to IID bit errors due to time limit and leave cases with burst errors for future research. Typically EPS has BER smaller than 10^{-12} after forward error correction (FEC) at the Physical Layer.² The BER may be changing slowly with time, but can be considered constant for many application scenarios. The design of OFS Transport Layer protocols should take into consideration the range of expected BER values.

1.1.2 Round Trip Time

For different source-destination pairs, the propagation delays and/or round-trip times (RTTs) can be different. Even for the same source-destination pair, with different loading factors, the RTTs can also be different. The RTTs can be as small as a few milliseconds, for example, between MIT and Columbia University, and can also be easily over 100ms, for example, between MIT and Singapore. For small RTTs, the delay caused by retransmissions might not be a big problem, while for large RTTs, retransmissions can impose long delays that are very bad for applications with time deadlines. Therefore, in the design of upper layers in OFS we will also need to take the RTT into consideration.

1.2 OFS Higher Layers

In OFS, once the scheduler establishes a dedicated end-to-end lightpath between the source and destination, the data are transmitted via the all-optical data plane. The lightpath or connection is dedicated to the flow throughout the initial transmission. If retransmissions

² Bursts of higher error rates can occur due to switching and amplifier transients; these bursts are not considered in this work.

due to errors are done via OFS, a new lightpath is requested from the scheduler. End-to-end data reliability is achieved with the coordinated efforts of different layers. We now briefly discuss the Data Link Control (DLC) and Transport Layers.

The customary purpose of a DLC is to convert the unreliable bit pipe at the Physical Layer into a higher-level, virtual communication link for sending data asynchronously but error-free in both directions [23]. The DLC layer places overhead control bits called a header at the beginning of each block/flow (e.g. each frame of a SONET frame, or each block of the flow if segmentation is done at the Transport Layer), and some more overhead bits called a trailer at the end of each flow (or block), resulting in a longer string of bits called a frame. Some of these overhead bits are used to determine if errors have occurred in the transmitted frames, and some identify the beginning and ending of frames. FEC codes such as the Reed-Solomon (RS) code, turbo-code or low-density parity check (LDPC) code are usually used in the header or trailer to allow both error detection and correction. In many cases, FEC is not an option but a necessity to reduce the bit error rate to an acceptable range.

To ensure data transfer reliability after FEC, error detection and error recovery via retransmissions are usually done at the Transport Layer. Error detection can be implemented either at the sender with comparison of a backward flow or at the receiver with an error detection code (e.g. cyclic redundancy check (CRC) code). Error detection by comparison of a backward flow (which we call Error Detection by Backward Comparison or EDBC) works by sending the received data back to the sender and comparing it with the original data. Should any error be found in the comparison process, the sender notifies the receiver that the flow transmitted is erroneous, and the whole flow is retransmitted after rescheduling. Otherwise, the sender sends an acknowledgement (ACK) message via EPS to the receiver to confirm that the received data is error-free. With error detection codes, should any error be found by the receiver, a request for retransmission of part of, or even the whole, transaction may be made via negative acknowledgement (NACK) to the transmitter. Retransmissions are

then done until the whole flow is received error-free. When the flow is segmented to smaller blocks before transmission, retransmissions can be done via either OFS or EPS, with differing performance, and is analyzed in detail in Chapter 7.

There is also a very small probability that the ACKs/NACKs are in error or never received. In the former case, the best approach is to use a strong FEC code together with an error detection code (e.g. CRC or checksum) on the ACK/NACK signals and then use a time-out to catch the rare events of not being able to correct for errors. In the latter case, timeouts can be used to alert the sender and receiver that the ACKs/NACKs have not been received and request that the receiver retransmit the ACKs/NACKs.

Section 1.3 gives a discussion of the performance metrics for different Transport Layer protocols. In Chapter 2 to Chapter 6, we look at various options available at the OFS transport layer (together with the DLC layer) to ensure end-to-end error-free communications.

1.3 Performance Metrics

In this work, we focus on the *delay* metric to compare performance of different Transport Layer protocols. Although there are other metrics that may also be used, such as throughput, cost (e.g. processing cost), network resource usage, and data efficiency, in this thesis we constrain our analysis to delay due to time limit.

In our discussions on Transport Layer protocols, we assume the control plane of OFS uses EPS. Furthermore, we assume that in the event of uncorrectable but detected errors,

- if retransmissions are done via OFS, then rescheduling is needed before each retransmission;

- if retransmissions are done via EPS, then no rescheduling is needed; instead retransmissions are purely dealt with by EPS (e.g. using TCP) until the flow data are successfully received at the destination.

Although it is possible to split the data that require retransmissions and retransmit them on different planes in parallel processes, we do not discuss that case. That is, we will assume that only one plane (either EPS or OFS, but not both) is used for each transmission. However, the data plane used can be different across different transmissions.

1.3.1 Delay

The *delay* of an OFS flow is defined as the time from the moment the user first requests transmission via OFS for a particular flow, until the last bit of the flow is successfully passed to the application layer at the destination, with possible retransmissions. The total delay of an OFS flow includes delay caused by the first transmission via OFS and also possible delay caused by retransmissions via either OFS or EPS when the flow has uncorrectable (but detected) errors. For the first transmission, the delay consists of processing delay, scheduling/queuing delay at the scheduler, transmission delay and propagation delay. For each retransmission, the expected delay can be the same as the delay of the first transmission if the whole flow is retransmitted via OFS, and can be different if only part of the flow is retransmitted and/or if the retransmission is via EPS.

The processing delay is the time it takes to process the flow/block header(s). At the transmitter, it includes possible delay caused by segmentation, framing and encoding, and at the receiver it includes delay by decoding and reassembly. For this work we ignore and assume the processing delay to be small when compared with typical flow durations ($> 1s$). We will discuss processing delays again in Chapter 6 when discussing a protocol that employs forward error correction. The scheduling delay consists of the request packets

propagation and processing delays from the sender and receiver and the request queuing delay at the scheduler (more details can be found in Chapter 4 of Guy Weichenberg's PhD thesis [1]). The request propagation and processing delay is at least one EPS round trip time, and the flow queuing delay depends much on the traffic condition or the WAN wavelength channel utilization (also called loading factor, see Section 3.2 below), defined as the average percentage of time that a WAN OFS channel is busy for data transmissions. A plot of the queuing delay and service time vs. WAN wavelength channel utilization is shown in Chapter 4 of [1] with three different types of flow length distributions: constant, exponential and heavy-tail. In all three length distributions, the request queuing delay and service time grow with increasing loading factor. We discuss the case of constant flow lengths in Chapter 3. The propagation delay is determined by the fiber distance between the source and destination, and speed of light in optical fibers. The transmission delay is the amount of time it takes to push the flow bits onto the fiber, and is determined by the flow size and the link rate.

For retransmissions, the delays can be very different depending on the data planes and/or mechanisms used, which we will discuss further when comparing different Transport Layer protocols.

When the first transmission and retransmissions are separable in time, i.e. they do not overlap, and the retransmission process is initiated by the transmitter as soon as it recognizes that retransmission is needed, we can quantify the total delay D_t as follows:

$$D_t = D_f + D_r$$

where D_f is the delay caused by the first transmission and D_r the delay by retransmissions until the flow is error-free ($D_r = 0$ when there is no retransmission). D_f basically consists of all the delay when there is no retransmission needed, including the scheduling delay, processing delay (e.g. possible error checking delay), transmission delay, propagation delay

and acknowledgment packet delay. As briefly discussed above, D_r can be very different depending on the retransmission strategies, which we will discuss more in Chapter 3-6.

There are also cases where the first transmission and retransmissions can run in parallel, such as the case of retransmitting data via EPS while the first transmission is still going on via OFS. We will discuss those cases separately under the corresponding protocol section.

1.3.2 Other Metrics

Examples of other metrics that may also be used to evaluate performances of OFS protocols include:

- *Network Resource Usage (NRU)*, which consists of both OFS and EPS resource usage. For OFS, the most precious network resource is the wavelength channels, and we can approximate the OFS network resource usage as a function of *OFS channel usage (OCU)*. Since every reserved wavelength is fully utilized and dedicated to a particular flow, OCU is directly proportional to the total amount of time that a flow uses the OFS wavelength channel(s) until the flow is error free at the receiver. In the case of retransmissions via EPS plane, there should be a second term that captures the EPS network resources used besides OCU, which we call *EPS resource usage (ERU)*, defined as the amount of EPS resources used until the flow is error-free at the receiver.
- *Processing Cost*, which consists of both per-byte and per-packet/ block processing cost. A block here is defined as a segment of a flow. Under the condition that there is no bit in error, the more blocks a flow is segmented into, the more processing cost is introduced. When there are bit errors, there is a relatively complex relationship between the processing cost and number of blocks a flow is segmented into.

1.4 Key Contributions and Results

This thesis addresses the problem of end-to-end data transfer reliability for optical flow switching, and proposes effective and efficient algorithms.

After comparing four classes of OFS protocols in Chapter 3-6 which are natural extensions of the previous ones, we find out that the protocol with forward error correction and segmentation (FEC-S) gives the best performance in terms of minimized delay over OFS, especially when the BER is high. With error reduction capability through forward error correction codes, FEC-S protocol is a promising protocol to reduce the probability of error occurrences and hence retransmissions and total delay. It is shown that, with FEC-S, the total number of transmissions can be reduced to 1 even if the original bit error rate is quite high (e.g. 10^{-6}). Nevertheless, error reduction is at the cost of adding redundancy and extra decoding delay, which increases monotonically with increasing segment or block size. With proper choice of block size given a certain flow size, the total delay can be minimized to the extent that there is almost no retransmission and the redundancy added is negligible compared to the flow size. The minimum delay is found when the block size is between 10^4 bits and $L_f/100$ bits (assume the flow size $L_f \geq 10^6$ bits), and is almost independent of the decoding delay coefficient.

The above results can serve as an important guidance for what protocol to use when OFS performance is measured in terms of total delay.

We also compare delay performance of OFS and EPS by comparing FEC-S over OFS against TCP over EPS. We draw “preference maps” (regions where OFS is better than EPS and regions where EPS is better) based on the file size, BER, propagation delay and loading factor. Comparison results show that OFS is preferred over EPS when the files are large and/or when the propagation delay is large and/or when the loading factor is large. Preference maps in Fig. 7.22-7.25 can serve as important guidance for protocol choice in practice given

different file sizes, propagation delays and loading factors³. This work can provide part of the answer to the important question below: OFS was claimed to be good for "large" transactions, but how large is "large"?

1.5 Thesis Organization

The rest of the thesis is organized as follows.

In Chapter 2, we model linear and exponential bounds of TCP using Markov chains and find the total delays for different BERs. We also briefly discuss EPS queuing delay as a function of router service speed and loading factor.

In Chapter 3, we discuss the OFS protocol with error detection by backward comparison (i.e. EDBC). We first describe the algorithm. We also include discussions of OFS queuing delay in this chapter.

In Chapter 4, we try to reduce the queuing delay incurred by EDBC and discuss the OFS protocol with error detection code and no segmentation (i.e. EDC-NS), and analyze its performance in terms of delay.

In Chapter 5, we extend the EDC-NS protocol to the OFS protocol with error detection code and segmentation (i.e. EDC-S). We then discuss the optimal block size to use for minimum delay.

In Chapter 6, we discuss the OFS protocol with forward error correction and segmentation (FEC-S). We start with some coding preliminaries by relating the code rate with the probability of errors through the random coding exponent. We then analyze its performances in terms of delay.

³ The BER is assumed to be 10^{-8} in Fig. 7.22-7.25 for illustration purposes. For different BERs, similar preference maps can be drawn, with the protocol boundary lines being shifted in different directions.

In Chapter 7, we first compare the four classes of protocols in Chapter 3-6, i.e. EDBC, EDC-NS, EDC-S, FEC-S, and then compare the best protocol among these four with TCP over EPS. Based on the comparison, we draw the preference maps with different file sizes, BERs, propagation delays and loading factors.

In Chapter 8, we discuss the previous results in a larger context of the Transport Layer problems. We state the problems we address and what we do not, and then discuss the usefulness of our results in a larger context.

Finally in Chapter 9, we conclude the thesis with a summary of our contributions and discussions of future research directions.

Chapter 2

Transmission Control Protocol (TCP) with Electronic Packet Switching (EPS)

In EPS, TCP can perform well the functions of congestion control, rate matching and end-to-end data reliability. Congestion control is performed through four intertwined phases known as *Slow Start*, *Congestion Avoidance*, *Fast Retransmit* and *Fast Recovery*, with packet loss in the network being interpreted as congestion. Rate matching between the end users is done by ensuring that the rate at which new packets are injected into the network (controlled by the congestion window) is the rate at which the acknowledgements are returned by the other end (controlled by the receiver). End-to-end data transfer reliability is ensured using Automatic Repeat reQuest (ARQ) mechanism of TCP together with error detections and corrections in lower layers.

Section 2.1 gives a description of standard TCP. Section 2.2 discusses the delay performance of TCP in the cases of zero and non-zero probabilities of errors. For the rest of this work packet loss due to congestion is NOT considered.

2.1 Standard TCP Description

Denote n_w as the TCP window size. Standard TCP works in the following manner [21]:

Slow Start:

1. After the TCP connection is established between the sender A and receiver B , the window size is initialized to 1 so that A sends one framed packet to B , and then waits for ACK, i.e. the positive acknowledgement packet.
2. Upon receiving the ACK, A then increases its window size to twice the size for the last transmission. This step repeats until n_w reaches 64.

Congestion Avoidance:

3. For each RTT, if there is no packet loss, increase n_w by 1 without exceeding 128. That is, n_w is then increased by 1 when the sender receives successful ACKs for all packets sent in the last RTT until n_w reaches the maximum value of 128.

Fast Retransmit/Recovery:

If there are 3 duplicate feedbacks of the same request number (RN), TCP assumes there is packet loss caused by congestion in the network (no matter whether there is really congestion), and sets n_w to be $n_w/2$. TCP timeouts when there is neither ACK nor NACK (i.e. negative acknowledgement) for $RTT + 4\sigma$, where σ is the standard deviation of the RTT, and goes back to the *Slow Start* phase (reset $n_w = 1$).

Let t be the time since TCP connection establishment. It can be readily shown that if there is no congestion packet loss, TCP window size is lower bounded by t/RTT , and upper bounded by both $57 + \lceil t/RTT \rceil$ and $2^{\lceil t/RTT \rceil - 1}$, where $\lceil \cdot \rceil$ is the ceiling operator that takes the smallest integer value larger than or equal to the argument inside. Fig. 2.1 is a plot of window size vs. the number of RTTs $n_t = \lceil \frac{t}{RTT} \rceil$ when there is no packet loss.

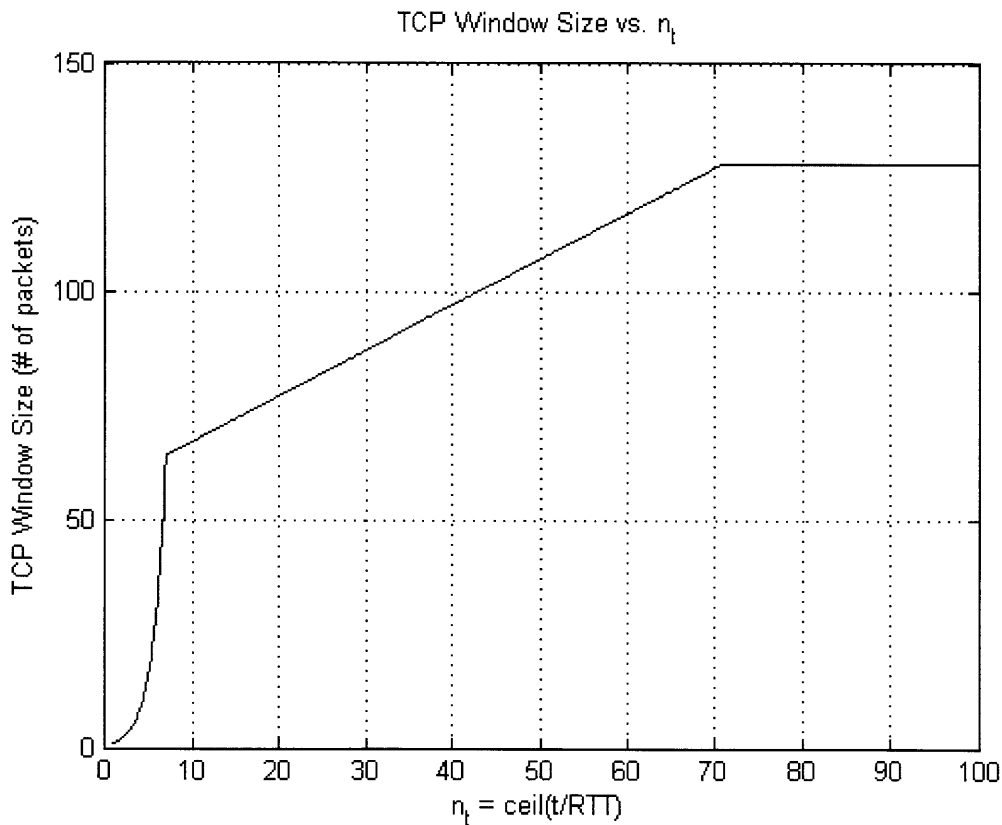


Figure 2.1: Plot of TCP window size vs. number of RTTs $n_t = \lceil \frac{t}{RTT} \rceil$ since TCP connection establishment. Standard TCP with no packet loss is assumed.

Note that the main reasons for packet loss are packet errors and router congestion. In this thesis, we assume that there is no congestion in the EPS networks by assuming infinite

buffers at each router and there is no packet drop due to limited buffer space; that is, the only reason for packet loss is assumed to be packet errors. The assumption of infinite buffers is not valid in real networks but still gives us useful answers for the purpose of comparing total delay caused by EPS and OFS protocols. Finite buffers may cause packet drop and hence window closing, leading to longer delay for TCP. With the assumption of infinite buffers we are looking at an optimistic version of TCP in terms of delay. We show in Chapter 7 that even this optimistic version of TCP does not perform so well in terms of total delay when compared to new protocols that are designed for sending very large files at very high rates such as using OFS.

Packet errors are caused by bit errors. When the BER is zero, TCP window size always stays at its maximum (i.e. 128 packets for standard TCP) whenever it reaches that value. When $BER > 0$, the average window size is smaller than the maximum window size and this case is treated in Section 2.2.2. We assume selective repeat ARQ [23] is used.

2.2 Delay Analysis of Standard TCP over EPS

2.2.1 Delay of TCP with $BER = 0$

We first analyze delay when the bit error rate is zero. Recall that $n_t = \lceil t/RTT \rceil$ is the number of round trip times (RTTs) until time t since TCP connection establishment. T_{pg}^{EPS} is the (average) one-way EPS delay between the sender and the receiver, including the processing delay T_{pc} , the propagation delay T_{pg} , the queuing delay T_q , and the transmission delay T_{tr} . The processing delay is the delay due to processing at the sender/receiver and the router, the (one-way) propagation delay is the time of flight from the sender to the receiver, the queuing delay is due to the queuing at the routers' buffers, and the total transmission delay is the time it takes to pump the bits onto the link and given by the file size divided by the EPS link rate.

In this work, we do not consider processing delay in our analysis. We used the highly simplified mode for the queue at each router of an M/M/1 queue; that is, both the packet arrival and departure processes at each router can be modeled to be Poisson processes with one server (i.e. one router). Let the arrival rate be λ and the service rate be μ . The loading factor is $S = \lambda/\mu$. The queuing delay at one router can be estimated to be

$$T_q = \frac{S}{\mu - \lambda} = \frac{1}{\mu} \cdot \frac{S}{1 - S} \quad (2.1)$$

It can be seen that as S increases, T_q also increases. Especially when $S \rightarrow 1$, T_q goes to infinity. We discuss more this effect in Chapter 7 when we compare delay of TCP over EPS and that of new Transport Layer protocols over OFS.

For example, for a router with speed limited to be $R_{router} = 2.5$ Gbps, the service rate is approximately

$$\mu = \frac{R_{router}}{L_p} = \frac{2.5 \times 10^9}{1.2 \times 10^4} \approx 2.08 \times 10^5 \text{ packets/second} \quad (2.2)$$

where $L_p = 1.2 \times 10^4$ bits is the EPS packet size. For a loading factor of $S = 0.9$, the queuing delay is approximately

$$T_q \approx 43.3 \times 10^{-6} \text{ seconds}$$

Assume there is one router in every 600 km. The total queuing delay depends on the number of routers (i.e. the distance) between the sender and receiver. We also assume the network is symmetric. In other words, the forward delay and reverse delay are the same, i.e. $RTT = 2T_{pg}^{EPS}$.

From the standard TCP description in Section 2.1, for $n_t \leq 7$, i.e. $t \leq 14T_{pg}^{EPS}$, the window size is exponentially increasing in each subsequent RTT until it reaches 64. This is the *Slow Start* phase, and the total number of data packets sent is

$$\sum_{k=1}^{n_t} 2^{k-1} = \sum_{k=0}^{n_t-1} 2^k \leq \sum_{k=0}^6 2^k = 127 \quad (2.3)$$

For $7 < n_t \leq 71$, the TCP window size is increased by one for each RTT until it reaches 128. This is the *Congestion Avoidance* phase, and the total number of data packets sent (excluding ACKs) is

$$\begin{aligned} & \sum_{k=1}^7 2^{k-1} + \sum_{k=8}^{n_t} (57 + k) \\ &= 127 + \sum_{k=1}^{n_t-7} (64 + k) \\ &= 127 + \frac{(n_t - 7)(n_t + 122)}{2} \\ &= \frac{1}{2}n_t^2 + \frac{115}{2}n_t - 300 \end{aligned} \quad (2.4)$$

For the special case of $n_t = 71$, the window size has just reached 128, and the total number of packets sent is

$$\frac{1}{2} \times 71^2 + \frac{115}{2} \times 71 - 300 = 6,303 \quad (2.5)$$

For $n_t > 71$, the window size remains to be 128. The total number of packets sent is

$$\sum_{k=1}^7 2^{k-1} + \sum_{k=8}^{71} (57 + k) + (n_t - 71) \times 128 = 128n_t - 2,785 \quad (2.6)$$

The size of each full EPS packet is $L_p = 1.2 \times 10^4$ bits. The number of packets for a flow of size L_f is $n_p = \left\lceil \frac{L_f}{L_p} \right\rceil = \left\lceil \frac{L_f}{1.2 \times 10^4} \right\rceil$. For $L_f \leq 127L_p = 127 \times 1.2 \times 10^4 = 1.524$ Mbits, i.e. $n_p \leq 127$, the number of RTTs required to transmit L_f will be

$$n_t = \lceil \log_2(n_p + 1) \rceil = \left\lceil \log_2 \left(\left\lceil \frac{L_f}{L_p} \right\rceil + 1 \right) \right\rceil \quad (2.7)$$

For 75.636 Mbits $= 6,303L_p \geq L_f > 127L_p = 1.524$ Mbits, i.e. $6,303 \geq n_p > 127$, we let n_t be the smallest integer that satisfies

$$\frac{1}{2}n_t^2 + \frac{115}{2}n_t - 300 \geq n_p = \left\lceil \frac{L_f}{L_p} \right\rceil \quad (2.8)$$

Solving for n_t gives

$$n_t = \left\lceil \sqrt{3906.25 + 2n_p} - 57.5 \right\rceil = \left\lceil \sqrt{3906.25 + 2 \left\lceil \frac{L_f}{L_p} \right\rceil} - 57.5 \right\rceil \quad (2.9)$$

For $L_f > 6,303L_p = 75.636$ Mbits, i.e. $n_p > 6,303$, we let n_t be the smallest integer that satisfies

$$128n_t - 2,785 \geq n_p = \left\lceil \frac{L_f}{L_p} \right\rceil \quad (2.10)$$

Solving for n_t gives

$$n_t = \left\lceil \frac{\left\lceil \frac{L_f}{L_p} \right\rceil + 2,785}{128} \right\rceil \quad (2.11)$$

We assume TCP connections are established using the 3-way handshake method, which is described below:

If A wants to establish a TCP connection with B,

- 1) *A first sends a TCP synchronize packet (SYN) to B.*
- 2) *After B receives A's SYN, B sends a synchronize-acknowledgement (SYN-ACK).*
- 3) *After A receives the SYN-ACK, A sends an SYN-ACK-acknowledgement (SYN-ACK-ACK).*
- 4) *When B receives SYN-ACK-ACK, TCP connection is established between A and B.*

So in general it takes 3 one-way EPS delays to establish a TCP connection. TCP shutdown is done in a way similar to the above 3-way handshake process.

With the TCP connection setup time ($\sim 3T_{pg}^{EPS}$), the total amount of time required to transmit flow of size L_f over EPS is

$$\begin{aligned}
 D_t &= 3T_{pg}^{EPS} + 2T_{pg}^{EPS}n_t \\
 &= \begin{cases} \left(3 + 2 \left\lceil \log_2 \left(\left\lceil \frac{L_f}{L_p} \right\rceil + 1 \right) \right\rceil \right) T_{pg}^{EPS}, & \text{when } L_f \leq 127L_p \\ \left(3 + 2 \left\lceil \sqrt{3906.25 + 2 \left\lceil \frac{L_f}{L_p} \right\rceil} - 57.5 \right\rceil \right) T_{pg}^{EPS}, & \text{when } 6,303L_p \geq L_f > 127L_p \\ \left(3 + 2 \left\lceil \frac{\left\lceil \frac{L_f}{L_p} \right\rceil + 2,785}{128} \right\rceil \right) T_{pg}^{EPS}, & \text{when } L_f > 6,303L_p \end{cases} \quad (2.12)
 \end{aligned}$$

Fig. 2.2 is a plot of D_t vs. L_f normalized to T_{pg}^{EPS} , the one-way forward delay from A to B . Fig. 2.3 is a plot of D_t vs. L_f for $RTT = 2T_{pg}^{EPS} = 91$ ms between MIT and Stanford University.

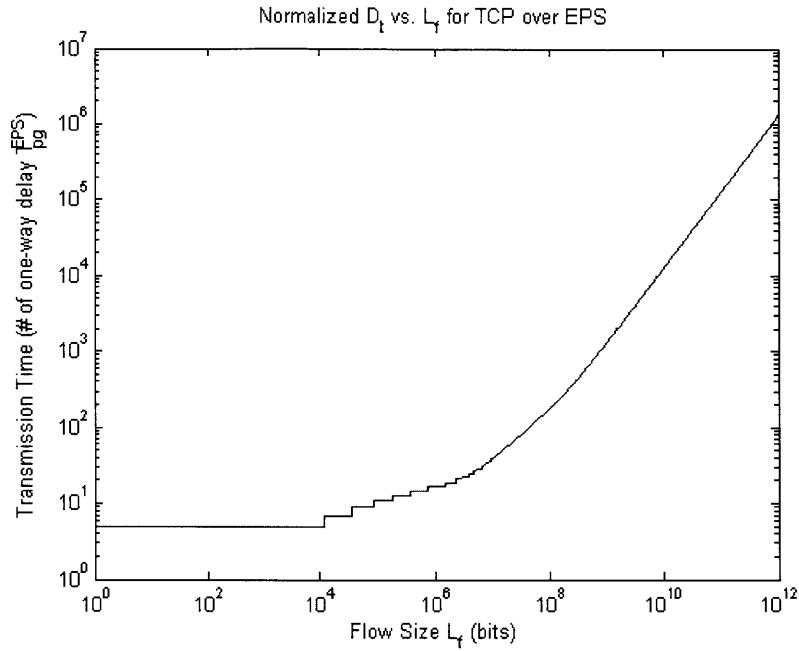


Figure 2.2: Normalized total delay D_t vs. L_f for standard TCP over EPS when the probability of error is zero.

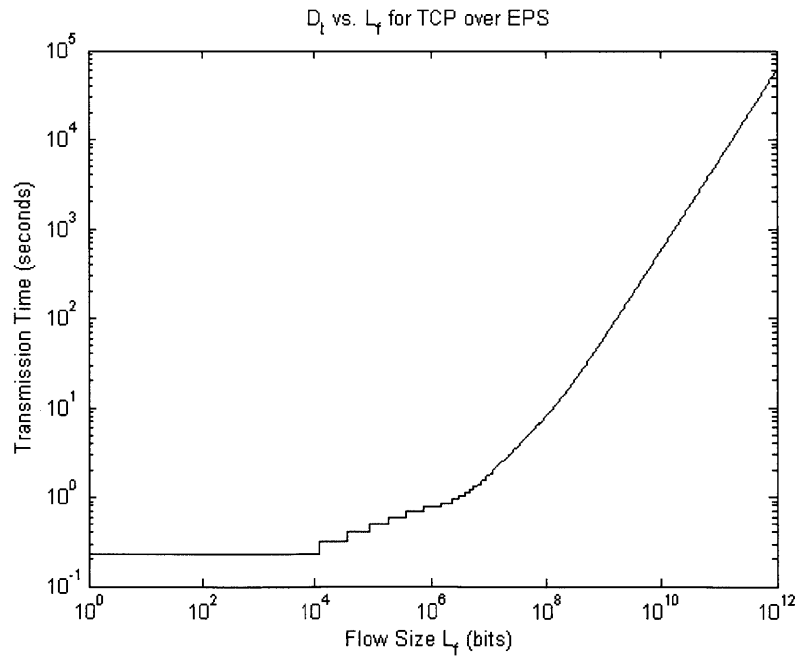


Figure 2.3: Total delay D_t vs. L_f for standard TCP over EPS when the probability of error is zero. The RTT is assumed to be 91 ms.

It can be seen from Fig. 2.2 and 2.3 that when the file size gets large enough (e.g. $> 10^8$ bits), the transmission time (or total delay) becomes linear with the file size. The normalized total delay for a file of size L_f bits is approximately given by $\frac{L_f}{128 \times L_p} = \frac{L_f}{1.536 \times 10^6}$ RTTs. This is because after a certain number of RTTs (71 RTTs for BER = 0), the TCP window size stays at its maximum value 128, and the network is at a constant rate of $\frac{128 \times L_p}{RTT}$. For example, for RTT = 91ms, a file of size $L_f = 10^{12}$ bits has total delay of $\frac{L_f \times RTT}{1.536 \times 10^6} = \frac{10^{12} \times 91 \times 10^{-3}}{1.536 \times 10^6} \approx 5.92 \times 10^4$ seconds.

2.2.2 Delay of TCP with BER > 0

When the probability of error in the packets sent is not negligible and/or when there is congestion in the network, the delay can be longer because of possible packet losses and window closing. In actual operation, TCP is sometimes in the Slow Start (exponential increase) phase and sometimes in the Congestion Avoidance (linear increase) phase. Linear window increase allows for fewer packets to be sent per unit time compared to exponential window increase. Thus, letting the window increase be linear yields a lower bound on TCP throughput, and letting the window increase be exponential yields an upper bound. If a packet loss occurs due to packet errors and/or network congestion, the window is halved. The upper and lower bound analysis correspond to that of the modified TCP in Section 6.3-6.4 of Eddy Lee's Ph.D. dissertation [27]. The Markov chains for linear window increase and exponential window increase are depicted below in Fig. 2.4 and 2.5. The transitions occur every RTT and the states represent a measure of the window size. For linear increase in Fig. 6.4, state n represents a window size of n , and the maximum number of states in the Markov chain is $n_{max} = M$ ($= 128$ in our discussions), where M is the maximum possible number of

packets in flight. Also, for exponential increase in Fig. 2.5, state n represents a window size of 2^{n-1} and the maximum number of states is $n_{max} = \log_2 M + 1$.

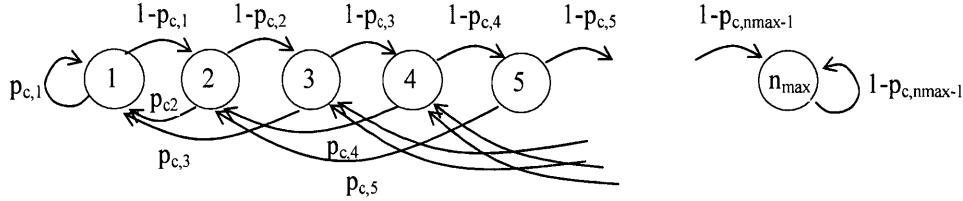


Figure 2.4: TCP linear increase (lower bound) Markov chain. State n represents a window size of n , and the maximum number of states in the Markov chain is $n_{max} = M$ [27].

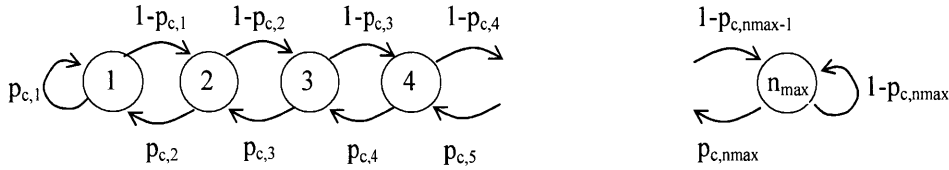


Figure 2.5 [27]: TCP exponential increase (upper bound) Markov chain. State n represents a window size of 2^{n-1} and the maximum number of states is $n_{max} = \log_2 M + 1$ [27].

The expected number of packets sent in K^{th} RTT and K RTTs can be found by (6.30-6.32) and (6.3) of [27], which depends on p_c , the probability that any given packet is lost due to packet errors and/or network congestion. The value of p_c not only depends on the bit error rate, but also the network capacity and buffer size [18]. In our analysis, we shall assume the network capacity is not limitation here and the buffer size is infinite; that is, p_c is only determined by the EPS bit error rate p_e , i.e.

$$p_c = 1 - (1 - p_e)^{Lp} \tag{2.13}$$

The following two pages of results up to (2.18) are captured from [27]. The expected number of packets sent in the m^{th} round-trip time is given by [27]

$$E[\text{number of packets sent in } m^{\text{th}} \text{ RTT}] = \begin{cases} \sum_{i=1}^{n_{\max}} i p_i(m) & \text{for linear increase(lower bound) of TCP} \\ \sum_{i=1}^{n_{\max}} 2^{i-1} p_i(m) & \text{for exponential increase (upper bound) of TCP} \end{cases}$$

where $p_i(m)$ is the probability of being in state i in the m^{th} RTT. The $p_i(m)$ can be obtained from the following evolution of probability distribution across the states:

$$\begin{aligned} \bar{p}(m) &= \bar{p}(m-1)P \\ &= \bar{p}(1)P^{(m-1)} \end{aligned} \quad (2.14)$$

where $\bar{p}(m)$ is a row vector of probabilities of being in the n_{\max} states in the m^{th} RTT, P is the probability transition matrix for the Markov chain, and $P^{(m-1)}$ is the matrix product of P with itself $(m-1)$ times and represents the transition matrix from any given RTT to $(m-1)$ RTTs later. TCP starts with an initial window size of 1. Thus, $\bar{p}(1)=[1 \ 0 \ 0 \ \dots \ 0]$.

Transition matrix of TCP linear increase Markov Chain is [27]

$$P = \begin{bmatrix} p_{c,1} & (1-p_{c,1}) & 0 & \dots & & & & & 0 \\ p_{c,2} & 0 & (1-p_{c,2}) & 0 & \dots & & & & \cdot \\ p_{c,3} & 0 & 0 & (1-p_{c,3}) & 0 & \dots & & & \cdot \\ 0 & p_{c,4} & 0 & 0 & \dots & & & & \cdot \\ \cdot & p_{c,5} & 0 & & & & & & \\ \cdot & 0 & p_{c,6} & & & & & & \\ & & p_{c,7} & & & & & & \\ & \cdot & 0 & \dots & 0 & & & & \\ & \cdot & & & 0 & p_{c,n_{\max}-2} & 0 & & 0 \\ & \cdot & & & 0 & p_{c,n_{\max}-1} & 0 & 0 & \dots & 0 & (1-p_{c,n_{\max}-1}) \\ 0 & \dots & & & & 0 & p_{c,n_{\max}} & 0 & \dots & 0 & (1-p_{c,n_{\max}}) \end{bmatrix} \quad (2.15)$$

where the $p_{c,n_{\max}}$ entry in the last row is in column $\frac{n_{\max}}{2}$, and

$$\begin{aligned}
p_{c,n} &= \Pr(\text{at least one of the packets sent in state } n \text{ is dropped due to congestion}) \\
&= 1 - \Pr(\text{none of the packets sent in stage } n \text{ is dropped due to congestion}) \\
&= \begin{cases} 1 - (1 - p_c)^n & \text{for linear window increase Markov chain} \\ 1 - (1 - p_c)^{2^{n-1}} & \text{for exponential window increase Markov chain} \end{cases}
\end{aligned} \tag{2.16}$$

Transition matrix of Modified TCP exponential increase Markov Chain is [27]

$$P = \begin{bmatrix} p_{c,1} & (1-p_{c,1}) & 0 & 0 & \dots & 0 \\ p_{c,2} & 0 & (1-p_{c,2}) & 0 & \dots & \cdot \\ 0 & p_{c,3} & 0 & (1-p_{c,3}) & 0 & \dots & \cdot \\ 0 & 0 & p_{c,4} & 0 & & & \cdot \\ \cdot & & 0 & \dots & & & 0 \\ \cdot & & & & & & 0 \\ \cdot & & & & 0 & (1-p_{c,n_{\max}-1}) & \\ 0 & 0 & \dots & & 0 & p_{c,n_{\max}} & (1-p_{c,n_{\max}}) \end{bmatrix} \tag{2.17}$$

The expected number of packets sent in K round-trip times is given by [27]

$$E[\text{number of packets sent in } K \text{ RTTs}] = \sum_{m=1}^K E[\text{number of packets sent in } m^{\text{th}} \text{ RTT}]$$

With packet size L_p , we can estimate the expected file size transmitted in K RTTs:

$$\begin{aligned}
L_f &= L_p \times E[\text{number of packets sent in } K \text{ RTTs}] \\
&= \sum_{m=1}^K L_p \times E[\text{number of packets sent in } m^{\text{th}} \text{ RTT}] \\
&= \begin{cases} L_p \sum_{m=1}^K \sum_{i=1}^{n_{max}} i p_i(m) & \text{for linear increase (lower bound) of TCP} \\ L_p \sum_{m=1}^K \sum_{i=1}^{n_{max}} 2^{i-1} p_i(m) & \text{for exponential increase (upper bound) of TCP} \end{cases} \quad (2.18)
\end{aligned}$$

We can express the above expression in matrix form:

$$L_f = L_p \sum_{m=1}^K \vec{N} \overline{p(m)}^T \quad (2.19)$$

where \vec{N} is the state row vector with each element being the number of packets corresponding to that state number. That is, for linear increase, $N(i) = i$, and for exponential increase $N(i) = 2^{i-1}$. $\overline{p(m)}^T$ is the transpose of $\overline{p(m)}$. Furthermore,

$$L_f = L_p \sum_{m=1}^K \vec{N} (\overline{p(1)} P^{m-1})^T = L_p \sum_{m=1}^K \vec{N} (P^{m-1})^T \overline{p(1)}^T \quad (2.20)$$

Then K is the smallest integer that gives

$$\sum_{m=1}^K \vec{N} (P^{m-1})^T \overline{p(1)}^T \geq \frac{L_f}{L_p} \quad (2.21)$$

By using the above approach, we can also simulate transient behaviors of standard TCP by using the Markov chain model. The only differences lie in the state row vector \vec{N} and the number of states.

Fig. 2.6 (a) - 2.10(c) show the results for the expected number of packets sent in the K^{th} RTT, K RTTs, and delay vs. file size for different bit error rates: 0, 10^{-10} , 10^{-8} , 10^{-6} , and 10^{-5} .

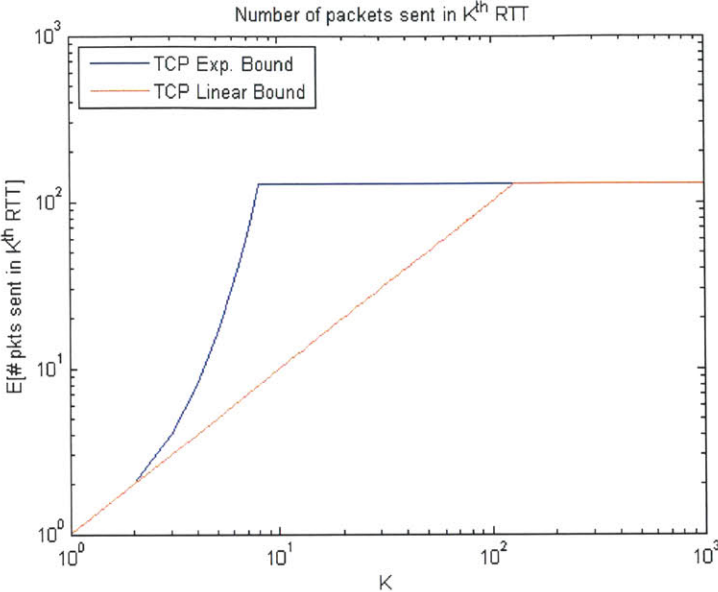


Figure 2.6 (a): Expected number of packets in the K^{th} RTT when $p_e = 0$. The maximum window size is assumed to be 128.

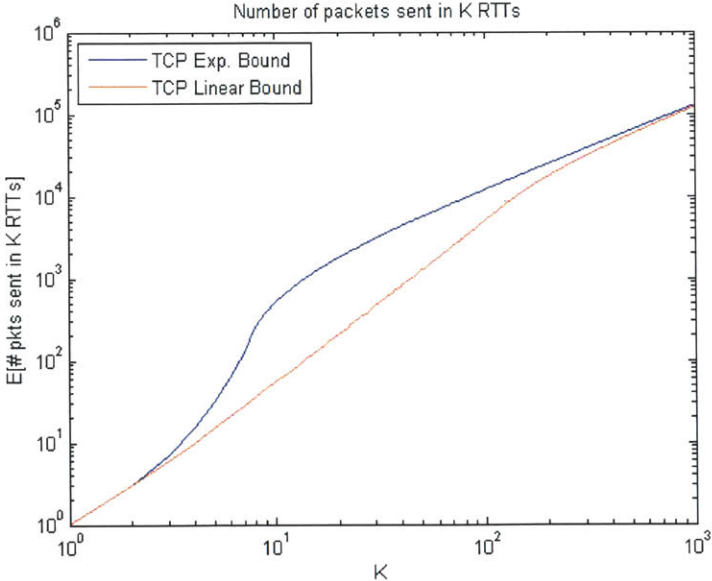


Figure 2.6 (b): Expected number of packets in K RTTs when $p_e = 0$. The maximum window size is assumed to be 128.

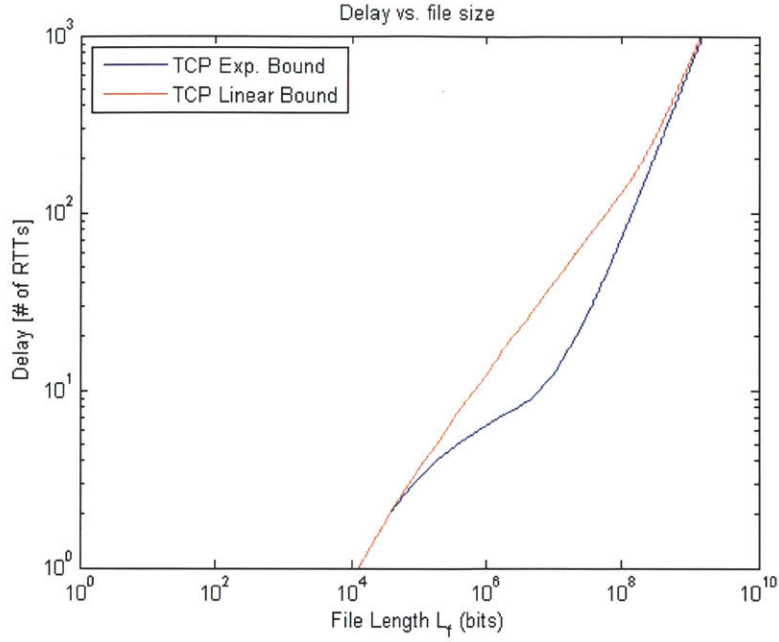


Figure 2.6 (c): Delay (number of RTTs) vs. file size when $p_e = 0$. The maximum window size is assumed to be 128.

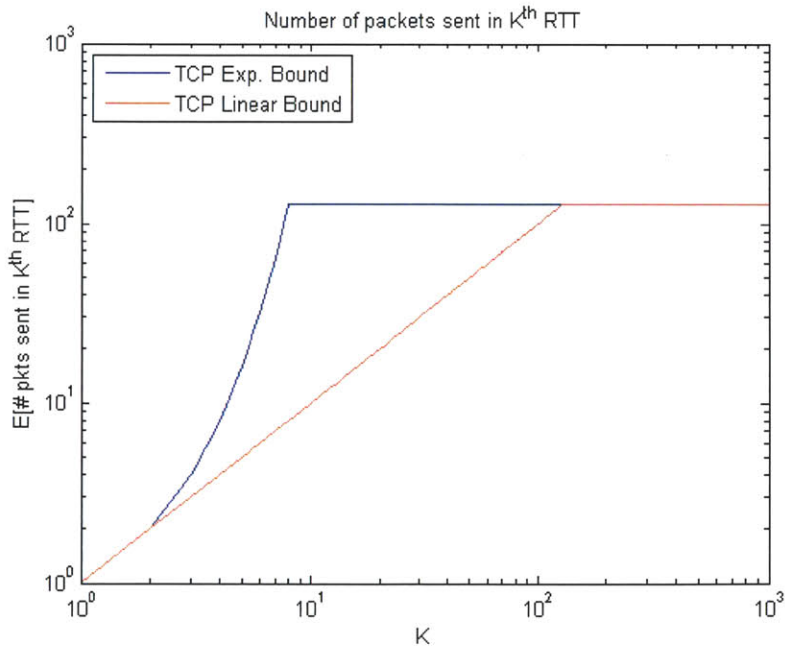


Figure 2.7 (a): Expected number of packets in the K^{th} RTT when $p_e = 10^{-10}$. The maximum window size is assumed to be 128.

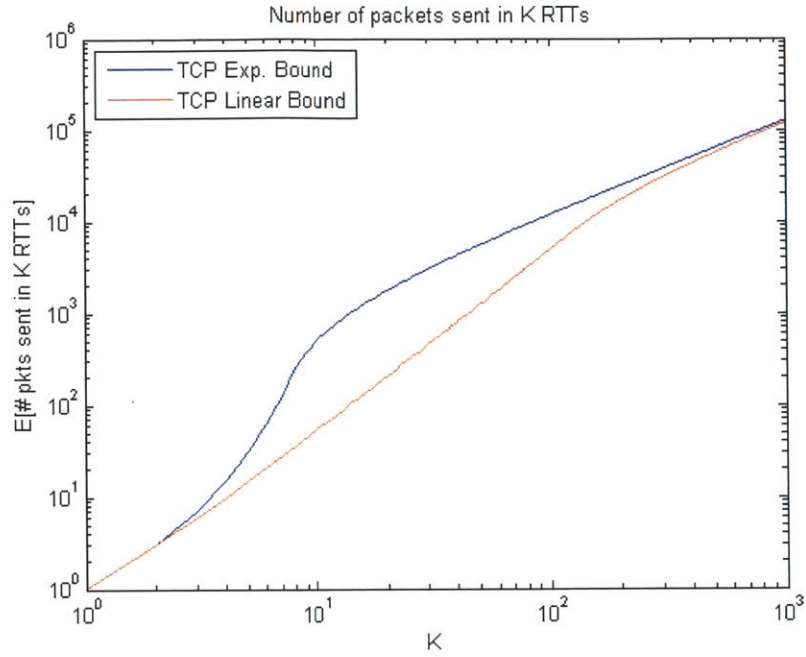


Figure 2.7 (b): Expected number of packets in K RTTs when $p_e = 10^{-10}$. The maximum window size is assumed to be 128.

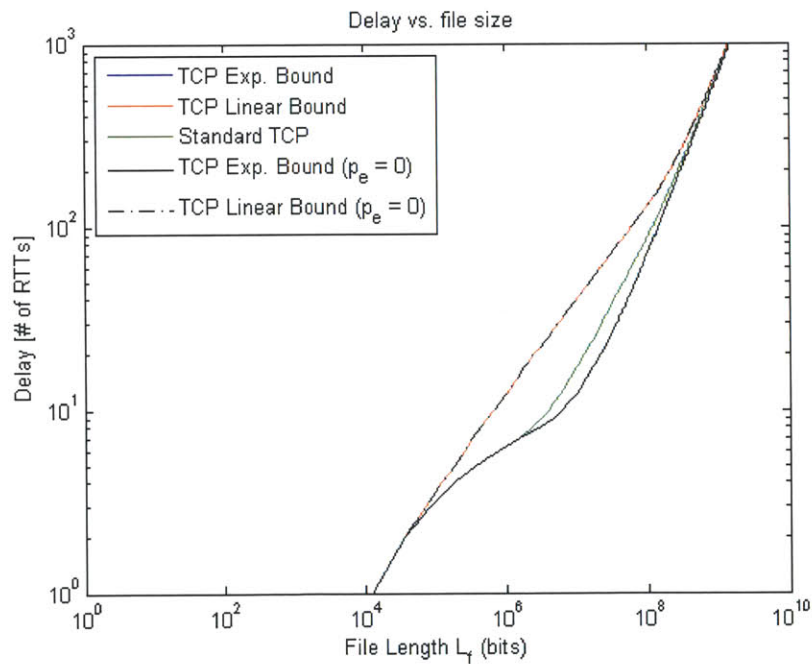


Figure 2.7 (c): Delay (number of RTTs) vs. file size when $p_e = 10^{-10}$. The maximum window size is assumed to be 128.

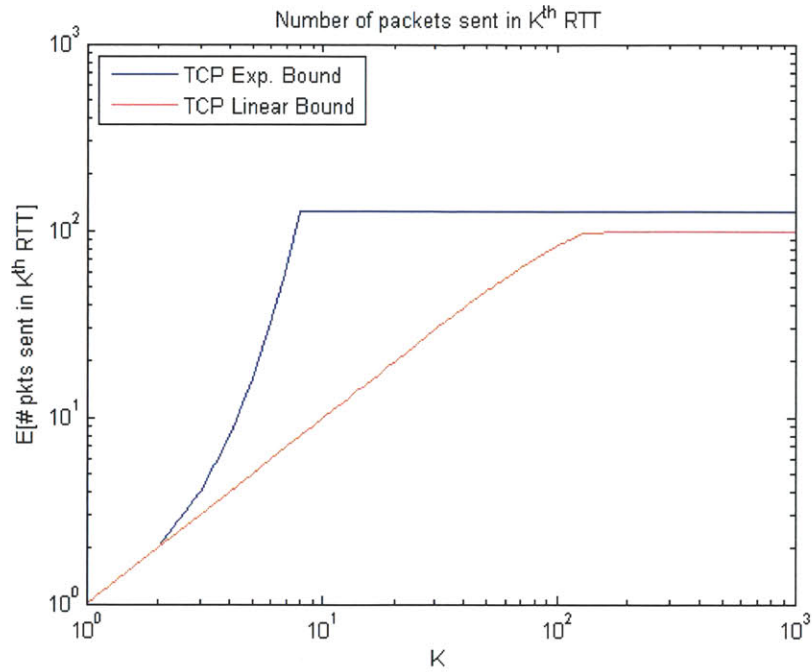


Figure 2.8 (a): Expected number of packets in the K^{th} RTT when $p_e = 10^{-8}$. The maximum window size is assumed to be 128.

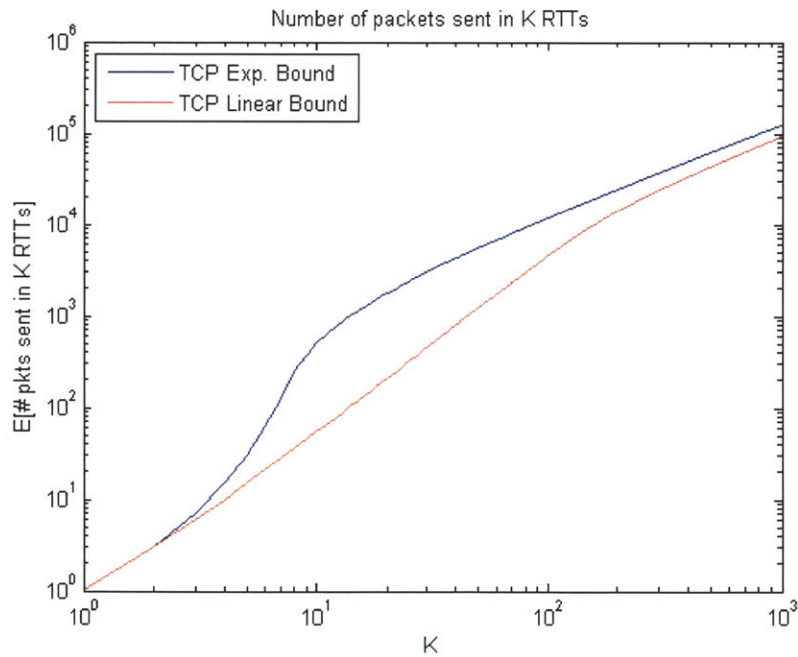


Figure 2.8 (b): Expected number of packets in K RTTs when $p_e = 10^{-8}$. The maximum window size is assumed to be 128.

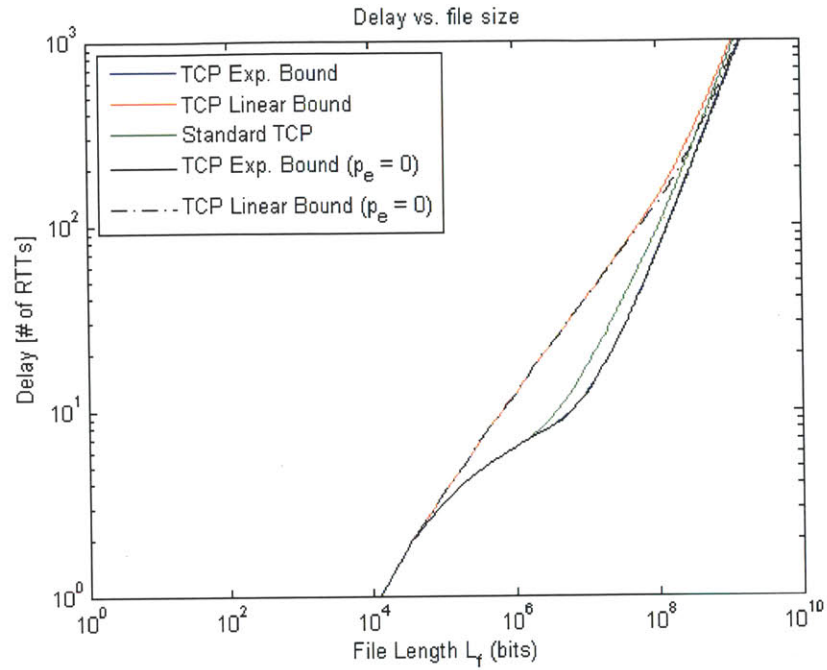


Figure 2.8 (c): Delay (number of RTTs) vs. file size when $p_e = 10^{-8}$. The maximum window size is assumed to be 128.

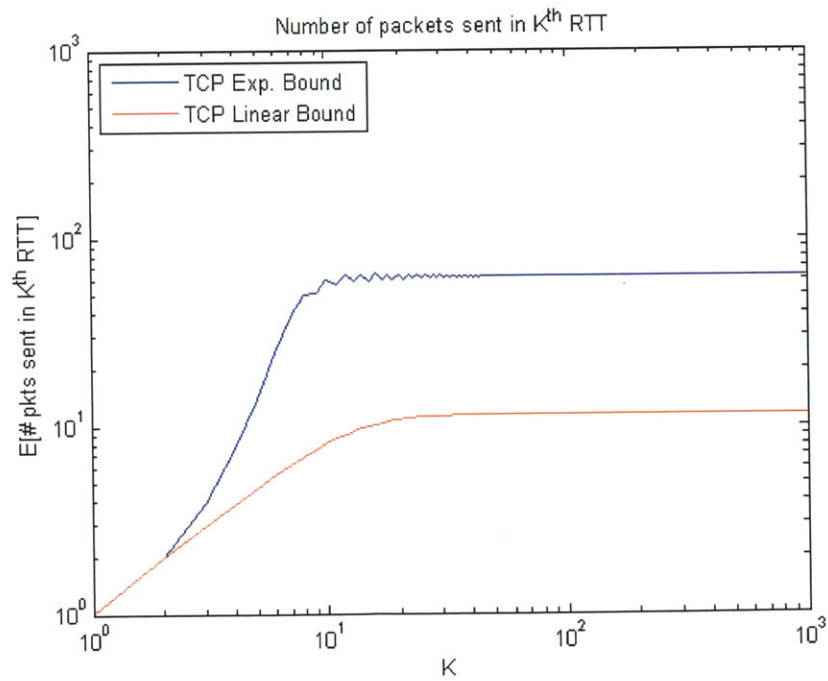


Figure 2.9 (a): Expected number of packets in the K^{th} RTT when $p_e = 10^{-6}$. The maximum window size is assumed to be 128.

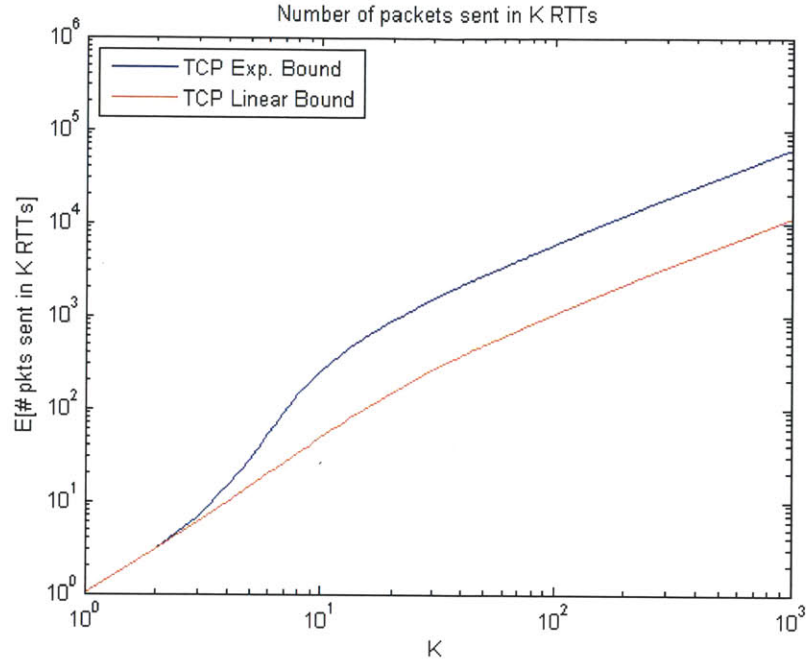


Figure 2.9 (b): Expected number of packets in K RTTs when $p_e = 10^{-6}$. The maximum window size is assumed to be 128.

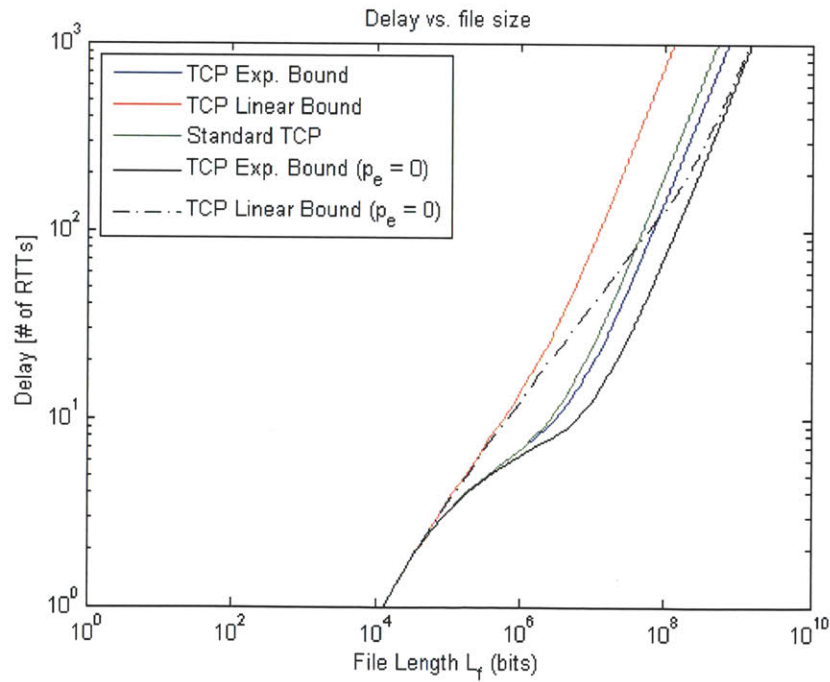


Figure 2.9 (c): Delay (number of RTTs) vs. file size when $p_e = 10^{-6}$. The maximum window size is assumed to be 128.

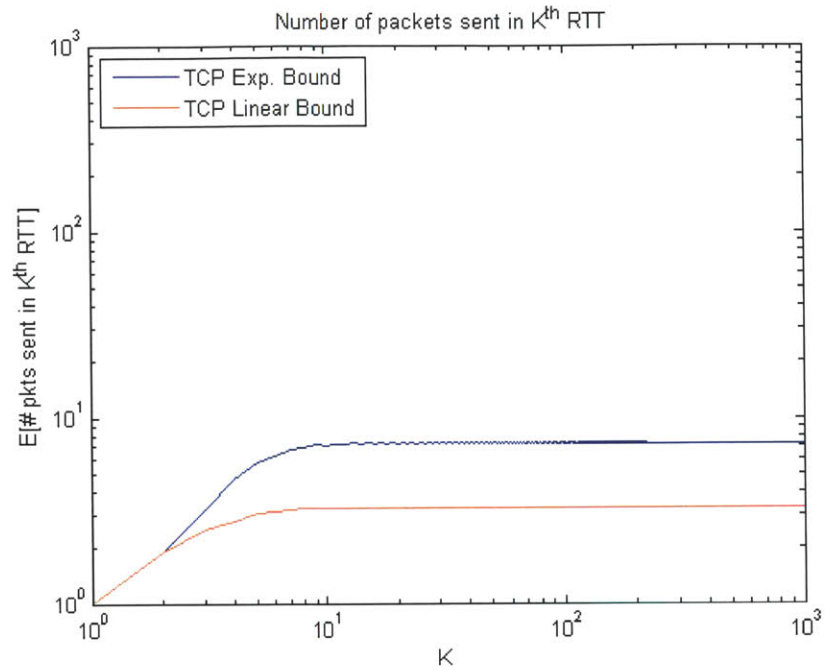


Figure 2.10 (a): Expected number of packets in the K^{th} RTT when $p_e = 10^{-5}$. The maximum window size is assumed to be 128.

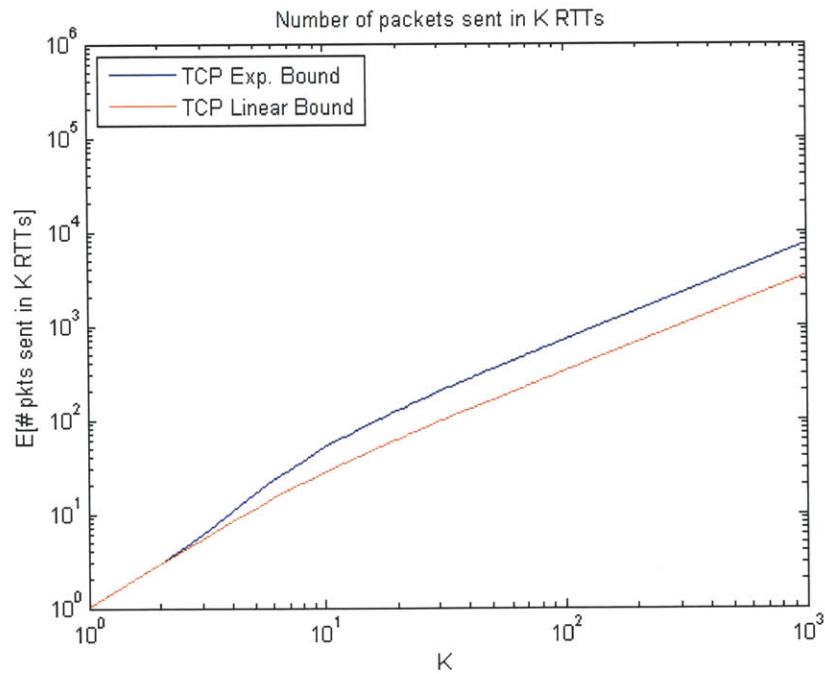


Figure 2.10 (b): Expected number of packets in K RTTs when $p_e = 10^{-5}$. The maximum window size is assumed to be 128.

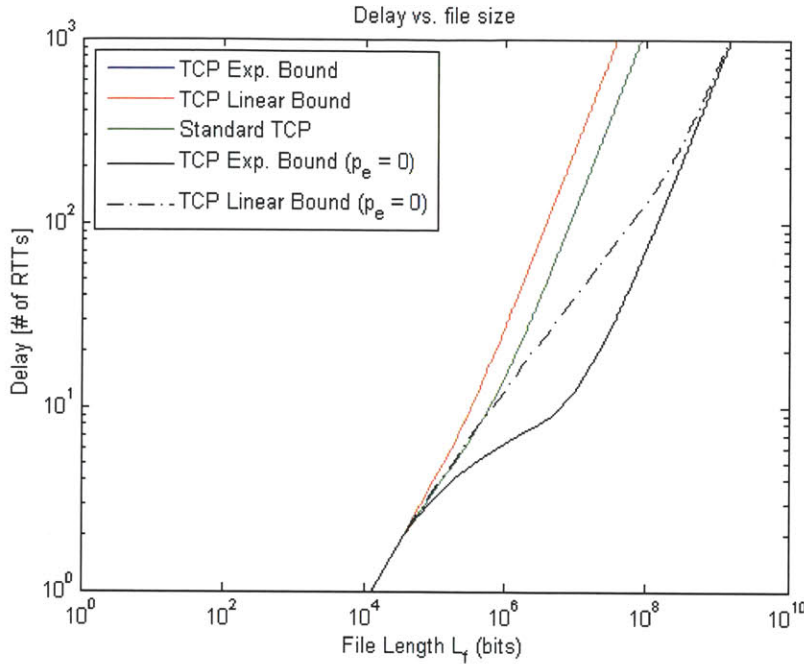


Figure 2.10 (c): Delay (number of RTTs) vs. file size when $p_e = 10^{-5}$. The maximum window size is assumed to be 128.

It can be seen from Fig. 2.6 (a) to Fig. 2.10 (c) that, as the bit error rate increases, the expected number of packets in the K^{th} RTT decreases, the total number of packets sent in K RTTs decreases, and the expected delay increases for both the upper bound and lower bound. All of these are expected. This is because: when the bit error rate becomes larger, more packets are expected to be in error, and more packets are expected to be lost. As a result, more often is the window size halved, which leads to a smaller average window size and less number of packets transmitted in a given amount of time. Therefore, a longer delay is experienced.

It is also interesting to note that as the BER increases, performance of the standard TCP gets closer to the exponential bound. This is expected, as with higher and higher BER, TCP window closing occurs more often and it tends to stay in the *Slow Start* phase most of the times, which has exponential increase behavior similar to that of exponential bound.

Furthermore, although we did not plot here, it is also important to note that when the maximum window size is larger than 128 for non-standard TCP, the expected number of packets sent in the K^{th} RTT and in the K RTTs also increase. The amount of increase for the upper and lower bound depends on the maximum value of the window size and the bit error rate. In general, the larger the bit error rate is, the less obvious the increase is. The reason is that as the bit error rate becomes high (e.g. 10^{-5}), most of the times the window size is kept at a small value much less than the maximum window size. The maximum window size itself does not contribute much to the average window size or the delay.

2.3 Summary of Chapter 2

In this chapter, we analyzed the total delay when standard TCP over EPS is used to transmit a file for a channel with bit errors and assuming the network does not have congestion packet loss, which is only a very crude approximation of the Internet. When the BER and congestion level are zero, the TCP window size can increase up to the maximum value (e.g. 128 for the standard TCP) after a few round trips, and the total delay is approximately proportional to the RTT, which includes the processing delay, transmission delay, propagation delay and queuing delay (we will see these delays separately in Chapter 7 when comparing EPS with OFS). When the BER is not zero, the average window size decreases with increasing BER, and it takes more round trip times to transmit the same file and hence incurs a longer delay.

In Chapter 3-6, we shall propose four types of OFS protocols and compare their performances in terms of delay in Chapter 7.

Chapter 3

Protocol with Error Detection by Backward Comparison (EDBC)

In today's high speed optical links, the raw Physical Layer transmission delivers a bit error rate that is too high for any higher layer protocol to function correctly. Therefore, forward error correction is built into the Physical Layer transmission and reception hardware to reduce this BER significantly, e.g. from $10^{-6} \sim 10^{-10}$ to 10^{-12} or lower.

As discussed in Chapter 1, to ensure data transfer reliability in case there are residual errors after FEC at the Physical Layer, error detection and error recovery via retransmissions are usually done at the Transport Layer. Error detection can be implemented either at the sender with comparison of a backward flow (EDBC) with the original file or at the receiver with an

error detection code (e.g. CRC code). In this Chapter, we focus on the former algorithm and analyze its performance in terms of delay. We will discuss the latter algorithm in Chapter 4.

EDBC works by sending the received data back to the sender using a backward path and comparing it with the original data, e.g. using XOR operations. Should any error be found in the comparison process, the sender notifies the receiver via EPS that the flow transmitted was erroneous, and the whole flow is retransmitted via OFS after rescheduling. Otherwise, the sender sends an ACK message via EPS to the receiver to confirm that the received data was error-free.

Section 3.1 gives a detailed description of the EDBC algorithm and Section 3.2 discusses the delay performance with zero and non-zero BER.

3.1 EDBC Algorithm Description and Flowchart

Assumptions and Definitions:

- Source A wants to send a file to destination B via OFS. At the sender side, files are the user data passed from the application layer to the transport layer for transmission. They are added with necessary header information, including start-of-flow, length-of-flow and other necessary information to form flows that are actually transmitted via OFS. At the receiver side, flows need to be decoded with headers removed to recover the "files", which may or may not be the same with the original files sent out by A , because of potential bit errors on the links.
- The forward path refers to the OFS wavelength channel from A to B , and backward path the OFS channel from B to A .

- Whenever A establishes/closes a TCP connection with its scheduler, A 's scheduler then establishes/closes a TCP connection with B 's scheduler. The same happens for B 's scheduler. (For simplicity, this is not explicitly shown in the flowchart)
- Whenever EPS is used for data communications, TCP with end-to-end reliability is assumed to be employed⁴. If a TCP connection closes at any time, A and/or B close all their connections and go back to the start of the algorithm. (We will draw a separate flowchart for this case.)
- There is no flow blocking or dropping for all OFS protocols we consider in this thesis. This assumption corresponds to the assumption of infinite router buffers for EPS. In general OFS can have protocols that have blocking to make resource utilization more efficient or less delays for the served flows.

Overall Algorithm Description:

For easy analysis, we divide the overall algorithm into three phases: Preparation, transmission and conclusion. In the preparation phase, all TCP connections are set up and the forward path and reverse (or backward) path are reserved. In the transmission phase, flow data is transmitted on the forward path and backward path (after set up) with data received from the backward path compared with the original file at the sender. In the conclusion phase, the sender decides whether there is any error in the data it received, from which it concludes whether any error was in the data it previously transmitted. It then retransmits the whole flow if any error is found. More specifically, the detailed algorithm is described below:

⁴ Although there are other protocols that can be used over EPS, we do not consider them in this work.

Phase I: Preparation

1. TCP connections between A and A 's scheduler, between A 's scheduler and B 's scheduler, between B 's scheduler and B , and between B and A are established in sequence.
 - 1a. A first establishes a TCP connection with A 's (ingress) scheduler at the network node connecting A 's MAN and the WAN. A then sends an informative message to its scheduler, saying that A wants to send data to B via OFS.
 - 1b. A 's scheduler then establishes a TCP connection with B 's (egress) scheduler at the node connecting the WAN and B 's MAN. A 's scheduler then sends a message to B 's scheduler, saying that A wants to send data to B via OFS.
 - 1c. B 's scheduler establishes a TCP connection with B and sends the same message in the same way as above.
 - 1d. B then establishes a TCP connection with A .
2. After all connections are established, A requests the scheduler to set up a forward path by telling the scheduler the length of the flow that needs to be transmitted and other necessary information.
3. The scheduler then reserves a wavelength channel from A to B , whenever available, for the duration of the flow (possibly with some guard time), and tells A and B the reserved OFS channel wavelength, planned start time and duration of the channel reservation. Assume all exchanges of messages in the scheduling process happen on the OFS control

plane, which can be EPS. The detailed scheduling algorithm is described on page 142 in Guy Weichenberg's PhD thesis [1].

Phase II: Transmission

4. At the start time specified by the scheduler, A starts transmitting the encoded flow to B via the reserved OFS channel, i.e. the forward path previously set up.
5. Upon decoding the data received from A , B recognizes the start-of-flow, and immediately requests the scheduler to set up a backward path from B to A . At the same time, B continues decoding the received flow, and stores a copy of the file after decoding. If B does not see start-of-flow within the OFS propagation delay plus some guard time after the start time, B timeouts and closes its TCP connections with A and scheduler. When A detects that B closes its TCP connection, A closes the TCP connection with its scheduler. At the same time, B clears all data it previously received from A . Go back to Step 1.
6. The scheduler then reserves a wavelength channel from B to A using the same algorithm as in Step 3, and tells B and A the reserved channel, planned start time and duration of the channel reservation.
7. At the specified start time for the backward path, B encodes the file again exactly as what A does (i.e. with start-of-flow, length-of-flow and other information included in the header) and starts sending back the encoded data to A on the backward path.
8. A decodes flow data received from B on the backward path, and compares bit by bit with the original file at A .

If A does not see start-of-flow within the OFS propagation delay plus some guard time after the start time, A timeouts and closes TCP connections with B and A 's scheduler. When B detects that A closes its TCP connection, B also closes the TCP connection with its scheduler. At the same time, B clears all data it previously received from A . Go back to Step 1.

Phase III: Conclusion

9. When A finishes receiving the flow (A knows the start-of-flow and length-of-flow, and hence when it is finished), it transmits the ACK/NACK message via EPS as below:

9a. If any difference is found, A sends a NACK message to B via EPS, saying that the OFS data previously received by B are probably erroneous. Meanwhile, A closes the TCP connection with A 's scheduler, which then closes TCP connection with B 's scheduler.

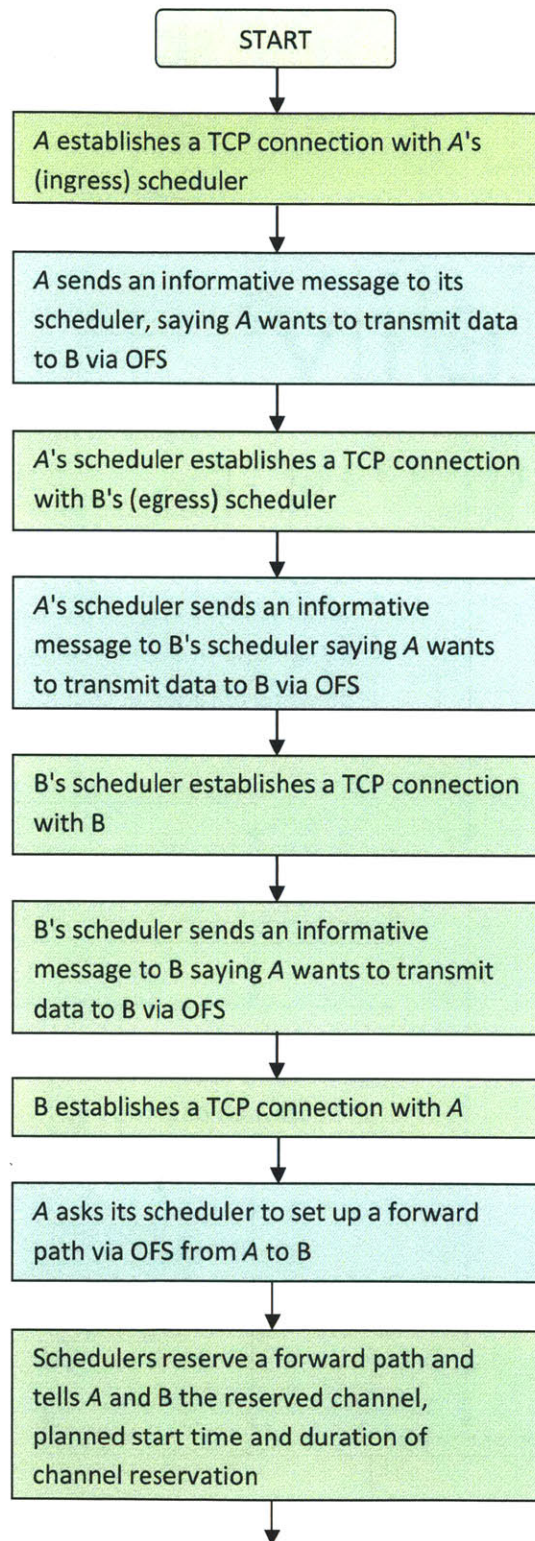
Upon receiving NACK, B closes TCP connections with A and B 's scheduler, and discards the file it previously stored. Go back to Step 1.

9b. If no bit difference is found for the whole flow, A sends an ACK message to B via EPS to confirm that the OFS data previously received by B are correct. Meanwhile, A closes TCP connection with A 's scheduler. A 's scheduler then closes TCP connection with B 's scheduler.

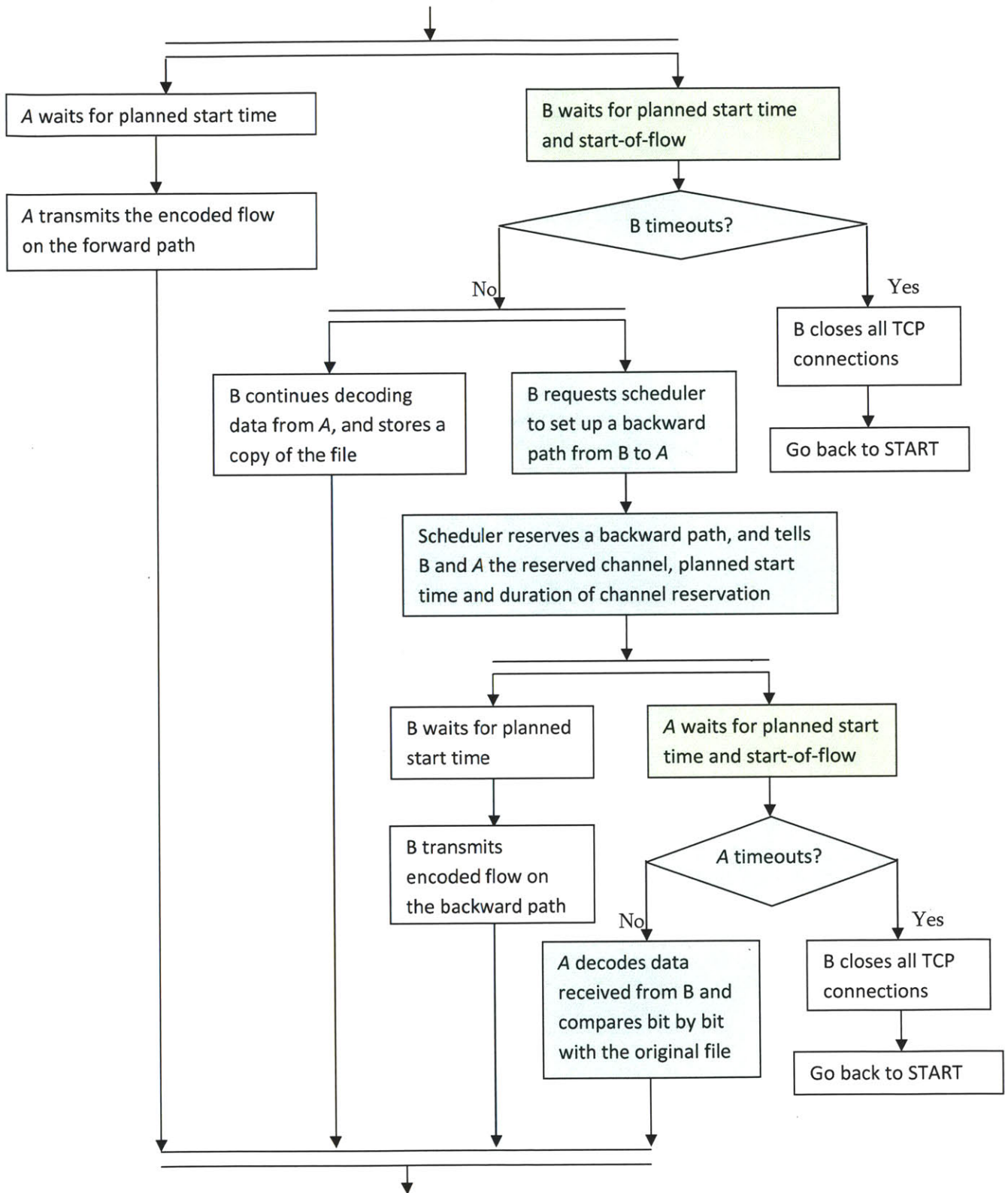
Upon receiving ACK, B sends back a handshake message via EPS to A , saying that B received the ACK message. Meanwhile B closes TCP connection with the scheduler, and passes the previously stored file to the application layer. Upon receiving the handshake message from B , A closes the TCP connection with B . Algorithm terminates.

Flowchart:

Phase I: Preparation



Phase II: Transmission



Phase III: Conclusion

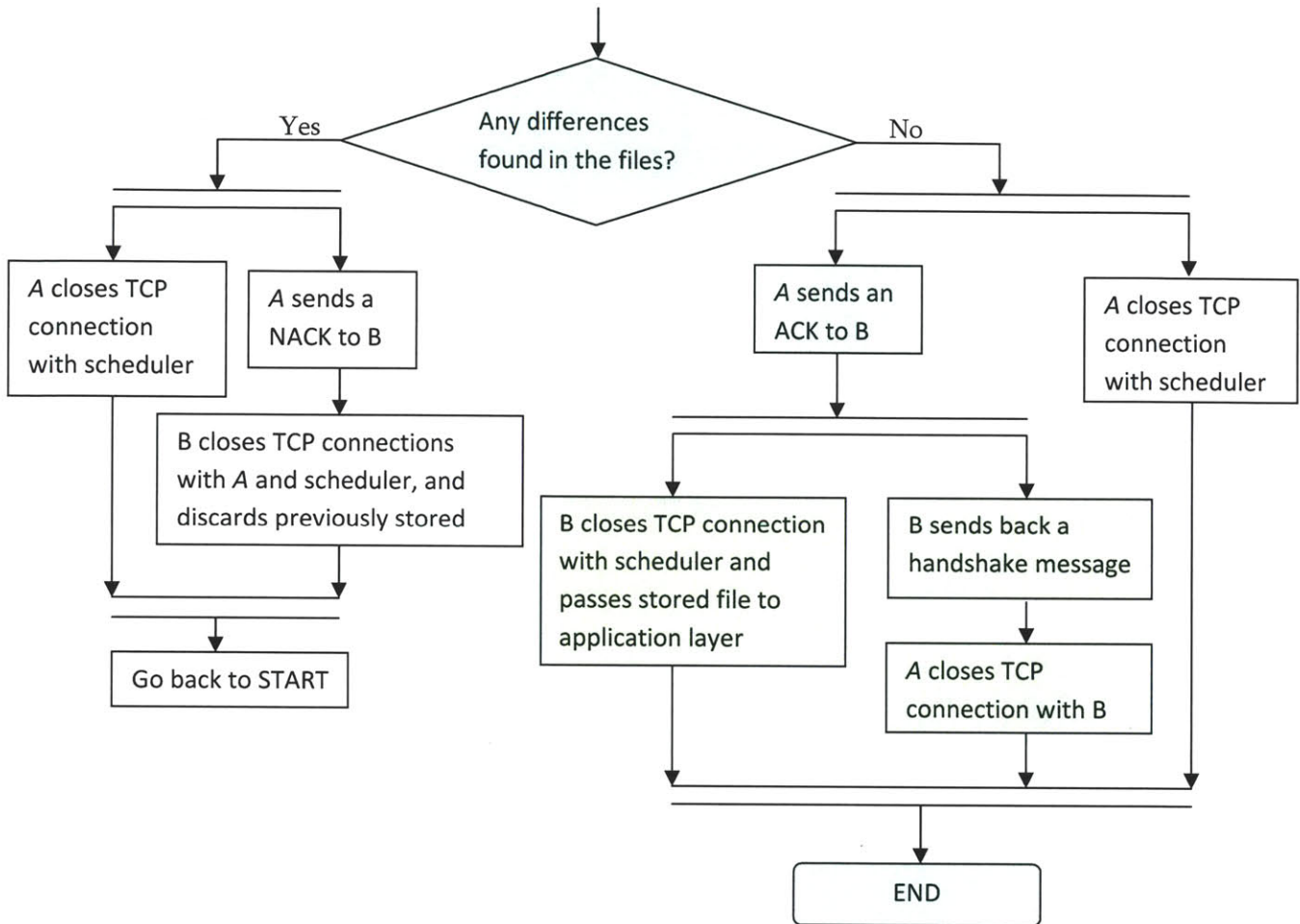
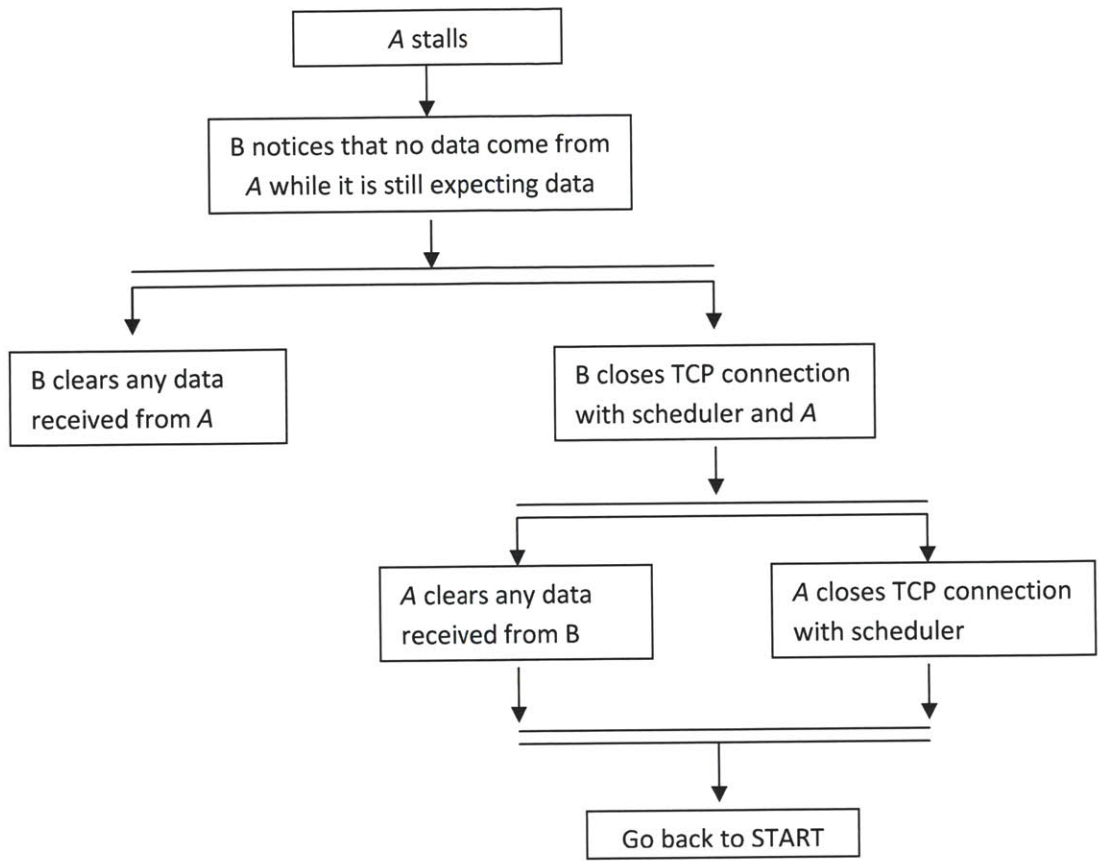
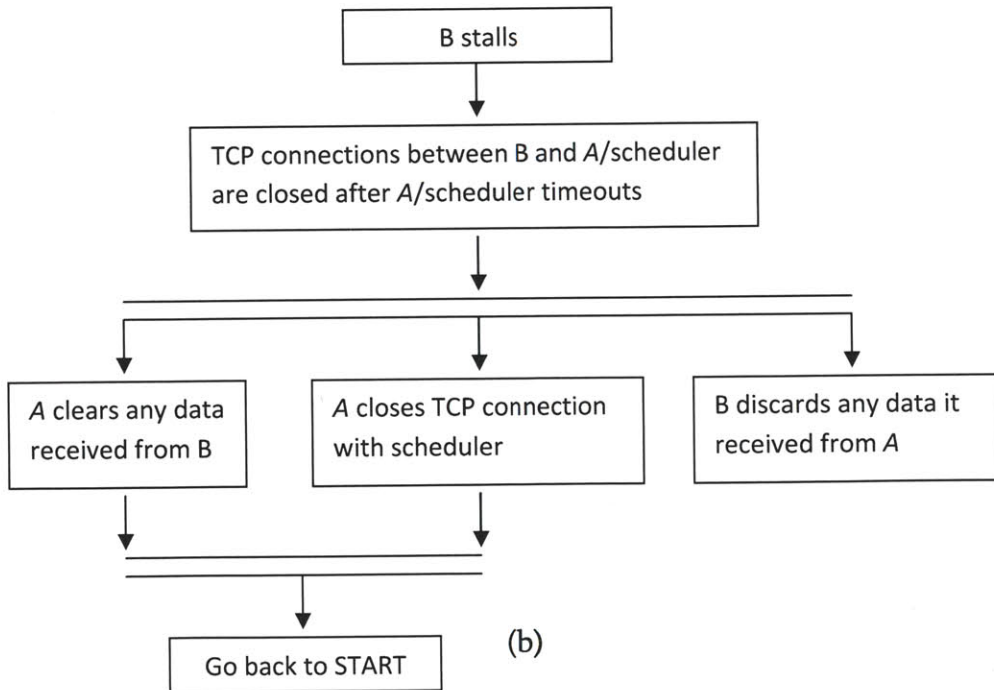


Figure 3.1: EDBC flowchart with phase I, II and III shown separately. The green boxes show the "green path" when there is no transmission error of any kind.

In the case where *A* and/or *B* stall in any of the steps above, all TCP connections are closed and data are cleared at *A* and *B*. The system then goes back to START. This is illustrated in the following flowchart (Fig. 3.2).



(a)



(b)

Figure 3.2: Flowchart for cases (a) when *A* stalls (b) when *B* stalls

3.2 Delay Analysis of EDBC Algorithm

We will analyze first the delay of EDBC in the simple case when BER (which we denote as p_e) is 0, and then in the case when BER $p_e > 0$.

3.2.1 Delay of EDBC with BER = 0

For a flow of length L_f bits and OFS link rate R_{OFS} bps, when there is no transmission error of any kind (i.e. perfect sender, channel and receiver), the timeline follows the "green path" in Fig. 3.1. The overall delay of the transmission is then

$$\begin{aligned}
 D_f = & D_{TCP} + (D_{pg}^{sch} + D_q) + (T_{pg} + T_{pc}^{OFS}) + (D_{pg}^{sch} + D_q) \\
 & + \left(T_{pg}^{OFS} + \frac{L_f}{R_{OFS}} + T_{pc}^{OFS} \right) + \left(T_q + T_{pg} + \frac{L_p}{R_{EPS}} + T_{pc}^{EPS} \right) + \tau
 \end{aligned} \tag{3.1}$$

where

- D_{TCP} is the time taken to establish all TCP connections between the sender, receiver and scheduler before scheduling processes can begin (please see a detailed explanation about this term below),
- D_{pg}^{sch} is the delay (including the transmission delay, propagation delay, queuing delay at the router, and processing delay) in the process of scheduling (at least one EPS RTT between sender and receiver),
- D_q is the queuing delay for an open channel as determined by the reservation scheduler. It is the time from the moment the scheduler receives the request for channel reservation until the moment the flow transmission starts on the reserved channel,
- T_{pg} (or interchangeably T_{pg}^{OFS}) is the propagation delay between the sender and receiver defined by the ratio of the fiber distance to the speed of light (note that this is different from T_{pg}^{EPS} which is defined as the one-way EPS delay that consists of transmission delay, propagation delay, queuing delay and processing delay),

- T_{pc}^{OFS} and T_{pc}^{EPS} are the OFS and EPS processing delays at the sender and/or receiver added to the overall delay respectively,
- T_q is the EPS queuing delay at the router,
- L_p is the size of the ACK/NACK message and other EPS packets,
- R_{EPS} is the EPS link rate, and
- τ is the time taken for the receiver to pass the decoded data to the application layer.

Here is an explanation of the delay expression in (3.1):

- D_{TCP} is the time taken to establish all TCP connections before scheduling processes can begin. Firstly, D_{TCP} includes the time to sequentially establish TCP connections between A and A 's (ingress) scheduler, between A 's scheduler and B 's (egress) scheduler, between B 's scheduler and B , and between B and A , which takes approximately $3T_{pg}^{EPS} + 3T_{pg}^{EPS} = 6T_{pg}^{EPS}$. Here we assumed that the propagation delay between A and B via the schedulers is roughly the same with that directly between A and B . Secondly, D_{TCP} also includes the time to send informative messages following each TCP connection establishment, except for that between B and A . Therefore, it takes in total $8T_{pg}^{EPS}$ to establish TCP connections with necessary communications.
- $(D_{pg}^{sch} + D_q)$ is the overall scheduling delay of the forward or backward path, which consists of scheduling propagation delay D_{pg}^{sch} and OFS resources queuing delay D_q ,
- $(T_{pg}^{OFS} + T_{pc}^{OFS})$ is the delay from the start time of the forward flow to the time the receiver sees start-of-flow,
- $(T_{pg}^{OFS} + \frac{L_f}{R_{OFS}} + T_{pc}^{OFS})$ is the delay from the start time of the backward flow to the time the whole flow is fully transmitted on the backward path, and
- $(T_q + T_{pg} + \frac{L_p}{R_{EPS}} + T_{pc}^{EPS})$ is the time it takes to process (at both sender and receiver) and transmit the EPS ACK/NACK packet. Remember that in Chapter 2, $T_{pg}^{EPS} = T_q + T_{pg} + \frac{L_p}{R_{EPS}} + T_{pc}^{EPS}$.

Given a particular flow, certain OFS and EPS link rates and a source-destination pair, we shall consider D_{TCP} , $\frac{L_f}{R_{OFS}}$, $\frac{L_p}{R_{EPS}}$, T_{pg}^{OFS} , T_{pg}^{EPS} and τ , as some known constants. T_{pc}^{OFS} and T_{pc}^{EPS} are not considered in this work. τ is considered negligible compared to other delay terms. D_{pg}^{sch} and D_q depend on the scheduling algorithms used and traffic condition. We shall adapt Guy Weichenberg's scheduling algorithm [1], and treat D_q as a random variable that depends on traffic statistics and the loading factor.

Also, in a typical optical network, $T_{pg}^{EPS} \gg \frac{L_p}{R_{EPS}}$ (refer to Table 3.1 for an example). We can neglect $\frac{L_p}{R_{EPS}}$ and only count T_{pg}^{EPS} . We can then approximate D_t as

$$\begin{aligned}
 D_t &\approx D_{TCP} + (D_{pg}^{sch} + D_q) + T_{pg} + (D_{pg}^{sch} + D_q) + \left(T_{pg}^{OFS} + \frac{L_f}{R_{OFS}}\right) + T_{pg}^{EPS} \\
 &\approx D_{TCP} + 2(D_{pg}^{sch} + D_q) + \left(2T_{pg}^{OFS} + \frac{L_f}{R_{OFS}}\right) + T_{pg}^{EPS}
 \end{aligned} \tag{3.2}$$

The following two pages of results until (3.12) are captured from [1]. Here the total queuing delay D_q is approximated by expression (4.10) in [1]:

$$D_q \approx \left[\bar{Y} + \frac{\lambda_c \bar{X}^2}{2(1 - \lambda_c \bar{X})} \right] \left[\frac{\widehat{W}_{M, w_m}(\lambda_m \bar{X}, \rho_X)}{\widehat{W}_{M, 1}(\lambda_c \bar{X}, \rho_X)} \right] \tag{3.3}$$

where \bar{Y} is the average time spent at the head of the primary queue reserving resources in both the source and destination distribution networks (DNs), X is the service time of a primary request, λ_c is the arrival rate of OFS flows for a source-destination MAN pair, normalized by the number of provisioned wavelength channels w_m , λ_m is the arrival rate of OFS flows for a source-destination MAN pair, $\widehat{W}_{M, k}(\cdot)$ is the average queuing delay of M/M/k queuing system as a function of offered load ρ_o , and

$$\widehat{W}_{M,k}(k\rho_o, p_L) = \frac{\bar{L}P_Q}{k(1-\rho_o)} \quad (3.4)$$

where L is the flow transmission time in seconds (note that L_f is the flow length in bits), and P_Q is given by the Erlang C formula:

$$P_Q = \frac{(k\rho_o)^k}{k!(1-\rho_o)} \left[\sum_{i=0}^{k-1} \frac{(k\rho_o)^i}{i!} + \frac{(k\rho_o)^k}{k!(1-\rho_o)} \right]^{-1} \quad (3.5)$$

It can be seen from (3.4) and (3.5) that the average queuing delay $\widehat{W}_{M,k}(k\rho_o, p_L)$ only depends on the first moment of the distribution p_L of the flow transmission time L .

For secondary requests, we will consider the (worse) case where they are sent sequentially after a primary request reaches the head of its queue, instead of being sent simultaneously [2]. This will give the upper bound for \bar{Y} and \bar{Y}^2 , which we denote as \bar{Y}_u and \bar{Y}_u^2 respectively. Then from (4.4) and (4.5) of [1],

$$\bar{X} \approx \bar{L} + \bar{Y}_u = \bar{L} + \bar{Z}_s + \bar{Z}_d \quad (3.6)$$

$$\bar{X}^2 \approx \overline{(L + Y_u)^2} = \bar{L}^2 + \bar{Z}_s^2 + \bar{Z}_d^2 + 2\bar{L} \cdot \bar{Z}_s + 2\bar{L} \cdot \bar{Z}_d + 2\bar{Z}_s \cdot \bar{Z}_d \quad (3.7)$$

where Z_s is the time spent by a secondary source request in its queue prior to reaching the head of the queue; and Z_d is the time spent by a secondary destination request in its queue prior to reaching the head of its queue. The first, second and third moments for Z_d are given by [1]

$$\bar{Z}_d = \frac{f\lambda_c\bar{L}^2}{2(\tilde{n}_a - f\lambda_c\bar{L})} \quad (3.8)$$

$$\bar{Z}_d^2 = \frac{(f\lambda_c\bar{L}^2)^2}{2(\tilde{n}_a - f\lambda_c\bar{L})^2} + \frac{f\lambda_c\bar{L}^3}{3(\tilde{n}_a - f\lambda_c\bar{L})} \quad (3.9)$$

$$\overline{Z_d^3} = \frac{(f\lambda_c\overline{L^2})^2}{(\tilde{n}_a - f\lambda_c\overline{L})^2} \left\{ \frac{3(f\lambda_c\overline{L^2})^2}{4(\tilde{n}_a - f\lambda_c\overline{L})} + f\lambda_c\overline{L^3} \right\} + \frac{f\lambda_c\overline{L^4}}{4(\tilde{n}_a - f\lambda_c\overline{L})} \quad (3.10)$$

The first and second moments of Z_s are

$$\overline{Z_s} = \frac{f\lambda_c(\overline{L^2} + 2\overline{L} \cdot \overline{Z_d} + \overline{Z_d^2})}{2(\tilde{n}_a - f\lambda_c\overline{L} - f\lambda_c\overline{Z_d})} \quad (3.11)$$

$$\overline{Z_s^2} = \frac{1}{2} \left\{ \frac{f\lambda_c(\overline{L^2} + 2\overline{L} \cdot \overline{Z_d} + \overline{Z_d^2})}{\tilde{n}_a - f\lambda_c\overline{L} - f\lambda_c\overline{Z_d}} \right\}^2 + \frac{f\lambda_c(\overline{L^3} + 3\overline{L^2} \cdot \overline{Z_d} + 3\overline{L} \cdot \overline{Z_d^2} + \overline{Z_d^3})}{3(\tilde{n}_a - f\lambda_c\overline{L} - f\lambda_c\overline{Z_d})} \quad (3.12)$$

where the previous expressions (3.8)-(3.10) for the moments of Z_d should be substituted in.

The WAN wavelength channel utilization or the WAN loading factor is defined by $S = \lambda_c\overline{L}$, by keeping the same notational convention with [1]. The WAN loading factor is defined as the fraction of time that an OFS channel in the WAN is busy for data transmissions.

In this work, for simplicity, we only consider constant transaction lengths or flow durations. More specifically, we assume all the flows for OFS are of constant size except for the one being considered, which may vary within a certain range. The problem is set in this way so that we can express easily D_q as a function of the average flow length and loading factors without losing much generality. Nevertheless, accurate plots of D_q depends a lot on the actual traffic statistics (e.g. exponential, heavy-tail, or others), which may be changing as more Internet applications are evolving.

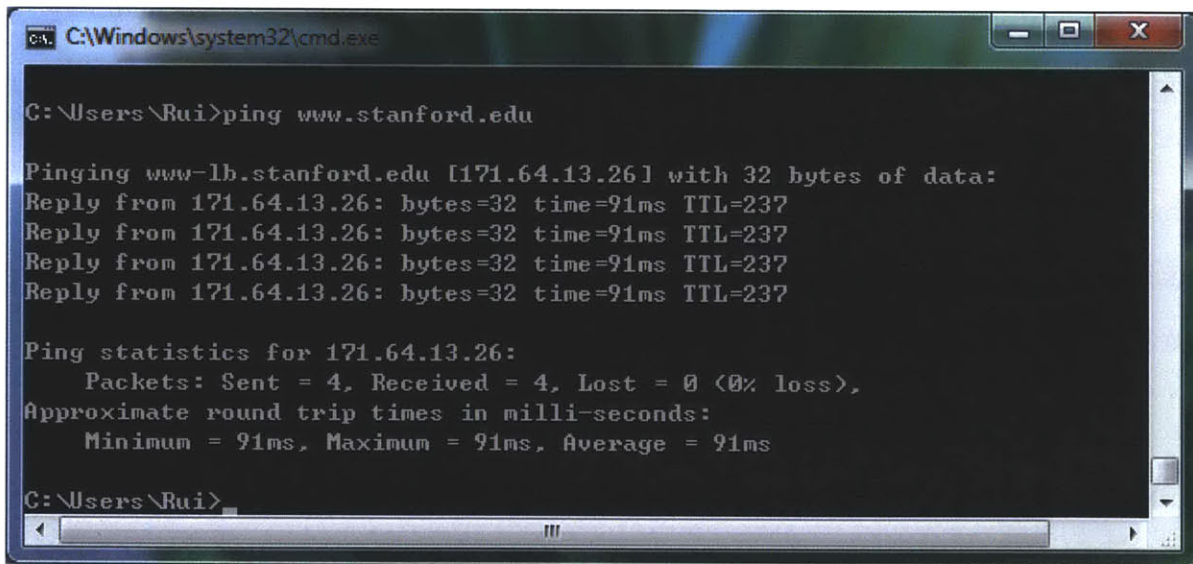
With other parameters known, we can then plot the total delay (in seconds) as a function of WAN loading factor. To do so, we estimated other parameters in Table 3.1 below by using MIT and Stanford as an example. With the "ping" command in Windows OS, the EPS RTT between MIT and Stanford was estimated to be approximately 91ms, as shown in Fig. 3.3.

The fiber distance between MIT and Stanford is about 5,000km, and the all optical OFS propagation delay can be approximated by

$$T_{pg} \approx \frac{\text{fiber distance}}{\text{speed of light in fiber}} \approx \frac{5,000\text{km}}{2 \times 10^5\text{km/s}} = 25 \times 10^{-3}\text{s} = 25\text{ms} \quad (3.13)$$

L_p is assumed to be 10 Kb on average (pessimistic approximation for EPS packet size which has a maximum value of 1.5 KB = 12 Kb), and R_{EPS} is assumed to be 50Mbps (for the author's computer at MIT, this is roughly the speed).

D_{TCP} is estimated to be around 8 times the EPS propagation delay, as discussed above. The scheduling propagation delay is in general more than two times the EPS propagation delay between A and B , which consists of the time for A to send its primary request to its MAN's scheduling node to reserve WAN channel, the time it takes for A 's scheduler to inform both A and B to reserve channels in their respective distribution networks (and get acknowledgements from both A and B), and also the time for A and B 's secondary requests to propagate to their respective MANs' scheduling nodes.



```
C:\Windows\system32\cmd.exe
C:\Users\Rui>ping www.stanford.edu

Pinging www-lb.stanford.edu [171.64.13.26] with 32 bytes of data:
Reply from 171.64.13.26: bytes=32 time=91ms TTL=237
Reply from 171.64.13.26: bytes=32 time=91ms TTL=237
Reply from 171.64.13.26: bytes=32 time=91ms TTL=237
Reply from 171.64.13.26: bytes=32 time=91ms TTL=237

Ping statistics for 171.64.13.26:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 91ms, Maximum = 91ms, Average = 91ms

C:\Users\Rui>
```

Figure 3.3: Estimation of EPS RTT between MIT and Stanford University using the "ping" command in Windows OS.

Table 3.1: Values of parameters used

Parameters	Assumed Value	Remarks
T_{pg}^{EPS}	45.5 ms	E.g. estimated EPS RTT from MIT to Stanford is 91ms
T_{pg}	25 ms	The fiber distance between MIT and Stanford is about 5,000km
L_p	≤ 12 Kb	Maximum EPS packet size 1.5 KB = 12 Kb
R_{EPS}	50 Mbps	Network speed of the author's desktop computer at MIT
R_{OFS}	10 Gbps	Future OFS link rate can be higher
D_{TCP}	364 ms	$= 8T_{pg}^{EPS}$
D_{pg}^{sch}	91 ms	$= 2T_{pg}^{EPS}$

Fig. 3.4 (a)-(d) are the plots of total delay (in seconds) vs. WAN loading factor when the flow duration is $L = 10s, 1s, 100ms,$ and $10ms$ respectively. Fig. 3.5 shows the plots when we normalize the total delay to the flow duration or transaction length for different flow durations $L = 10s, 1s, 100ms,$ and $10ms$.

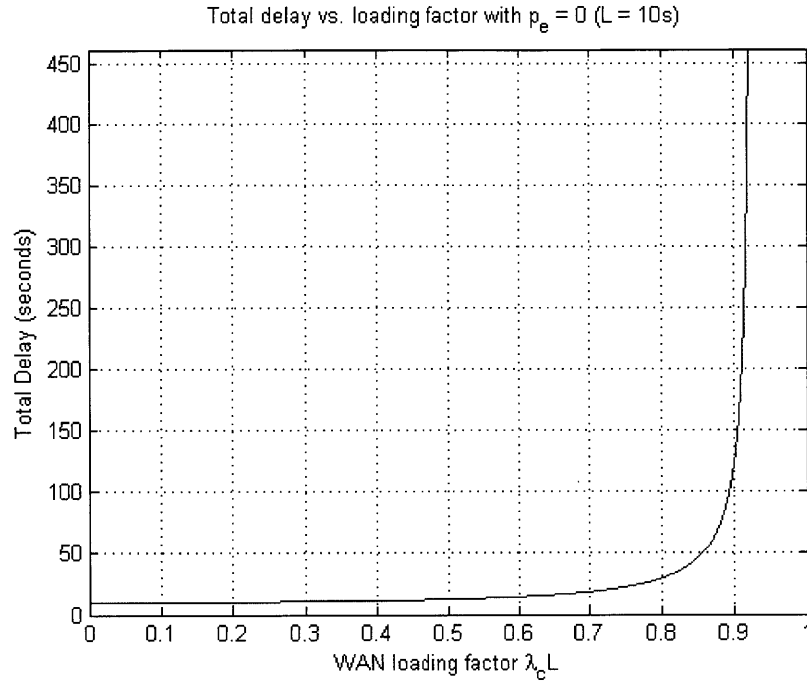


Fig. 3.4 (a): EDBC total delay vs. loading factor with $L = 10s$, $p_e = 0$, $T_{pg}^{EPS} = 45.5ms$, $T_{pg}^{OFS} = 25ms$, $R_{OFS} = 10Gbps$, $D_{TCP} = 8T_{pg}^{EPS}$, and $D_{pg}^{sch} = 2T_{pg}^{EPS}$.

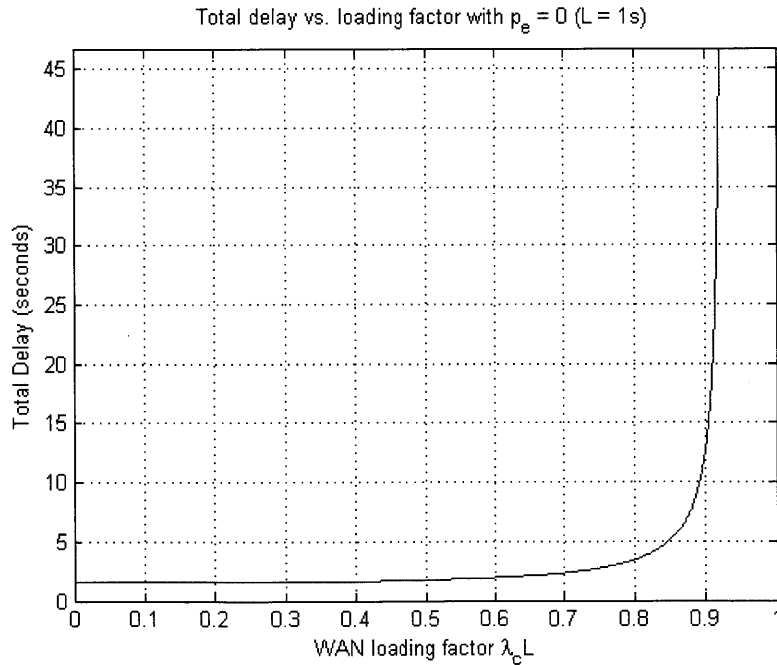


Fig. 3.4 (b): EDBC total delay vs. loading factor with $L = 1s$, $p_e = 0$, $T_{pg}^{EPS} = 45.5ms$, $T_{pg}^{OFS} = 25ms$, $R_{OFS} = 10Gbps$, $D_{TCP} = 8T_{pg}^{EPS}$, and $D_{pg}^{sch} = 2T_{pg}^{EPS}$.

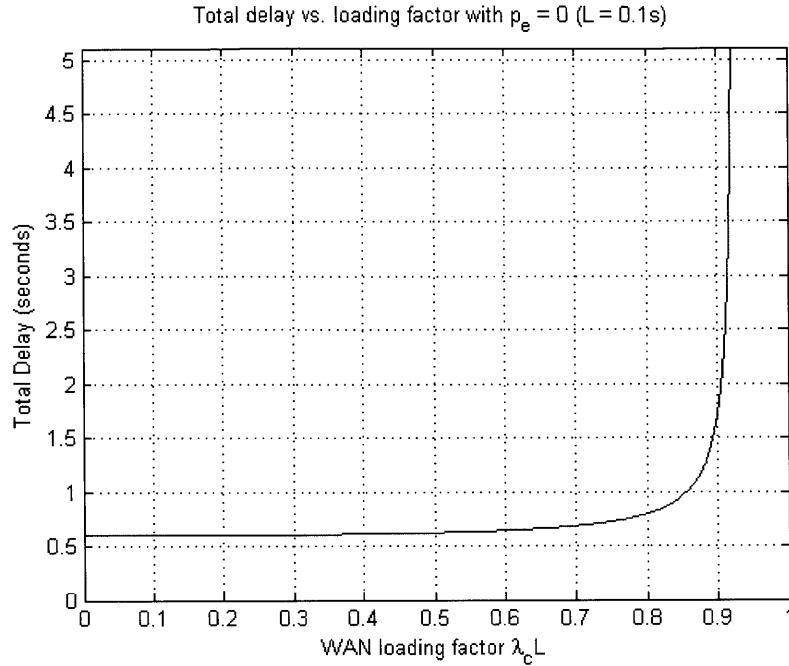


Fig. 3.4 (c): EDBC total delay vs. loading factor with $L = 100$ ms, $p_e = 0$, $T_{pg}^{EPS} = 45.5$ ms, $T_{pg}^{OFS} = 25$ ms, $R_{OFS} = 10$ Gbps, $D_{TCP} = 8T_{pg}^{EPS}$, and $D_{pg}^{sch} = 2T_{pg}^{EPS}$.

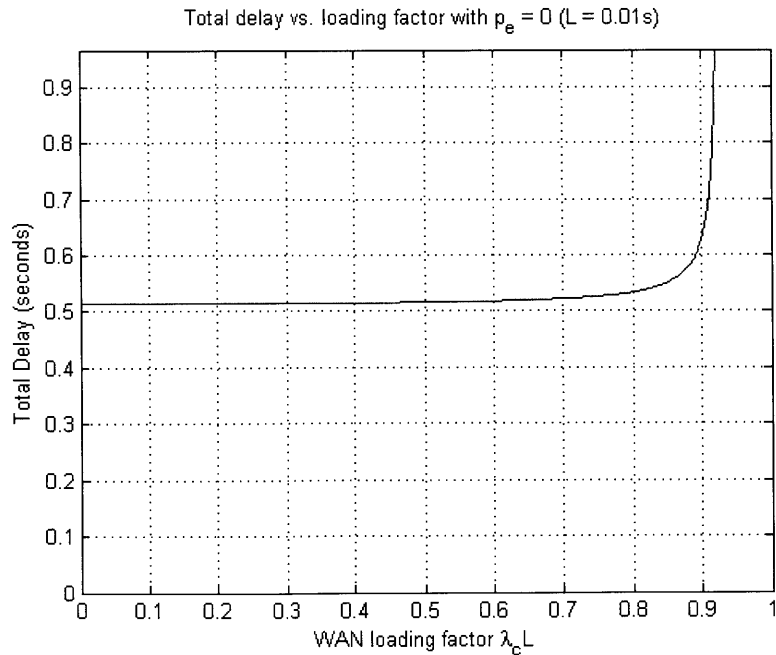


Fig. 3.4 (d): EDBC total delay vs. loading factor with $L = 10$ ms, $p_e = 0$, $T_{pg}^{EPS} = 45.5$ ms, $T_{pg}^{OFS} = 25$ ms, $R_{OFS} = 10$ Gbps, $D_{TCP} = 8T_{pg}^{EPS}$, and $D_{pg}^{sch} = 2T_{pg}^{EPS}$.

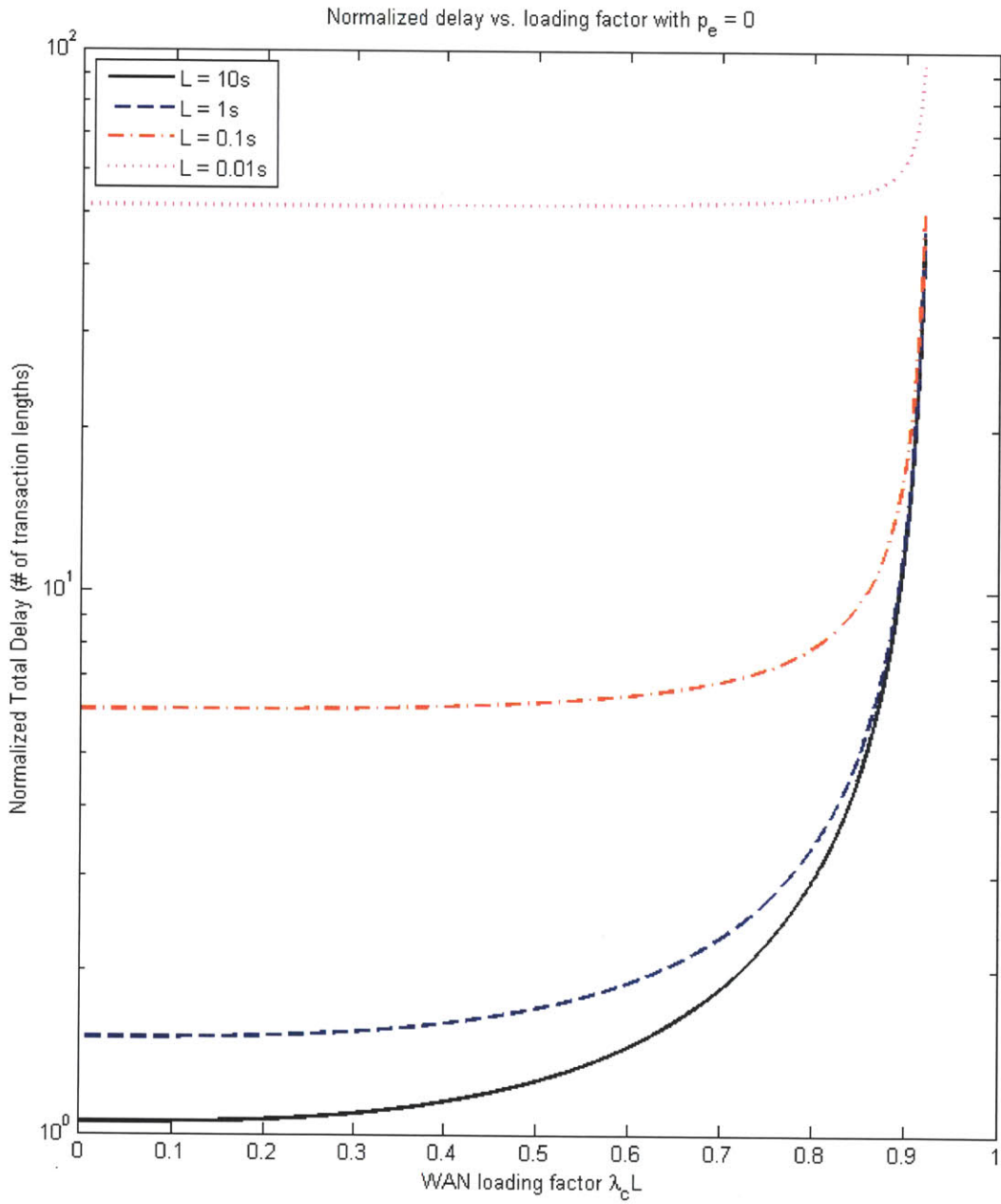


Figure 3.5: EDBC normalized delay (number of transaction lengths) vs. loading factor for different transaction lengths. $p_e = 0$, $T_{pg}^{EPS} = 45.5ms$, $T_{pg}^{OFS} = 25ms$, $R_{OFS} = 10Gbps$, $D_{TCP} = 8T_{pg}^{EPS}$, and $D_{pg}^{sch} = 2T_{pg}^{EPS}$.

It can be readily seen from Fig. 3.5 that as the transaction length becomes small enough (say, on the order of tens of ms), the normalized total delay becomes large (easily on the order of 10 times the transaction length). When the transaction length is on the order of seconds (or larger), the normalized total delay tends to be small (on the order of several transaction lengths). This difference is caused by the delay contributions of the process of TCP connection setup and closure, scheduling, queuing, and acknowledgement packets that are transferred via EPS.

We will now take into account non-zero bit error rates and retransmissions, and analyze the overall delay.

3.2.2 Delay of EDBC with BER > 0

Denote $p_{e,f}$ as the flow error rate defined as the probability that at least one bit of a flow is in error. With bit error rate p_e after Physical Layer error correction for OFS, the probability that a flow of size L_f bits is in error is then

$$p_{e,f} = 1 - (1 - p_e)^{L_f} \approx L_f p_e \quad (3.14)$$

where the approximation holds when $L_f p_e \ll 1$.

Denote $p_{e,p}$ as the packet error rate defined as the probability that at least one bit of an EPS packet is in error. With bit error rate p_e after error correction for EPS, the probability that an EPS packet of size L_p is in error is then

$$p_{e,p} = 1 - (1 - p_e)^{L_p} \approx L_p p_e \quad (3.15)$$

where the approximation holds when $L_p p_e \ll 1$, which is generally true for today's low bit error rate EPS networks. Typically, p_e for EPS is on the order of $10^{-12} \sim 10^{-14}$ after FEC, and L_p has a maximum value of $1.5\text{KB} = 12\text{kb} = 1.2 \times 10^4$ bits, which gives a $p_{e,p}$ on the order of

$10^{-8} \sim 10^{-10}$. For easy analysis without introducing too much unnecessary detail, we shall assume that the EPS communication has almost no error induced and only focus on the case where potential errors are only due to the imperfect OFS channels and/or the sender or receiver. Therefore, retransmissions are only needed when there is at least an error in the flow.

For OFS protocol EDC-NS, the total number of transmissions (including retransmissions) is a geometric random variable with mean $1/(1 - p_{e,f})$, and the expected retransmission delay incurred by this algorithm is then

$$E[D_r] = \left(\frac{1}{1 - p_{e,f}} - 1 \right) (E[D_f] + 3T_{pg}^{EPS}) \quad (3.16)$$

where the $3T_{pg}^{EPS}$ term is added because after sender A sends a NACK message to receiver B , B uses the 3-way handshake method to close the TCP connection with A , only after which A can start to retransmit the file (see the flowchart in Fig. 3.1).

The total expected delay from the moment the flow is requested for transmission until it is successfully received is then

$$E[D_t] = E[D_f + D_r] = E[D_f] + E[D_r] = \frac{E[D_f] + 3p_{e,f}T_{pg}^{EPS}}{1 - p_{e,f}} \quad (3.17)$$

$$E[D_t] \approx \frac{D_{TCP} + 2(D_{pg}^{sch} + D_q) + \left(2T_{pg}^{OFS} + \frac{L_f}{R_{OFS}} \right) + T_{pg}^{EPS} + 3(1 - (1 - p_e)^{L_f})T_{pg}^{EPS}}{(1 - p_e)^{L_f}} \quad (3.18)$$

As discussed earlier in this section, $D_{TCP} \approx 8T_{pg}^{EPS}$, $D_{pg}^{sch} \approx 2T_{pg}^{EPS}$, we can further approximate the total expected delay as

$$E[D_t] \approx \frac{2D_q + \left(2T_{pg}^{OFS} + \frac{L_f}{R_{OFS}} \right) + (16 - 3(1 - p_e)^{L_f})T_{pg}^{EPS}}{(1 - p_e)^{L_f}} \quad (3.19)$$

Expression (3.19) implies that as the BER increases, the total delay increases for the same file size L_f . Also, as the file size L_f increases, $(1 - p_e)^{L_f}$ becomes exponentially small, leading to an exponential increase in total delay.

Fig. 3.6 shows the plots of total delay versus loading factor, similar to the cases with no channel errors. Here we assume the OFS BER is $p_e = 10^{-14}$, and $R_{OFS} = 10$ Gbps. Fig. 3.7 shows the plots of normalized delay vs. loading factor.

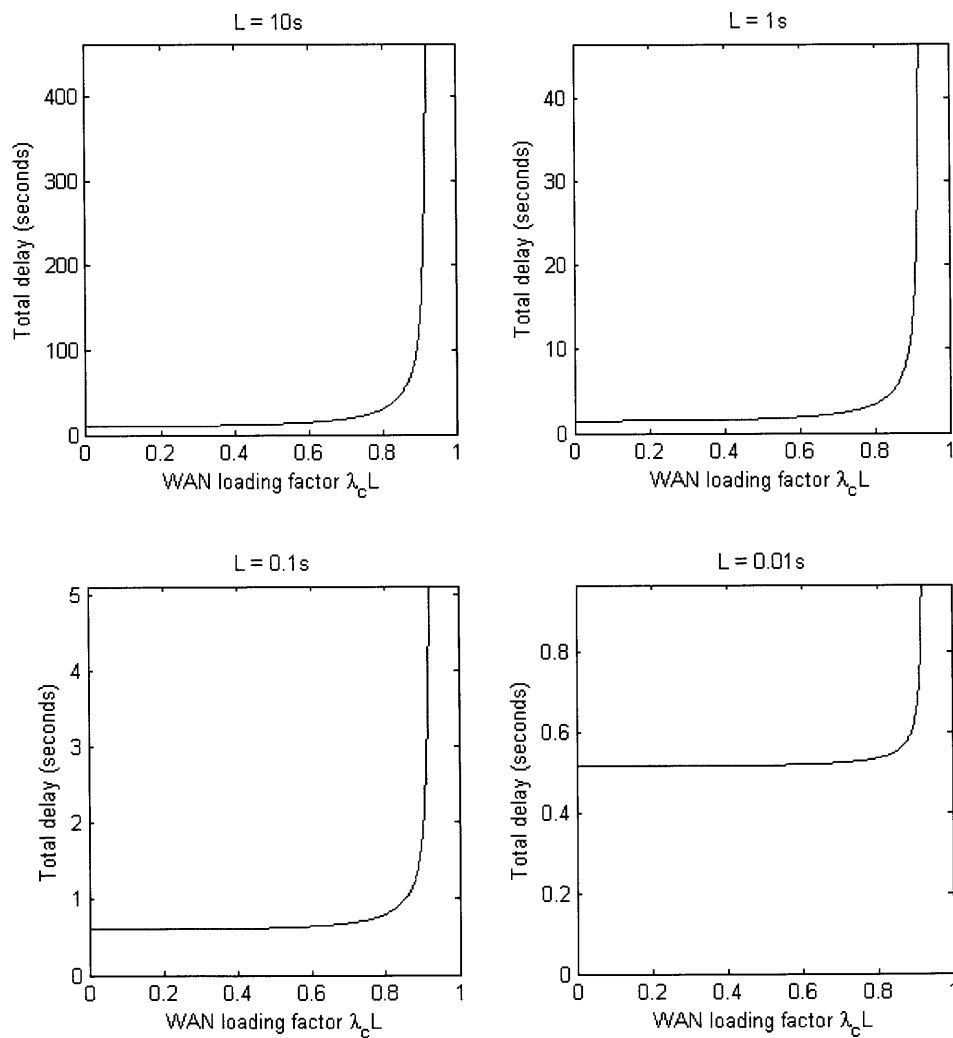


Figure 3.6: EDBC total delay vs. loading factor with $L = 10s$, $p_e = 10^{-14}$, $T_{pg}^{EPS} = 45.5ms$, $T_{pg}^{OFS} = 25ms$, $R_{OFS} = 10Gbps$, $D_{TCP} = 8T_{pg}^{EPS}$, and $D_{pg}^{sch} = 2T_{pg}^{EPS}$.

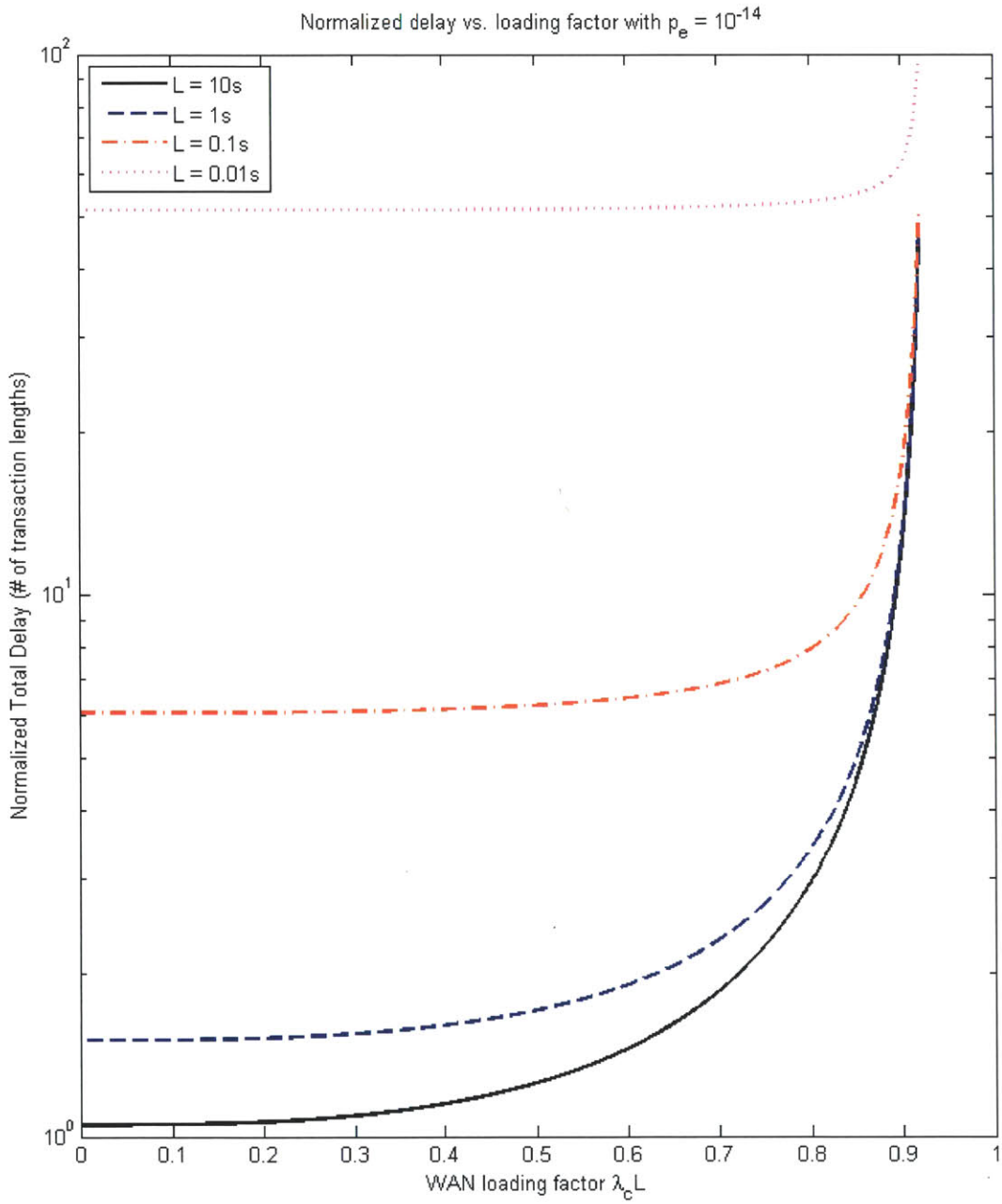


Figure 3.7: EDBC normalized delay (number of transaction lengths) vs. loading factor for different transaction lengths. $p_e = 10^{-14}$, $T_{pg}^{EPS} = 45.5ms$, $T_{pg}^{OFS} = 25ms$, $R_{OFS} = 10Gbps$, $D_{TCP} = 8T_{pg}^{EPS}$, and $D_{pg}^{sch} = 2T_{pg}^{EPS}$.

It is not surprising to see that Fig. 3.7 looks almost the same with Fig. 3.6 when the BER is zero. This is because $p_{e,f} \approx L_f p_e \ll 1$, and only with negligible probability that the file will be retransmitted. In the case where $L_f p_e$ is on the order of 10^{-1} , the probability of retransmission is high, and the plots for delays for large transactions and small transactions look different, as depicted in Fig. 3.8 and 3.9 (the two plots are the total delay and normalized delay vs. loading factor with $p_e = 10^{-12}$):

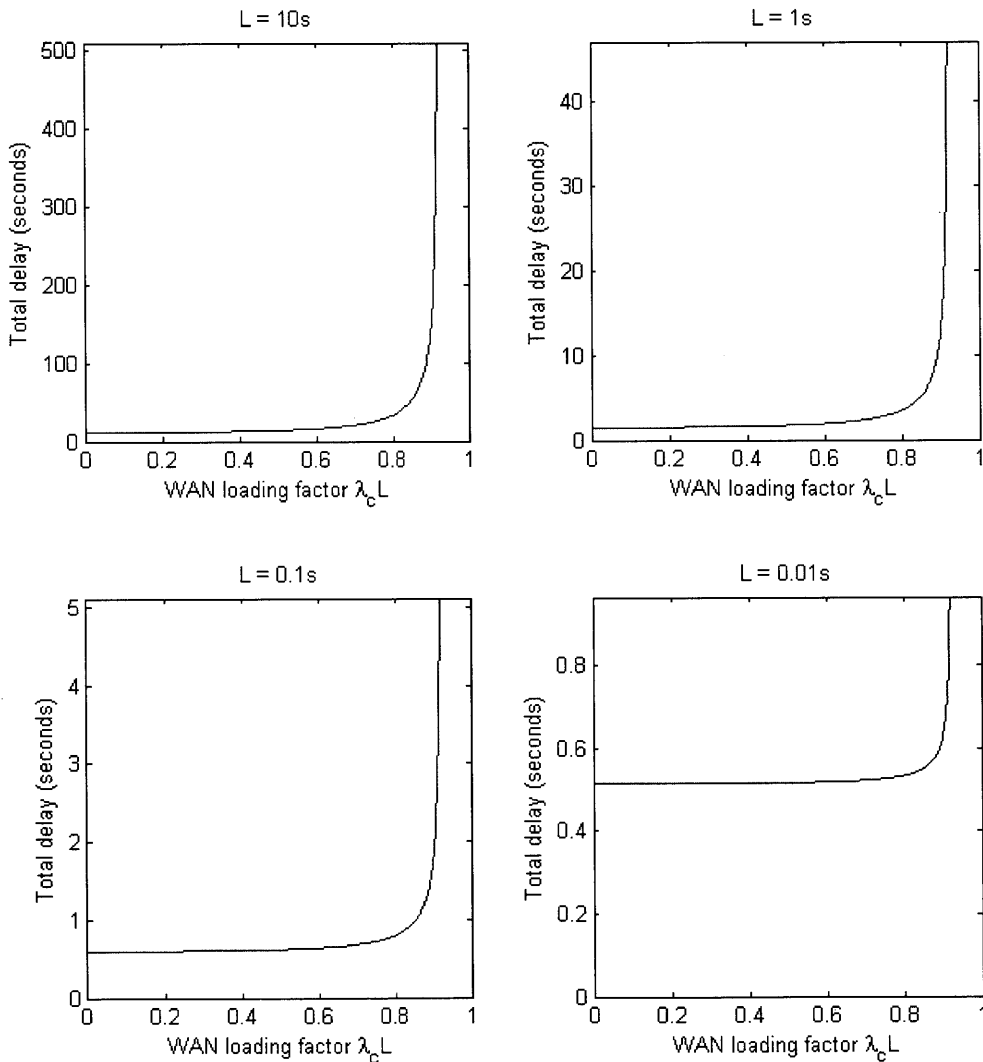


Figure 3.8: EDBC total delay vs. loading factor with $L = 10s$, $p_e = 10^{-12}$, $T_{pg}^{EPS} = 45.5ms$, $T_{pg}^{OFS} = 25ms$, $R_{OFS} = 10Gbps$, $D_{TCP} = 8T_{pg}^{EPS}$, and $D_{pg}^{sch} = 2T_{pg}^{EPS}$.

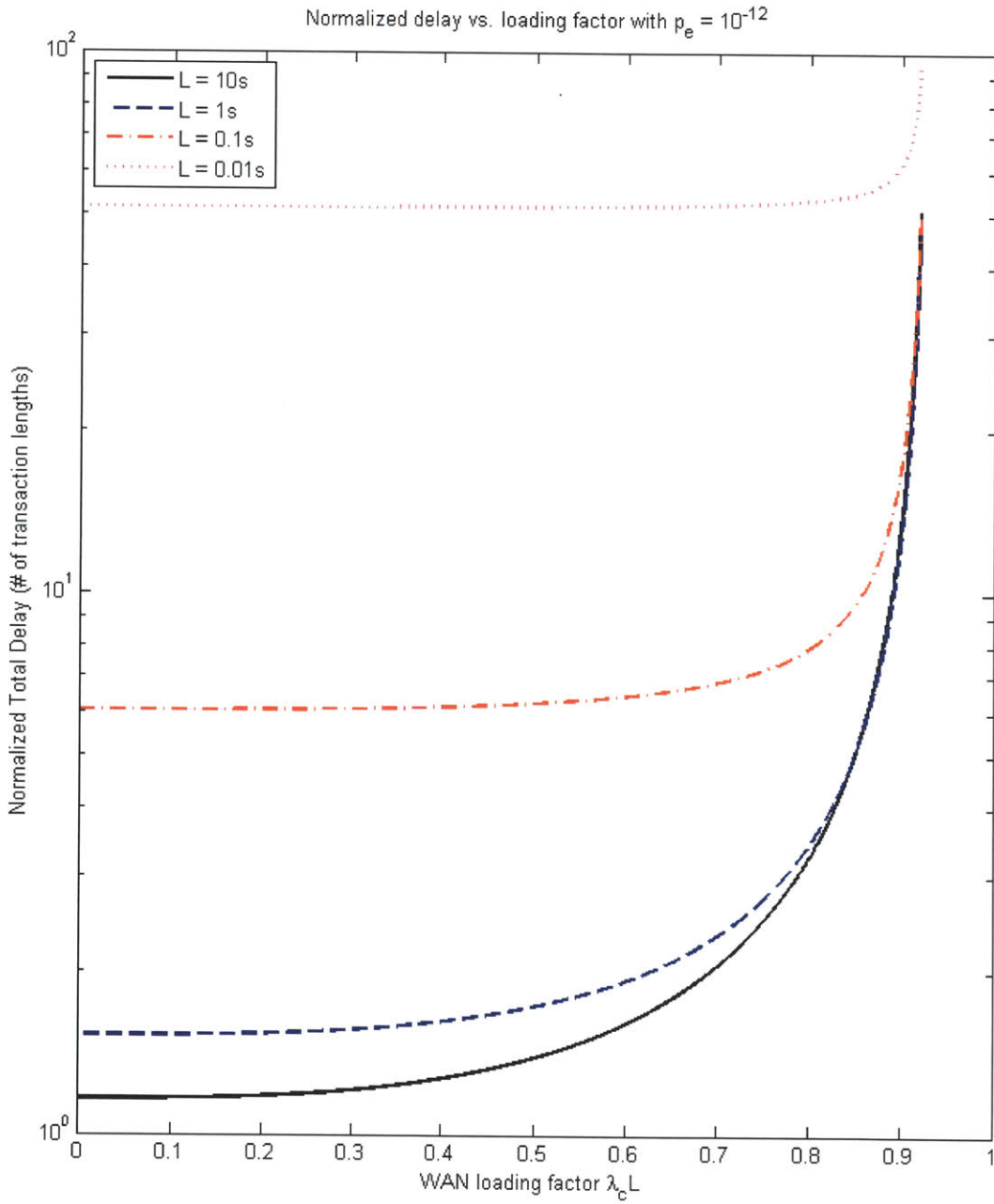


Figure 3.9: EDBC normalized delay (number of transaction lengths) vs. loading factor for different transaction lengths. $p_e = 10^{-12}$, $T_{pg}^{EPS} = 45.5ms$, $T_{pg}^{OFS} = 25ms$, $R_{OFS} = 10Gbps$, $D_{TCP} = 8T_{pg}^{EPS}$, and $D_{pg}^{sch} = 2T_{pg}^{EPS}$.

With further increased $p_e = 10^{-10}$, we have the following plots for total delay (Fig. 3.10) and normalized delay (Fig. 3.11) vs. loading factor:

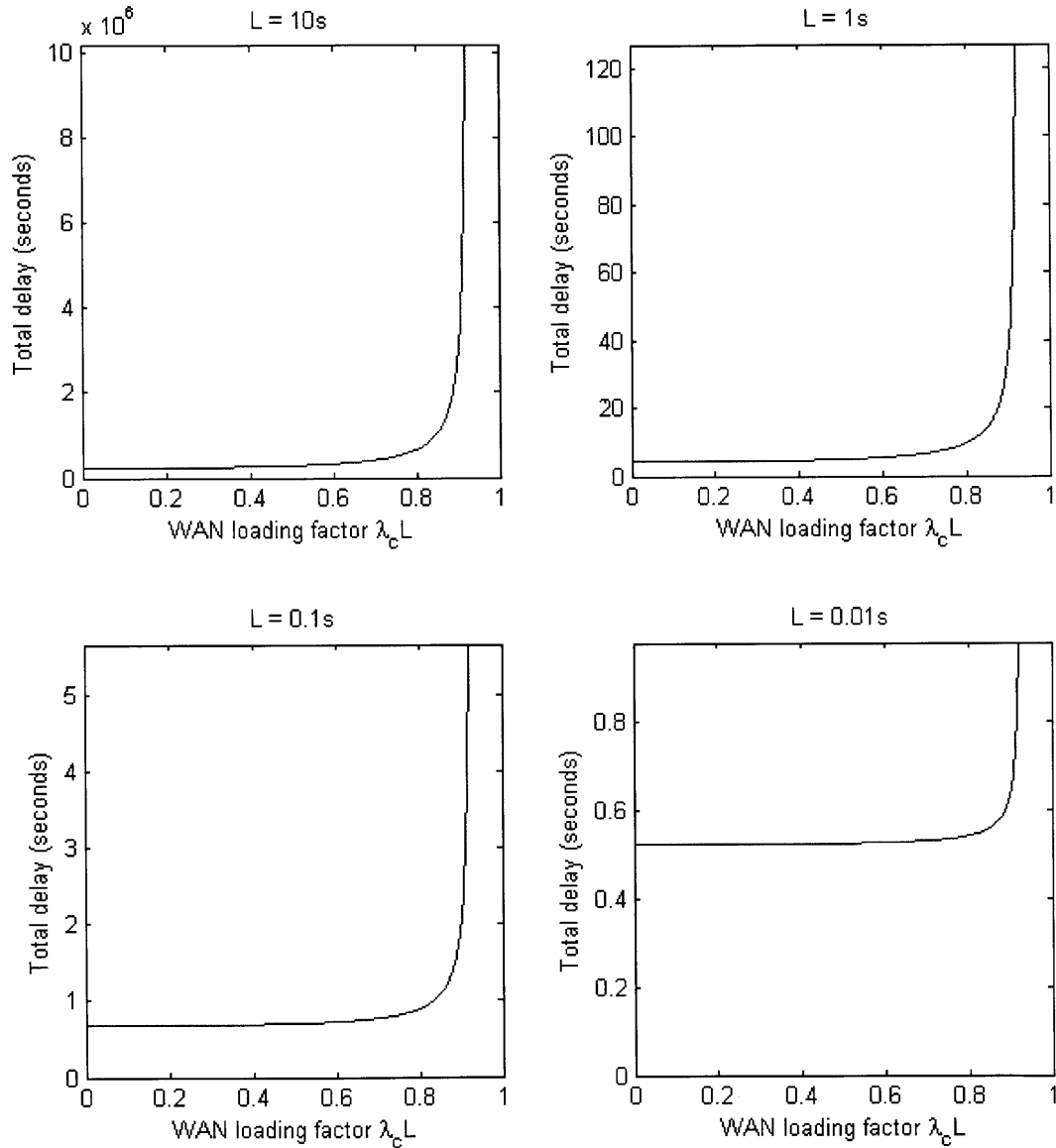


Figure 3.10: EDBC total delay vs. loading factor with $L = 10s$, $p_e = 10^{-10}$, $T_{pg}^{EPS} = 45.5ms$, $T_{pg}^{OFS} = 25ms$, $R_{OFS} = 10Gbps$, $D_{TCP} = 8T_{pg}^{EPS}$, and $D_{pg}^{sch} = 2T_{pg}^{EPS}$.

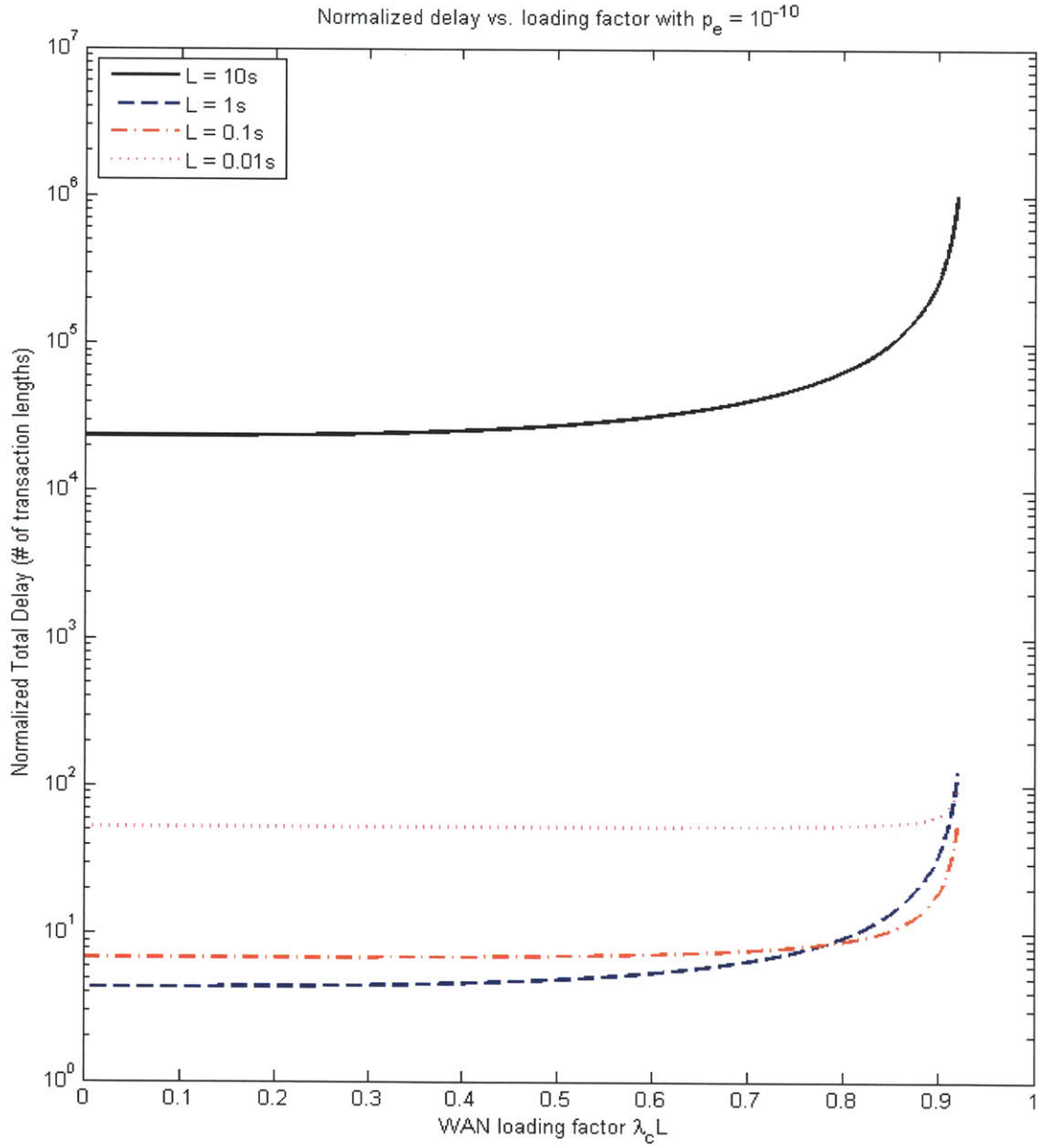


Figure 3.11: EDBC normalized delay (number of transaction lengths) vs. loading factor for different transaction lengths. $p_e = 10^{-10}$, $T_{pg}^{EPS} = 45.5ms$, $T_{pg}^{OFS} = 25ms$, $R_{OFS} = 10Gbps$, $D_{TCP} = 8T_{pg}^{EPS}$, and $D_{pg}^{sch} = 2T_{pg}^{EPS}$.

It can be seen from Fig. 3.12 that the normalized delay for $L = 10$ s is extremely high ($> 2 \times 10^4$ times of the transaction length). The reason is that the flow error rate

$$p_{e,f} = 1 - (1 - p_e)^{L_f} = 1 - (1 - 10^{-10})^{10 \times 10^{10}} \approx 0.9999546$$

is extremely high in this case, and there need to be a lot of retransmissions.

Similarly, for $L = 1$ s, the flow error rate is

$$p_{e,f} = 1 - (1 - p_e)^{L_f} = 1 - (1 - 10^{-10})^{1 \times 10^{10}} \approx 0.6321206$$

and a fair amount of retransmission is also needed, leading to a large normalized delay on the order of 100 times the transaction length. The cases of $L = 100$ ms and $L = 10$ ms are not bad in comparison in terms of normalized delay. These simple examples point to the need for segmentation of the flow based on channel bit error rates.

3.3 Summary of Chapter 3

In this chapter, we proposed and analyzed the OFS protocol EDBC. We looked at the total delay and normalized delay of EDBC for different BERs and loading factors. In the delay expression of EDBC in (3.19), the total queuing delay is $2D_q$, because of queuing in both forward and backward directions. We shall discuss in Chapter 4 another protocol that does not send back data on the backward path, but uses error detection code instead.

The plots in Fig. 3.5, 3.7, 3.9 and 3.11 showed that the normalized delay is smaller for larger files when the number of retransmissions are not too many, i.e. when $p_e L_f \ll 1$. When $p_e L_f$ gets close to 1, however, many retransmissions are needed to make the whole flow error-free. It may be necessary to divide the flow into some smaller blocks so that each smaller block has smaller probability of errors. In this way, the number of retransmissions can also be smaller. We shall discuss this class of protocols in Chapter 5 and 6.

Chapter 4

Protocol with Error Detection Code and No Segmentation (EDC-NS)

As discussed in Chapter 3, the use of error detection by backward comparison requires two channels be reserved in sequence and hence two-way queuing delay in each (re)transmission. Thus extra use of resources (backward flow) and queuing delay are introduced, which can be avoided if we employ some error detection codes into the flow so that the receiver can immediately determine whether the flow received is erroneous from the code.

In the case of error detection codes being used at the sender and receiver, there are different Transport Layer protocols depending on whether the flow is segmented into smaller blocks before transmission. In this chapter, we will consider the case where there is no

segmentation. We call this protocol EDC-NS for short, which stands for Error Detection Code and No Segmentation.

EDC-NS works in the following way: After scheduling, the whole file to be transmitted via OFS is first encoded with an error detection code (e.g. CRC code), and with the redundancy added, the receiver can detect (almost) any error in the flow. If any error is detected, the receiver requests retransmission of the whole file by sending a negative acknowledgement to the sender. The entire process starts again until the whole file is received error-free.

Similar to the case of the EDBC protocol, we divide the overall EDC-NS protocol into three phases: preparation, transmission, and conclusion. We will adopt the same assumptions and definitions as that for the EDBC protocol, except that there is no backward OFS path from the receiver B to the sender A , and that the flow is encoded with error detection code (e.g. CRC code) before transmission.

The detailed algorithm description and flowchart are discussed in Section 4.1. The delay performance is analyzed in Section 4.2.

4.1 EDC-NS Algorithm Description and Flowchart

The detailed three-phase EDC-NS algorithm is described below:

Phase I: Preparation

1. TCP connections between A and A 's scheduler, between A 's scheduler and B 's scheduler, between B 's scheduler and B , and between B and A are established in sequence.

- 1a. A first establishes a TCP connection with A 's (ingress) scheduler at the network node connecting A 's MAN and the WAN. A then sends an informative message to its scheduler, saying that A wants to send data to B via OFS.
- 1b. A 's scheduler then establishes a TCP connection with B 's (egress) scheduler at the node connecting the WAN and B 's MAN. A 's scheduler then sends a message to B 's scheduler, saying that A wants to send data to B via OFS.
- 1c. B 's scheduler establishes a TCP connection with B and sends the same message in the same way as above.
- 1d. B then establishes a TCP connection with A .
2. After all connections are established, A requests the scheduler to set up a forward path by telling the scheduler the length of the flow that needs to be transmitted and other necessary information.
3. The scheduler then reserves a wavelength channel from A to B , whenever available, for the duration of the flow (possibly with some guard time), and tells A and B the reserved OFS channel wavelength, planned start time and duration of the channel reservation. Assume all exchanges of messages in the scheduling process happen on the OFS control plane, which can be EPS. The detailed scheduling algorithm is described on page 142 in Guy Weichenberg's PhD thesis [1].

Phase II: Transmission

4. At the start time specified by the scheduler, A starts transmitting the encoded flow to B via the reserved OFS channel, i.e. the forward path previously set up.
5. Upon decoding the data received from A , B recognizes the start-of-flow. If B does not see start-of-flow within the OFS propagation delay plus some guard time after the start time, B timeouts and closes its TCP connections with A and scheduler. When A detects that B closes its TCP connection, A closes its TCP connection with scheduler. At the same time, B clears all data it previously received from A . Go back to Step 1.

Phase III: Conclusion

6. B continues decoding the received flow, and stores a copy of the file after decoding. At the same time, B also checks whether there is any error in the received flow by using the error detection code.

6a. If any error is detected in the flow, B sends a NACK message to A via EPS after the flow transmission is finished, saying that the OFS data previously received by B are erroneous. Meanwhile, B discards any data received from A and closes its TCP connection with the scheduler.

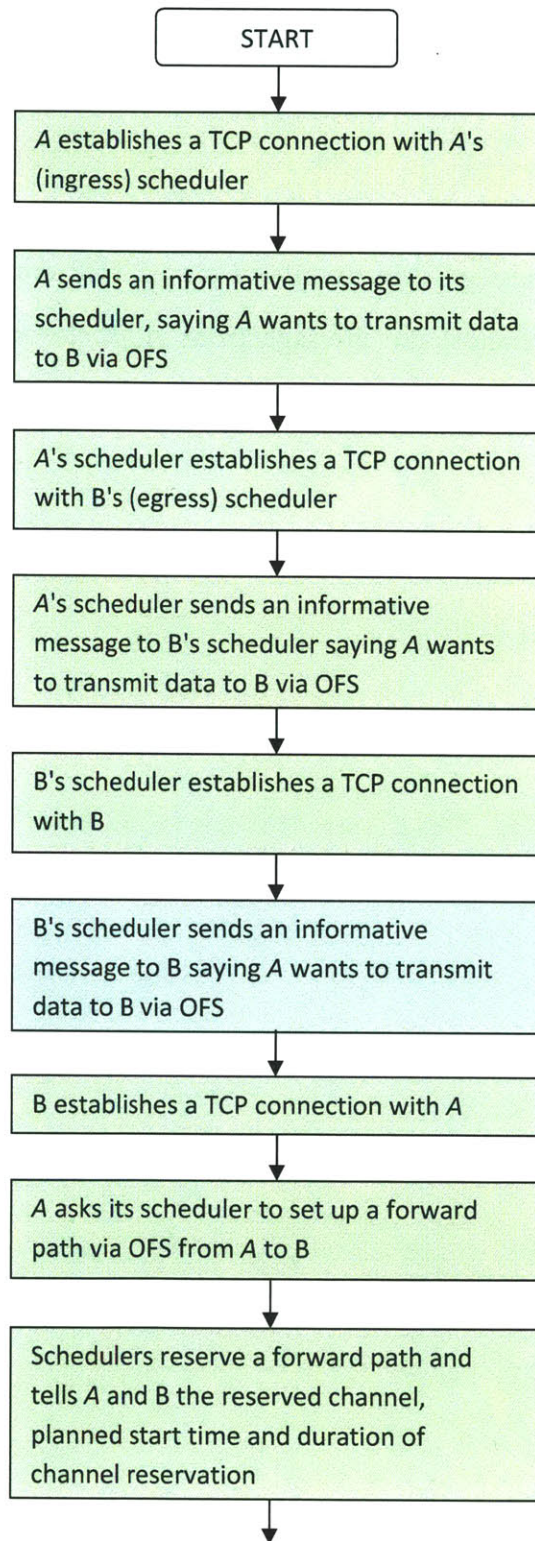
Upon receiving the NACK, A sends back a handshake message via EPS to B , saying that it received the NACK message. Meanwhile A closes its TCP connection with the scheduler. Upon receiving the handshake message from A , B closes its TCP connection with A . Go back to Step 1.

6b. If no error is found for the whole flow, B sends back an ACK message to A via EPS to confirm that the OFS data previously received by B is correct. Meanwhile, B closes the TCP connection with the scheduler, and passes the previously stored file to the application layer.

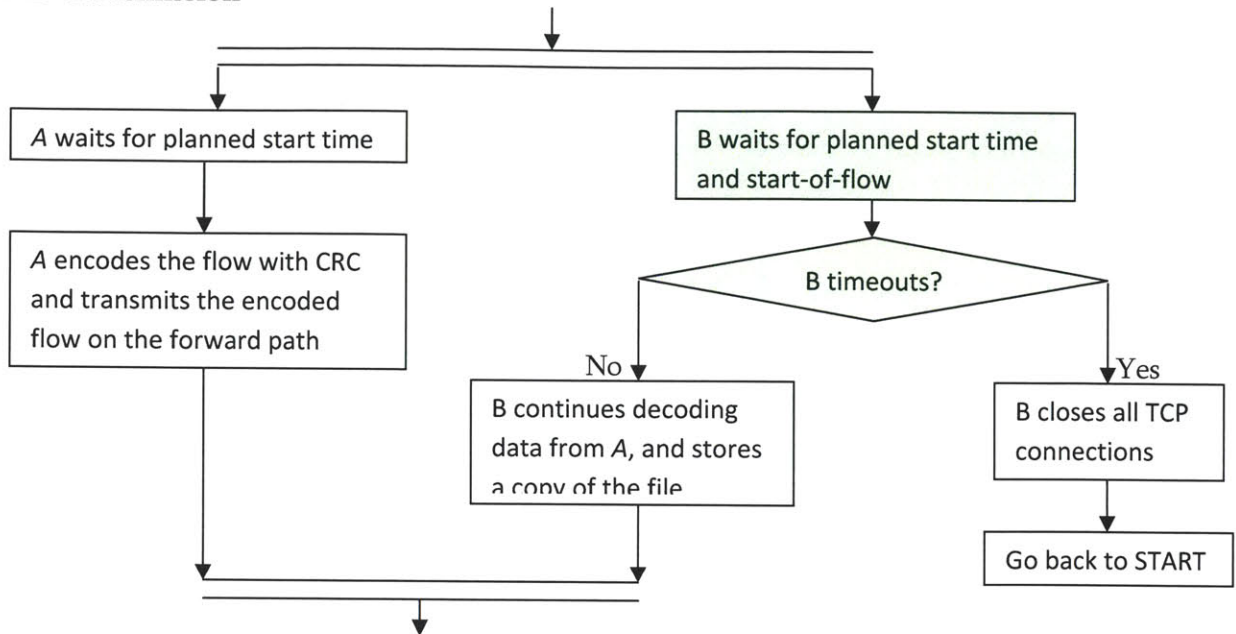
Upon receiving the ACK, A sends back a handshake message via EPS to B , saying that it received the ACK message. Meanwhile A closes TCP connections with the scheduler. Upon receiving the handshake message from A , B closes its TCP connection with A . Algorithm terminates.

Flowchart:

Phase I: Preparation



Phase II: Transmission



Phase III: Conclusion

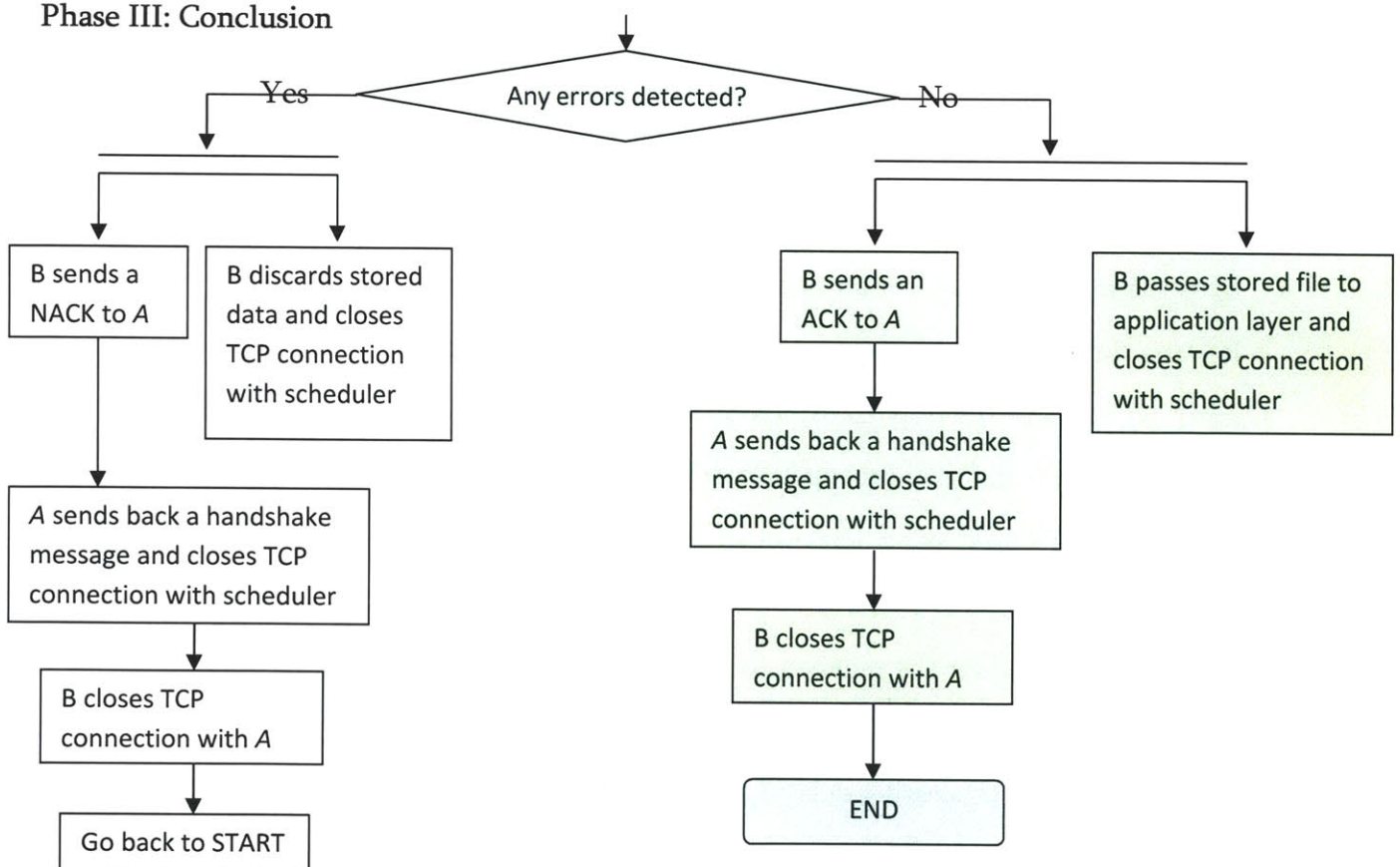


Figure 4.1: EDC-NS protocol flowchart. There are three different phases: Preparation, Transmission and Conclusion. The "green path" shows the algorithm flow when there is no bit error.

4.2 Delay Analysis of EDC-NS Protocol

For a flow of length L_f and OFS link rate R_{OFS} , when there is no transmission error of any kind (i.e. perfect sender, channel, receiver conditions), the timeline follows the "green path" in Fig. 4.1. The overall delay of the transmission is then

$$D_f = D_{TCP} + (D_{pg}^{sch} + D_q) + \left(T_{pg}^{OFS} + T_{pc}^{OFS} + \frac{L_f}{R_{OFS}} \right) + \tau \quad (4.1)$$

where all the terms used on the right hand side of the above expression have similar interpretations to those used in EDBC protocol in Section 3.2.1, i.e.

- D_{TCP} is the time taken to establish all three TCP connections between the sender, receiver and scheduler before scheduling processes can begin (see a detailed explanation about this term in Section 3.2.1),
- D_{pg}^{sch} is the delay (including the EPS packet transmission delay, propagation delay, queuing delay at the router, and processing delay) in the process of scheduling (at least one EPS RTT between sender and receiver),
- D_q is the queuing delay for an open channel as determined by the reservation scheduler. It is the time from the moment the scheduler receives the request for channel reservation until the moment the flow transmission starts on the reserved channel,
- T_{pg}^{OFS} is the OFS one-way propagation delay between the sender and receiver,
- T_{pc}^{OFS} is the OFS processing delays at the sender and/or receiver added to the overall delay, and is neglected in this thesis (see Section 3.2), and
- τ is the time taken for the receiver to pass the decoded data to the application layer.

When we ignore the processing delay T_{pc}^{OFS} and τ , the delay can be approximated by

$$D_f \approx D_{TCP} + (D_{pg}^{sch} + D_q) + \left(T_{pg}^{OFS} + \frac{L_f}{R_{OFS}} \right) \quad (4.2)$$

With bit error rate p_e defined as the probability that a bit is in error, the probability that the flow of size L_f bits is in error is then

$$p_{e,f} = 1 - (1 - p_e)^{L_f} \approx L_f p_e \quad (4.3)$$

where the approximation holds when $L_f p_e \ll 1$.

Typically, for an n -bit well designed CRC [26, 28] applied to a data block of arbitrary length, it can detect any odd number of errors, any single error burst not longer than n bits, and a fraction of $1 - 2^{-n}$ of all longer error bursts [26, 28]. That is, the probability of undetected error will be roughly 2^{-n} when the message is somehow randomly corrupted by noise. For $n = 32$, this corresponds to $2^{-32} \approx 2.33 \times 10^{-10}$, which is extremely small. For OFS with small error probability, the case of single bit error, which has a probability of $\binom{L_f}{1} p_e (1 - p_e)^{L_f - 1} = L_f p_e (1 - p_e)^{L_f - 1}$, can always be detected. The cases with two or more bits in error are extremely rare in our region of interest, and have a probability of

$$p_{e,f} - \binom{L_f}{1} p_e (1 - p_e)^{L_f - 1} = 1 - (1 - p_e)^{L_f} - L_f p_e (1 - p_e)^{L_f - 1} \quad (4.4)$$

With $p_e L_f \ll 1$, the above expression can be approximated by $p_e^2 (L_f - 1) \ll L_f p_e (1 - p_e)^{L_f - 1}$. That is, the probability of two or more errors is much smaller than the probability of one single error. Therefore, in our discussions below, we shall assume that any error(s) in the message received will be detected by the CRC code and ignore the case of undetected error.

The total number of transmissions and retransmissions due to potential errors in the flow is a geometric random variable with mean $\frac{1}{1 - p_{e,f}}$, and the expected retransmission delay due to errors incurred by the EDC-NS protocol is given by

$$E[D_r] \approx \left(\frac{1}{1 - p_{e,f}} - 1 \right) (E[D_f] + 5T_{pg}^{EPS}) \quad (4.5)$$

where the $5T_{pg}^{EPS}$ term is added because when B detects an error it will then send a NACK to A , which takes T_{pg}^{EPS} to arrive at A , and get handshake message from A , which takes another T_{pg}^{EPS} , after which B will then close TCP connection with A using the 3-way handshake method discussed in Section 3.2, which takes $3T_{pg}^{EPS}$. The total expected number of sending NACK and TCP connection closures before B receives the flow data correctly is $\frac{1}{1-p_{e,f}} - 1$, which gives a total added EPS delay of $\left(\frac{1}{1-p_{e,f}} - 1\right) (5T_{pg}^{EPS})$ due to retransmissions.

The total expected delay is then

$$E[D_t] = E[D_f + D_r] = E[D_f] + E[D_r] \quad (4.6)$$

That is,

$$\begin{aligned} E[D_t] &\approx \frac{E[D_f] + 5p_{e,f}T_{pg}^{EPS}}{1 - p_{e,f}} \\ &= \frac{D_{TCP} + (D_{pg}^{sch} + D_q) + \left(T_{pg}^{OFS} + \frac{L_f}{R_{OFS}}\right) + 5p_{e,f}T_{pg}^{EPS}}{(1 - p_e)^{L_f}} \end{aligned} \quad (4.7)$$

As discussed in Section 3.2, $D_{TCP} \approx 8T_{pg}^{EPS}$, $D_{pg}^{sch} \approx 2T_{pg}^{EPS}$ and by ignoring the $\frac{L_p}{R_{EPS}}$ term, we can further approximate the total expected delay as

$$E[D_t] \approx \frac{D_q + \left(T_{pg}^{OFS} + \frac{L_f}{R_{OFS}}\right) + (15 - 5(1 - p_e)^{L_f})T_{pg}^{EPS}}{(1 - p_e)^{L_f}} \quad (4.8)$$

Below is a plot of normalized delay vs. WAN loading factor with parameters in Table 3.1, and different bit error rates: $p_e = 0, 10^{-14}, 10^{-12},$ and 10^{-10} respectively. Four different flow durations are used in each plot: $L = 10s, 1s, 100ms,$ and $10ms$.

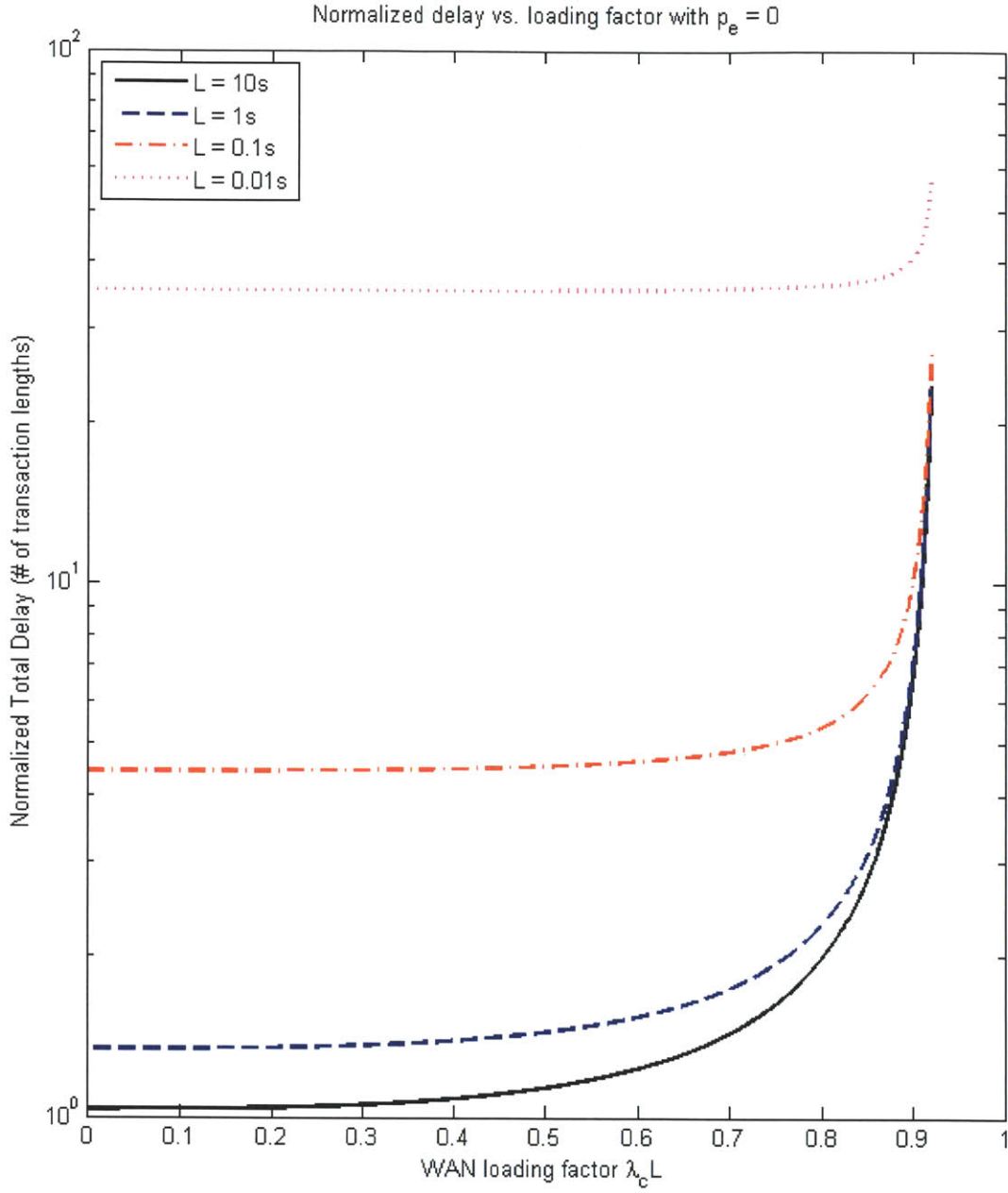


Figure 4.2: EDC-NS normalized delay (number of transaction lengths) vs. loading factor for different transaction lengths. $p_e = 0$, $T_{pg}^{EPS} = 45.5ms$, $T_{pg}^{OFS} = 25ms$, $R_{OFS} = 10Gbps$, $D_{TCP} = 8T_{pg}^{EPS}$, and $D_{pg}^{sch} = 2T_{pg}^{EPS}$.

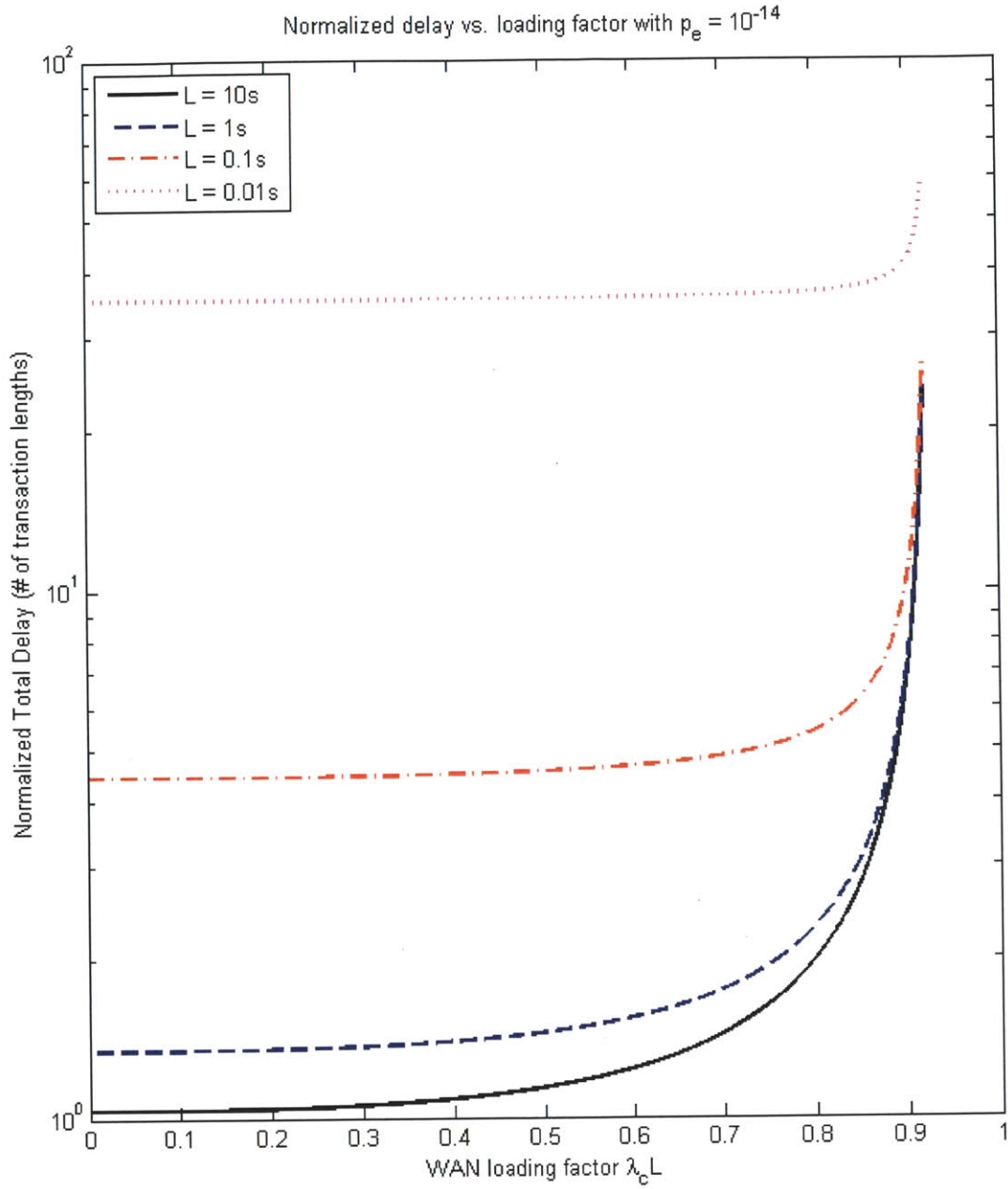


Figure 4.3: EDC-NS normalized delay (number of transaction lengths) vs. loading factor for different transaction lengths. $p_e = 10^{-14}$, $T_{pg}^{EPS} = 45.5ms$, $T_{pg}^{OFS} = 25ms$, $R_{OFS} = 10Gbps$, $D_{TCP} = 8T_{pg}^{EPS}$, and $D_{pg}^{sch} = 2T_{pg}^{EPS}$.

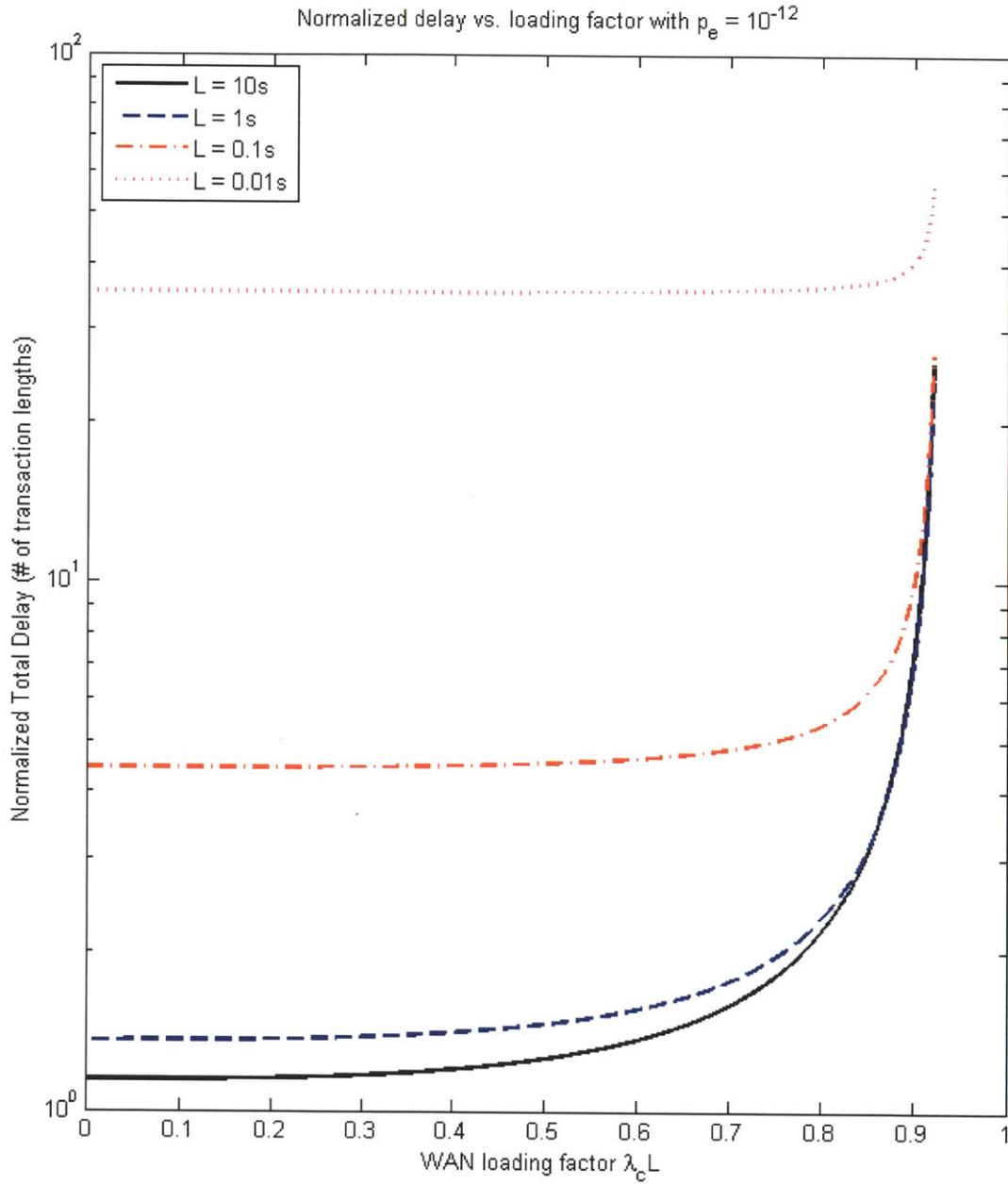


Figure 4.4: EDC-NS normalized delay (number of transaction lengths) vs. loading factor for different transaction lengths. $p_e = 10^{-12}$, $T_{pg}^{EPS} = 45.5ms$, $T_{pg}^{OFS} = 25ms$, $R_{OFS} = 10Gbps$, $D_{TCP} = 8T_{pg}^{EPS}$, and $D_{pg}^{sch} = 2T_{pg}^{EPS}$.

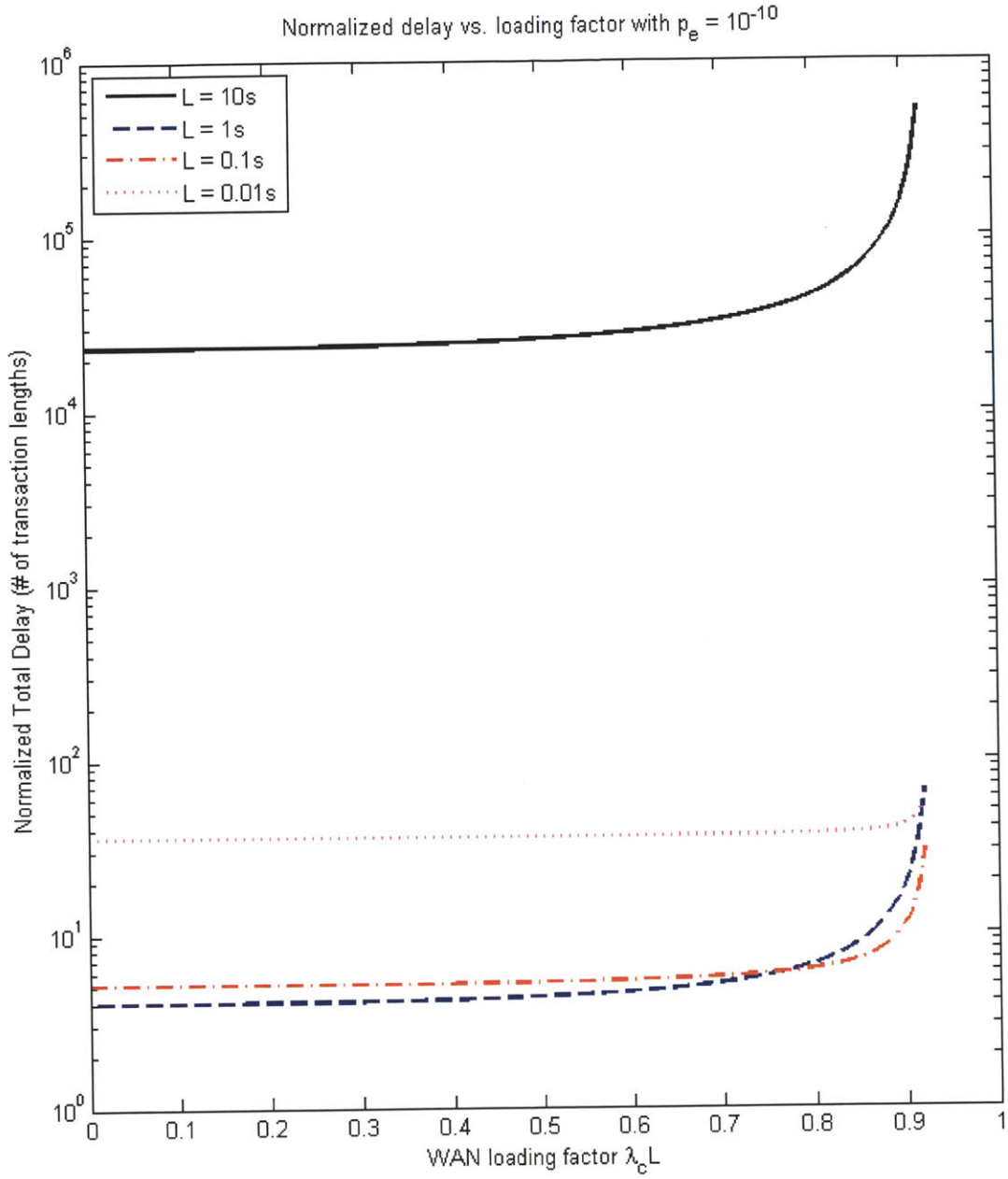


Figure 4.5: EDC-NS normalized delay (number of transaction lengths) vs. loading factor for different transaction lengths. $p_e = 10^{-10}$, $T_{pg}^{EPS} = 45.5ms$, $T_{pg}^{OFS} = 25ms$, $R_{OFS} = 10Gbps$, $D_{TCP} = 8T_{pg}^{EPS}$, and $D_{pg}^{sch} = 2T_{pg}^{EPS}$.

From Fig. 4.2-4.4, it can be seen that the longer the flow duration (e.g. from 0.01s to 10s) in a certain range, the lower the normalized delay (note that this is only true when $p_e L_f \ll 1$). This is because the queuing delay is on the order of a few hundred milliseconds and contributes more to the normalized delay when the flow duration is small. This is true for the bit error rates in the range of $0 \sim 10^{-12}$. As the bit error rate increases further, as shown in Fig. 4.5 with $BER = 10^{-10}$, the normalized delay of flow with duration of 1s (i.e. $L_f = 10$ Gbits) starts to intersect with the delay curve for the case of 1ms when the loading factor is around 0.75. Also, the normalized delay of flow with duration of 10s goes above three other curves. This is because $p_e L_f = p_e L R_{OFS} = 10^{-10} \times 10 \times 10^{10} = 10 > 1$, and the probability of errors for such a long flow is very high, leading to lots of retransmissions and hence a long delay.

4.3 Summary of Chapter 4

In this chapter, we discussed the EDC-NS protocol. To conclude, when $p_e L_f \ll 1$, a longer flow duration results in a lower normalized delay. Otherwise, when $p_e L_f$ is not much smaller than 1, a longer flow duration causes lots of retransmissions and hence a longer normalized delay. This means that the EDC-NS protocol works well only when $p_e L_f \ll 1$.

In the case that $p_e L_f$ is not much smaller than 1, it will be necessary to divide the flow into smaller segments or blocks and encode each block separately. Only erroneous blocks need to be retransmitted. This may greatly reduce the overall delay. We will next take a look at this type of protocols in Chapter 5 and 6.

Chapter 5

Protocol with Error Detection Code and Segmentation (EDC-S)

Instead of transmitting the whole flow, we also have the choice of segmenting the files and encoding each segmented block with an error detection code (e.g. CRC code) before transmission over OFS. Only erroneous blocks, instead of the whole flow, are retransmitted. We abbreviate the protocol with error detection code and segmentation as EDC-S, and the protocol with forward error correction code and segmentation as FEC-S. In this chapter, we will look at EDC-S and leave the discussions about FEC-S in Chapter 6.

The EDC-S algorithm works in the following manner: After scheduling, the file to be transmitted is first segmented into smaller blocks of pre-defined length (which can be assumed to be much larger than the header size), and each block is encoded with an error

detection code. Also, each block is assigned a block number, so that when the block is in error, the receiver and transmitter both know which block to retransmit. These blocks are then transmitted in sequence via OFS. After the whole flow is received, if at least one bit is found erroneous in any of the blocks, the receiver requests retransmission of the erroneous block(s) via OFS or EPS. If retransmissions are via OFS, the previous steps are repeated until all blocks are error-free. In the case that all retransmissions are via EPS, the retransmitted blocks have to be packetized into IP packets, framed, and then transmitted via EPS. For transmissions via EPS, we assume TCP is used and leave analyses of other protocols over EPS for future work.

For retransmissions, it is also possible to adaptively choose between OFS and EPS depending on the conditions at the time of retransmissions, such as the amount of retransmitted data and the network loading factor. We will suggest the optimal strategy to use for different situations after comparing all OFS protocols with TCP in Chapter 7.

Similar to the case of EDBC protocol, we divide the overall EDC-S protocol into three phases: preparation, transmission, and conclusion. We will adopt the same assumptions and definitions as that of EDC-NS algorithm.

The detailed algorithm description and flowchart are discussed in Section 5.1. The performance analysis of EDC-S algorithm with retransmissions via OFS is done in Section 5.2.

5.1 EDC-S Algorithm Description and Flowchart

The detailed three-phase EDC-S algorithm is described below (note that Phase I and Phase II of the EDC-S algorithm is exactly the same with that of EDC-NS in Chapter 4):

Phase I: Preparation

1. TCP connections between A and A 's scheduler, between A 's scheduler and B 's scheduler, between B 's scheduler and B , and between B and A are established in sequence in this step.
 - 1a. A first establishes a TCP connection with A 's (ingress) scheduler at the network node connecting A 's MAN and the WAN. A then sends an informative message to its scheduler, saying that A wants to send data to B via OFS.
 - 1b. A 's scheduler then establishes a TCP connection with B 's (egress) scheduler at the node connecting the WAN and B 's MAN. A 's scheduler then sends a message to B 's scheduler, saying that A wants to send data to B via OFS.
 - 1c. B 's scheduler establishes a TCP connection with B and sends the same message in the same way as above.
 - 1d. B then establishes a TCP connection with A .
2. When the TCP connection between B and A is established, A requests the scheduler to set up a forward path by telling the scheduler the length of the flow that needs to be transmitted and other necessary information.

The scheduler then reserves a wavelength channel from A to B , whenever available, for the duration of the flow (possibly with some guard time), and tells A and B the reserved OFS channel, planned start time, duration of the channel reservation and the recommended segmentation size. Assume all exchanges of messages in the scheduling

process happen on the OFS control plane, which can be EPS. The detailed scheduling algorithm is similar to the one given in the last chapter expect for the announcement of segmentation size by the scheduler.

Phase II: Transmission

3. At the start time specified by the scheduler, *A* starts transmitting the encoded flow to *B* via the reserved OFS channel, i.e. the forward path previously set up.
4. Upon decoding the data received from *A*, *B* recognizes the start-of-flow. If *B* does not see start-of-flow within the OFS propagation delay plus some guard time after the start time, *B* timeouts and closes its TCP connections with *A* and scheduler. When *A* detects that *B* closes its TCP connection, *A* closes its TCP connection with scheduler. At the same time, *B* clears all data it previously received from *A*. Go back to Step 1.

Phase III: Conclusion

5. *B* continues decoding the received flow, and stores a copy of the file after decoding. At the same time, *B* also checks whether there is any error in the received flow by using the error detection code.
 - 6a. If any error is detected in any block, *B* records down the block number. After the last block is decoded, *B* sends a NACK message to *A* via EPS that contains block numbers of all erroneous blocks. The NACK message says that the OFS data previously received by *B* are erroneous, and the blocks with the specified numbers need to be retransmitted. Upon receiving the NACK, *A* sends back a handshake message via EPS to *B*, saying that it received the NACK message.

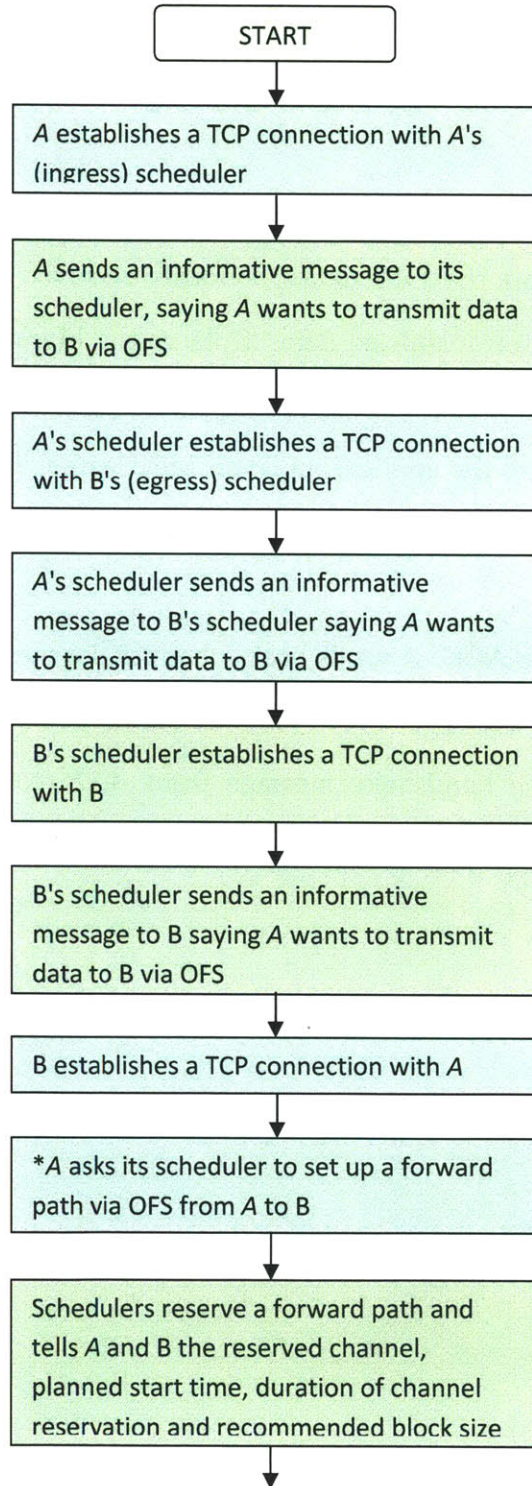
For retransmissions via OFS, A requests the scheduler to reserve a wavelength channel for retransmissions of erroneous blocks. Go back to Step 1. For retransmissions via EPS, A retransmits the data to B using TCP via EPS.

6b. If no error is detected for all received blocks, B sends back an ACK message to A via EPS to confirm that the OFS data previously received by B are correct. If the blocks received by B are retransmitted data, B places the blocks in the correct positions of the original file, and passes the combined file to the application layer. Otherwise, B passes the stored file directly to the application layer. Meanwhile, B closes the TCP connection with the scheduler.

Upon receiving the ACK, A sends back a handshake message via EPS to B , saying that it received the ACK message. Meanwhile A closes TCP connections with the scheduler. Upon receiving the handshake message from A , B closes its TCP connection with A . Algorithm terminates.

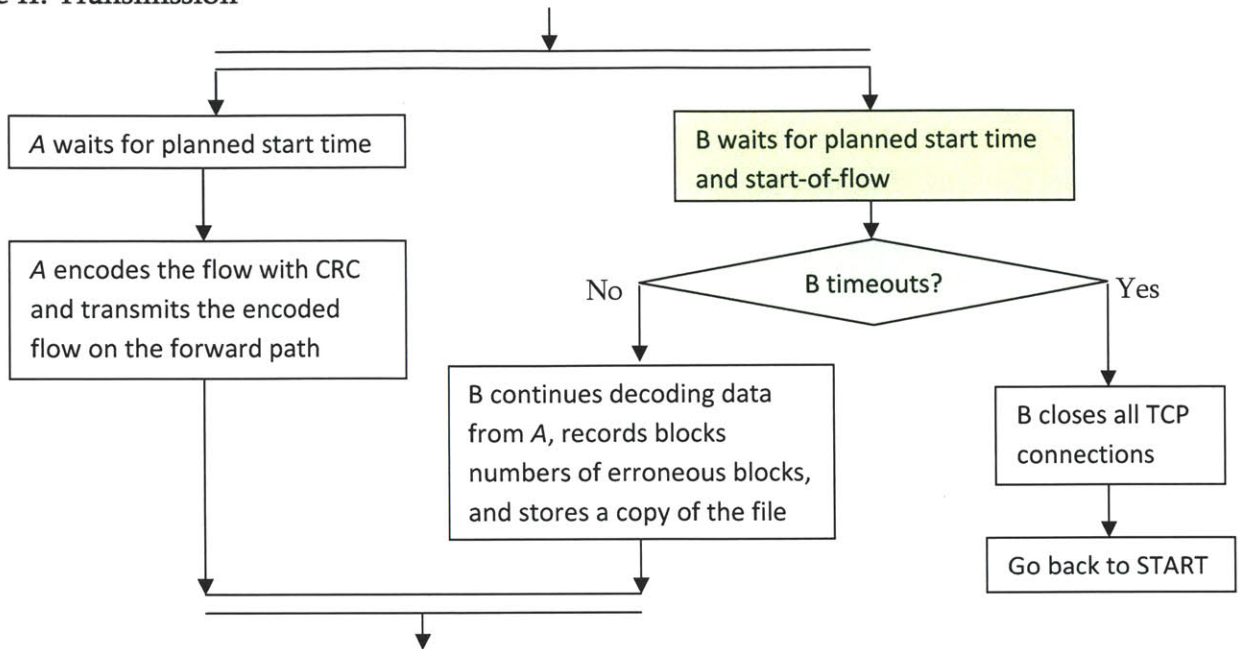
Flowchart:

Phase I: Preparation



* This is the step to return to when there is error(s) in received block(s) in Phase III.

Phase II: Transmission



Phase III: Conclusion

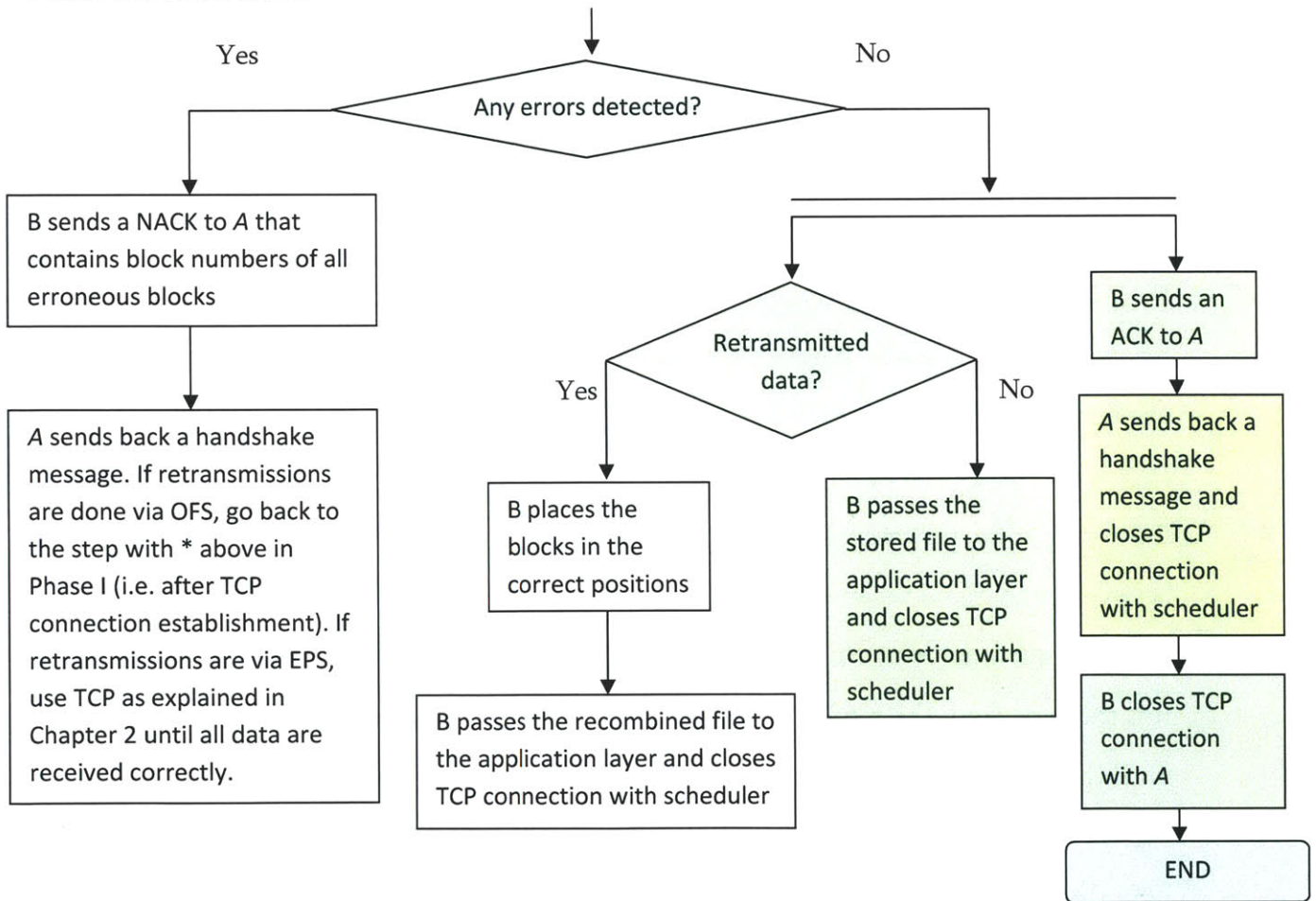


Figure 5.1: EDC-S protocol flowchart. The "green path" is the default path if there is no error.

5.2 Performance Analysis of EDC-S with Retransmissions via OFS (EDC-S (OFS))

We denote the EDC-S algorithm with retransmissions via OFS by EDC-S (OFS) and analyze its delay performance below.

5.2.1 Delay of EDC-S (OFS)

For a flow of length L_f bits and OFS link rate R_{OFS} bps, when there is no transmission error of any kind (i.e. perfect sender, channel, receiver conditions), the timeline follows the "green path" in Fig. 5.1. The overall delay of the flow transmission can be expressed as

$$D_f = D_{TCP} + (D_{pg}^{sch} + D_q) + \left(T_{pg}^{OFS} + T_{pc}^{OFS} + \frac{L_f}{R_{OFS}} \right) + \tau \quad (5.1)$$

where all the terms used on the right hand side of the above expression have similar interpretations to those used in EDC-NS protocol in Section 4.2, i.e.

- D_{TCP} is the time taken to establish all three TCP connections between the sender, receiver and scheduler before scheduling processes can begin (please see a detailed explanation about this term in Section 2.2.1),
- D_{pg}^{sch} is the delay (including the transmission delay, propagation delay, queuing delay at the router, and processing delay) in the process of scheduling (at least one EPS RTT between sender and receiver),
- D_q is the queuing delay for an open channel as determined by the reservation scheduler. It is the time from the moment the scheduler receives the request for channel reservation until the moment the flow transmission starts on the reserved channel,
- T_{pg}^{OFS} is the propagation delay between the sender and receiver defined by the ratio of the fiber distance to the speed of light (note that this is different from T_{pg}^{EPS} which is defined

as the one-way EPS delay that consists of transmission delay, propagation delay, queuing delay and processing delay),

- T_{pc}^{OFS} is the OFS processing delays at the sender and/or receiver added to the overall delay, and is neglected in this thesis (see Section 3.2), and
- τ is the time taken for the receiver to pass the decoded data to the application layer.

When we ignore the processing delay T_{pc}^{OFS} and τ , the delay can be approximated by

$$D_f \approx D_{TCP} + (D_{pg}^{sch} + D_q) + \left(T_{pg}^{OFS} + \frac{L_f}{R_{OFS}} \right) \quad (5.2)$$

Let the segmented block length be L_b bits (useful data) and the overhead for each block be L_h bits. The actual size of each block is then $L'_b = L_b + L_h$. Let the number of blocks for a flow of length L_f bits be $n_b = \lceil \frac{L_f}{L_b} \rceil$. Then the total number of bits actually transferred is $n_b L'_b = n_b(L_b + L_h) \geq L_f$ with equality if and only if L_b is divisible by L_f and $L_h = 0$.

With bit error rate p_e , the block error rate $p_{e,b}$ can be expressed by

$$p_{e,b} = 1 - (1 - p_e)^{L'_b} = 1 - (1 - p_e)^{L_b + L_h} \approx (L_b + L_h)p_e \quad (5.3)$$

where the approximation holds when $L'_b p_e \ll 1$ (e.g. $L'_b p_e \leq 0.01$), which we will assume to be the case.

The total expected number of blocks that actually need to be transmitted is $\frac{n_b}{1 - p_{e,b}}$. The probability that there is at least one block in error for the first transmission is

$$1 - (1 - p_{e,b})^{n_b} \approx n_b p_{e,b} \approx n_b L'_b p_e \approx L_f p_e \quad (5.4)$$

when $n_b p_{e,b} \ll 1$, $L_b p_e \ll 1$ and $L_b \gg L_h$, where L_h is the header length. However, in the case where $n_b p_{e,b} \ll 1$ is not true, the above approximation fails. This is the case where we need segmentation, and where EDC-S has advantages over EDC-NS.

Let N_t be a random variable that denotes the total number of transmissions (including retransmissions) required to make all blocks error-free. It can be shown that (see Appendix A) the probability of transmitting once (i.e. without retransmissions) is

$$P(N_t = 1) = (1 - p_{e,b})^{n_b} \quad (5.5)$$

The probability of transmitting twice is

$$P(N_t = 2) = (1 - p_{e,b})^{n_b} [(1 + p_{e,b})^{n_b} - 1] \quad (5.6)$$

and in general for $k \geq 2$,

$$P(N_t = k) = (1 - p_{e,b})^{n_b} [(1 + p_{e,b} + p_{e,b}^2 + \dots + p_{e,b}^{k-1})^{n_b} - (1 + p_{e,b} + p_{e,b}^2 + \dots + p_{e,b}^{k-2})^{n_b}] \quad (5.7)$$

$$= (1 - p_{e,b})^{n_b} \left[\left(\sum_{i=0}^{k-1} p_{e,b}^i \right)^{n_b} - \left(\sum_{i=0}^{k-2} p_{e,b}^i \right)^{n_b} \right] \quad (5.8)$$

The expected number of transmissions is then

$$E[N_t] = \sum_{k=1}^{\infty} k P(N_t = k) \quad (5.9)$$

$$= (1 - p_{e,b})^{n_b} + \sum_{k=2}^{\infty} k (1 - p_{e,b})^{n_b} \left[\left(\sum_{i=0}^{k-1} p_{e,b}^i \right)^{n_b} - \left(\sum_{i=0}^{k-2} p_{e,b}^i \right)^{n_b} \right] \quad (5.10)$$

$$= (1 - p_{e,b})^{n_b} \left\{ 1 + \sum_{k=2}^{\infty} k \left[\left(\sum_{i=0}^{k-1} p_{e,b}^i \right)^{n_b} - \left(\sum_{i=0}^{k-2} p_{e,b}^i \right)^{n_b} \right] \right\} \quad (5.11)$$

where $p_{e,b} = 1 - (1 - p_e)^{L_b}$ and $n_b = \left\lceil \frac{L_f}{L_b} \right\rceil$. Equivalently,

$$E[N_t] = (1 - p_e)^{\lfloor L_b + L_h \rfloor} \times \left\{ 1 + \sum_{k=2}^{\infty} k \left[\left(\sum_{i=0}^{k-1} (1 - (1 - p_e)^{L_b + L_h})^i \right)^{\lfloor \frac{L_f}{L_b} \rfloor} - \left(\sum_{i=0}^{k-2} (1 - (1 - p_e)^{L_b + L_h})^i \right)^{\lfloor \frac{L_f}{L_b} \rfloor} \right] \right\} \quad (5.12)$$

Fig. 5.2 shows plots of $E[N_t]$ versus L_b for $L_h = 0$, $L_f = 10^{10}$, 10^{11} , 10^{12} , and 10^{13} bits, and $p_e = 10^{-10}$, 10^{-8} and 10^{-6} . The file sizes and BERs are chosen so that $p_e L_f \ll 1$ is no longer valid, and EDC-NS in Chapter 4 no longer works well in all these cases.

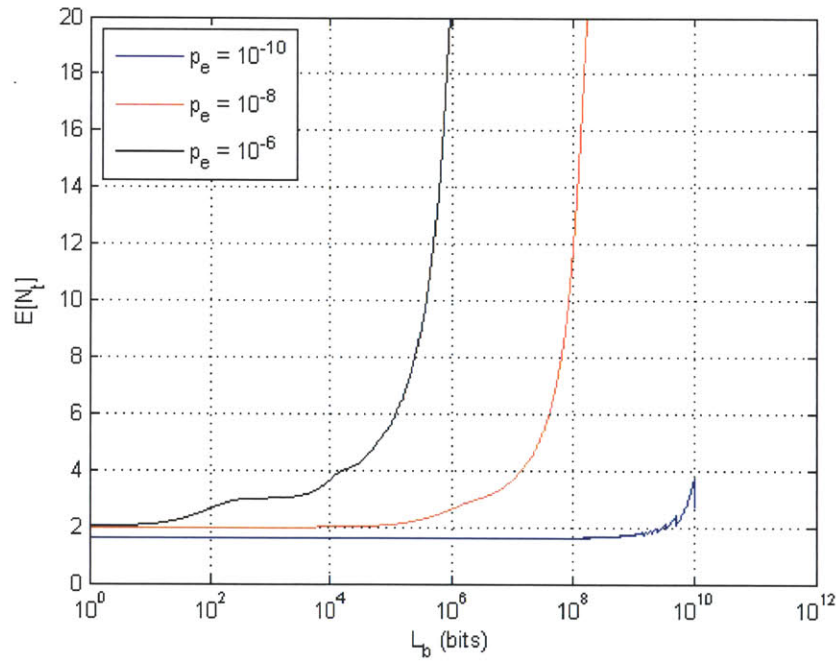


Figure 5.2 (a): Plot of $E[N_t]$ vs. L_b for $L_f = 10^{10}$ bits, $L_h = 0$. The blue, red and black curves (from right to left) are for cases where $p_e = 10^{-10}$, 10^{-8} and 10^{-6} respectively.

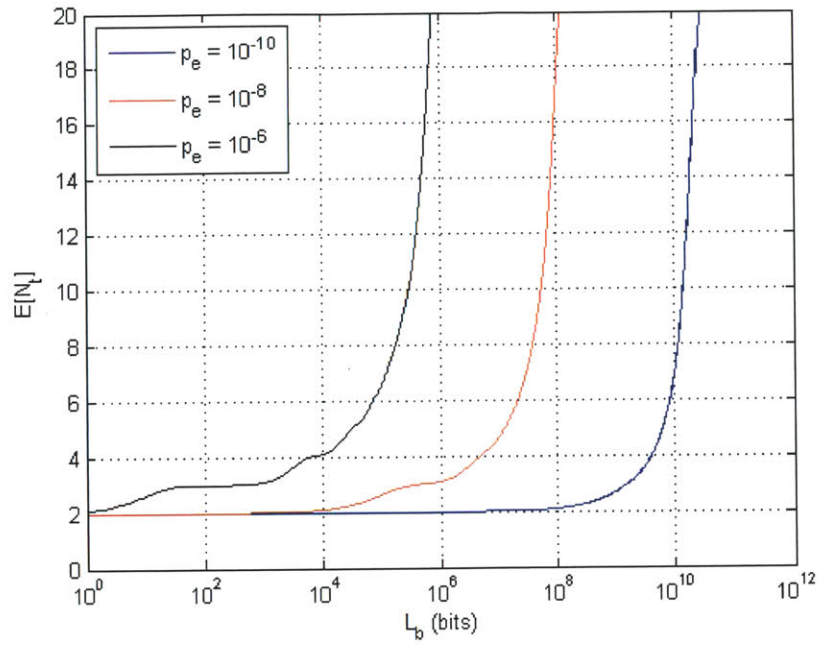


Figure 5.2 (b): Plot of $E[N_t]$ vs. L_b for $L_f = 10^{11}$ bits, $L_h = 0$. The blue, red and black curves (from right to left) are for cases where $p_e = 10^{-10}$, 10^{-8} and 10^{-6} respectively.

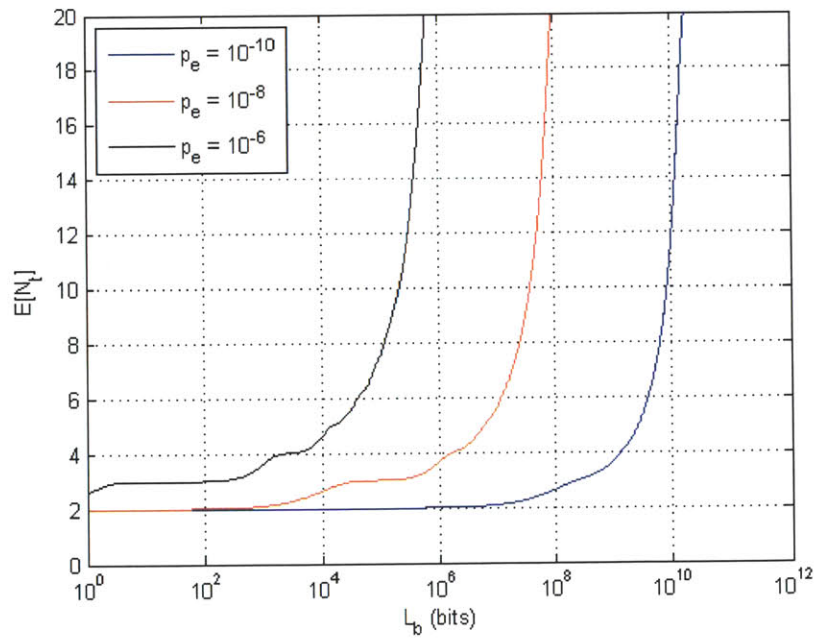


Figure 5.2 (c): Plot of $E[N_t]$ vs. L_b for $L_f = 10^{12}$ bits, $L_h = 0$. The blue, red and black curves (from right to left) are for cases where $p_e = 10^{-10}$, 10^{-8} and 10^{-6} respectively.

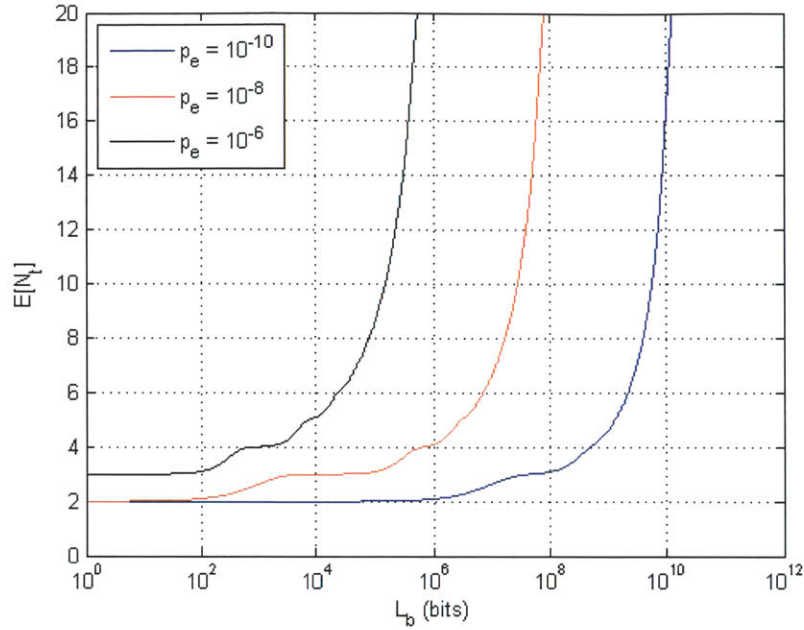


Figure 5.2 (d): Plot of $E[N_t]$ vs. L_b for $L_f = 10^{13}$ bits, $L_h = 0$. The blue, red and black curves (from right to left) are for cases where $p_e = 10^{-10}$, 10^{-8} and 10^{-6} respectively.

In Fig. 5.2 (a)-(d) when $L_h = 0$, for all three BERs $p_e = 10^{-10}$, 10^{-8} and 10^{-6} , the total expected number of transmissions $E[N_t]$ is monotonically non-decreasing with increasing block size L_b . The reason for this trend is that as the block size increases, the block error rate also increases, resulting in potentially more retransmissions and a non-decreasing total expected number of transmissions. $E[N_t]$ goes up especially quickly when $p_e L_b \rightarrow 1$, which is also expected for the same reasoning as above. For practical protocol design, the results tend to imply that we should use smaller block size for the sake of minimizing $E[N_t]$ in order to minimize the total delay when $L_h = 0$.

It can also be seen from Fig. 5.2 (a)-(d) that for high BER like $p_e = 10^{-6}$, as the file size increases from 10^{10} bits to 10^{13} bits, the minimum $E[N_t]$ goes up from 2 in 5.2 (a) to 3 in 5.2 (d). This is because as the file size increases, for the same block size, there are more blocks to be transmitted and retransmitted, resulting in a larger expected number of transmissions.

Such increase is not so obvious for lower bit error rate (e.g. 10^{-8} and 10^{-10}), as $p_e L_f$ is not too large compared to the case of higher bit error rate, which potentially results in less retransmissions.

We will now analyze the practical case when $L_h > 0$. Fig. 5.3 shows the plots under the same conditions as that in Fig. 5.2 except that $L_h = 320$ bits. Note that the number "320" comes from IPv6 in which the header size is 40 bytes = 320 bits. For OFS, the header size may not necessarily be 320 bits, but these plots can still serve as a good illustration of what happens when $L_h > 0$.

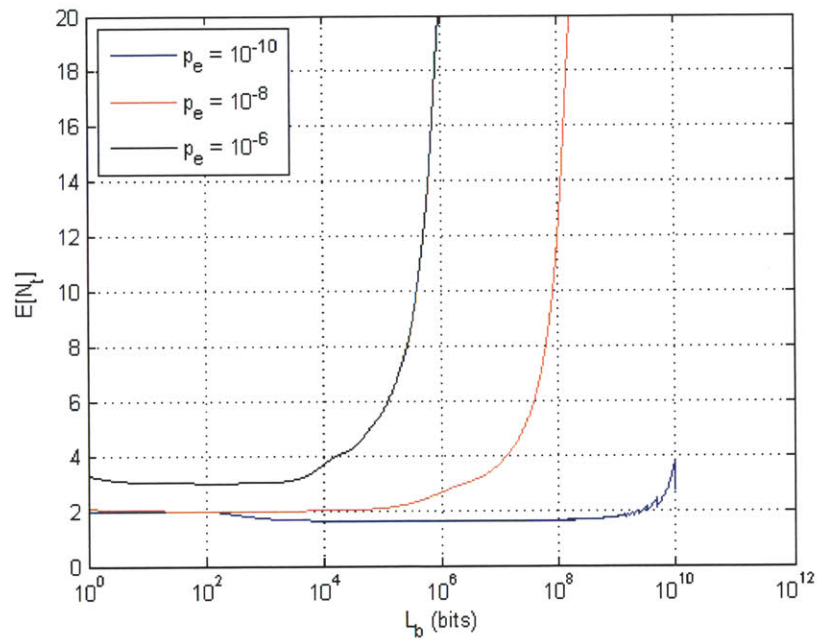


Figure 5.3 (a): Plot of $E[N_t]$ vs. L_b for $L_f = 10^{10}$ bits, $L_h = 320$ bits. The blue, red and black curves (from right to left) are for cases where $p_e = 10^{-10}$, 10^{-8} and 10^{-6} respectively.

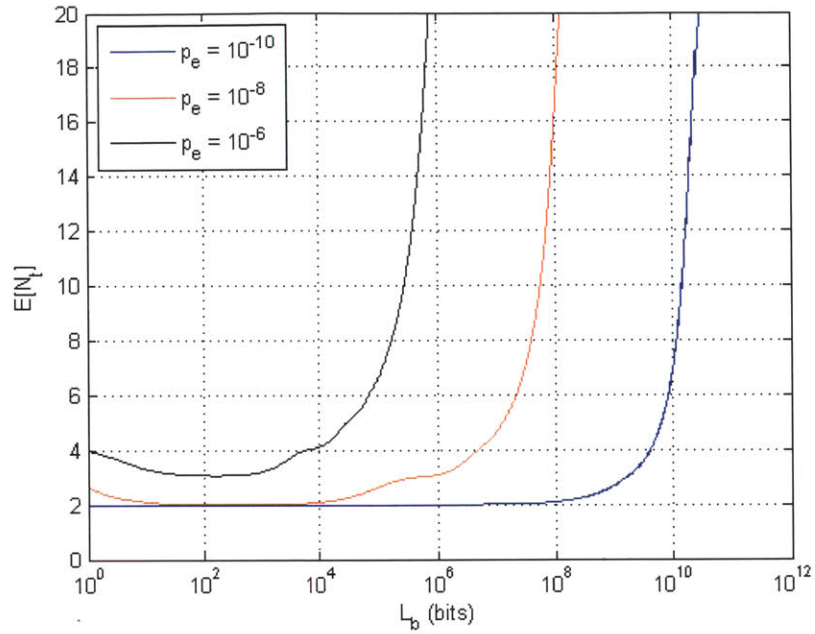


Figure 5.3 (b): Plot of $E[N_t]$ vs. L_b for $L_f = 10^{11}$ bits, $L_h = 320$ bits. The blue, red and black curves (from right to left) are for cases where $p_e = 10^{-10}$, 10^{-8} and 10^{-6} respectively.

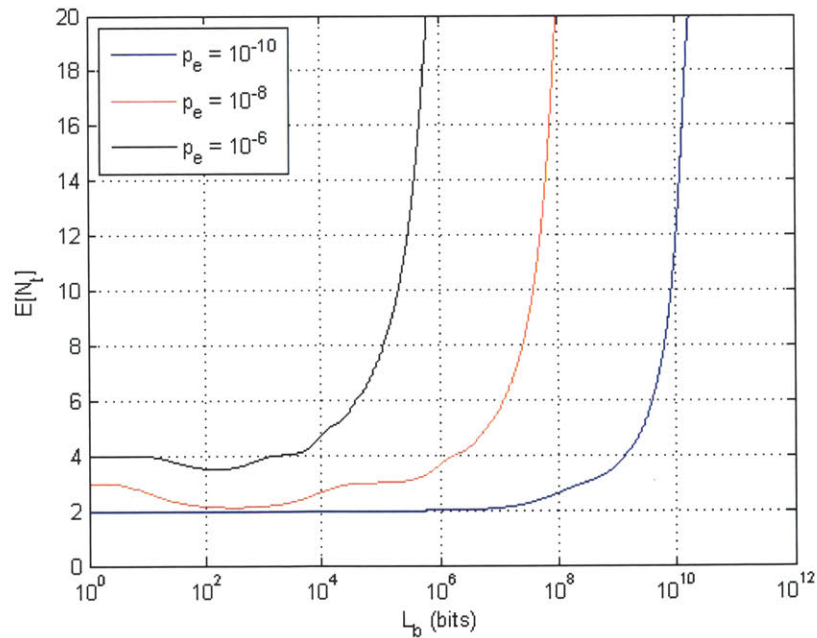


Figure 5.3 (c): Plot of $E[N_t]$ vs. L_b for $L_f = 10^{12}$ bits, $L_h = 320$ bits. The blue, red and black curves (from right to left) are for cases where $p_e = 10^{-10}$, 10^{-8} and 10^{-6} respectively.

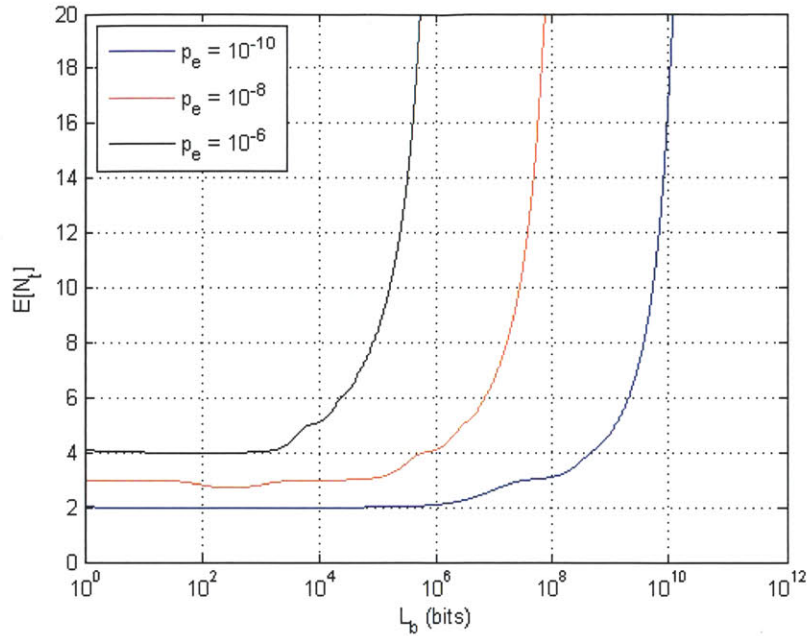


Figure 5.3 (d): Plot of $E[N_t]$ vs. L_b for $L_f = 10^{13}$ bits, $L_h = 320$ bits. The blue, red and black curves (from right to left) are for cases where $p_e = 10^{-10}$, 10^{-8} and 10^{-6} respectively.

In Fig. 5.3 (a)-(d) when $L_h = 320$ bits, for all three BERs $p_e = 10^{-10}$, 10^{-8} and 10^{-6} , the total expected number of transmissions $E[N_t]$ first decreases and then increases as the block size L_b increases. The reason for this trend is that as the block size is small, the header size is relatively large and plays an important role in the retransmission process. For example, the case of $L_b = 30$ bits in Fig. 5.3 corresponds to the case of $L_b = 350$ bits in Fig. 5.2. At some block size, the total number of blocks is reduced while the block error rate is increased, but the overall effect is that the required number of transmissions is decreased. But as the block size increases further, the increase in block error rate starts to dominate more than the reduction in the number of blocks.

It can also be seen from Fig. 5.3 (a)-(d) that for high BER like $p_e = 10^{-6}$, as the file size increases from 10^{10} bits to 10^{13} bits, the minimum $E[N_t]$ goes up from 3 in 5.3 (a) to 4 in 5.3 (d), for similar reasons to that in the case of Fig. 5.2. For practical protocol design of EDC-S,

to minimize the total delay, we should try to find the block size that gives a relatively small value for the total expected number of transmissions. However, it should be noted that the block size that minimizes $E[N_t]$ does not necessarily minimize the total delay, as we have also take into consideration the extra redundancy added because of the non-zero header size. We will discuss how to minimize the total delay below.

With $p_{e,b} \geq 0$, the total expected number of blocks actually transmitted is $n_b/(1 - p_{e,b})$. The expected total delay is then

$$E[D_t] \approx D_{TCP} + E[N_t](D_{pg}^{sch} + D_q + T_{pg}^{OFS}) + (E[N_t] - 1)T_{pg}^{EPS} + \frac{L_b + L_h}{R_{OFS}} \cdot \frac{n_b}{1 - p_{e,b}} \quad (5.13)$$

$$= D_{TCP} + E[N_t](D_{pg}^{sch} + D_q + T_{pg}^{OFS} + T_{pg}^{EPS}) - T_{pg}^{EPS} + \frac{L_b + L_h}{R_{OFS}} \cdot \frac{n_b}{1 - p_{e,b}} \quad (5.14)$$

where $E[N_t](D_{pg}^{sch} + D_q + T_{pg}^{OFS})$ in (5.13) is the delay caused by scheduling and queuing, $(E[N_t] - 1)T_{pg}^{EPS}$ is the delay caused by sending back the NACKs from the receiver when one or more blocks are erroneous, and $\frac{L_b + L_h}{R_{OFS}} \cdot \frac{n_b}{1 - p_{e,b}}$ is the delay caused by flow transmission(s), including both the first transmission and possible retransmissions.

Substituting $p_{e,b} = 1 - (1 - p_e)^{L_b}$, $n_b = \left\lceil \frac{L_f}{L_b} \right\rceil$ and expression of $E[N_t]$ in (5.12) into (5.14), we obtain

$$E[D_t] \approx D_{TCP} - T_{pg}^{EPS} + \frac{L_b + L_h}{R_{OFS}} \cdot \frac{\left\lceil \frac{L_f}{L_b} \right\rceil}{(1 - p_e)^{L_b}} + (D_{pg}^{sch} + D_q + T_{pg}^{OFS} + T_{pg}^{EPS}) \times (1 - p_e)^{(L_b + L_h) \left\lceil \frac{L_f}{L_b} \right\rceil} \times \left\{ 1 + \sum_{k=2}^{\infty} k \left[\left(\sum_{i=0}^{k-1} (1 - (1 - p_e)^{L_b + L_h})^i \right)^{\left\lceil \frac{L_f}{L_b} \right\rceil} - \left(\sum_{i=0}^{k-2} (1 - (1 - p_e)^{L_b + L_h})^i \right)^{\left\lceil \frac{L_f}{L_b} \right\rceil} \right] \right\} \quad (5.15)$$

In the special case where there is no segmentation, i.e. $L_b = L_f$, it can be shown that (5.15) can be simplified to

$$E[D_t] \approx D_{TCP} + (1 - p_e)^{L_f} \left\{ 1 + \sum_{k=2}^{\infty} k \left[\sum_{i=0}^{k-1} (1 - (1 - p_e)^{L_f})^i - \sum_{i=0}^{k-2} (1 - (1 - p_e)^{L_f})^i \right] \right\} (D_{pg}^{sch} + D_q + T_{pg}^{OFS} + T_{pg}^{EPS}) - T_{pg}^{EPS} + \frac{L_f}{R_{OFS}(1 - p_e)^{L_f}} \quad (5.16)$$

$$\approx D_{TCP} + (1 - p_e)^{L_f} \left\{ 1 + \sum_{k=2}^{\infty} k (1 - (1 - p_e)^{L_f})^{k-1} \right\} (D_{pg}^{sch} + D_q + T_{pg}^{OFS} + T_{pg}^{EPS}) - T_{pg}^{EPS} + \frac{L_f}{R_{OFS}(1 - p_e)^{L_f}} \quad (5.17)$$

$$\approx D_{TCP} + \frac{D_{pg}^{sch} + D_q + T_{pg}^{OFS} + T_{pg}^{EPS}}{(1 - p_e)^{L_f}} - T_{pg}^{EPS} + \frac{L_f}{R_{OFS}(1 - p_e)^{L_f}} \quad (5.18)$$

$$= D_{TCP} + \frac{D_{pg}^{sch} + D_q + T_{pg}^{OFS} + T_{pg}^{EPS} + \frac{L_f}{R_{OFS}}}{(1 - p_e)^{L_f}} - T_{pg}^{EPS} \quad (5.19)$$

which reduces to a delay expression similar to (4.8) in the case of no segmentation in Section 4.2.1. Note, however, that the above expression is slightly different from (4.8) because of the TCP connection establishment and shutdown methods. That is, in Section 4.2.1, the TCP connection is shut down and re-established between retransmissions if the received flow is erroneous. But in EDC-S (OFS), TCP connection is maintained throughout the session until the flow is correctly received at the receiver even if there are retransmissions. Nevertheless, we can still treat EDC-NS as a special case of EDC-S (OFS) when the segment size is the flow size.

With the parameter values in Table 3.1, we can plot $E[D_t]$ vs. L_b for $L_f = 10^{10}$ bits and $p_e = 10^{-10}$, 10^{-8} and 10^{-6} in Fig. 5.4 (a) when $L_h = 0$, and in Fig. 5.4 (b) when $L_h = 320$ bits. Fig. 5.5-5.7 show the same plots except that $L_f = 10^{11}$, 10^{12} and 10^{13} bits.

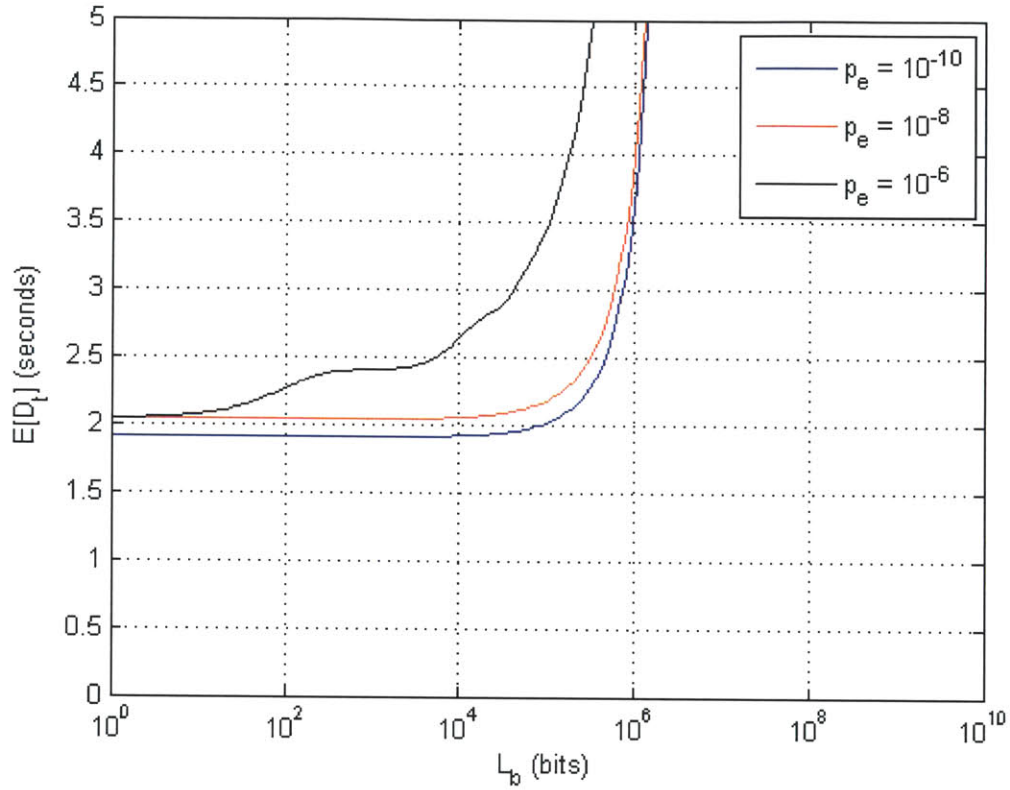


Figure 5.4 (a): Plot of $E[D_t]$ vs. L_b for $L_h = 0$, $L_f = 10^{10}$ bits, and $p_e = 10^{-10}$, 10^{-8} and 10^{-6} . $T_{pg}^{EPS} = 45.5\text{ms}$, $T_{pg}^{OFS} = 25\text{ms}$, $R_{OFS} = 10\text{Gbps}$, $D_{TCP} = 8T_{pg}^{EPS}$, $D_{pg}^{sch} = 2T_{pg}^{EPS}$, and $D_q = 0.089\text{s}$. Note that the loading factor is assumed to be 0.5 with average transaction length of 1s, resulting in a queuing delay of approximately 0.089s. Refer to Fig. 3.5 for a plot of normalized total delay vs. loading factor.

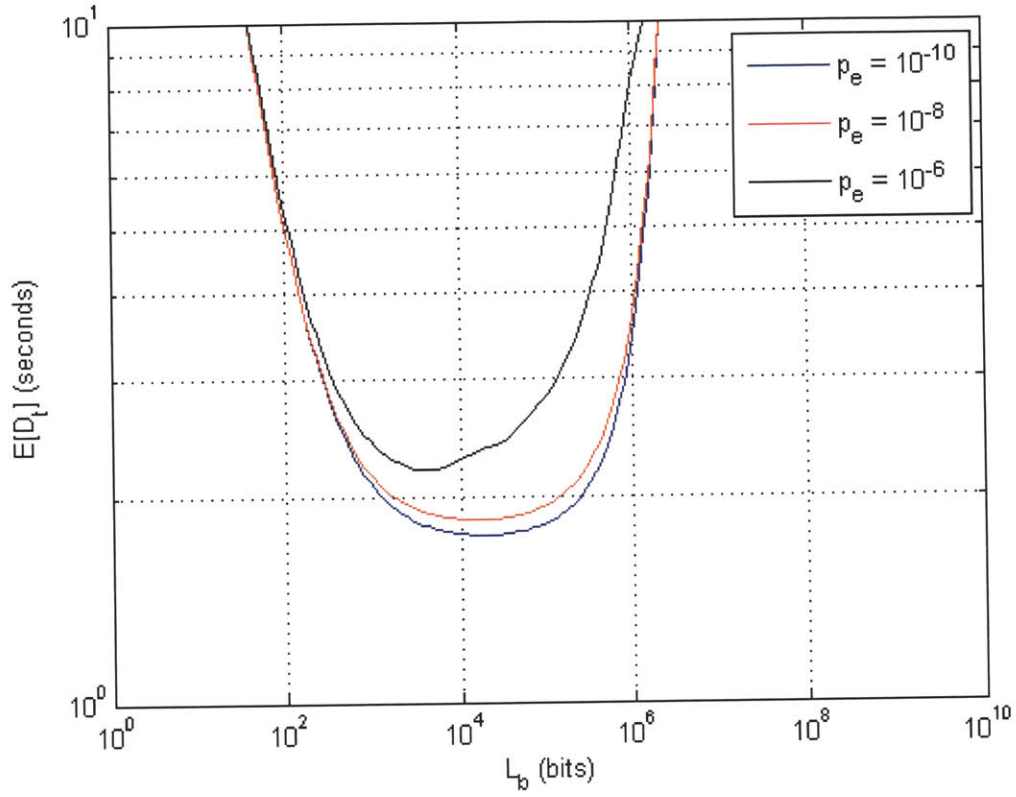


Figure 5.4 (b): Plot of $E[D_t]$ vs. L_b for $L_h = 320$ bits, $L_f = 10^{10}$ bits, and $p_e = 10^{-10}$, 10^{-8} and 10^{-6} . $T_{pg}^{EPS} = 45.5\text{ms}$, $T_{pg}^{OFS} = 25\text{ms}$, $R_{OFS} = 10\text{Gbps}$, $D_{TCP} = 8T_{pg}^{EPS}$, $D_{pg}^{sch} = 2T_{pg}^{EPS}$, and $D_q = 0.089\text{s}$. Note that the loading factor is assumed to be 0.5 with average transaction length of 1s, resulting in a queuing delay of approximately 0.089s. Refer to Fig. 3.5 for a plot of normalized total delay vs. loading factor.

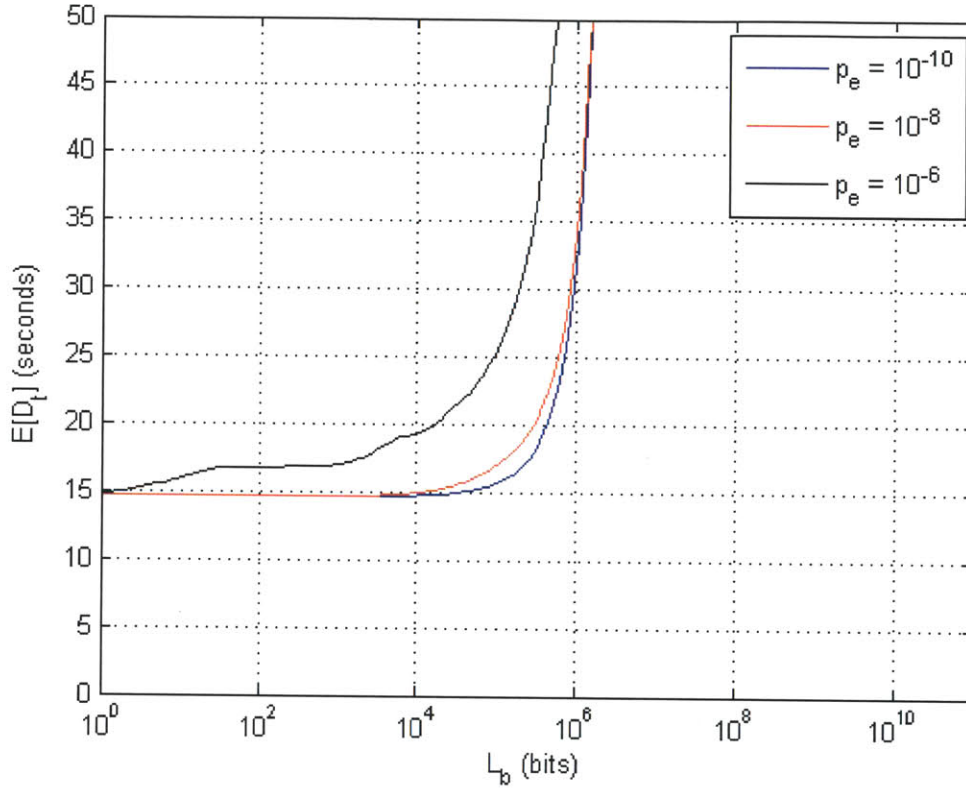


Figure 5.5 (a): Plot of $E[D_t]$ vs. L_b for $L_h = 0$, $L_f = 10^{11}$ bits, and $p_e = 10^{-10}$, 10^{-8} and 10^{-6} . $T_{pg}^{EPS} = 45.5\text{ms}$, $T_{pg}^{OFS} = 25\text{ms}$, $R_{OFS} = 10\text{Gbps}$, $D_{TCP} = 8T_{pg}^{EPS}$, $D_{pg}^{sch} = 2T_{pg}^{EPS}$, and $D_q = 0.89\text{s}$. Note that the loading factor is assumed to be 0.5 with average transaction length of 10s, resulting in a queuing delay of approximately 0.89s. Refer to Fig. 3.5 for a plot of normalized total delay vs. loading factor.

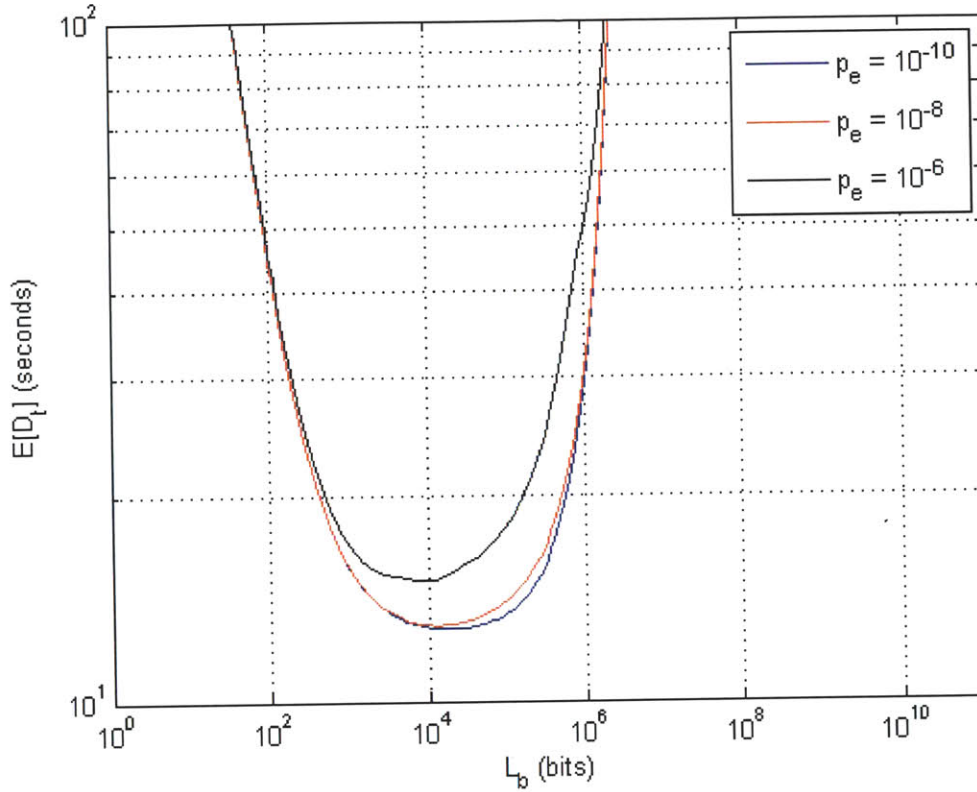


Figure 5.5 (b): Plot of $E[D_t]$ vs. L_b for $L_h = 320$ bits, $L_f = 10^{11}$ bits, and $p_e = 10^{-10}$, 10^{-8} and 10^{-6} . $T_{pg}^{EPS} = 45.5\text{ms}$, $T_{pg}^{OFS} = 25\text{ms}$, $R_{OFS} = 10\text{Gbps}$, $D_{TCP} = 8T_{pg}^{EPS}$, $D_{pg}^{sch} = 2T_{pg}^{EPS}$, and $D_q = 0.89\text{s}$. Note that the loading factor is assumed to be 0.5 with average transaction length of 10s, resulting in a queuing delay of approximately 0.89s. Refer to Fig. 3.5 for a plot of normalized total delay vs. loading factor.

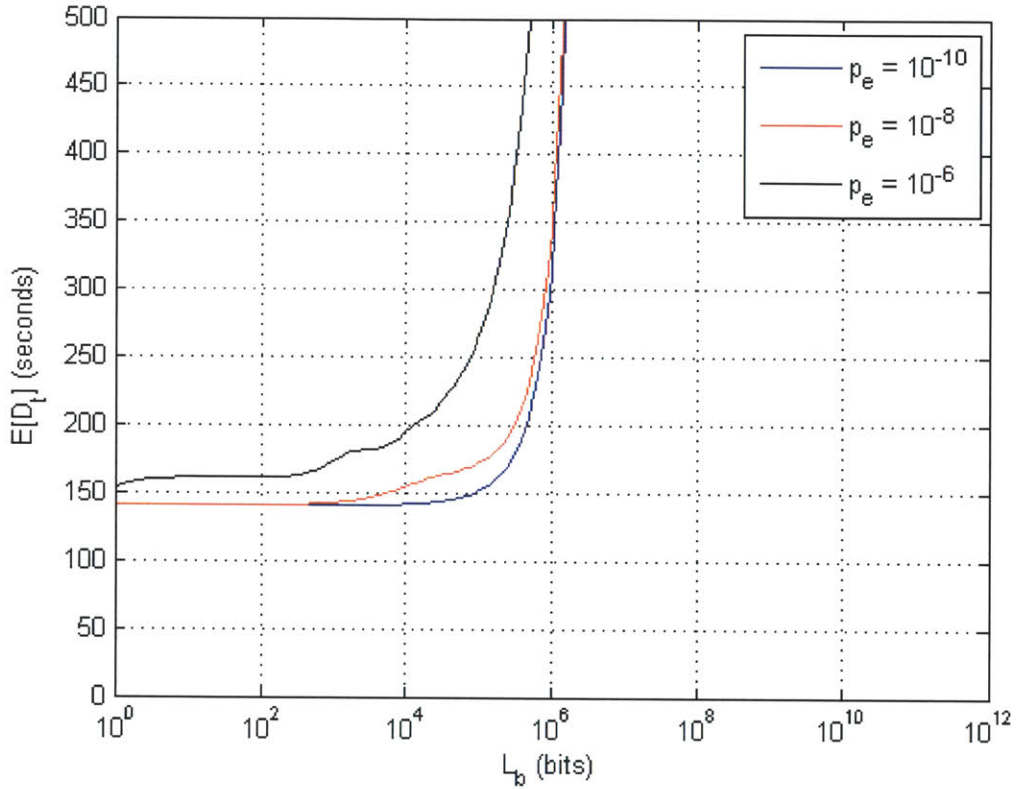


Figure 5.6 (a): Plot of $E[D_t]$ vs. L_b for $L_h = 0$, $L_f = 10^{12}$ bits, and $p_e = 10^{-10}$, 10^{-8} and 10^{-6} . $T_{pg}^{EPS} = 45.5\text{ms}$, $T_{pg}^{OFS} = 25\text{ms}$, $R_{OFS} = 10\text{Gbps}$, $D_{TCP} = 8T_{pg}^{EPS}$, $D_{pg}^{sch} = 2T_{pg}^{EPS}$, and $D_q = 8.9\text{s}$. Note that the loading factor is assumed to be 0.5 with average transaction length of 100s, resulting in a queuing delay of approximately 8.9s. Refer to Fig. 3.5 for a plot of normalized total delay vs. loading factor.

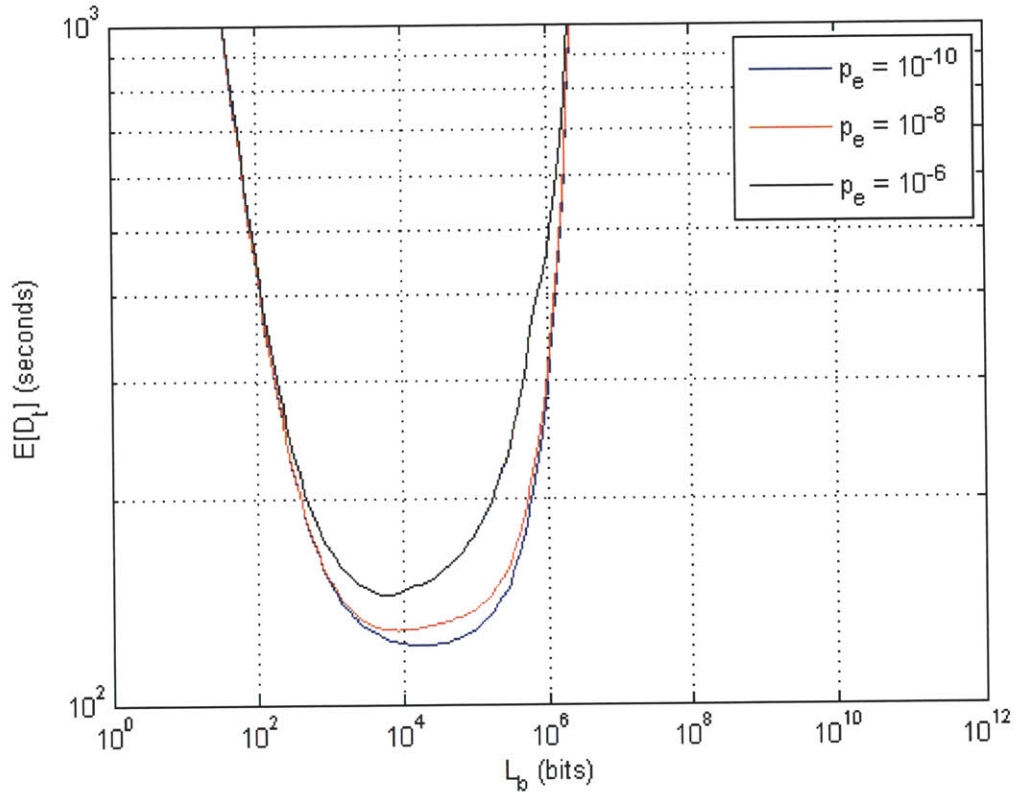


Figure 5.6 (b): Plot of $E[D_t]$ vs. L_b for $L_h = 320$ bits, $L_f = 10^{12}$ bits, and $p_e = 10^{-10}$, 10^{-8} and 10^{-6} . $T_{pg}^{EPS} = 45.5\text{ms}$, $T_{pg}^{OFS} = 25\text{ms}$, $R_{OFS} = 10\text{Gbps}$, $D_{TCP} = 8T_{pg}^{EPS}$, $D_{pg}^{sch} = 2T_{pg}^{EPS}$, and $D_q = 8.9\text{s}$. Note that the loading factor is assumed to be 0.5 with average transaction length of 100s, resulting in a queuing delay of approximately 8.9s. Refer to Fig. 3.5 for a plot of normalized total delay vs. loading factor.

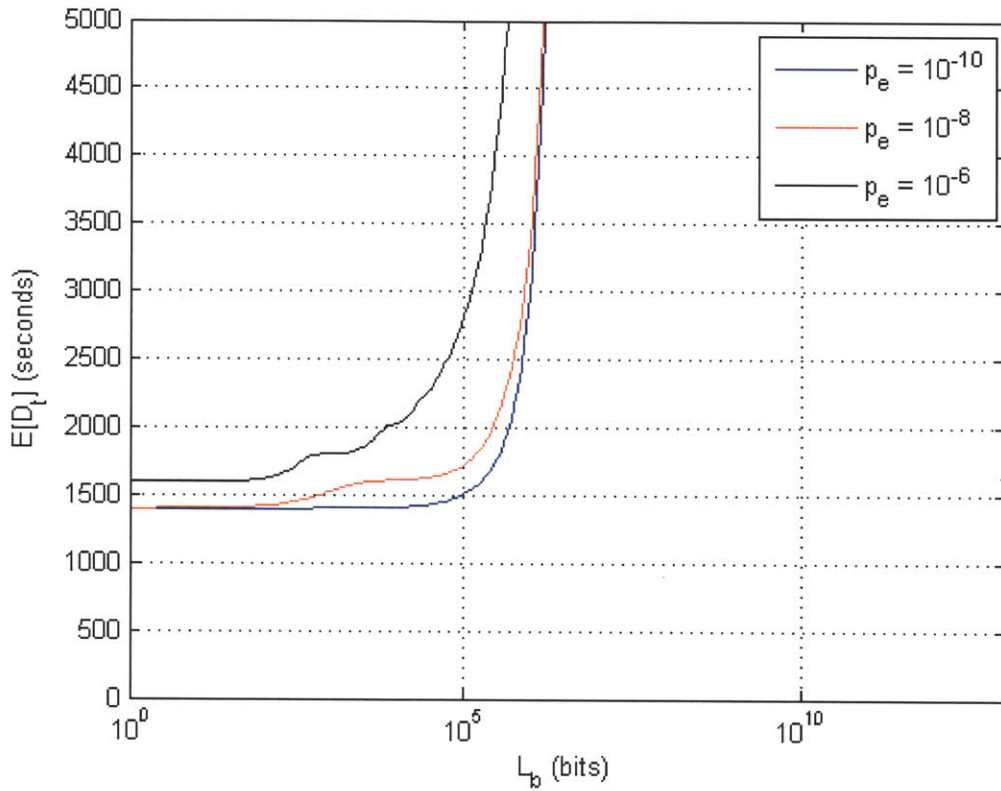


Figure 5.7 (a): Plot of $E[D_t]$ vs. L_b for $L_h = 0$, $L_f = 10^{13}$ bits, and $p_e = 10^{-10}$, 10^{-8} and 10^{-6} . $T_{pg}^{EPS} = 45.5\text{ms}$, $T_{pg}^{OFS} = 25\text{ms}$, $R_{OFS} = 10\text{Gbps}$, $D_{TCP} = 8T_{pg}^{EPS}$, $D_{pg}^{sch} = 2T_{pg}^{EPS}$, and $D_q = 89\text{s}$. Note that the loading factor is assumed to be 0.5 with average transaction length of 1000s, resulting in a queuing delay of approximately 89s. Refer to Fig. 3.5 for a plot of normalized total delay vs. loading factor.

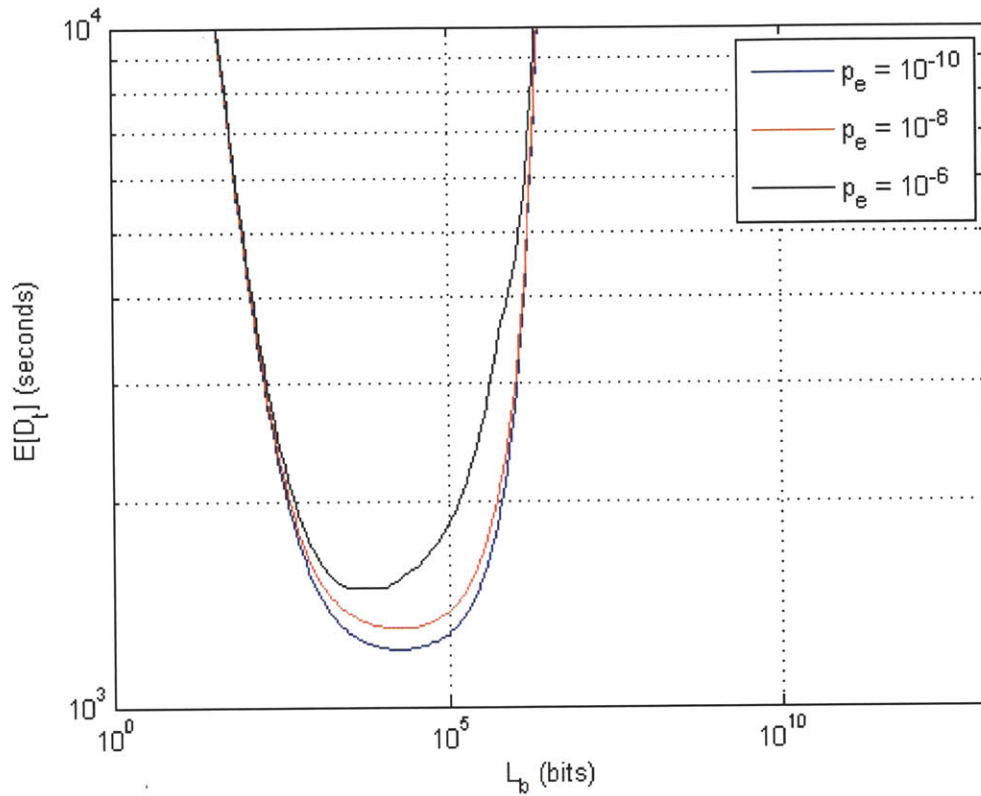


Figure 5.7 (b): Plot of $E[D_t]$ vs. L_b for $L_h = 320$ bits, $L_f = 10^{13}$ bits, and $p_e = 10^{-10}$, 10^{-8} and 10^{-6} . $T_{pg}^{EPS} = 45.5\text{ms}$, $T_{pg}^{OFS} = 25\text{ms}$, $R_{OFS} = 10\text{Gbps}$, $D_{TCP} = 8T_{pg}^{EPS}$, $D_{pg}^{sch} = 2T_{pg}^{EPS}$, and $D_q = 89\text{s}$. Note that the loading factor is assumed to be 0.5 with average transaction length of 1000s, resulting in a queuing delay of approximately 89s. Refer to Fig. 3.5 for a plot of normalized total delay vs. loading factor.

Fig. 5.4 (a), 5.5 (a), 5.6 (a) and 5.7 (a) correspond to the case where $L_h = 0$. There is no overhead delay introduced and the delay is approximately inversely proportional to $1 - p_{e,b} = (1 - p_e)^{L_b}$, which is monotonically decreasing with increasing L_b . Therefore, the delay is monotonically increasing with increasing L_b and starts to increase dramatically when $L_b \rightarrow 1/p_e$. Otherwise the curve is almost flat on the left part.

It can be seen from Fig. 5.4 (b), 5.5 (b), 5.6 (b), and 5.7 (b) that with non-zero overhead, as the block size increases, the delay first decreases, then keeps relatively flat, and then increases again. The reason is that when L_b is small, $n_b = \left\lceil \frac{L_f}{L_b} \right\rceil$ is large, and the total overhead $L_h n_b = L_h \left\lceil \frac{L_f}{L_b} \right\rceil$ is also large and cannot be ignored. This introduces large overhead delay added to the data delay without retransmissions. When L_b gets larger, $n_b = \left\lceil \frac{L_f}{L_b} \right\rceil$ gets smaller and the overhead $L_h n_b = L_h \left\lceil \frac{L_f}{L_b} \right\rceil$ also gets smaller. This corresponds to the left part of the plot in Fig. 5.4 (a). On the other hand, when L_b becomes so large that $p_{e,b} = 1 - (1 - p_e)^{L_b + L_h}$ becomes large, the total number of retransmissions also becomes large, leading to a large delay. This corresponds to the right part of the plot in Fig. 5.4 (a) and Fig. 5.4 (b). The central region corresponds to the case where $L_b \gg L_h$ and $p_{e,b} = 1 - (1 - p_e)^{L_b + L_h} \ll 1$.

Also, it is interesting to observe that when the header size is 320 bits (in fact for other smaller header size too) the optimal block size L_b is around 10^4 bits, which is very close to the maximum EPS packet size 1.2×10^4 bits, but does not increase much until after 10^5 bits. So pragmatically we will use 10^5 bits as the segmentation length to reduce the overhead for segmentation which is not accounted for in these analyses.

5.3 Summary of Chapter 5

In this chapter, we discussed the EDC-S protocol which is a natural extension of the EDC-NS protocol with the flexibility of segmenting the original flow into smaller blocks. Instead of retransmitting the whole flow that is erroneous in EDC-NS, we only need to retransmit those erroneous blocks in EDC-S. This will save quite an amount of transmission time, and also reduce the number of retransmissions due to a smaller probability of error for small blocks. Nevertheless, the use of error detection code without forward error correction code does not work too well when the BER is high. We may need to use FEC to reduce the BER for each block. The next chapter discusses the protocol with FEC and segmentation.

Chapter 6

Protocol with Forward Error Correction and Segmentation (FEC-S)

From previous chapters, it can be seen that EDC-NS can be treated as a special case of EDC-S with the whole flow treated as one segment. In that sense, EDC-S is at least as good as EDC-NS. In the case that the bit error rate is relatively high after error correction in DLC Layer, it may be necessary to use forward error correction again in the Transport Layer to reduce the bit error rate to an acceptable range to decrease the probability of retransmissions. This will potentially reduce the total delay. We call this protocol FEC-S, which stands for forward error correction and segmentation.

Nevertheless, there is some cost associated with further reducing the bit error rate using FEC in the Transport Layer. One type of cost is coding redundancy added to each block, resulting in a longer block and hence larger delay. Another type of cost is the extra encoding/decoding delay added to the original delay. We will examine the benefits and cost of using FEC in the Transport Layer in this chapter.

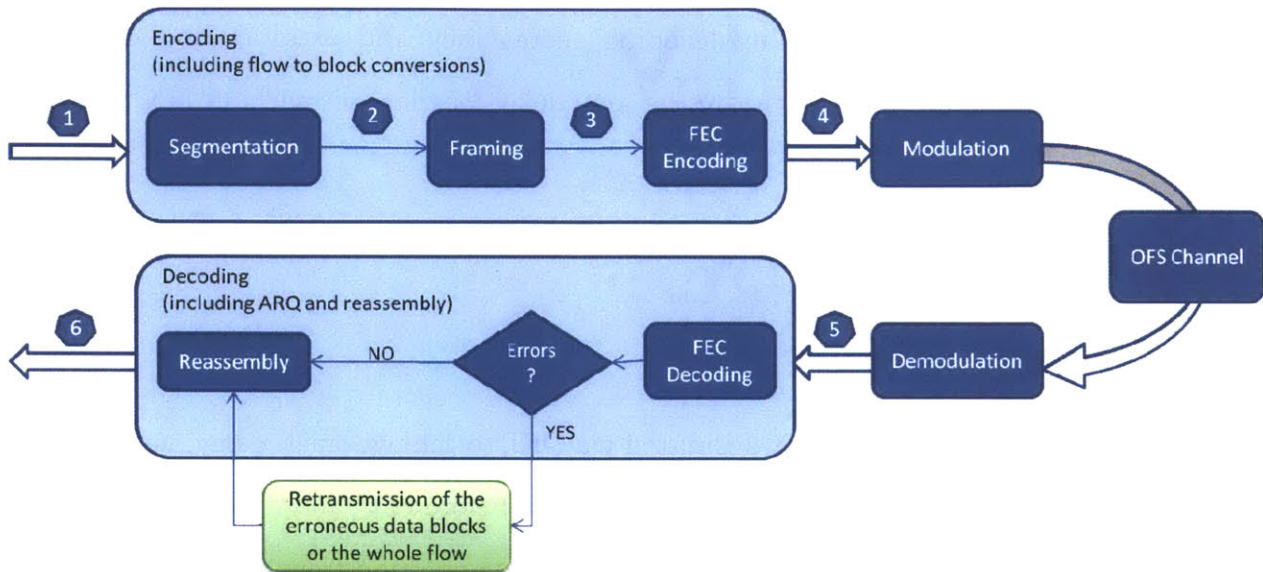
The FEC-S protocol works in the same way as the EDC-S protocol, except that an FEC is used in the Transport Layer in addition to the error detection code. That is: after scheduling, the file to be transmitted is first segmented into smaller blocks of pre-defined length (which can be assumed to be much larger than the header size), and each block is encoded with an error detection code. Also, each block is assigned a block number, so that when the block is in error, the receiver and transmitter both know which block to retransmit. These blocks are then encoded with FEC code and transmitted in sequence via OFS. After the whole flow is received, if an error is found in any of the blocks, the receiver requests retransmission of the erroneous block(s) via OFS or EPS.

Section 6.1 gives an overview of the FEC-S protocol, and Section 6.2 focuses on the delay analysis with some coding preliminaries.

6.1 FEC-S Algorithm Description

The FEC-S algorithm works in a similar way to Fig. 5.1, except that FEC is used in the Transport Layer in addition to the error detection code. Therefore, we will not draw the flowchart here. Instead, Fig. 6.1 shows the block diagram of how a flow is transmitted using FEC-S. From (1) to (2), the segmentation is basically done at the sender with the data passed from the application layer to the processor for further processing. From (2) to (3), each block is framed with error detection code and other information added, such as the block number. From (3) to (4), the FEC encoding is done to decrease the bit error rate. From (4) to (5), the

data is modulated, transmitted via OFS channels and demodulated at the receiver. From (5) to (6), the received data is decoded and checked against any error that may exist in the data blocks. If any error is found, the erroneous blocks are requested for retransmissions.



FEC: Forward Error Correction
 ARQ: Automatic Repeat reQuest

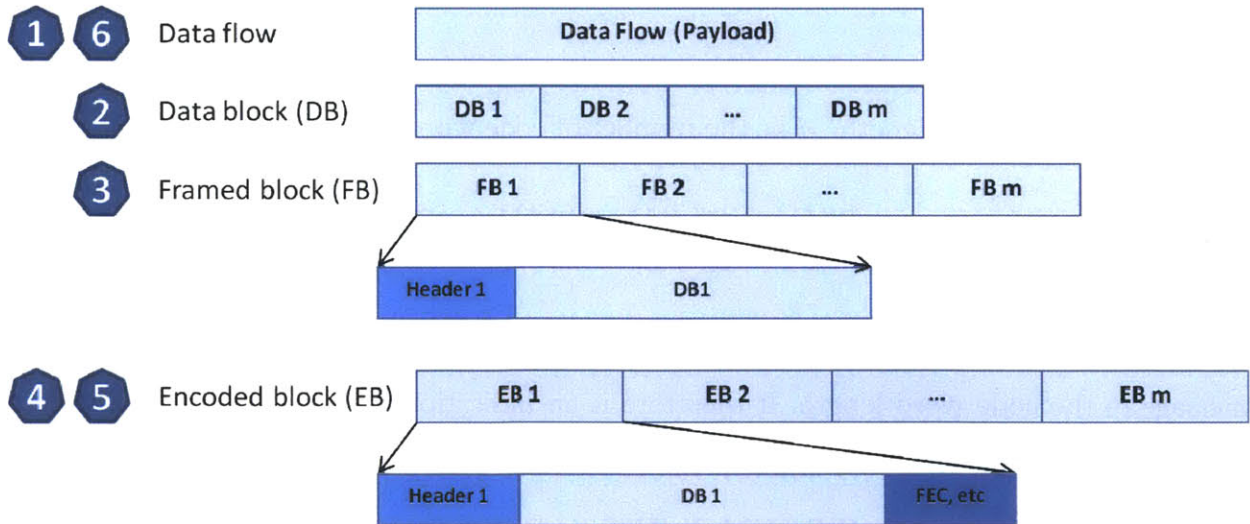


Figure 6.1: Illustration of flow transmission in the case of segmentations

6.2 Delay Analysis of FEC-S Algorithm with Retransmissions via OFS (FEC-S (OFS))

We denote the FEC-S algorithm with retransmissions via OFS by FEC-S (OFS), and analyze its performance below. When there is FEC used, the error probability of a block is decreased, but at the same time the encoding/decoding redundancy and extra coding delay are introduced. We are interested to compare the total delay for the cases without FEC and with FEC.

6.2.1 Preliminaries on Coding

Give a certain file that needs to be transmitted via OFS, to achieve small delay, naturally we will ask whether the file needs to be segmented or not before FEC, and if yes, how large the segment size should be. It is necessary to examine such a question from the coding point of view. We first take a look at the upper bound on probability of block errors as a function of the random coding exponent and block length [25]. The following two pages of results up to (6.15) are from [25].

Let N be the code word length, M be the number of code words, and

$$R = \frac{\ln(M)}{N} = \frac{\ln(2^{L_b})}{L_b + L_h} = \frac{\ln(2) L_b}{L_b + L_h} = \frac{\ln(2) L_b}{L'_b} \quad (6.1)$$

be the code rate. Let $R_2 = \frac{L_b}{L'_b} = \frac{R}{\ln(2)}$ be the normalized code rate that denotes the ratio of the message to the code word length. It therefore is an indication of the efficiency of the code. The closer R_2 is to 1, the more efficient the code is considered to be. Thus $M = e^{NR}$ and, for fixed rate, M varies exponentially with N . Unfortunately, varying N for fixed R can lead to non-integer values of e^{NR} . In [25], Gallager circumvented this detail with the following definition: *For any positive integer N and any positive number R , and (N,R) block code is a*

code of block-length N with $[e^{NR}]$ code words where, by the notation $[e^{NR}]$, we mean the smallest integer greater than or equal to e^{NR} .

According to Theorem 5.6.2 in [25], we have the following statement:

Let a discrete memoryless channel have transition probabilities $P(j|k)$ and, for any positive integer N and positive number R , consider the ensemble of (N, R) block codes in which each letter of each code word is independently selected with the probability assignment $Q(k)$. Then, for each message m , $1 \leq m \leq [e^{NR}]$, and all ρ , $0 \leq \rho \leq 1$, the ensemble average probability of decoding error using maximum-likelihood decoding satisfies

$$\overline{P_{e,m}} \leq \exp\{-N[E_0(\rho, \mathbf{Q}) - \rho R]\} \quad (6.2)$$

where

$$E_0(\rho, \mathbf{Q}) = -\ln \sum_{j=0}^{J-1} \left[\sum_{k=0}^{K-1} Q(k) P(j|k)^{\frac{1}{1+\rho}} \right]^{1+\rho} \quad (6.3)$$

where $(0, 1, \dots, K-1)$ is the input alphabet and $(0, 1, \dots, J-1)$ is the output alphabet.

Define the random coding exponent $E_r(R)$ by

$$E_r(R) = \max_{0 \leq \rho \leq 1} \max_{\mathbf{Q}} [E_0(\rho, \mathbf{Q}) - \rho R] \quad (6.4)$$

we then have

$$\overline{P_{e,m}} \leq \exp[-NE_r(R)]; \quad 1 \leq m \leq M = e^{NR} \quad (6.5)$$

$$\overline{P_e} \leq \exp[-NE_r(R)] \quad (6.6)$$

From (6.6) we see that, as the block length increases, the upper bound of block error rate decreases exponentially. Also, it is exponentially tight as the block length increases [25].

Furthermore, Gallager showed that for any discrete memoryless channel, any positive integer N , and any positive R , there exists an (N, R) block code for which

$$P_{e,m} < 4 \exp[-NE_r(R)]; \quad \text{each } m, \quad 1 \leq m \leq [e^{NR}] \quad (6.7)$$

In many cases (e.g. binary phase shift keying with coherent detection), the optical channel can be modeled as a binary symmetric channel, as shown in Fig. 6.2 [25].

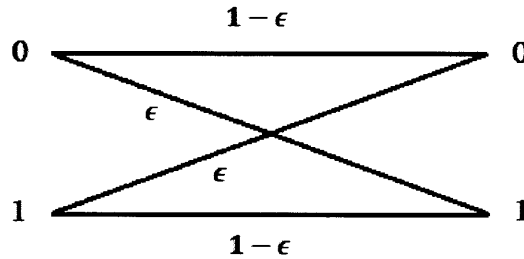


Figure 6.2: A binary symmetric channel with bit error rate ϵ .

Given a certain bit error rate, $E_r(R)$ depends on a number of factors such as ρ , \mathbf{Q} , and R , and is independent of N . Furthermore, $E_0(\rho, \mathbf{Q})$ is maximized over \mathbf{Q} by $Q(0) = Q(1) = 1/2$. For this \mathbf{Q} , we have [25]

$$E_0(\rho, \mathbf{Q}) = \rho \ln 2 - (1 + \rho) \ln \left[\epsilon^{\frac{1}{1+\rho}} + (1 - \epsilon)^{\frac{1}{1+\rho}} \right] \quad (6.8)$$

After some manipulations, we have [25]

$$R = \ln 2 - H(\delta) \quad (6.9)$$

$$E_r(R) = T_\epsilon(\delta) - H(\delta) \quad (6.10)$$

where the parameter δ is related to the parameter ρ by

$$\delta = \frac{\epsilon^{\frac{1}{1+\rho}}}{\epsilon^{\frac{1}{1+\rho}} + (1-\epsilon)^{\frac{1}{1+\rho}}} \quad (6.11)$$

and $H(\delta)$ and $T_\epsilon(\delta)$ are given by

$$H(\delta) = -\delta \ln \delta - (1-\delta) \ln (1-\delta) \quad (6.12)$$

$$T_\epsilon(\delta) = -\delta \ln \epsilon - (1-\delta) \ln (1-\epsilon) \quad (6.13)$$

The above equations are only valid for δ in the range $\epsilon \leq \delta \leq \sqrt{\epsilon}/(\sqrt{\epsilon} + \sqrt{1-\epsilon})$.

For $\delta > \sqrt{\epsilon}/(\sqrt{\epsilon} + \sqrt{1-\epsilon})$, where $\epsilon < 1/2$, we have

$$0 \leq R < \ln 2 - 2 \ln(\sqrt{\epsilon} + \sqrt{1-\epsilon}) \quad (6.14)$$

$$E_r(R) = \ln 2 - 2 \ln(\sqrt{\epsilon} + \sqrt{1-\epsilon}) - R \quad (6.15)$$

Based on the above equations, we can plot $E_r(R)$ vs. R as shown in Fig. 6.3-6.6 with BER $\epsilon = p_e = 10^{-1}, 10^{-2}, 10^{-3}$ and 10^{-8} , respectively.

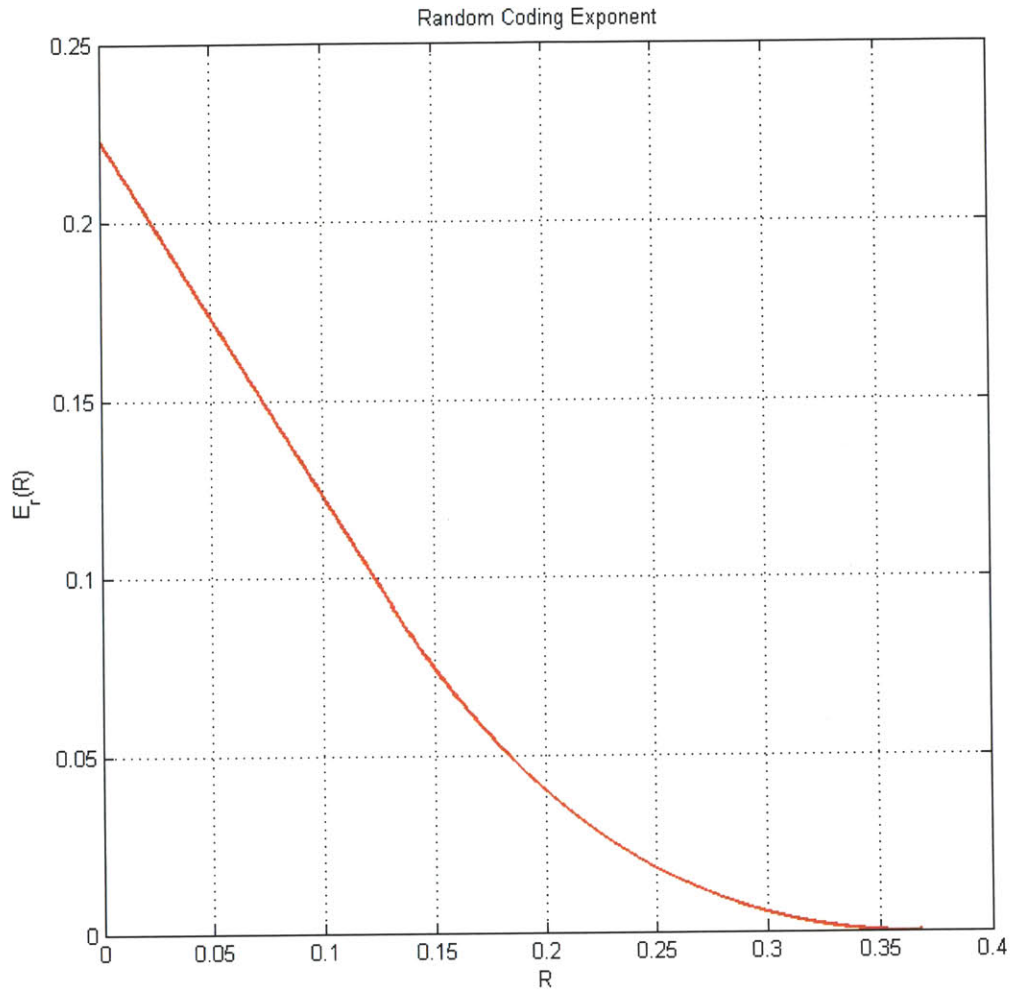


Fig. 6.3: Random coding exponent $E_r(R)$ vs. R with $\epsilon = p_e = 10^{-1}$. The maximum value of R is the capacity $\ln 2 + \epsilon \ln(\epsilon) + (1 - \epsilon) \ln(1 - \epsilon) \approx 0.368$.

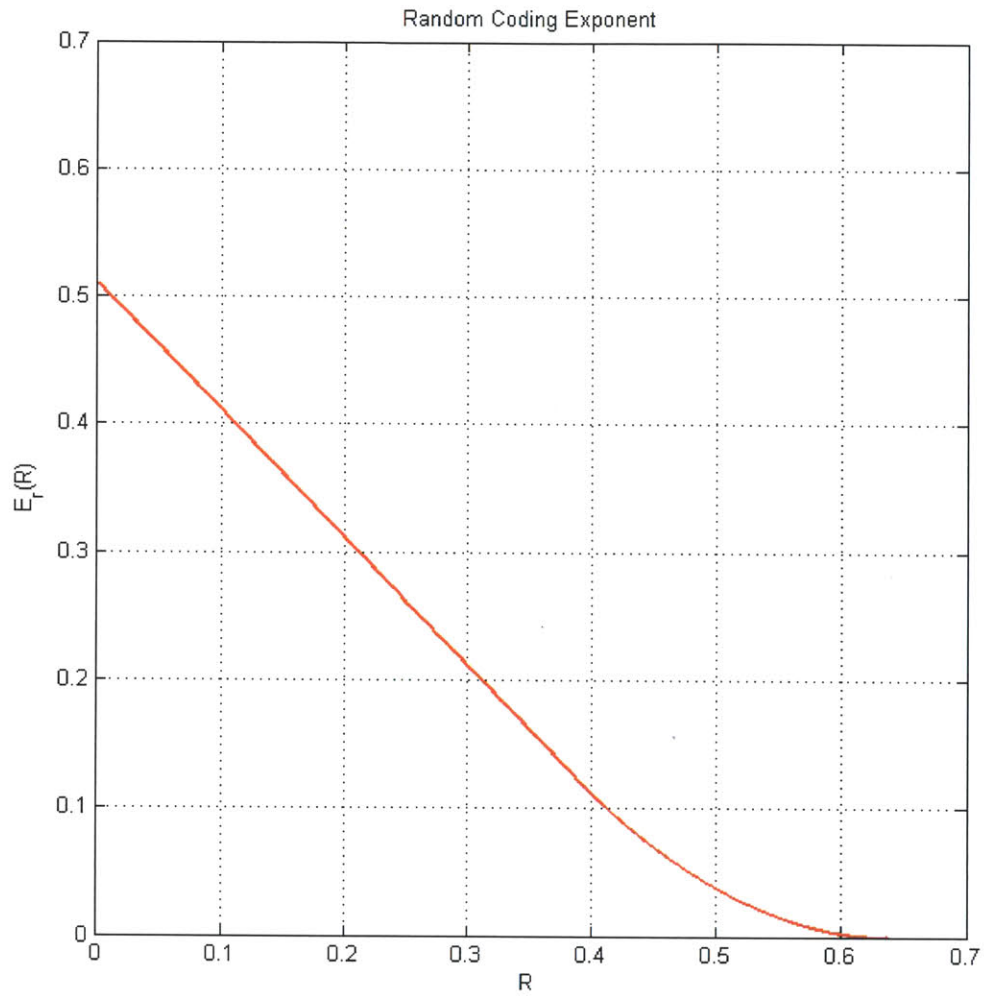


Fig. 6.4: Random coding exponent $E_r(R)$ vs. R with $\epsilon = p_e = 10^{-2}$. The maximum value of R is the capacity $\ln 2 + \epsilon \ln(\epsilon) + (1 - \epsilon) \ln(1 - \epsilon) \approx 0.637$.

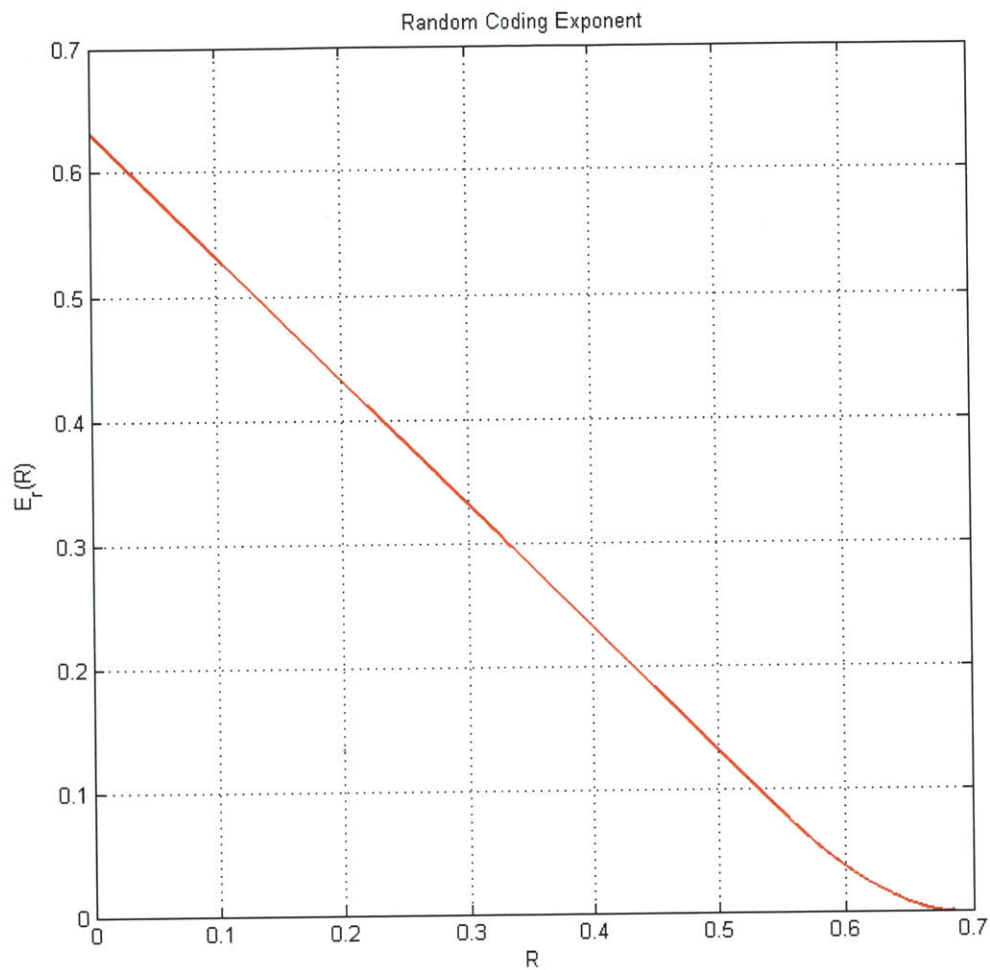


Fig. 6.5: Random coding exponent $E_r(R)$ vs. R with $\epsilon = p_e = 10^{-3}$. The maximum value of R is the capacity $\ln 2 + \epsilon \ln(\epsilon) + (1 - \epsilon) \ln(1 - \epsilon) \approx 0.685$.

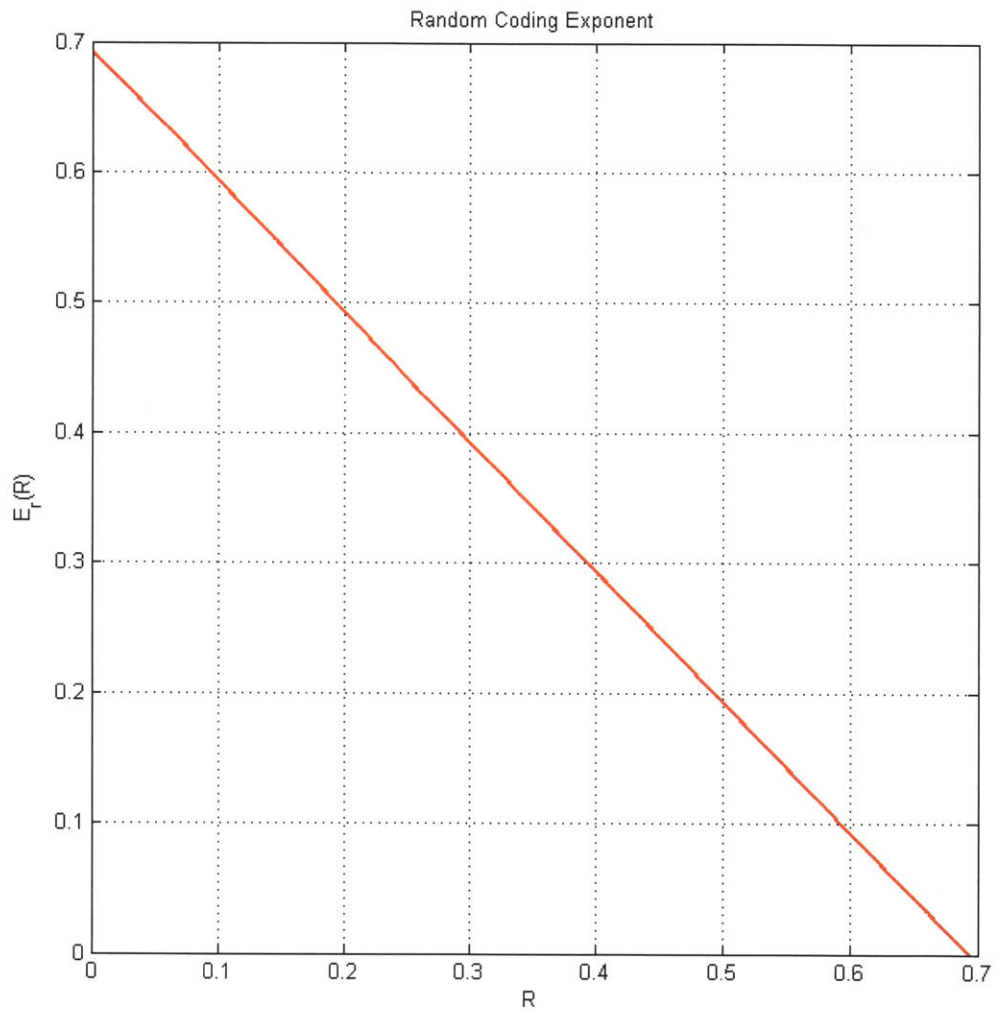


Fig. 6.6: Random coding exponent $E_r(R)$ vs. R with $\epsilon = p_e = 10^{-8}$. The maximum value of R is the capacity $\ln 2 + \epsilon \ln(\epsilon) + (1 - \epsilon) \ln(1 - \epsilon) \approx 0.693$.

From Fig. 6.3-6.6 it can be seen that for a given BER, the random coding exponent is a decreasing function of the code rate R . As the BER decreases, the capacity increases and gets closer to $\ln 2 \approx 0.693$, which corresponds to a normalized code rate $R_2 = \frac{L_b}{L'_b} = \frac{R}{\ln 2} = 1$. Here with $\text{BER} = 0$, basically the maximum achievable message to flow ratio is 1. However, as R gets larger, $E_r(R)$ gets smaller, which requires N to be larger to keep the same probability of errors after FEC, according to (6.5)-(6.7).

6.2.2 Segmentation or Not

Suppose we now have a flow of length L_f before any encoding, and its size becomes $L'_f > L_f$ after encoding. When there is no segmentation, assume the flow error rate is $p_{e,f} = 10^{-10}$ for instance. After dividing it into n_b blocks while keeping the same rate, the block error rate roughly becomes $p_{e,b} \approx p_{e,f}^{1/n_b}$ because the block size becomes $1/n_b$ of original flow size (remember that the block error rate is exponentially decreasing with increasing block length). We then have the following plots in Fig. 6.7 (a)-(e) when we segment the original flow into 2, 4, 8, 16, and 32 blocks, respectively.

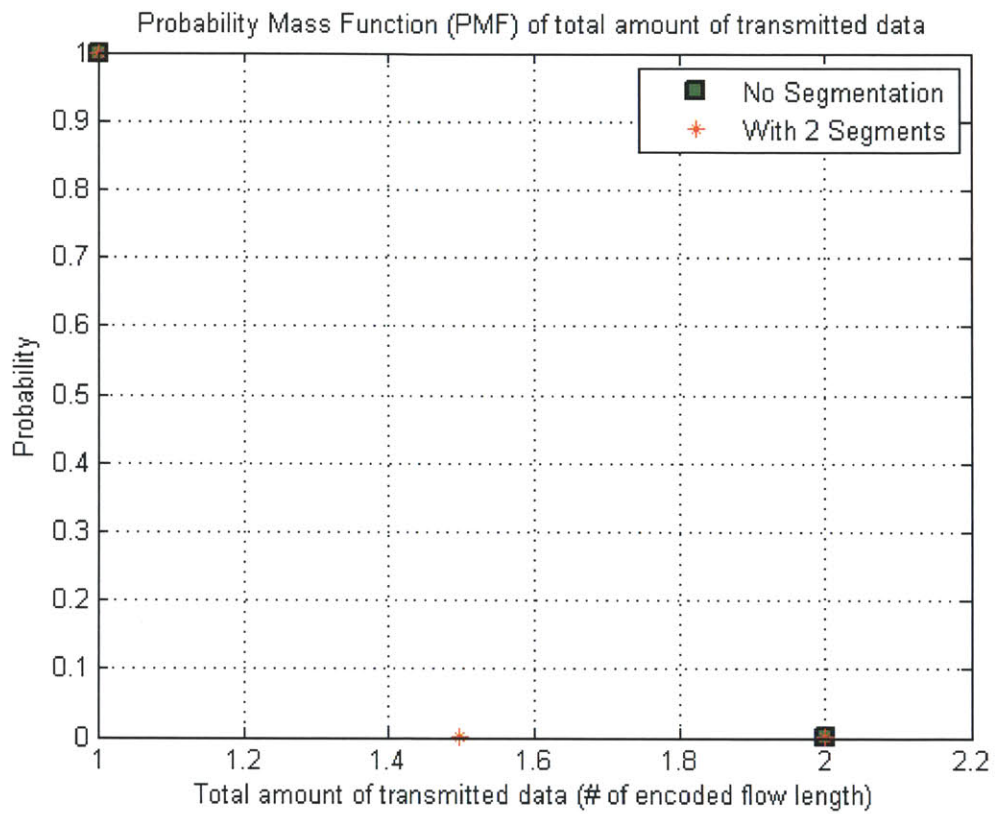


Figure 6.7 (a): Probability mass function of total amount of transmitted data which is normalized to the number of encoded flow length L'_f . The comparison is between the case with no segmentation ($p_{e,f} = 10^{-10}$) and that with 2 segments ($p_{e,b} = 10^{-5}$).

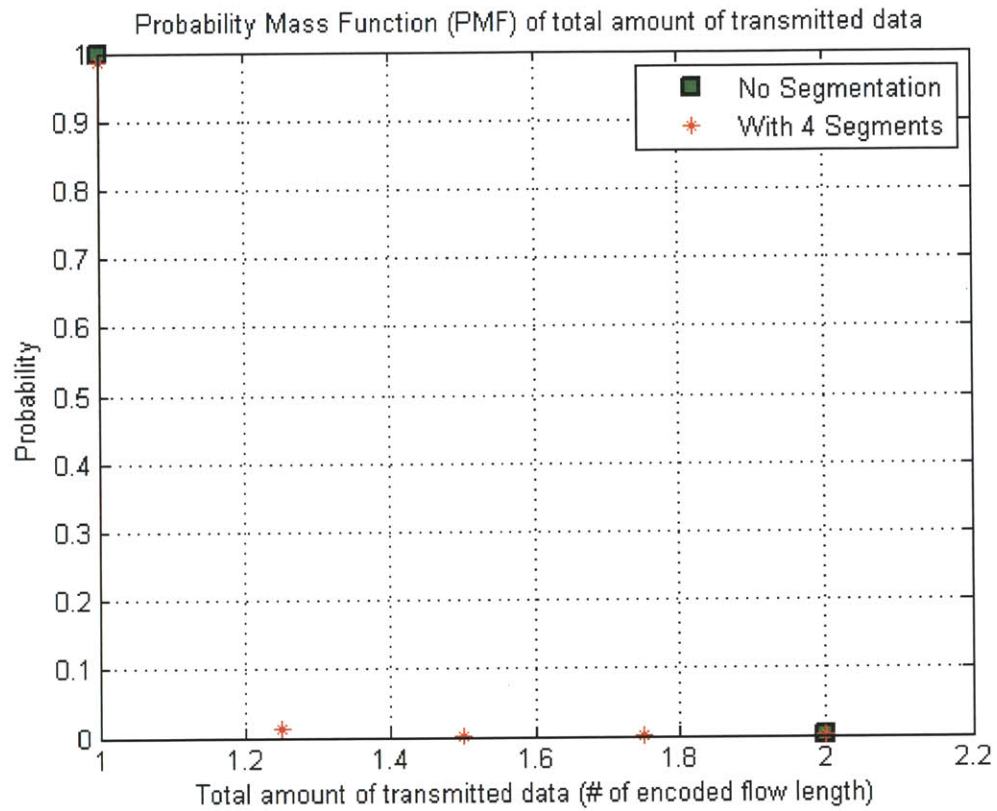


Figure 6.7 (b): Probability mass function of total amount of transmitted data which is normalized to the number of encoded flow length L'_f . The comparison is between the case with no segmentation ($p_{e,f} = 10^{-10}$) and that with 4 segments ($p_{e,b} = 10^{-2.5} \approx 0.0032$).

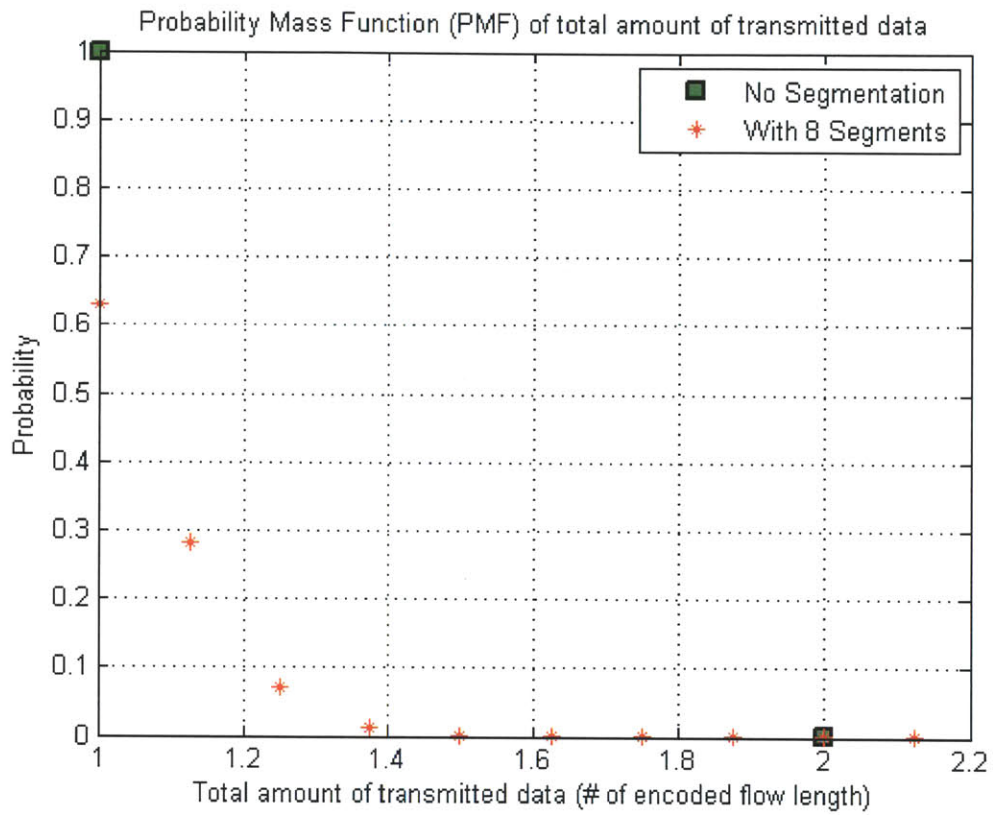


Figure 6.7 (c): Probability mass function of total amount of transmitted data which is normalized to the number of encoded flow length L'_f . The comparison is between the case with no segmentation ($p_{e,f} = 10^{-10}$) and that with 8 segments ($p_{e,b} = 10^{-1.25} \approx 0.0562$).

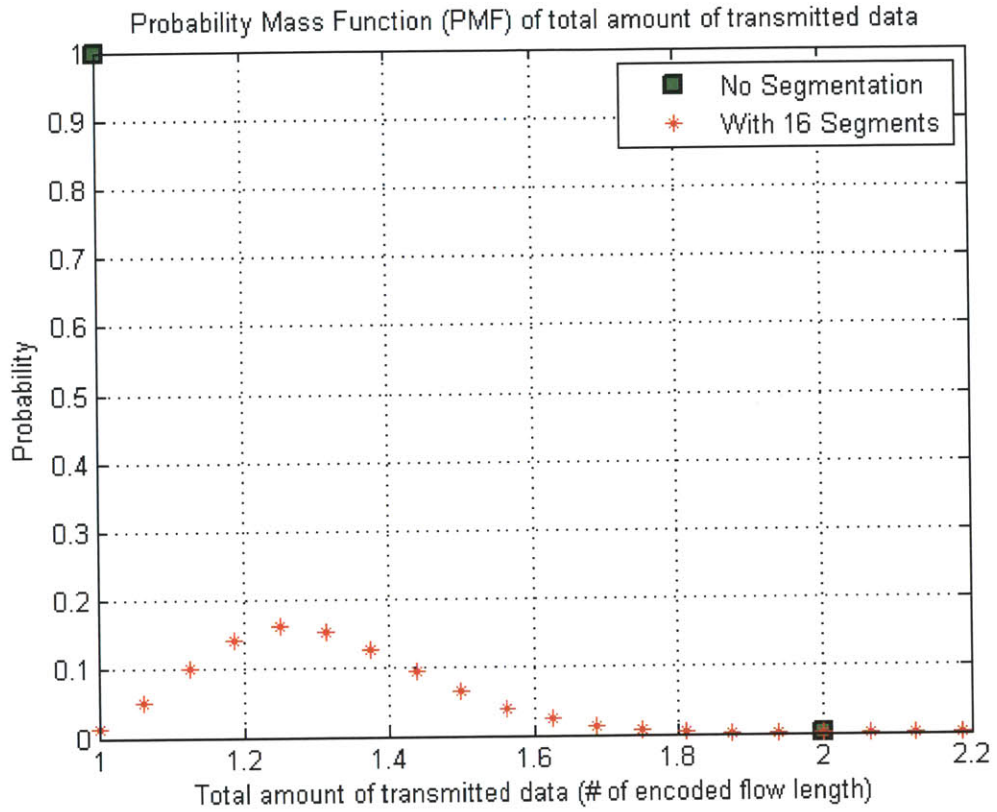


Figure 6.7 (d): Probability mass function of total amount of transmitted data which is normalized to the number of encoded flow length L'_f . The comparison is between the case with no segmentation ($p_{e,f} = 10^{-10}$) and that with 16 segments ($p_{e,b} = 10^{-5/8} \approx 0.2371$).

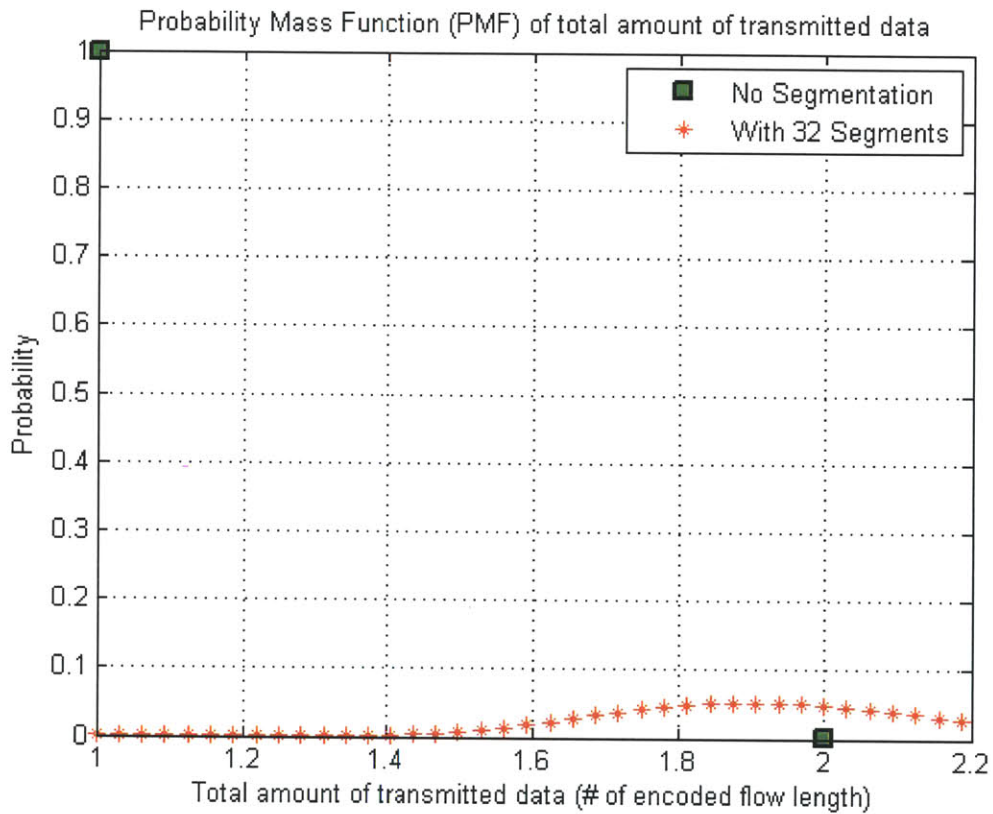


Figure 6.7 (e): Probability mass function of total amount of transmitted data which is normalized to the number of encoded flow length L'_f . The comparison is between the case with no segmentation ($p_{e,f} = 10^{-10}$) and that with 32 segments ($p_{e,b} = 10^{-5/16} \approx 0.4780$).

It can be seen from Fig. 6.7 (a)-(e) that as the original flow is segmented into different number of blocks (while the rate R is kept the same), the probability mass function of the total amount of transmitted data also changes. The trend is that the more blocks there are (i.e. the smaller the block length is), the point with maximum probability shifts more to the right (e.g. from 1 in Fig. 6.7 (a) to 1.25 in Fig. 6.7 (d) and to 1.9 in Fig. 6.7 (e)). Also, the probability of transmitting $2L'_f$ in the case of no segmentation is not larger than that in the case of segmentation.

Furthermore, when there are more and more segments, the probability of error for each block increases and the probability of retransmission also increases which is the probability that the total amount of transmitted data is more than L'_f . When we take into the scheduling and queuing delay into consideration, the case with segmentation tends to have longer delay than the case without segmentation because each retransmission requires one extra scheduling and queuing delay.

Note that the above analysis is true for any L_f in general, and we may tend to conclude that for least delay, we always prefer not to segment the flow into smaller blocks which can have larger probability of errors. However, we ignored one fact that as the block length becomes longer and longer, the decoding complexity also increases and the decoding time can also increase dramatically. Also, there is longer latency, defined as the time from the moment the first bit of the block is received until the moment it is decoded, because decoding can only start until all bits in that block are received. Moreover, we have not treated the case that when we do segmentation we would lower the code rate to decrease the block error rate back to the original unsegmented scheme at the expense of more transmission delay of a longer block.

Therefore, we should decide whether we should segment the flow into smaller blocks, and if so, what size the block size should be, and whether we should lower the code rate to prevent more retransmissions, depending on the latency requirements of the applications.

We will next include decoding delay into the analysis by assuming decoding in linear time. That is, decoding delay is proportional to the code word length. Note, however, that in practice the code may not be able to be decoded in linear time (the more accurate complexity analysis will be left for future work). A code word can be decoded while the next one is being received. If the decoding time is less than the transmission time of the same codeword, only the last codeword of the whole flow contributes to the total delay. We will assume this is the case given the fast processor speed nowadays.

Using similar deductions as shown in Chapter 5, the expected delay is

$$E[D_t] \approx D_{TCP} + E[N_t](D_{pg}^{sch} + D_q + T_{pg}^{OFS} + T_{pg}^{EPS}) - T_{pg}^{EPS} + \frac{L'_b \left\lfloor \frac{L_f}{L_b} \right\rfloor / R_2 + \Delta L'_b}{R_{OFS}(1 - p_{e,b})} \quad (6.16)$$

$$\approx D_{TCP} - T_{pg}^{EPS} + E[N_t](D_{pg}^{sch} + D_q + T_{pg}^{OFS} + T_{pg}^{EPS}) + \frac{(L_b + L_h) \left\lfloor \frac{L_f}{L_b} \right\rfloor / R_2 + \Delta(L_b + L_h)}{R_{OFS}(1 - p_{e,b})} \quad (6.17)$$

where

$$E[N_t] = (1 - p_{e,b})^{\left\lfloor \frac{L_f}{L_b} \right\rfloor} \times \left\{ 1 + \sum_{k=2}^{\infty} k \left[\left(\sum_{i=0}^{k-1} (p_{e,b})^i \right)^{\left\lfloor \frac{L_f}{L_b} \right\rfloor} - \left(\sum_{i=0}^{k-2} (p_{e,b})^i \right)^{\left\lfloor \frac{L_f}{L_b} \right\rfloor} \right] \right\} \quad (6.18)$$

$\frac{\Delta L'_b}{R_{OFS}(1 - p_{e,b})}$ captures the decoding delay, and Δ is the coefficient that captures the linear relationship between the decoding time and code word length. $R_2 = L_b/L'_b = R/\ln(2) \leq 1$ is the normalized code rate when the block size is L_b before encoding and L'_b after encoding. Here we assume L'_b is large so that the upper bound on the error probability is tight; that is, $p_{e,b} \approx \exp(-L'_b E_r(R))$.

One way to find the optimal point of $E[D_t]$ is taking the derivative of $E[D_t]$ with respect to L_b and solving for L_b by setting $\frac{\partial E[D_t]}{\partial L_b} = 0$. Nevertheless, it should be noted that $p_{e,b} = \exp(-L'_b E_r(R))$ is valid only for large $L'_b = L_b + L_h$, say $L'_b \geq 10^4$. When L'_b is small, we can no longer express $p_{e,b}$ explicitly by $\exp(-L'_b E_r(R))$. Thus, if the answer we find is a small value of L'_b (e.g. $< 10^4$ bits), we should pay special attention to whether it is valid.

Fig 6.8-6.11 are plots of $E[D_t]$ vs. L_b for different values of Δ and L_f .

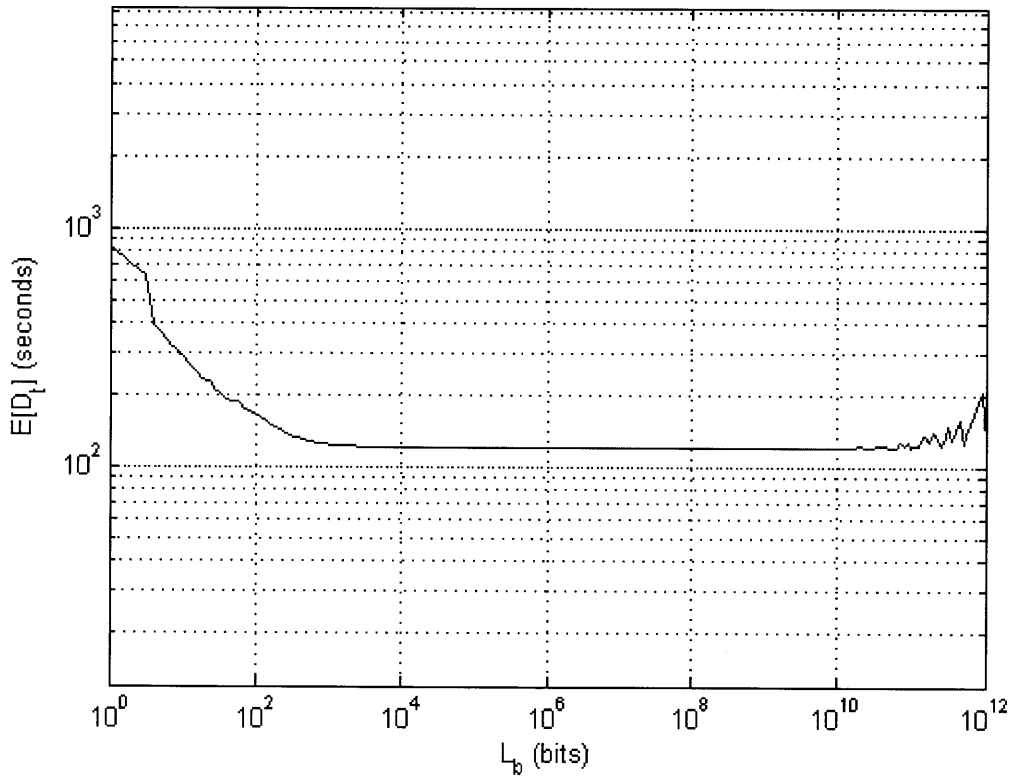


Figure 6.8: Plot of $E[D_t]$ vs. L_b when $\Delta = 0.1$, $L_f = 10^{12}$ bits, $L_h = 0$ and $p_e = 10^{-6}$. $T_{pg}^{EPS} = 45.5ms$, $T_{pg}^{OFS} = 25ms$, $R_{OFS} = 10Gbps$, $D_{TCP} = 8T_{pg}^{EPS}$, $D_{pg}^{sch} = 2T_{pg}^{EPS}$, and $D_q = 8.9s$. Note that the loading factor is assumed to be 0.5 with average transaction length 100s, resulting in a queuing delay of approximately 8.9s. Refer to Fig. 3.5 for a plot of normalized total delay vs. loading factor.

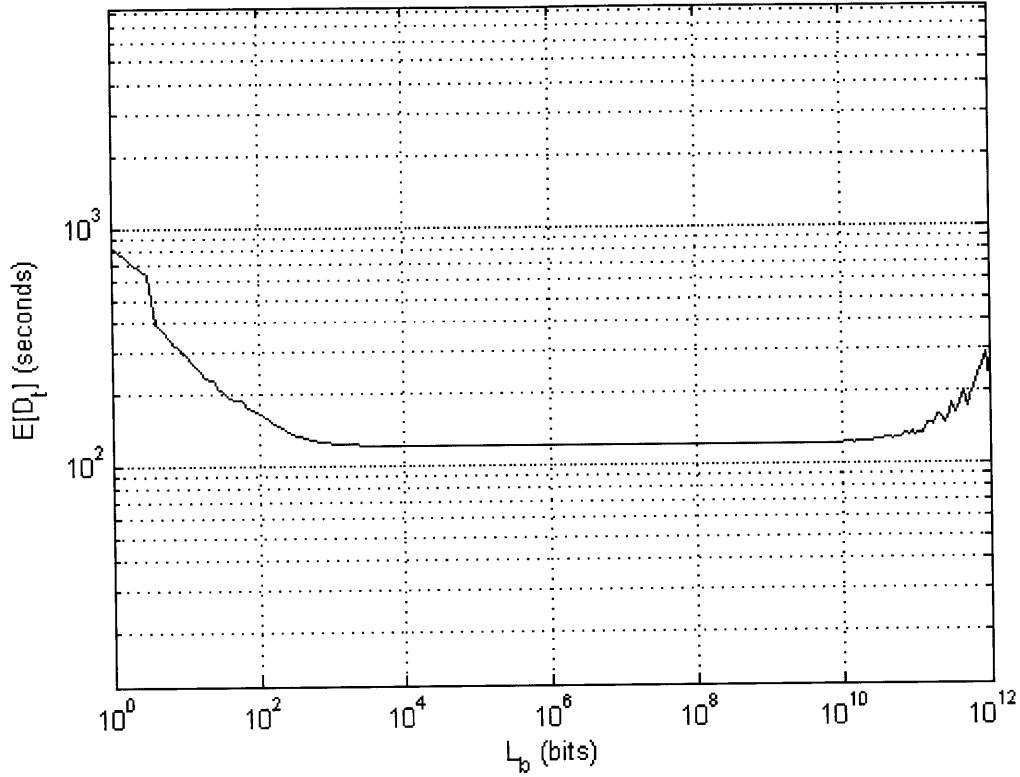


Figure 6.9: Plot of $E[D_t]$ vs. L_b when $\Delta = 1$, $L_f = 10^{12}$ bits, $L_h = 0$ and $p_e = 10^{-6}$. $T_{pg}^{EPS} = 45.5ms$, $T_{pg}^{OFS} = 25ms$, $R_{OFS} = 10Gbps$, $D_{TCP} = 8T_{pg}^{EPS}$, $D_{pg}^{sch} = 2T_{pg}^{EPS}$, and $D_q = 8.9s$. Note that the loading factor is assumed to be 0.5 with average transaction length 100s, resulting in a queuing delay of approximately 8.9s. Refer to Fig. 3.5 for a plot of normalized total delay vs. loading factor.

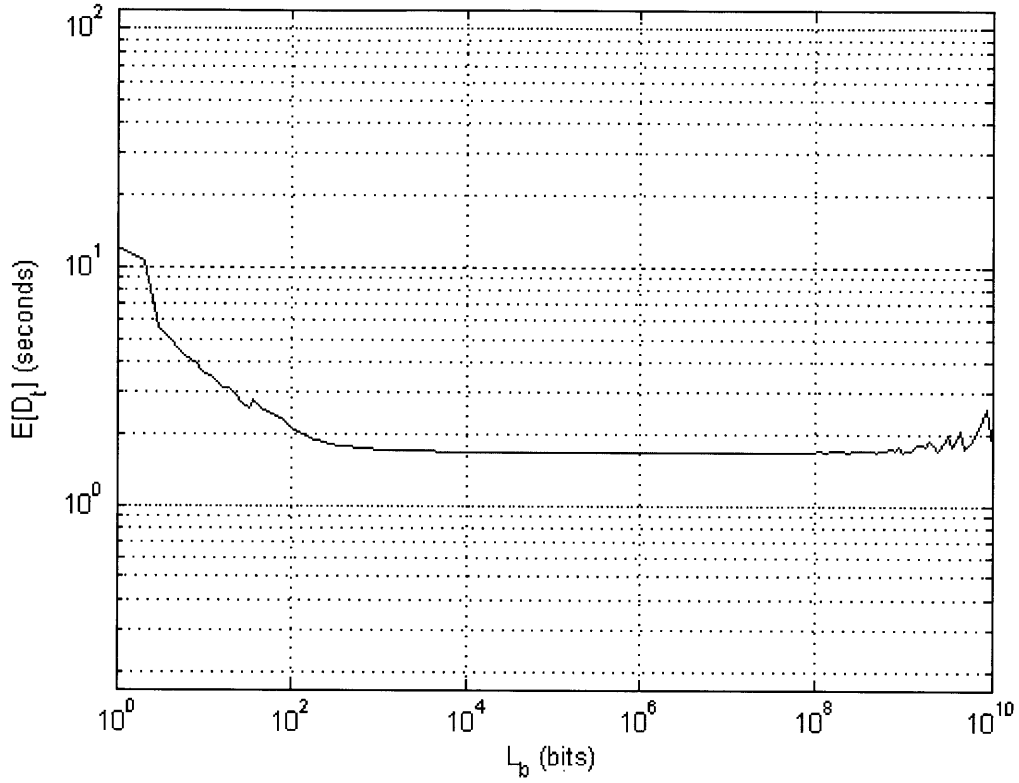


Figure 6.10: Plot of $E[D_t]$ vs. L_b when $\Delta = 0.1$, $L_f = 10^{10}$ bits, $L_h = 0$ and $p_e = 10^{-6}$. $T_{pg}^{EPS} = 45.5ms$, $T_{pg}^{OFS} = 25ms$, $R_{OFS} = 10Gbps$, $D_{TCP} = 8T_{pg}^{EPS}$, $D_{pg}^{sch} = 2T_{pg}^{EPS}$, and $D_q = 0.089s$. Note that the loading factor is assumed to be 0.5 with average transaction length 1s, resulting in a queuing delay of approximately 0.089s. Refer to Fig. 3.5 for a plot of normalized total delay vs. loading factor.

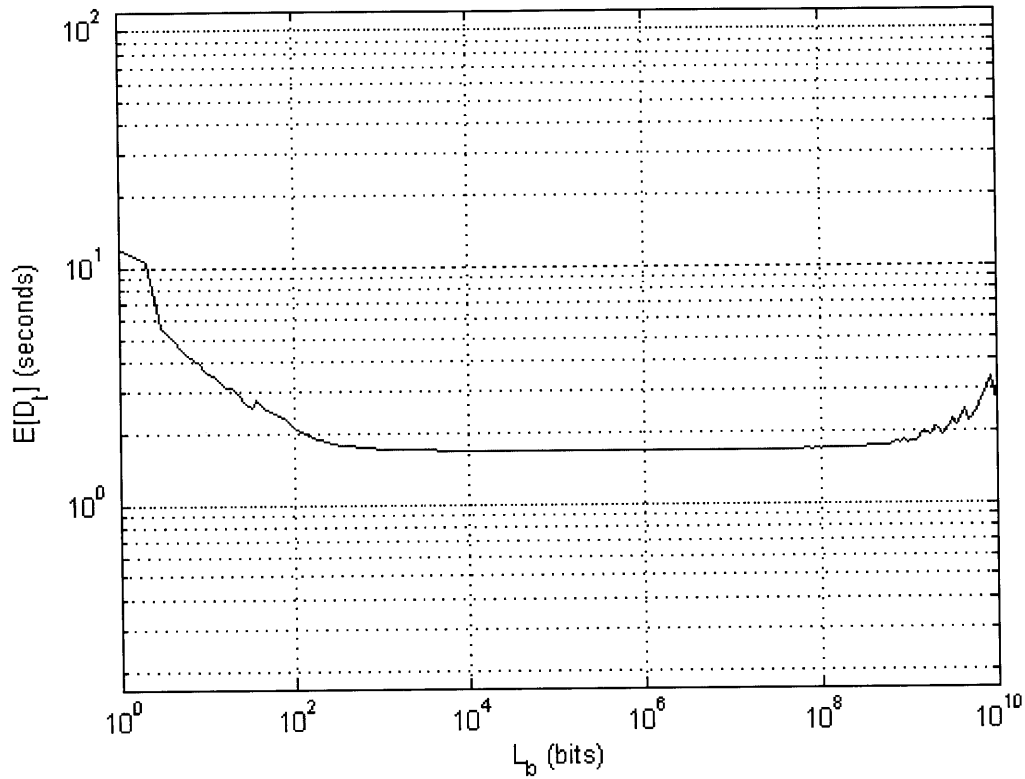


Figure 6.11: Plot of $E[D_t]$ vs. L_b when $\Delta = 1$, $L_f = 10^{10}$ bits, $L_h = 0$ and $p_e = 10^{-6}$. $T_{pg}^{EPS} = 45.5ms$, $T_{pg}^{OFS} = 25ms$, $R_{OFS} = 10Gbps$, $D_{TCP} = 8T_{pg}^{EPS}$, $D_{pg}^{sch} = 2T_{pg}^{EPS}$, and $D_q = 0.089s$. Note that the loading factor is assumed to be 0.5 with average transaction length 1s, resulting in a queuing delay of approximately 0.089s. Refer to Fig. 3.5 for a plot of normalized total delay vs. loading factor.

It can be seen from Fig. 6.8-6.11 that as L_b increases from 1 to the flow size, the total delay $E[D_t]$ first decreases when L_b is small (e.g. $< 10^4$ bits), then keeps relatively constant, and finally goes up again as L_b becomes close to L_f . For small L_b , the bit error rate is relatively high even after error correction or the redundancy is high for a low bit error rate, either of which can result in potentially large total delay. We can also observe that the curves are almost flat in regions where $L_b \in \left[10^4, \frac{L_f}{100}\right]$, which gives total delay $E[D_t]$ close to its minimum value, and the value of Δ does not affect much the optimal L_b . As the block size L_b gets closer and closer to L_f , the delay $E[D_t]$ starts to increase. This is because $p_{e,b} \rightarrow 0$ and the term

$$\frac{(L_b + L_h) \left\lceil \frac{L_f}{L_b} \right\rceil / R_2 + \Delta(L_b + L_h)}{R_{OFS}(1 - p_{e,b})} \approx \frac{(L_b + L_h) \left\lceil \frac{L_f}{L_b} \right\rceil / R_2 + \Delta(L_b + L_h)}{R_{OFS}} \quad (6.19)$$

starts to dominate and increases with increasing L_b .

The discussions above give us some guidance on practical FEC-S protocol designs. For $L_f \geq 10^6$, we should choose L_b so that $10^4 \leq L_b \ll L_f$ in order to minimize the total delay, as indicated (and inferred) from Fig. 6.8-6.11. For $10^4 \leq L_f < 10^6$, we can choose $L_b = 10^4$ bits. For all smaller $L_f < 10^4$ bits, no segmentation is needed, and $L_b = L_f$. That is,

$$L_b = \begin{cases} \in \left[10^4, \frac{L_f}{100}\right], & \text{when } L_f \geq 10^6 \\ 10^4, & \text{when } 10^4 \leq L_b \ll L_f \\ L_f, & \text{when } L_f < 10^4 \end{cases} \quad (6.20)$$

As for the types of FEC codes that may be used associated with the chosen block length, it is left for future work. What we do in this thesis only shows its feasibility to decrease transmission delay.

6.3 Summary of Chapter 6

In this chapter, we discussed FEC-S (OFS) protocol as a promising protocol to reduce the probability of errors and hence retransmissions and total delay. Nevertheless, the error reduction is at the cost of adding redundancy and extra decoding delay. With proper choices of block size and FEC code given a certain flow size, the total delay can be minimized (or be close to the minimum). The minimum delay is found when the block size is chosen according to (6.20), almost independent of the decoding delay coefficient. Note, however, that we assumed linear relationship between the decoding delay and code word length. Expression (6.20) may need to be adjusted if the relationship were not linear. We also have not treated the case when there are segmentations we would decrease the code rate to keep the retransmission probability low. This is more likely the way a practical design would end up with.

In Chapter 7, we shall compare all protocols that are presented so far, and provide practical guidance for OFS Transport Layer protocol choice to ensure end-to-end data transfer reliability.

Chapter 7

Comparison of Various Protocols

In this chapter, we compare various protocols discussed in Chapters 2-6, i.e. TCP, EDBC, EDC-NS, EDC-S and FEC-S, and show which protocol is preferred under what situation. That is, we will plot "preference maps"⁵ and show which protocol is preferred in each sub region of the map, whose x- and y-axes can be different independent variables. The preference maps can serve as guidance for protocol choice in practical optical network architecture design.

⁵ A preference map is a 2D plot that shows the regions where each protocol has optimum delay.

7.1 Delay Comparison of Various Protocols

We first compare the four protocols for OFS using delay as the metric, and then compare the best protocol with TCP over EPS under a delay metric.

7.1.1 Delay Comparison of Various Protocols for OFS

As discussed in Chapter 3-6, the expected total delays for each protocol over OFS are restated below:

For EDBC (see expression (3.19)):

$$E[D_t] \approx \frac{2D_q + \left(2T_{pg}^{OFS} + \frac{L_f}{R_{OFS}}\right) + (16 - 3(1 - p_e)^{L_f})T_{pg}^{EPS}}{(1 - p_e)^{L_f}} \quad (7.1)$$

For EDC-NS (see expression (4.8)):

$$E[D_t] \approx \frac{D_q + \left(T_{pg}^{OFS} + \frac{L_f}{R_{OFS}}\right) + (15 - 5(1 - p_e)^{L_f})T_{pg}^{EPS}}{(1 - p_e)^{L_f}} \quad (7.2)$$

For EDC-S (OFS) (see expression (5.15)):

$$E[D_t] \approx D_{TCP} - T_{pg}^{EPS} + E[N_t](D_{pg}^{sch} + D_q + T_{pg}^{OFS} + T_{pg}^{EPS}) + \frac{(L_b + L_h) \left\lceil \frac{L_f}{L_b} \right\rceil}{R_{OFS} (1 - p_{e,b})} \quad (7.3)$$

where

$$E[N_t] = (1 - p_{e,b})^{\left\lceil \frac{L_f}{L_b} \right\rceil} \times \left\{ 1 + \sum_{k=2}^{\infty} k \left[\left(\sum_{i=0}^{k-1} (p_{e,b})^i \right)^{\left\lceil \frac{L_f}{L_b} \right\rceil} - \left(\sum_{i=0}^{k-2} (p_{e,b})^i \right)^{\left\lceil \frac{L_f}{L_b} \right\rceil} \right] \right\} \quad (7.4)$$

and

$$p_{e,b} = 1 - (1 - p_e)^{L_b + L_h} \quad (7.5)$$

For FEC-S (OFS) (see expressions (6.17) and (6.18)):

$$E[D_t] \approx D_{TCP} - T_{pg}^{EPS} + E[N_t](D_{pg}^{sch} + D_q + T_{pg}^{OFS} + T_{pg}^{EPS}) + \frac{(L_b + L_h) \left\lfloor \frac{L_f}{L_b} \right\rfloor / R_2 + \Delta(L_b + L_h)}{R_{OFS}(1 - p_{e,b})} \quad (7.6)$$

where

$$E[N_t] = (1 - p_{e,b})^{\left\lfloor \frac{L_f}{L_b} \right\rfloor} \times \left\{ 1 + \sum_{k=2}^{\infty} k \left[\left(\sum_{i=0}^{k-1} (p_{e,b})^i \right)^{\left\lfloor \frac{L_f}{L_b} \right\rfloor} - \left(\sum_{i=0}^{k-2} (p_{e,b})^i \right)^{\left\lfloor \frac{L_f}{L_b} \right\rfloor} \right] \right\} \quad (7.7)$$

and

$$p_{e,b} \approx \exp(-(L_b + L_h)E_r(R)) \quad (7.8)$$

From expressions (7.1) and (7.2), it is apparent that the expected delay of EDC-NS is always smaller than the expected delay of EDBC, given the same set of parameters. As discussed in Section 5.2, (7.2) can be viewed as a special case of (7.3) when the number of segments is 1. Therefore, the optimal delay of EDC-NS in (7.3) is at least as good as that of EDC-S in (7.2). The comparison of EDC-S in (7.3) and FEC-S in (7.6) is less straightforward and more interesting, and will be our focus for discussion in this section.

In Fig. 7.1-7.4, we first compare the expected number of transmissions in expressions (7.4) and (7.7), which are important parameters for the total delay in expressions (7.3) and (7.6) respectively. Fig. 7.1-7.2 show the comparisons when the header of each block $L_h = 0$, and Fig. 7.3-7.4 show the comparisons when $L_h = 320$ bits. Note that the choice of "320 bits" for L_h is based on the header size of IPv6. In actual design of OFS, the size of each block header

can be smaller or larger. Nevertheless, the plots in Fig. 7.3-7.4 are for illustration purposes only, showing what happens to $E[N_t]$ when the header size is non-zero.

In Fig. 7.5-7.8, we next compare the total delays in expressions (7.3) and (7.6). Fig. 7.5-7.6 show the comparisons when $L_h = 0$, and Fig. 7.7-7.8 show that when $L_h = 320$ bits.

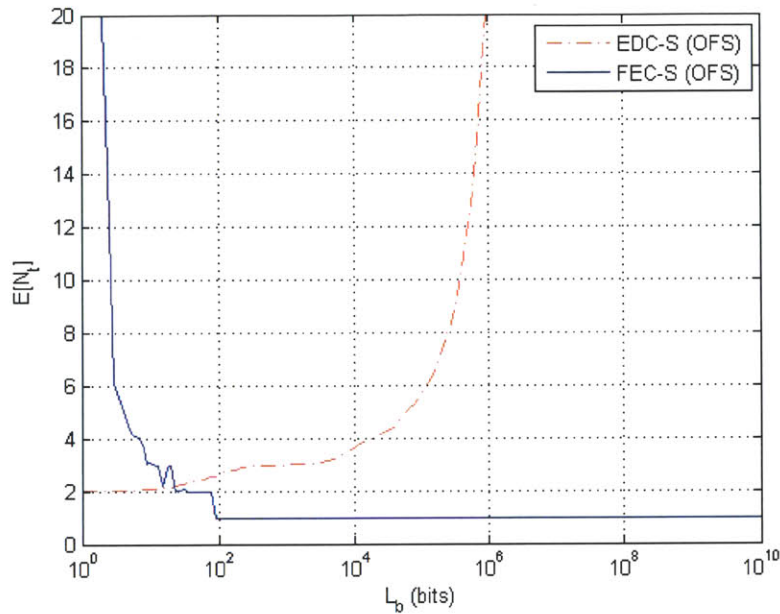


Figure 7.1: Comparison of EDC-S and FEC-S in terms of $E[N_t]$ vs. L_b for $L_f = 10^{10}$ bits, $L_h = 0$, and $p_e = 10^{-6}$. It can be seen that the minimum value of $E[N_t]$ for FEC-S is 1 while that for EDC-S is 2.

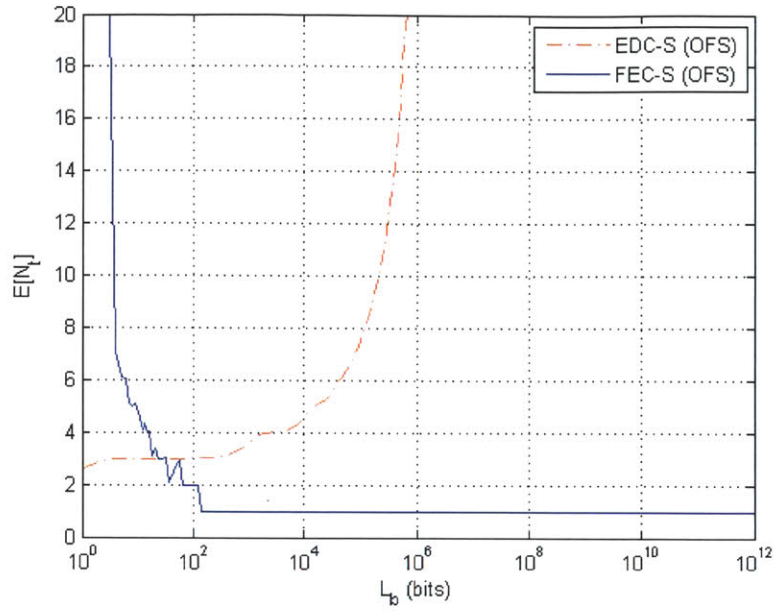


Figure 7.2: Comparison of EDC-S and FEC-S in terms of $E[N_t]$ vs. L_b for $L_f = 10^{12}$ bits, $L_h = 0$, and $p_e = 10^{-6}$. It can be seen that the minimum value of $E[N_t]$ for FEC-S is 1 while that for EDC-S is > 2 .

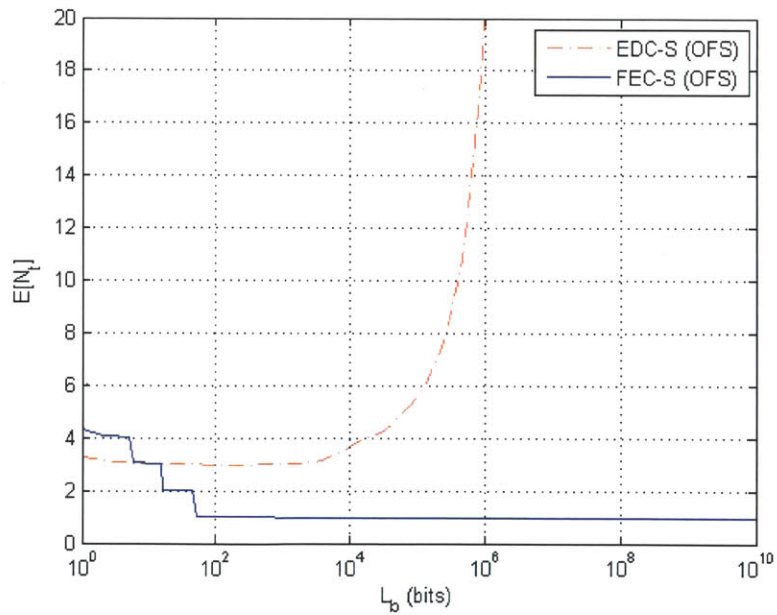


Figure 7.3: Comparison of EDC-S and FEC-S in terms of $E[N_t]$ vs. L_b for $L_f = 10^{10}$ bits, $L_h = 320$ bits, and $p_e = 10^{-6}$. It can be seen that the minimum value of $E[N_t]$ for FEC-S is 1 while that for EDC-S is > 3 .

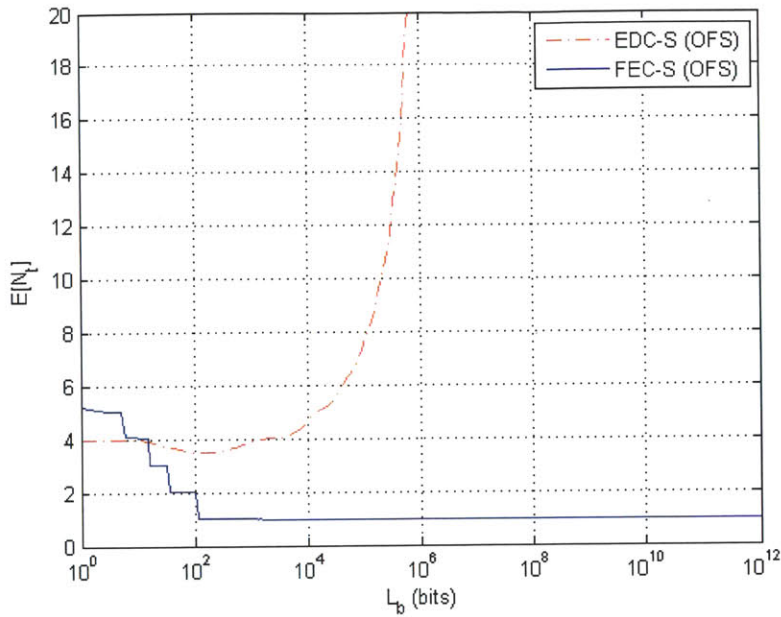


Figure 7.4: Comparison of EDC-S and FEC-S in terms of $E[N_t]$ vs. L_b for $L_f = 10^{12}$ bits, $L_h = 320$ bits, and $p_e = 10^{-6}$. It can be seen that the minimum value of $E[N_t]$ for FEC-S is 1 while that for EDC-S is > 3 .

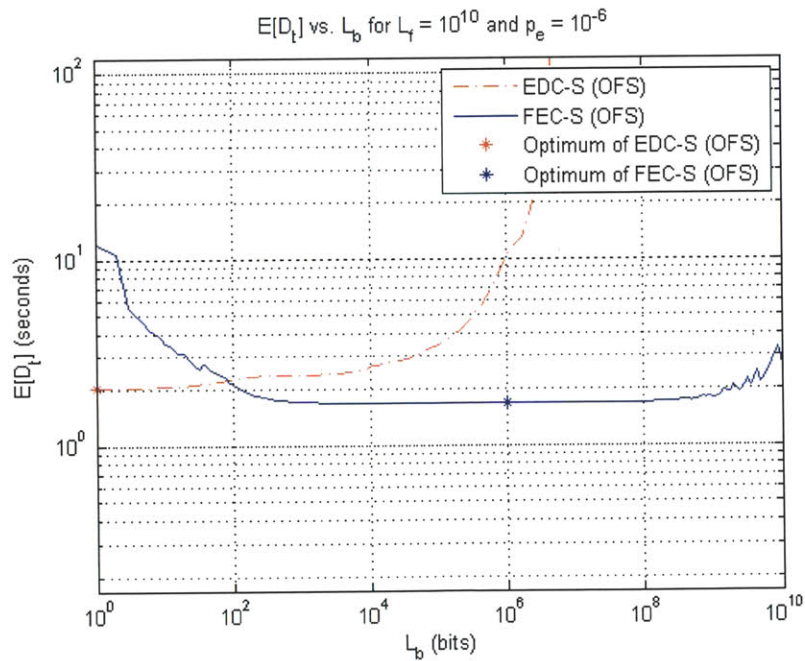


Figure 7.5: Comparison of EDC-S and FEC-S in terms of $E[D_t]$ vs. L_b for $L_f = 10^{10}$ bits, $L_h = 0$, and $p_e = 10^{-6}$.

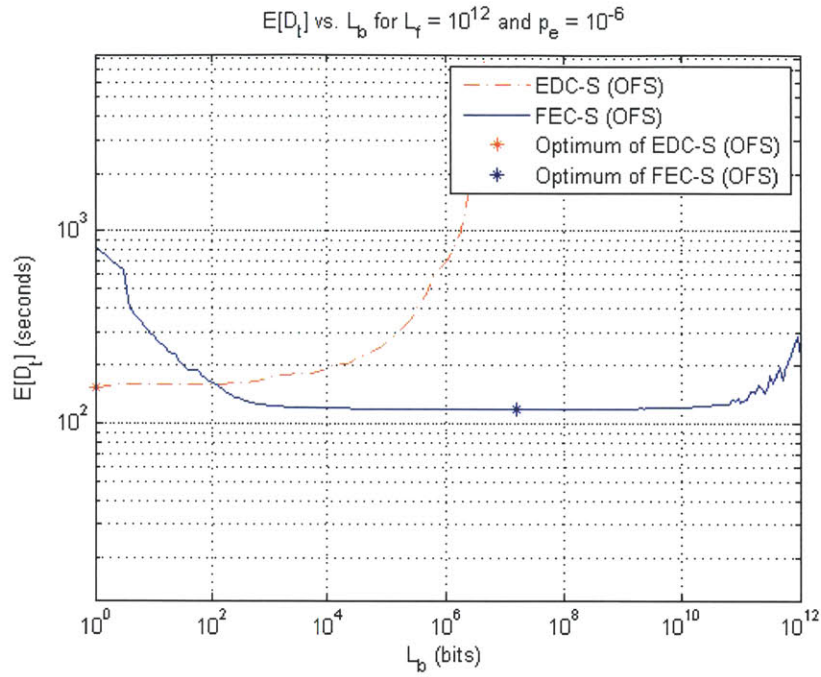


Figure 7.6: Comparison of EDC-S and FEC-S in terms of $E[D_t]$ vs. L_b for $L_f = 10^{12}$ bits, $L_h = 0$, and $p_e = 10^{-6}$.

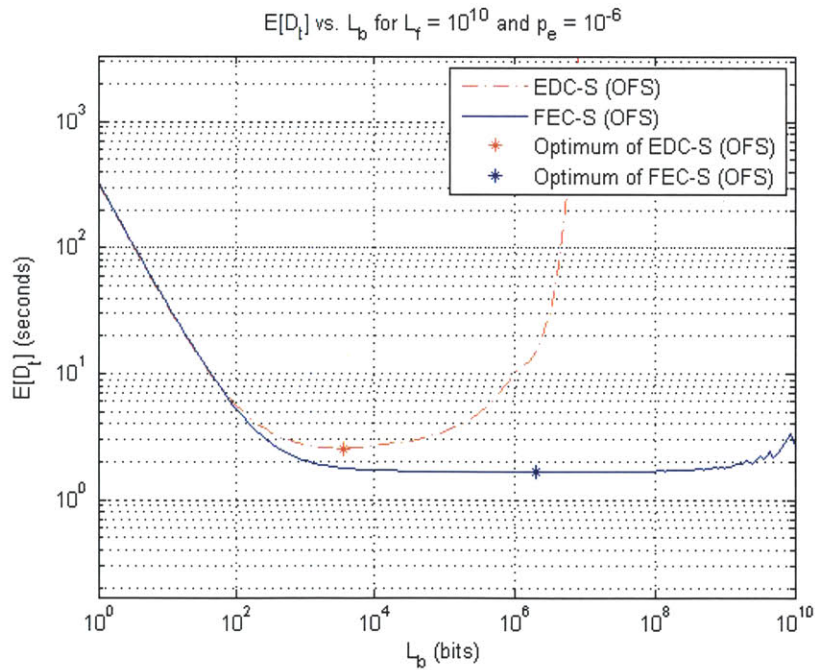


Figure 7.7: Comparison of EDC-S and FEC-S in terms of $E[D_t]$ vs. L_b for $L_f = 10^{10}$ bits, $L_h = 320$ bits, and $p_e = 10^{-6}$.

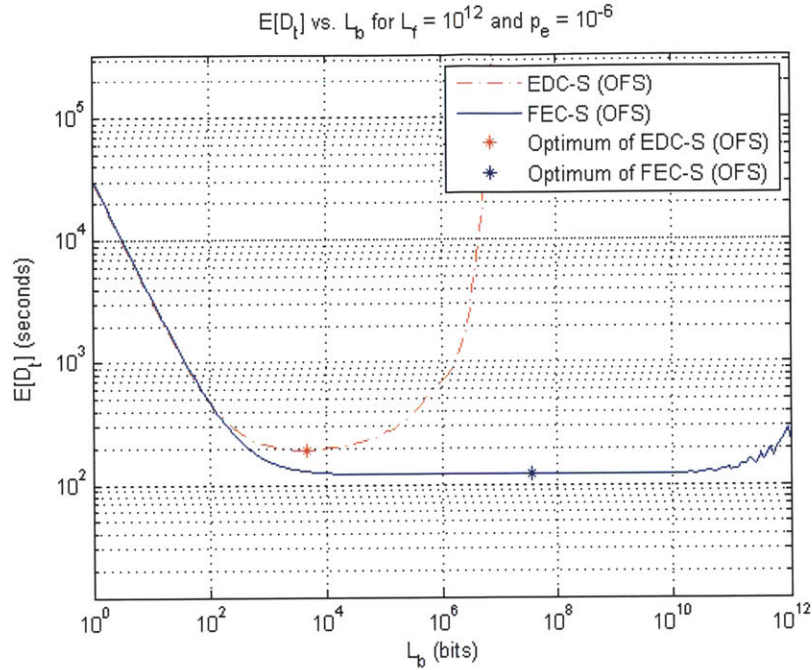


Figure 7.8: Comparison of EDC-S and FEC-S in terms of $E[D_t]$ vs. L_b for $L_f = 10^{12}$ bits, $L_h = 320$ bits, and $p_e = 10^{-6}$.

It can be seen from Fig. 7.1-7.4 that the minimum $E[N_t]$ for FEC-S is almost 1 when the block size reaches a certain value (e.g. $> 10^4$ bits). This is expected, as the probability of block error decreases exponentially for FEC-S as the block size is increased and quickly drops to a value close to 0. The minimum $E[N_t]$ for EDC-S can be larger than 2 when $L_f p_e \geq 1$, as very likely retransmissions are needed.

It can be seen from Fig. 7.5-7.8 that the minimum $E[D_t]$ for FEC-S is smaller than that for EDC-S. This is true in general because minimum $E[N_t]$ can be almost 1 for FEC-S under the condition that $L_b^* \ll L_f$, and $p_{e,b} \approx 0$, where L_b^* is the optimal block size that gives the minimum $E[D_t]$. We can make some approximations to (7.6) and have the following expressions:

$\min(E[D_t])$

$$\approx D_{TCP} - T_{pg}^{EPS} + E[N_t](D_{pg}^{sch} + D_q + T_{pg}^{OFS} + T_{pg}^{EPS}) + \frac{(L_b^* + L_h) \left\lceil \frac{L_f}{L_b^*} \right\rceil}{R_2} + \frac{\Delta(L_b^* + L_h)}{R_{OFS}(1 - p_{e,b})} \quad (7.9)$$

$$\approx D_{TCP} - T_{pg}^{EPS} + (D_{pg}^{sch} + D_q + T_{pg}^{OFS} + T_{pg}^{EPS}) + \frac{(L_b^* + L_h) \left\lceil \frac{L_f}{L_b^*} \right\rceil}{R_{OFS}} \quad (7.10)$$

$$\approx D_{TCP} + D_{pg}^{sch} + D_q + T_{pg}^{OFS} + \frac{(L_b^* + L_h) \left\lceil \frac{L_f}{L_b^*} \right\rceil}{R_{OFS}} \quad (7.11)$$

$$\approx D_{TCP} + D_{pg}^{sch} + D_q + T_{pg}^{OFS} + \frac{L_f}{R_{OFS}} \quad (7.12)$$

The approximation from (7.9) to (7.10) is because $E[N_t] = 1$, $R_2 \approx 1$, and $\left\lceil \frac{L_f}{L_b^*} \right\rceil \gg \Delta$ for $L_f \gg L_b^* > 10^4$ bits. The transition from (7.11) to (7.12) holds when $L_f \gg L_b^* \gg L_h$.

(7.12) gives the minimum total delay that a flow of size L_f can ever have when it is transmitted via OFS. *This implies that among all OFS protocols FEC-S (OFS) can be an optimal protocol that gives the minimum delay for a flow.*

It is of practical interest to find the optimal block size L_b^* for FEC-S that gives the minimum delay. When $p_e \leq 10^{-6}$, the *necessary* conditions for L_b^* to be the optimal block size are that

$$L_b^* > 10^4 \text{ bits, so that } E[N_t] = 1 \text{ and } R_2 \approx 1$$

$$L_f \gg L_b^*, \text{ so that } \left\lceil \frac{L_f}{L_b^*} \right\rceil \gg \Delta, \text{ and}$$

$$L_f \gg L_b^* \gg L_h, \text{ so that } (L_b^* + L_h) \left\lceil \frac{L_f}{L_b^*} \right\rceil \approx L_f.$$

The block size that gives this minimum delay is between 10^4 and $L_f/100$ bits as seen from the plots, with exact range depending on the actual flow size. For $L_f \geq 10^6$ bits, we should choose L_b so that $10^4 \leq L_b \ll L_f$ in order to minimize the total delay, as indicated (and inferred) from Fig. 6.7-6.10. For $10^4 \leq L_f < 10^6$, we can choose $L_b = 10^4$ bits. For all smaller $L_f < 10^4$ bits, no segmentation is needed, and $L_b = L_f$.

7.1.2 Delay Comparison of Best Protocol over OFS and TCP over EPS

We shall next compare the delay of FEC-S (OFS) with that of TCP over EPS. There are four independent parameters that we shall look at: the BER p_e , the propagation delay T_{pg} , the loading factor S and the flow size L_f . In our discussions below, we assume that the OFS line rate is $R_{OFS} = 10$ Gbps, EPS line rate is $R_{EPS} = 10$ Gbps, and the router speed is limited to $R_{router} = 2.5$ Gbps (corresponding to 25% of the full line rate in practice). In this sense, the effective loading factor for EPS is four times of that for OFS because of the router speed limitations. We also assume that there is one additional router for every 600km of fiber distance. For discussions of TCP delay, refer to Chapter 2 for more details.

Effect of BER on Delay Comparison of FEC-S (OFS) and TCP

Fig. 7.9-7.12 show the plots of FEC-S (OFS) and TCP when the BER varies from 0 to 10^{-10} , 10^{-8} and 10^{-6} . Note that T_{pg}^{EPS} is the one-way EPS delay and T_{pg} is only the propagation delay.

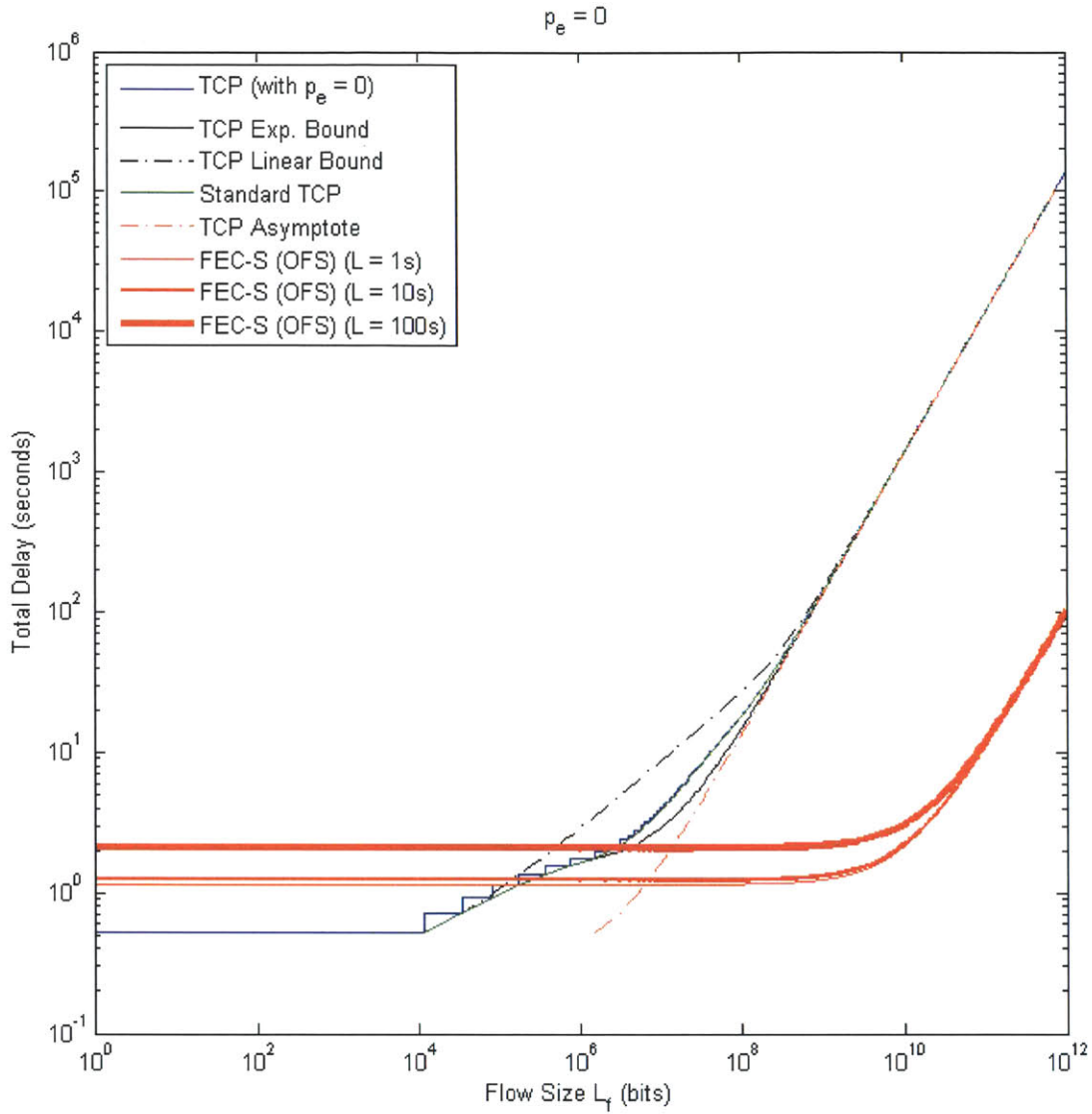


Figure 7.9: Delay comparison of TCP with FEC-S (OFS) when $p_e = 0$, $L_h = 320$ bits, $T_{pg} = 100$ ms, $R_{OFS} = R_{EPS} = 10$ Gbps, $R_{router} = 2.5$ Gbps, $D_{TCP} = 8T_{pg}^{EPS}$, and $D_{pg}^{sch} = 2T_{pg}^{EPS}$. Note that the loading factor is assumed to be 0.24, resulting in an OFS queuing delay D_q of approximately 0.00953 times of the average transaction lengths. It is assumed that there is one router every 600km. The three red solid lines are FEC-S (OFS) delays for different average transaction lengths (here we assume all other files have the same size (e.g. 10^{10} bits or 1s) except for the one being considered).

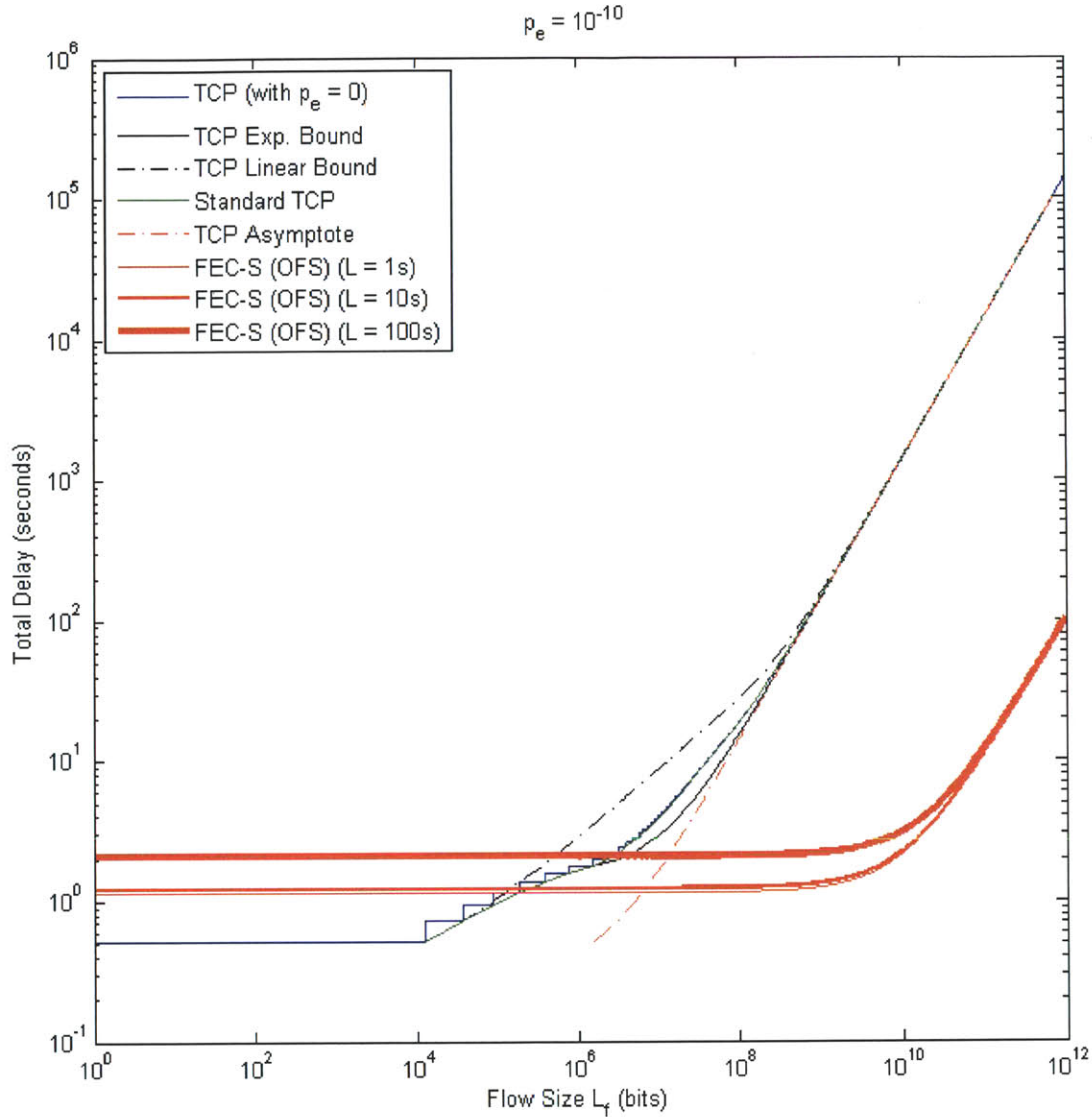


Figure 7.10: Delay comparison of TCP with FEC-S (OFS) when $p_e = 10^{-10}$, $L_h = 320$ bits, $T_{pg} = 100$ ms, $R_{OFS} = R_{EPS} = 10$ Gbps, $R_{router} = 2.5$ Gbps, $D_{TCP} = 8T_{pg}^{EPS}$, and $D_{pg}^{sch} = 2T_{pg}^{EPS}$. Note that the loading factor is assumed to be 0.24, resulting in an OFS queuing delay D_q of approximately 0.00953 times of the average transaction lengths. It is assumed that there is one router every 600km. The three red solid lines are FEC-S (OFS) delays for different average transaction lengths (here we assume all other files have the same size (e.g. 10^{10} bits or 1s) except for the one being considered).

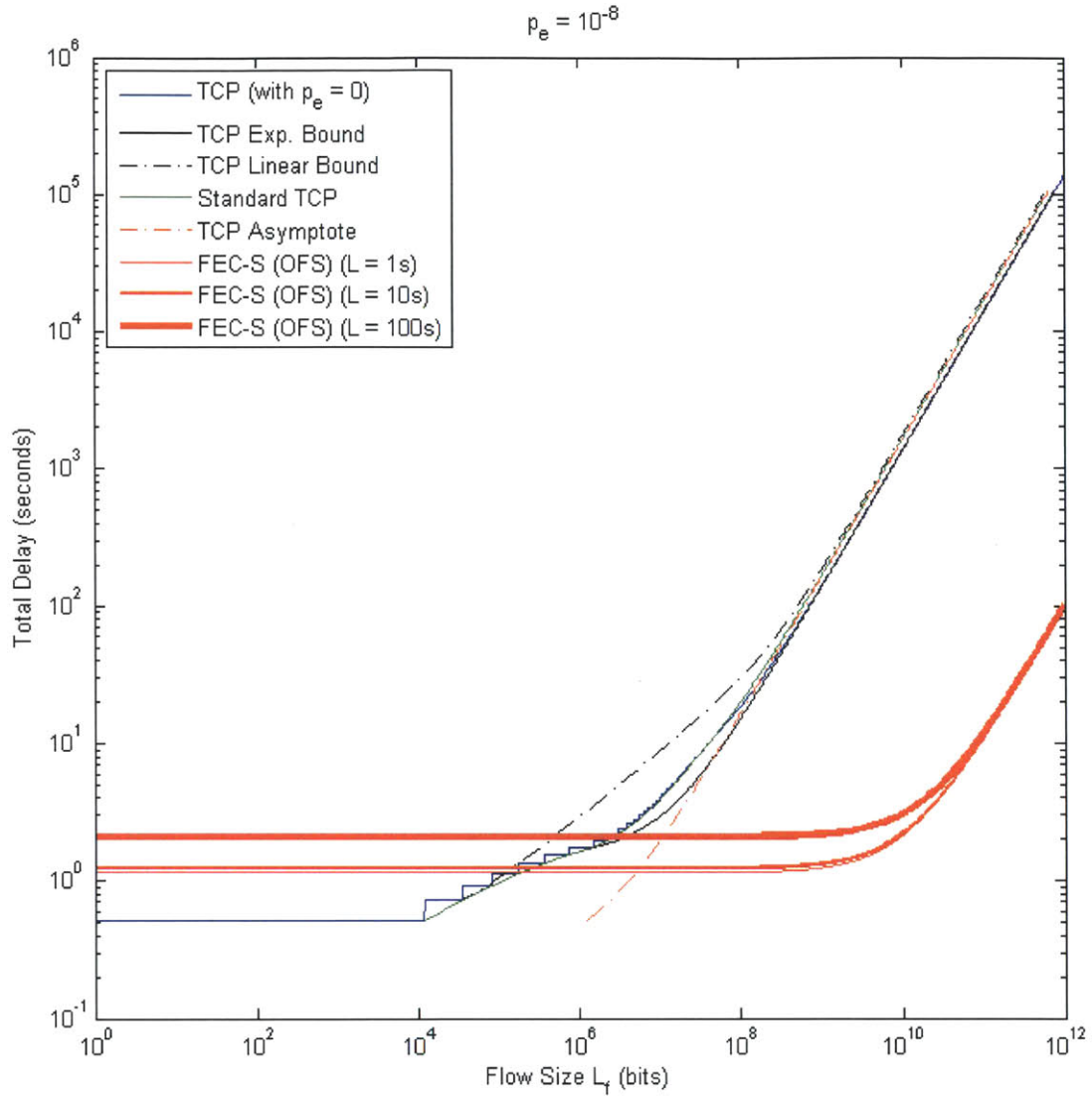


Figure 7.11: Delay comparison of TCP with FEC-S (OFS) when $p_e = 10^{-8}$, $L_h = 320$ bits, $T_{pg} = 100\text{ms}$, $R_{OFS} = R_{EPS} = 10\text{Gbps}$, $R_{router} = 2.5\text{Gbps}$, $D_{TCP} = 8T_{pg}^{EPS}$, and $D_{pg}^{sch} = 2T_{pg}^{EPS}$. Note that the loading factor is assumed to be 0.24, resulting in an OFS queuing delay D_q of approximately 0.00953 times of the average transaction lengths. It is assumed that there is one router every 600km. The three red solid lines are FEC-S (OFS) delays for different average transaction lengths (here we assume all other files have the same size (e.g. 10^{10} bits or 1s) except for the one being considered).

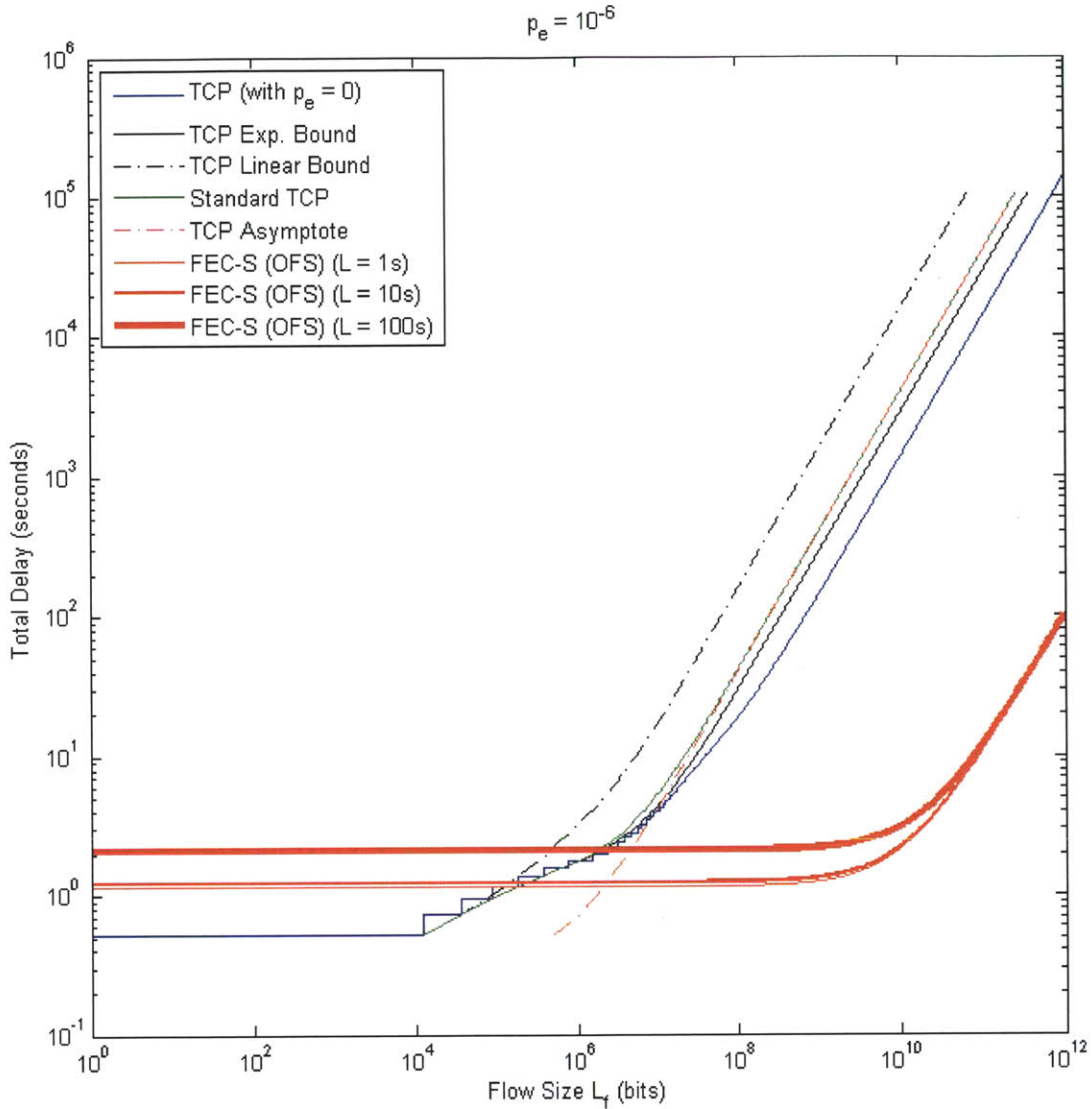


Figure 7.12: Delay comparison of TCP with FEC-S (OFS) when $p_e = 10^{-6}$, $L_h = 320$ bits, $T_{pg} = 100$ ms, $R_{OFS} = R_{EPS} = 10$ Gbps, $R_{router} = 2.5$ Gbps, $D_{TCP} = 8T_{pg}^{EPS}$, and $D_{pg}^{sch} = 2T_{pg}^{EPS}$. Note that the loading factor is assumed to be 0.24, resulting in an OFS queuing delay D_q of approximately 0.00953 times of the average transaction lengths. It is assumed that there is one router every 600km. The three red solid lines are FEC-S (OFS) delays for different average transaction lengths (here we assume all other files have the same size (e.g. 10^{10} bits or 1s) except for the one being considered).

It can be seen from Fig. 7.9-7.12 that when the BER increases, the total TCP delay increases for a given flow size with all other parameters remaining the same. That is, a larger BER gives a smaller cut-off flow size (i.e. the x-axis of the crossing point) where TCP delay and FEC-S (OFS) delay are the same.

For the same BER, with different average file lengths (i.e. 1s, 10s and 100s), the crossing point increases with increasing average file lengths for OFS. This is because OFS queuing delay is positively proportional to the average file lengths (see Chapter 3 for more information), while TCP is not affected.

Effect of Propagation Delay on Delay Comparison of FEC-S (OFS) and TCP

Besides the bit error rate, the actual crossing point of EDC-S (OFS) and TCP also depends on two other factors: the propagation delay and the loading factor (or queuing delay).

Fig. 7.13-7.16 are plots when we decrease the propagation delay by a factor of 10 (i.e. $T_{pg} = 10$ ms), with relevant parameters scaled accordingly.

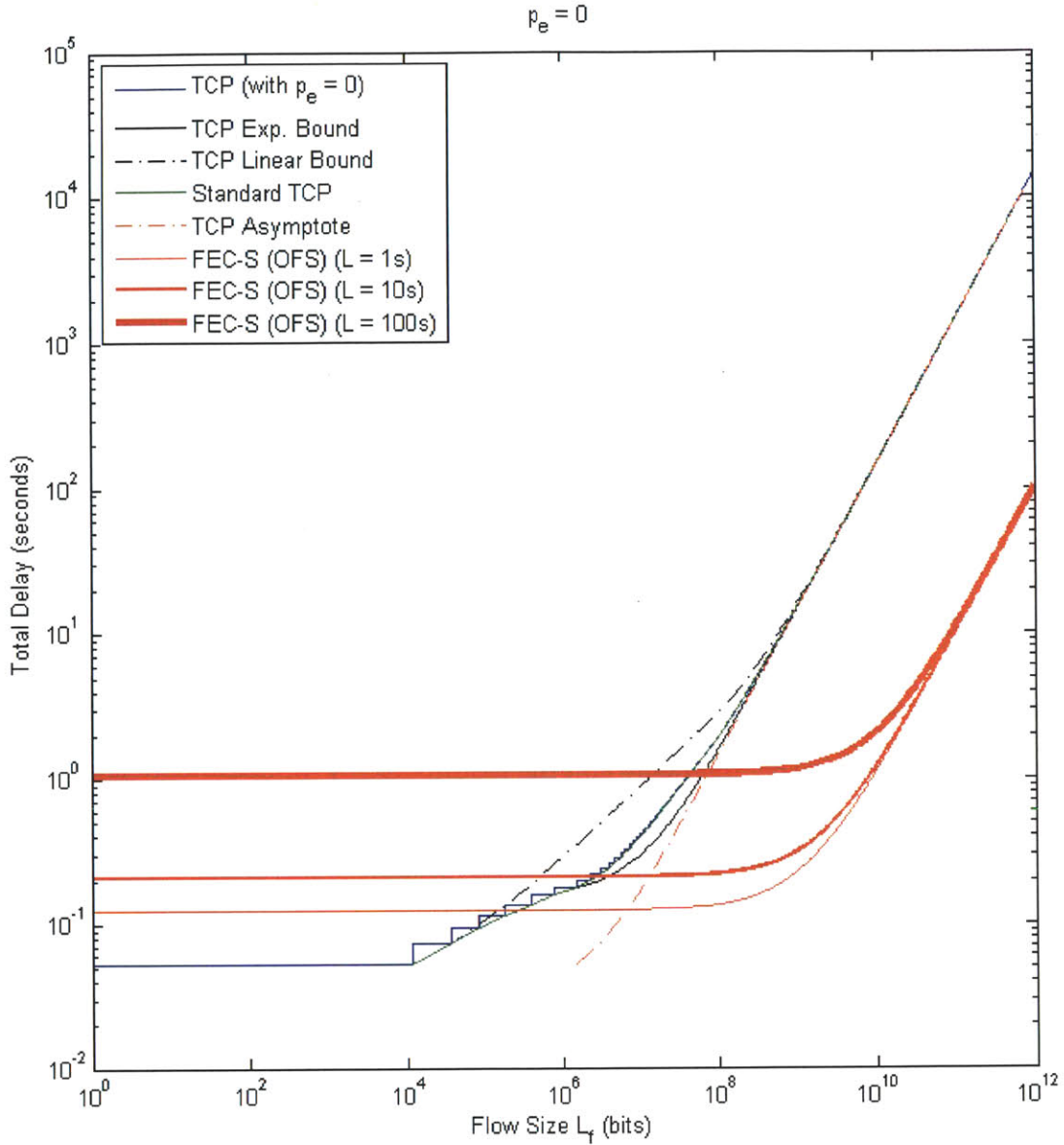


Figure 7.13: Delay comparison of TCP with FEC-S (OFS) when $p_e = 0$, $L_h = 320$ bits, $T_{pg} = 10$ ms, $R_{OFS} = R_{EPS} = 10$ Gbps, $R_{router} = 2.5$ Gbps, $D_{TCP} = 8T_{pg}^{EPS}$, and $D_{pg}^{sch} = 2T_{pg}^{EPS}$. Note that the loading factor is assumed to be 0.24, resulting in an OFS queuing delay D_q of approximately 0.00953 times of the average transaction lengths. It is assumed that there is one router every 600km. The three red solid lines are FEC-S (OFS) delays for different average transaction lengths (here we assume all other files have the same size (e.g. 10^{10} bits or 1s) except for the one being considered).

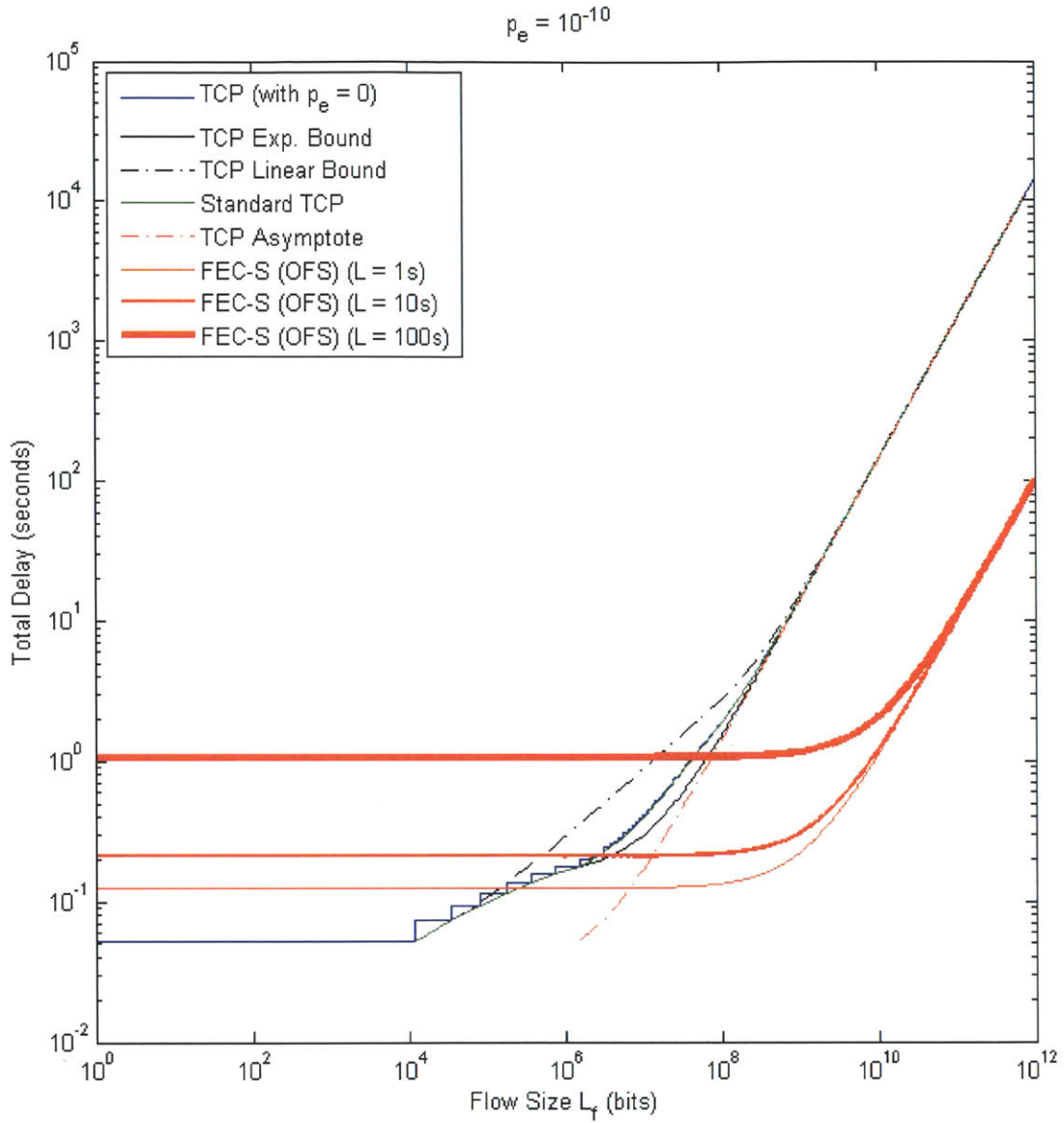


Figure 7.14: Delay comparison of TCP with FEC-S (OFS) when $p_e = 10^{-10}$, $L_h = 320$ bits, $T_{pg} = 10\text{ms}$, $R_{OFS} = R_{EPS} = 10\text{Gbps}$, $R_{router} = 2.5\text{Gbps}$, $D_{TCP} = 8T_{pg}^{EPS}$, and $D_{pg}^{sch} = 2T_{pg}^{EPS}$. Note that the loading factor is assumed to be 0.24, resulting in an OFS queuing delay D_q of approximately 0.00953 times of the average transaction lengths. It is assumed that there is one router every 600km. The three red solid lines are FEC-S (OFS) delays for different average transaction lengths (here we assume all other files have the same size (e.g. 10^{10} bits or 1s) except for the one being considered).

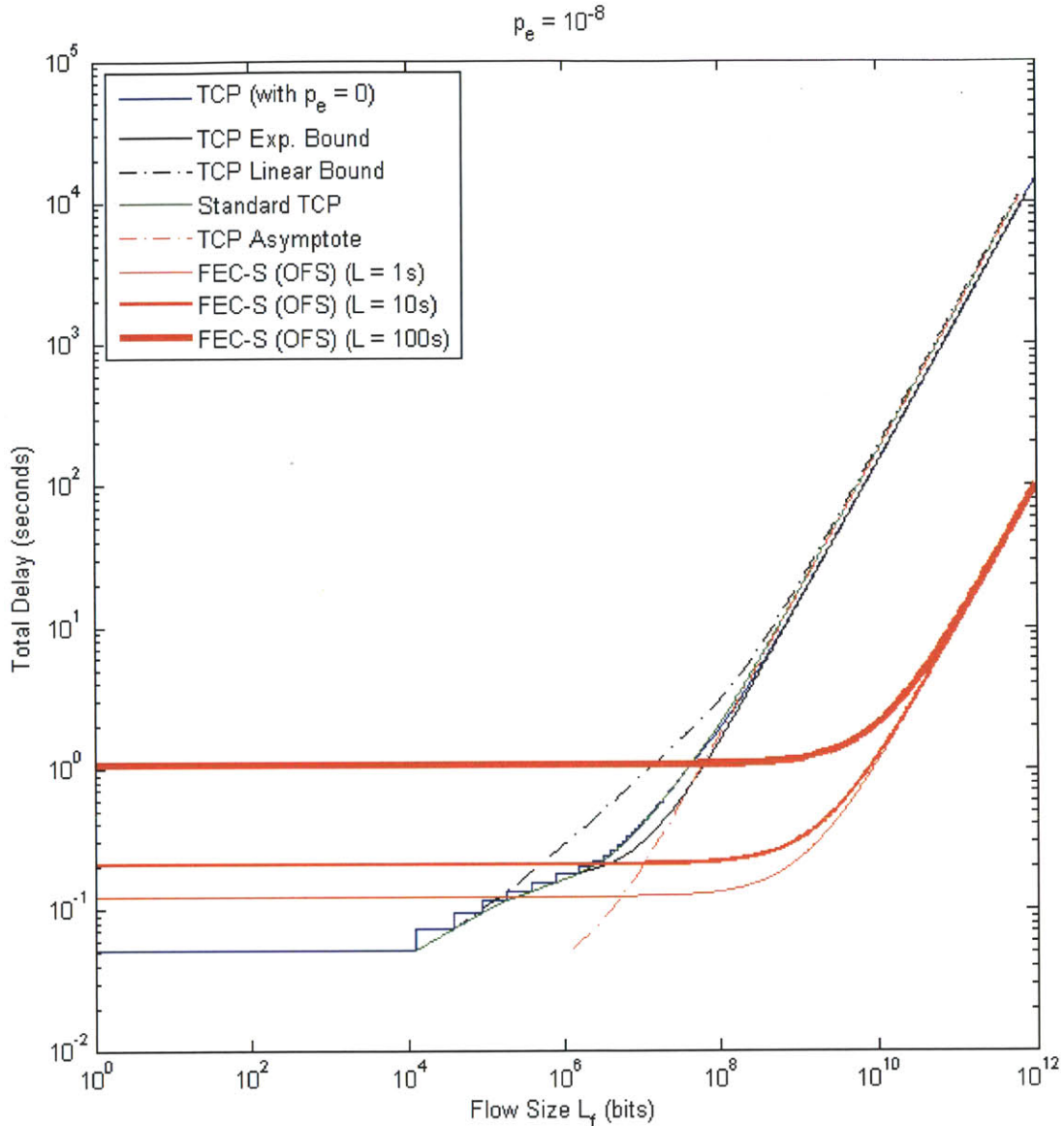


Figure 7.15: Delay comparison of TCP with FEC-S (OFS) when $p_e = 10^{-8}$, $L_h = 320$ bits, $T_{pg} = 10$ ms, $R_{OFS} = R_{EPS} = 10$ Gbps, $R_{router} = 2.5$ Gbps, $D_{TCP} = 8T_{pg}^{EPS}$, and $D_{pg}^{sch} = 2T_{pg}^{EPS}$. Note that the loading factor is assumed to be 0.24, resulting in an OFS queuing delay D_q of approximately 0.00953 times of the average transaction lengths. It is assumed that there is one router every 600km. The three red solid lines are FEC-S (OFS) delays for different average transaction lengths (here we assume all other files have the same size (e.g. 10^{10} bits or 1s) except for the one being considered).

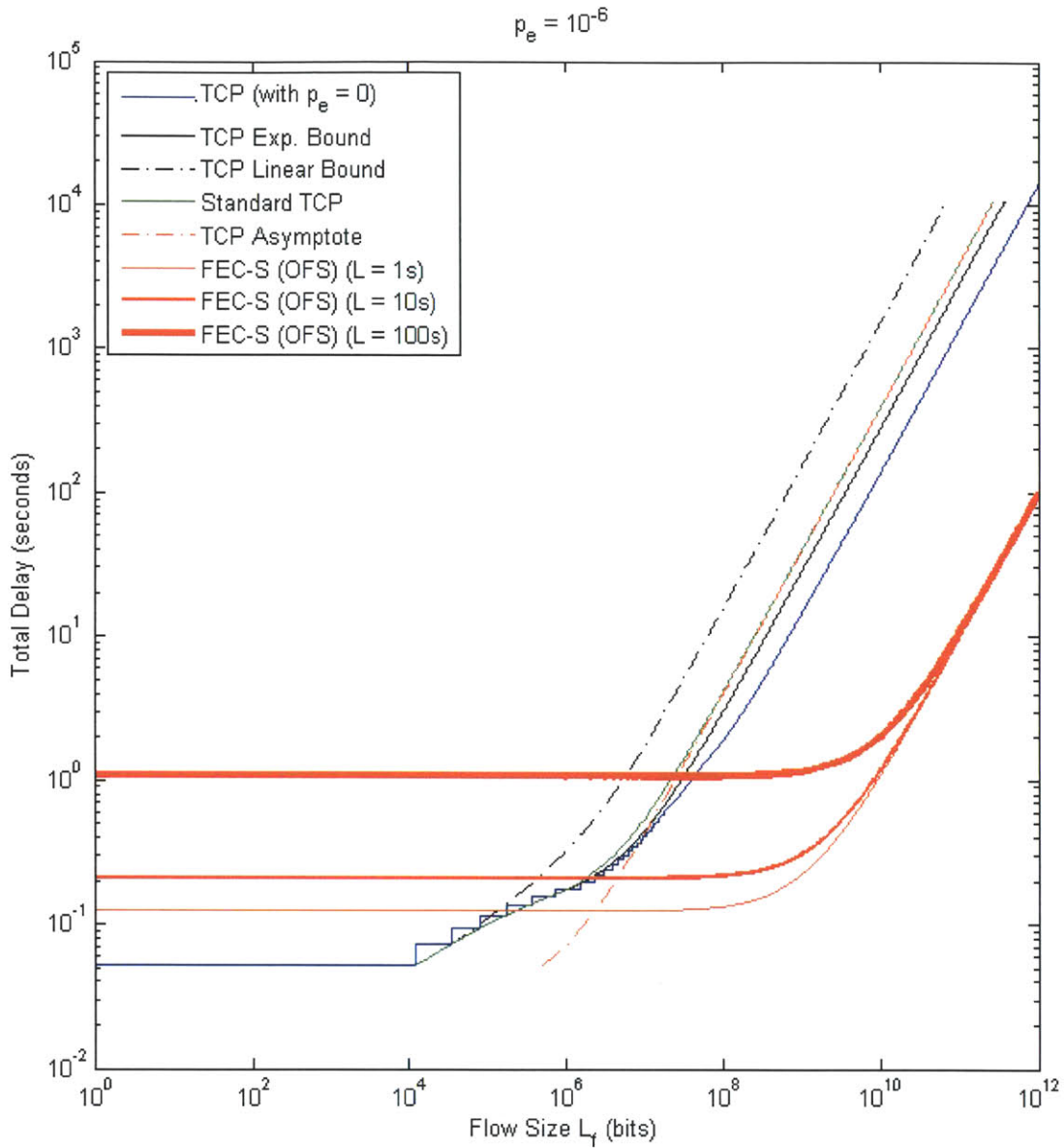


Figure 7.16: Delay comparison of TCP with FEC-S (OFS) when $p_e = 10^{-6}$, $L_h = 320$ bits, $T_{pg} = 10$ ms, $R_{OFS} = R_{EPS} = 10$ Gbps, $R_{router} = 2.5$ Gbps, $D_{TCP} = 8T_{pg}^{EPS}$, and $D_{pg}^{sch} = 2T_{pg}^{EPS}$. Note that the loading factor is assumed to be 0.24, resulting in an OFS queuing delay D_q of approximately 0.00953 times of the average transaction lengths. It is assumed that there is one router every 600km. The three red solid lines are FEC-S (OFS) delays for different average transaction lengths (here we assume all other files have the same size (e.g. 10^{10} bits or 1s) except for the one being considered).

It can be seen from Fig. 7.13-7.16 that, when compared to Fig. 7.9-7.12, the crossing point moves to the right for the same BER, i.e. with a bigger value. This is because TCP depends highly on the RTTs (and propagation delays) and will increase in rate when the RTT is reduced. On the other hand, FEC-S (OFS) is less affected by the RTT and stays relatively constant when the RTT changes within a small range.

Effect of Loading Factor on Delay Comparison of FEC-S (OFS) and TCP

Loading factor may also affect the crossing point by affecting both the delay of FEC-S (OFS) and TCP. When we change the loading factor from 0.24 to 0.2499, the queuing delay for both FEC-S (OFS) and TCP increase but by different amounts. We show their behavior plots in Fig. 7.17-7.20. For expressions of queuing delay as a function of loading factor, refer to Chapter 2 (for TCP over EPS) and 3 (for OFS) for more information.

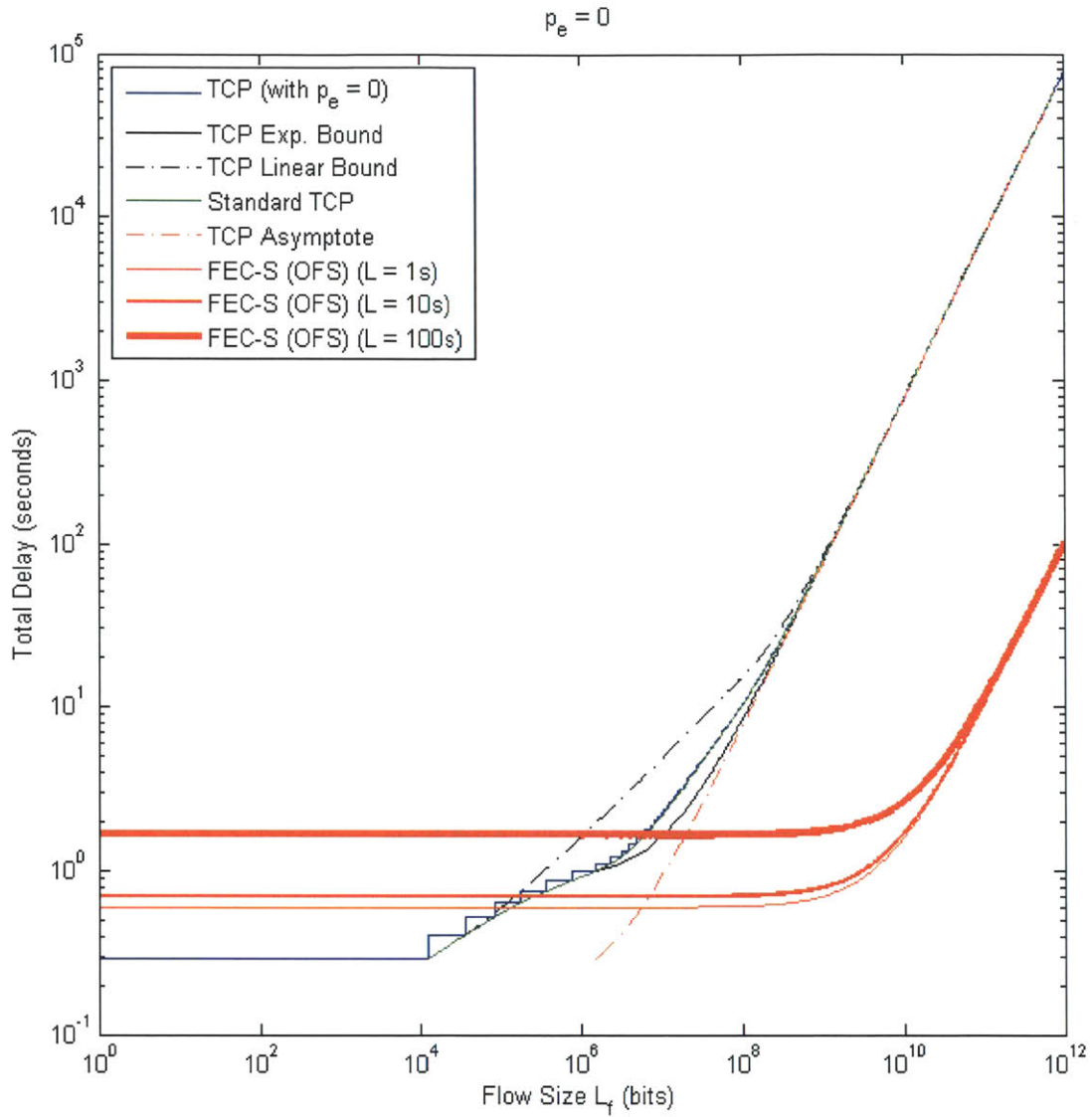


Figure 7.17: Delay comparison of TCP with FEC-S (OFS) when $p_e = 0$, $L_h = 320$ bits, $T_{pg} = 10\text{ms}$, $R_{OFS} = R_{EPS} = 10\text{Gbps}$, $R_{router} = 2.5\text{Gbps}$, $D_{TCP} = 8T_{pg}^{EPS}$, and $D_{pg}^{sch} = 2T_{pg}^{EPS}$. Note that the loading factor is assumed to be **0.2499**, resulting in an OFS queuing delay D_q of approximately 0.0107 times of the average transaction lengths. It is assumed that there is one router every 600km. The three red solid lines are FEC-S (OFS) delays for different average transaction lengths (here we assume all other files have the same size (e.g. 10^{10} bits or 1s) except for the one being considered).

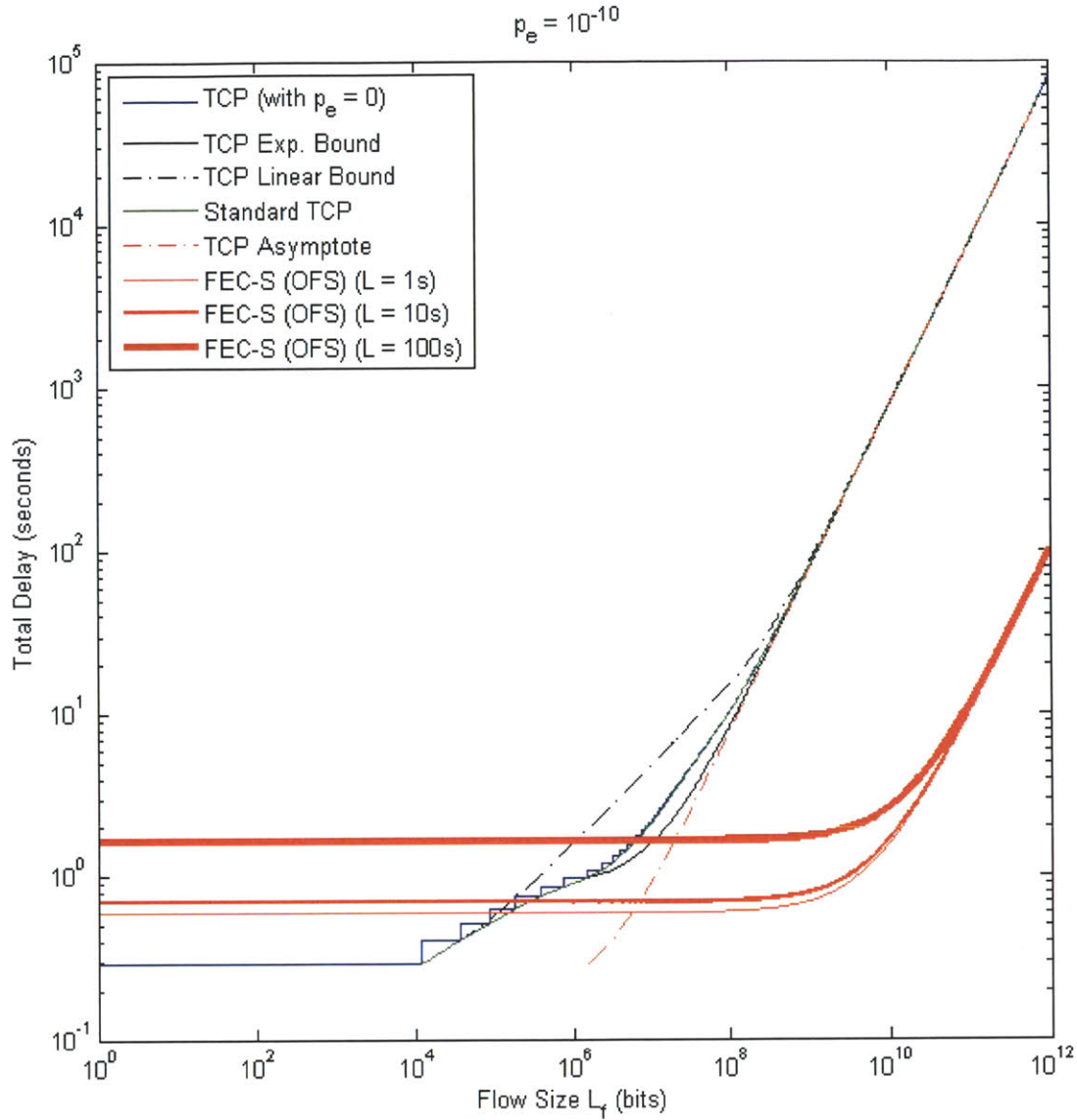


Figure 7.18: Delay comparison of TCP with FEC-S (OFS) when $p_e = 10^{-10}$, $L_h = 320$ bits, $T_{pg} = 10\text{ms}$, $R_{OFS} = R_{EPS} = 10\text{Gbps}$, $R_{router} = 2.5\text{Gbps}$, $D_{TCP} = 8T_{pg}^{EPS}$, and $D_{pg}^{sch} = 2T_{pg}^{EPS}$. Note that the loading factor is assumed to be **0.2499**, resulting in an OFS queuing delay D_q of approximately 0.0107 times of the average transaction lengths. It is assumed that there is one router every 600km. The three red solid lines are FEC-S (OFS) delays for different average transaction lengths (here we assume all other files have the same size (e.g. 10^{10} bits or 1s) except for the one being considered).

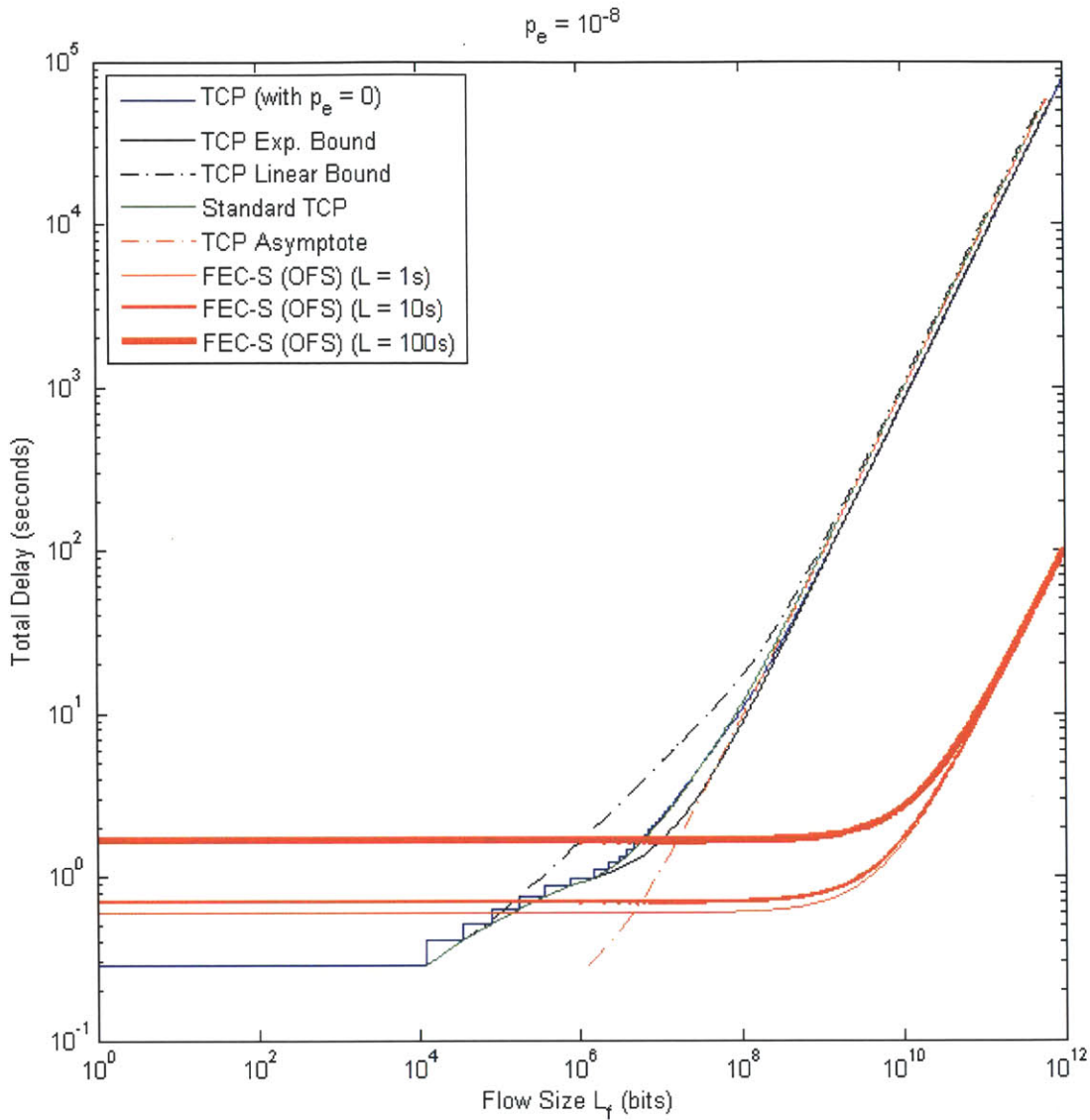


Figure 7.19: Delay comparison of TCP with FEC-S (OFS) when $p_e = 10^{-8}$, $L_h = 320$ bits, $T_{pg} = 10\text{ms}$, $R_{OFS} = R_{EPS} = 10\text{Gbps}$, $R_{router} = 2.5\text{Gbps}$, $D_{TCP} = 8T_{pg}^{EPS}$, and $D_{pg}^{sch} = 2T_{pg}^{EPS}$. Note that the loading factor is assumed to be 0.2499, resulting in an OFS queuing delay D_q of approximately 0.0107 times of the average transaction lengths. It is assumed that there is one router every 600km. The three red solid lines are FEC-S (OFS) delays for different average transaction lengths (here we assume all other files have the same size (e.g. 10^{10} bits or 1s) except for the one being considered).

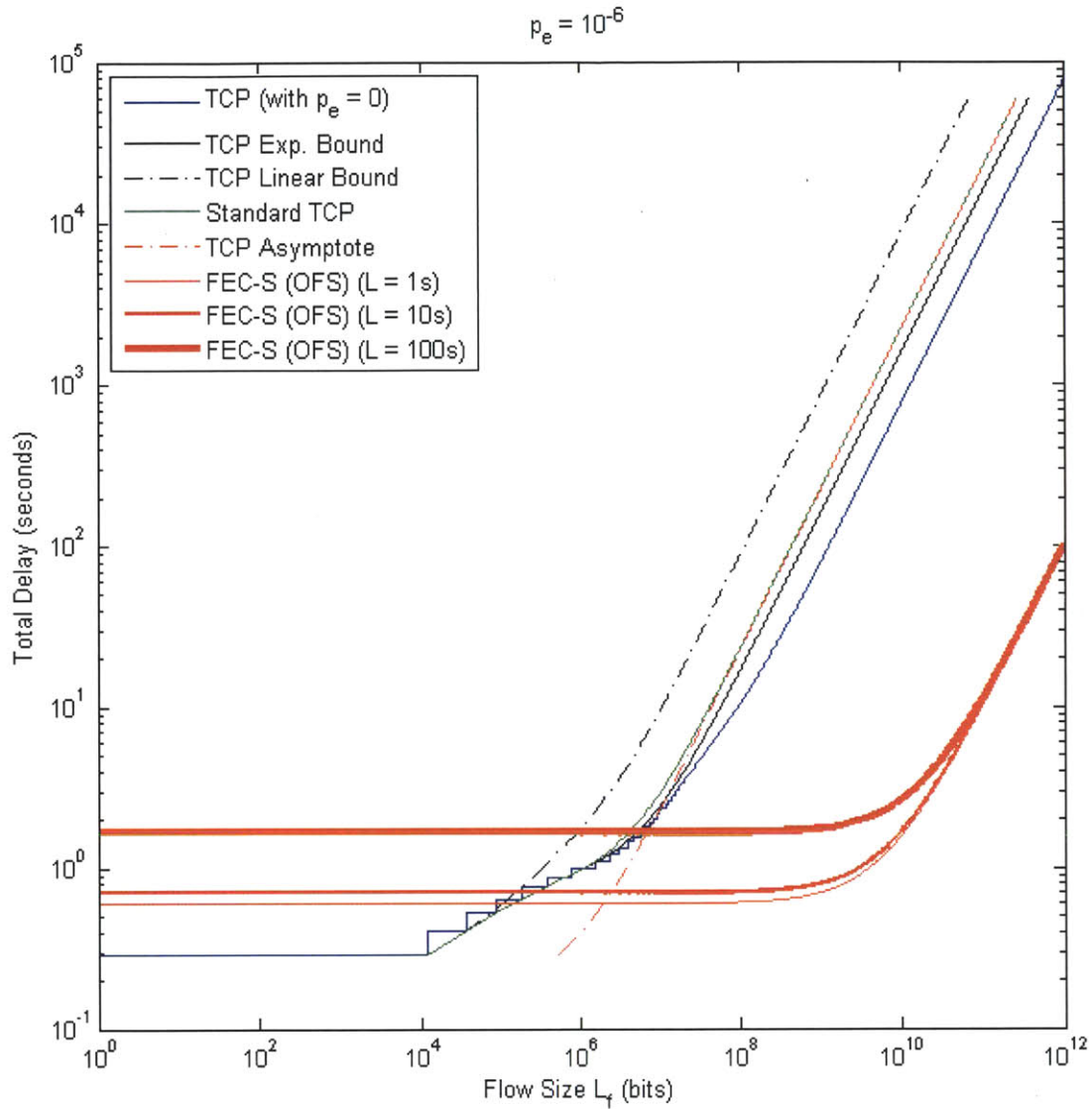


Figure 7.20: Delay comparison of TCP with FEC-S (OFS) when $p_e = 10^{-6}$, $L_h = 320$ bits, $T_{pg} = 10\text{ms}$, $R_{OFS} = R_{EPS} = 10\text{Gbps}$, $R_{router} = 2.5\text{Gbps}$, $D_{TCP} = 8T_{pg}^{EPS}$, and $D_{pg}^{sch} = 2T_{pg}^{EPS}$. Note that the loading factor is assumed to be **0.2499**, resulting in an OFS queuing delay D_q of approximately **0.0107** times of the average transaction lengths. It is assumed that there is one router every 600km. The three red solid lines are FEC-S (OFS) delays for different average transaction lengths (here we assume all other files have the same size (e.g. 10^{10} bits or 1s) except for the one being considered).

Plots in Fig. 7.13-7.16 and plots in Fig. 7.17-7.20 differ only in the loading factors. When we compare each pair of the plots, we can see that with only loading factor changed from 0.24 to 0.2499, the crossing point also shifts to the left. This is expected, as 0.2499 is very close to 0.25 and the effective loading factor for EPS is 0.9996 which corresponds to a nearly full load. But for OFS, the change is only from 0.00953 to 0.0107 for the queuing delay coefficients (See captions of Fig. 7.13-7.20). However, we should not conclude from here that a larger loading factor always results in a smaller crossing point. In fact when the router is not nearly fully loaded, the reverse is true: a larger effective loading factor (e.g. < 0.9 though) results in a larger crossing point, as the effect of loading factor on OFS is larger than that on EPS. We shall see further these effects in Section 7.2 below. Note we have been assuming the routers have infinite buffer size with no packet loss due to congestion. Routers in the field all have finite buffer and will drop packets during congestion triggering TCP window closing, further reducing transmission rate and increasing delay. Thus, The EPS delay given in this chapter is only an overly optimistic case.

7.2 Preference Maps

We can draw the following preference maps in Fig. 7.1-7.4 based on different values of flow lengths, propagation delays and loading factors. In Fig. 7.1-7.2, the five lines are the boundary lines between TCP and FEC-S (OFS). For a given OFS loading factor (e.g. $S = 0.5$), every (L_f, T_{pg}) pair corresponds to either the TCP (EPS) region below the line (e.g. the red line), or FEC-S (OFS) region above the line. Along the lines, both TCP and FEC-S protocols give the same total delay, defined as the time from the moment the data file is requested for transmission at the sender to the moment the file is correctly received and passed to the application layer at the receiver.

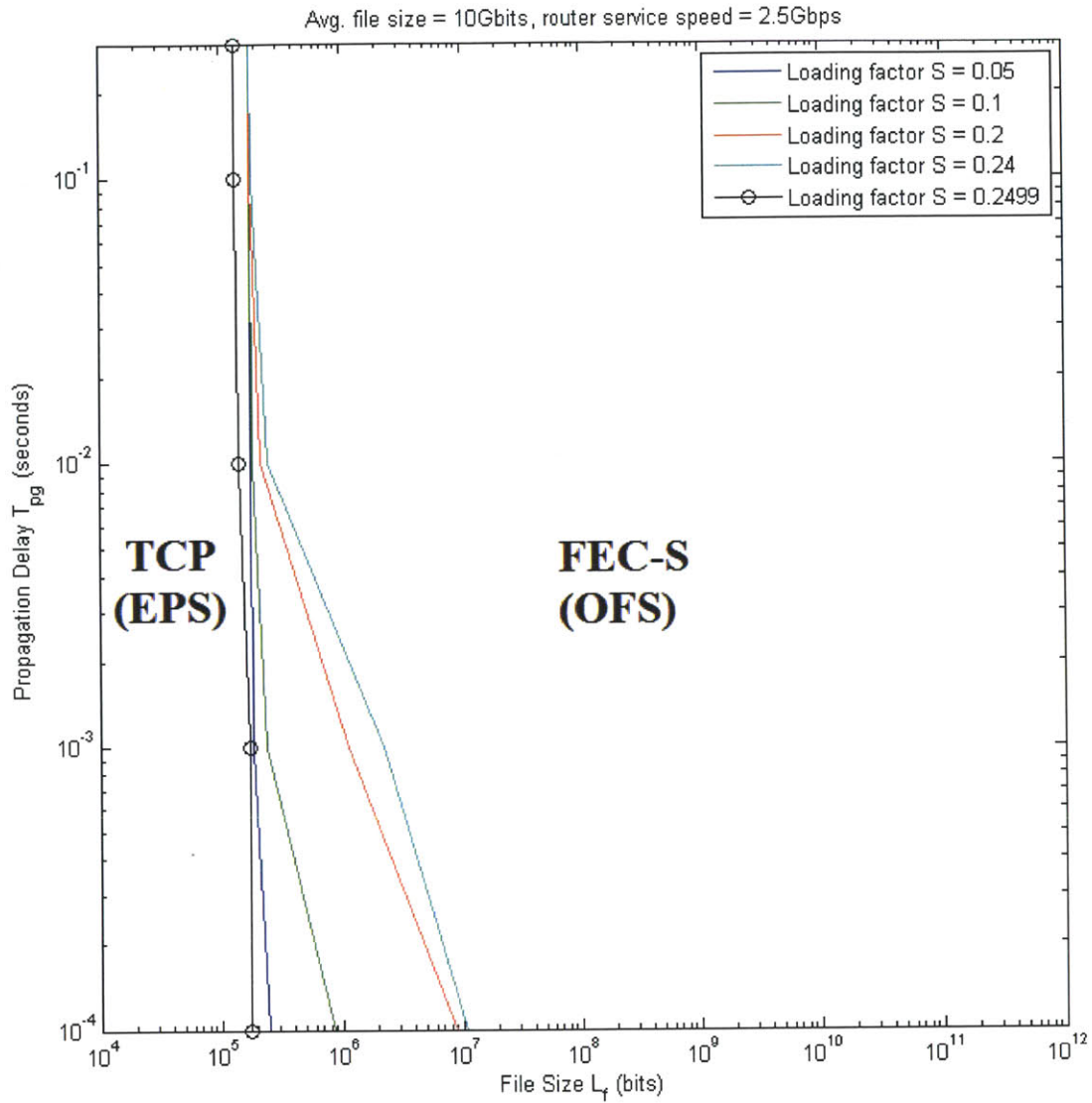


Figure 7.22 (a): Preference maps for EPS and OFS for different flow lengths L_f and propagation delays when $p_e = 10^{-8}$. The size of all other files is assumed to be **10Gbits** except for the one being considered, which can be of any size. The line rates are assumed to be 10Gbps for both EPS and OFS, but the router service speed is limited to be **2.5Gbps**. The four solid lines are the boundaries for TCP and FEC-S protocols with loading factors $S = 0.05, 0.1, 0.2$ and 0.24 respectively (which correspond to loading factors $0.2, 0.4, 0.8$ and 0.96 for router service speed 2.5Gbps). For each loading factor, if the pair (L_f, T_{pg}) falls into the region to the right of the boundary line, then FEC-S (OFS) is preferred; otherwise TCP over EPS is preferred. $R_{\text{OFS}} = 10\text{Gbps}$, $D_{\text{TCP}} = 8T_{\text{pg}}^{\text{EPS}}$, $D_{\text{pg}}^{\text{sch}} = 2T_{\text{pg}}^{\text{EPS}}$.

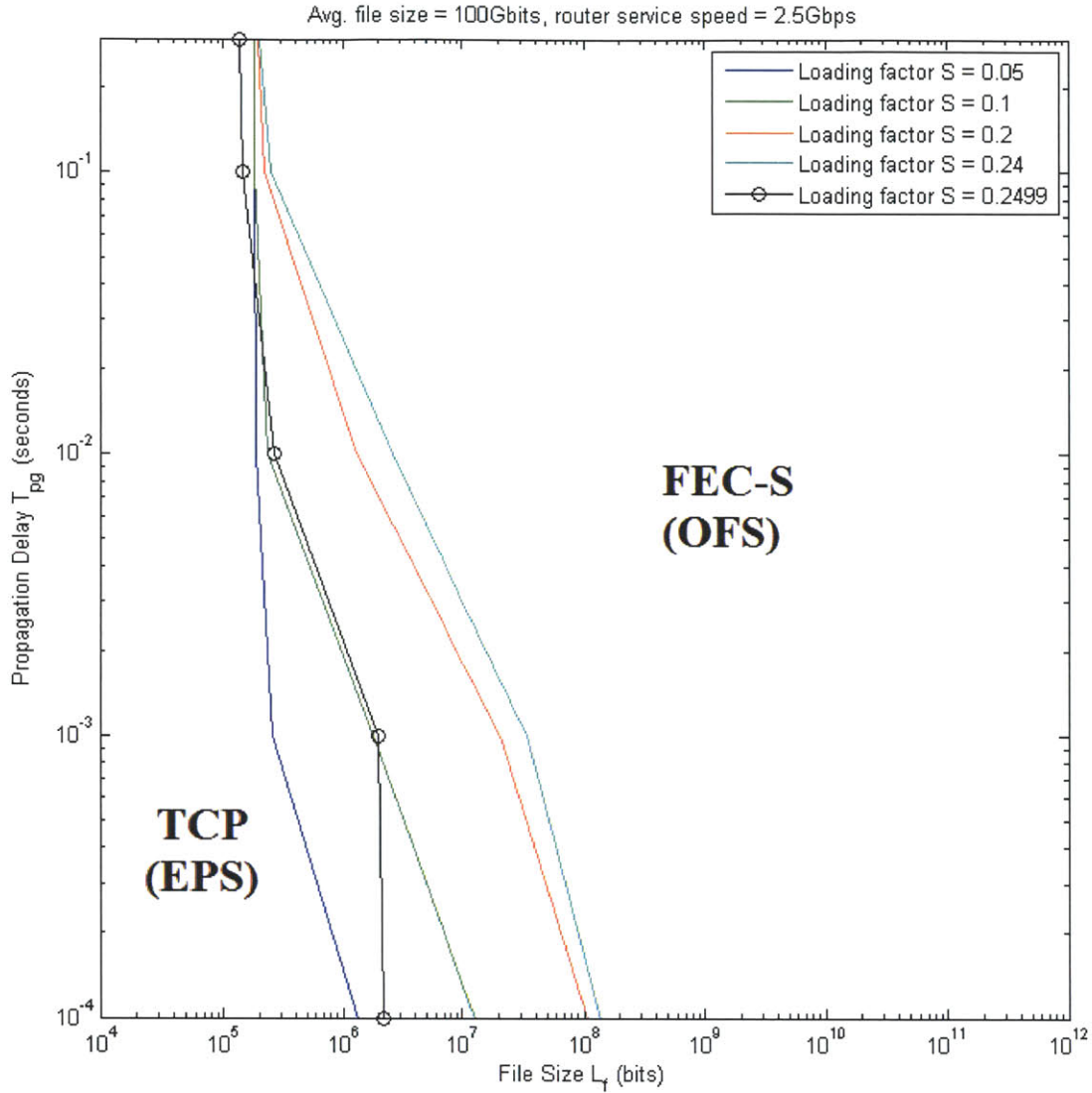


Figure 7.22 (b): Preference maps for EPS and OFS for different flow lengths L_f and propagation delays when $p_e = 10^{-8}$. The size of all other files is assumed to be **100Gbits** except for the one being considered, which can be of any size. The line rates are assumed to be 10Gbps for both EPS and OFS, but the router service speed is limited to be **2.5Gbps**. The five lines are the boundaries for TCP and FEC-S protocols with loading factors $S = 0.05, 0.1, 0.2, 0.24$ and 0.2499 respectively (which correspond to loading factors 0.2, 0.4, 0.8 and 0.96 for router service speed 2.5Gbps). For each loading factor, if the pair (L_f, T_{pg}) falls into the region to the right of the boundary line, then FEC-S (OFS) is preferred; otherwise TCP over EPS is preferred. $R_{OFS} = 10\text{Gbps}$, $D_{TCP} = 8T_{pg}^{EPS}$, $D_{pg}^{sch} = 2T_{pg}^{EPS}$.

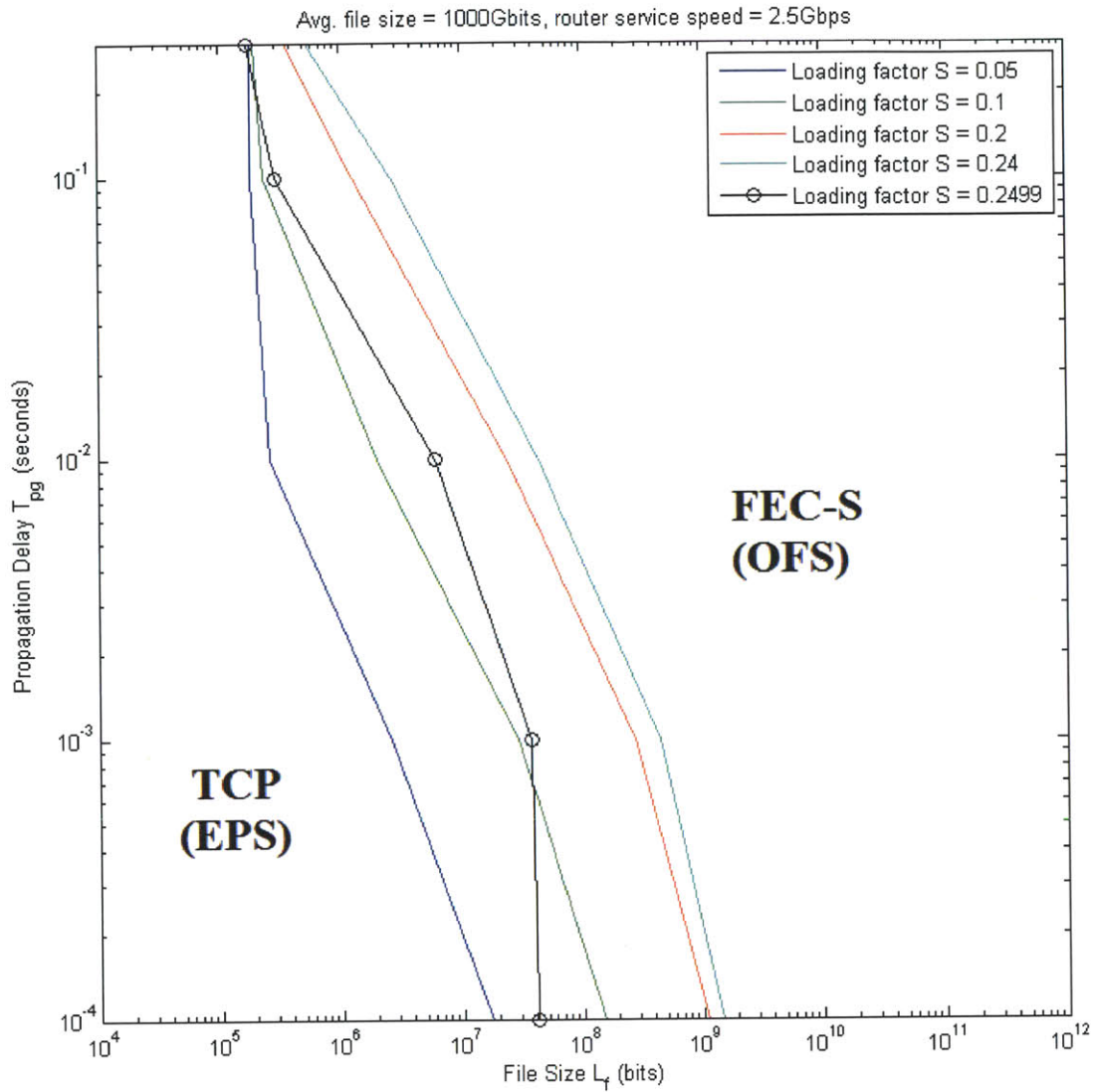


Figure 7.22 (c): Preference maps for EPS and OFS for different flow lengths L_f and propagation delays when $p_e = 10^{-8}$. The size of all other files is assumed to be 1000Gbits except for the one being considered, which can be of any size. The line rates are assumed to be 10Gbps for both EPS and OFS, but the router service speed is limited to be 2.5Gbps. The four solid lines are the boundaries for TCP and FEC-S protocols with loading factors $S = 0.05, 0.1, 0.2$ and 0.24 respectively (which correspond to loading factors 0.2, 0.4, 0.8 and 0.96 for router service speed 2.5Gbps). For each loading factor, if the pair (L_f, T_{pg}) falls into the region to the right of the boundary line, then FEC-S (OFS) is preferred; otherwise TCP over EPS is preferred. $R_{\text{OFS}} = 10\text{Gbps}$, $D_{\text{TCP}} = 8T_{\text{pg}}^{\text{EPS}}$, $D_{\text{pg}}^{\text{sch}} = 2T_{\text{pg}}^{\text{EPS}}$.

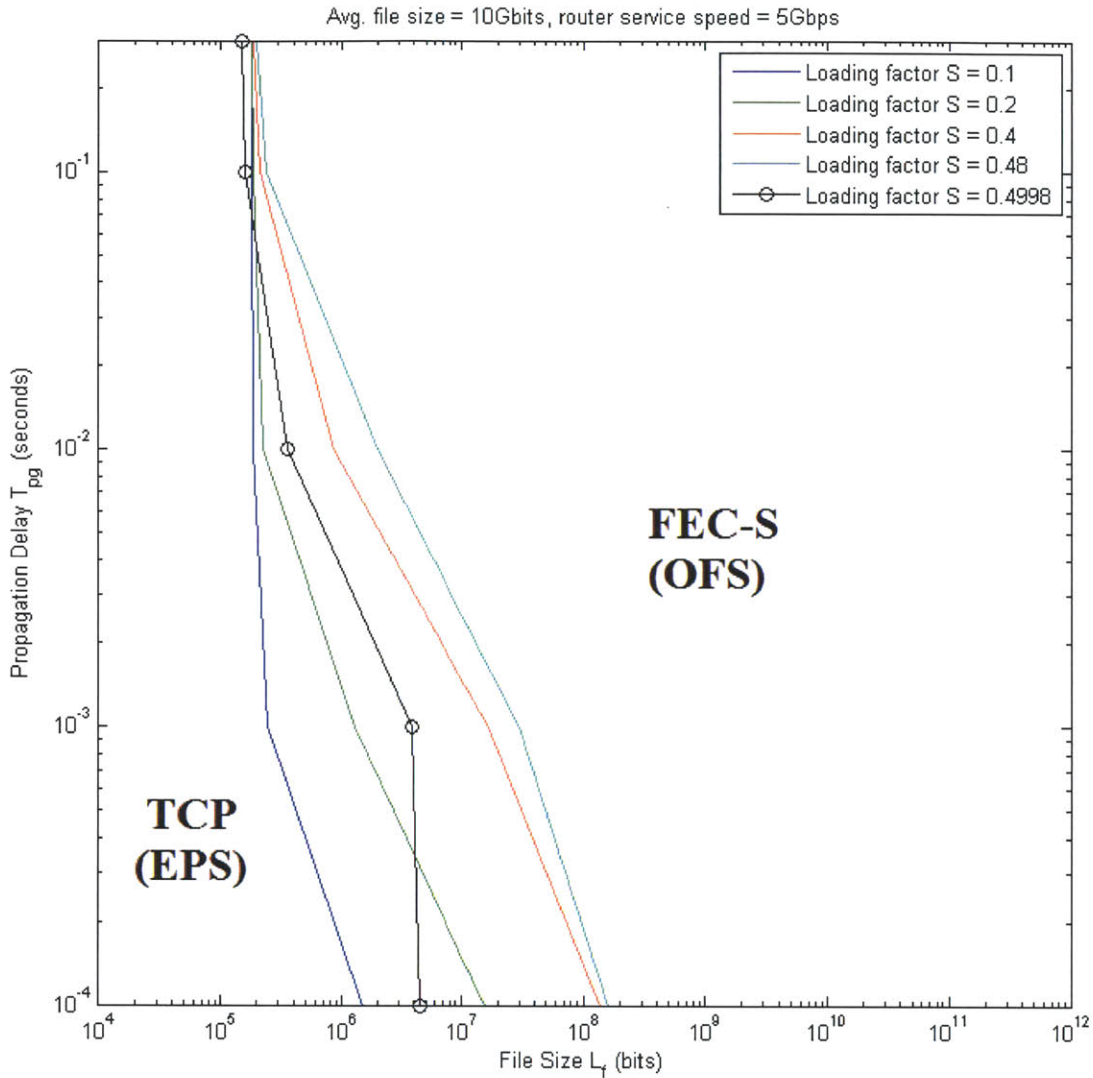


Figure 7.23 (a): Preference maps for EPS and OFS for different flow lengths L_f and propagation delays when $p_e = 10^{-8}$. The size of all other files is assumed to be 10Gbits except for the one being considered, which can be of any size. The line rates are assumed to be 10Gbps for both EPS and OFS, but the router service speed is limited to be 5Gbps. The four solid lines are the boundaries for TCP and FEC-S protocols with loading factors $S = 0.1, 0.2, 0.4$ and 0.48 respectively (which correspond to loading factors 0.2, 0.4, 0.8 and 0.96 for router service speed 2.5Gbps). For each loading factor, if the pair (L_f, T_{pg}) falls into the region to the right of the boundary line, then FEC-S (OFS) is preferred; otherwise TCP over EPS is preferred. $R_{OFS} = 10\text{Gbps}$, $D_{TCP} = 8T_{pg}^{EPS}$, $D_{pg}^{sch} = 2T_{pg}^{EPS}$.

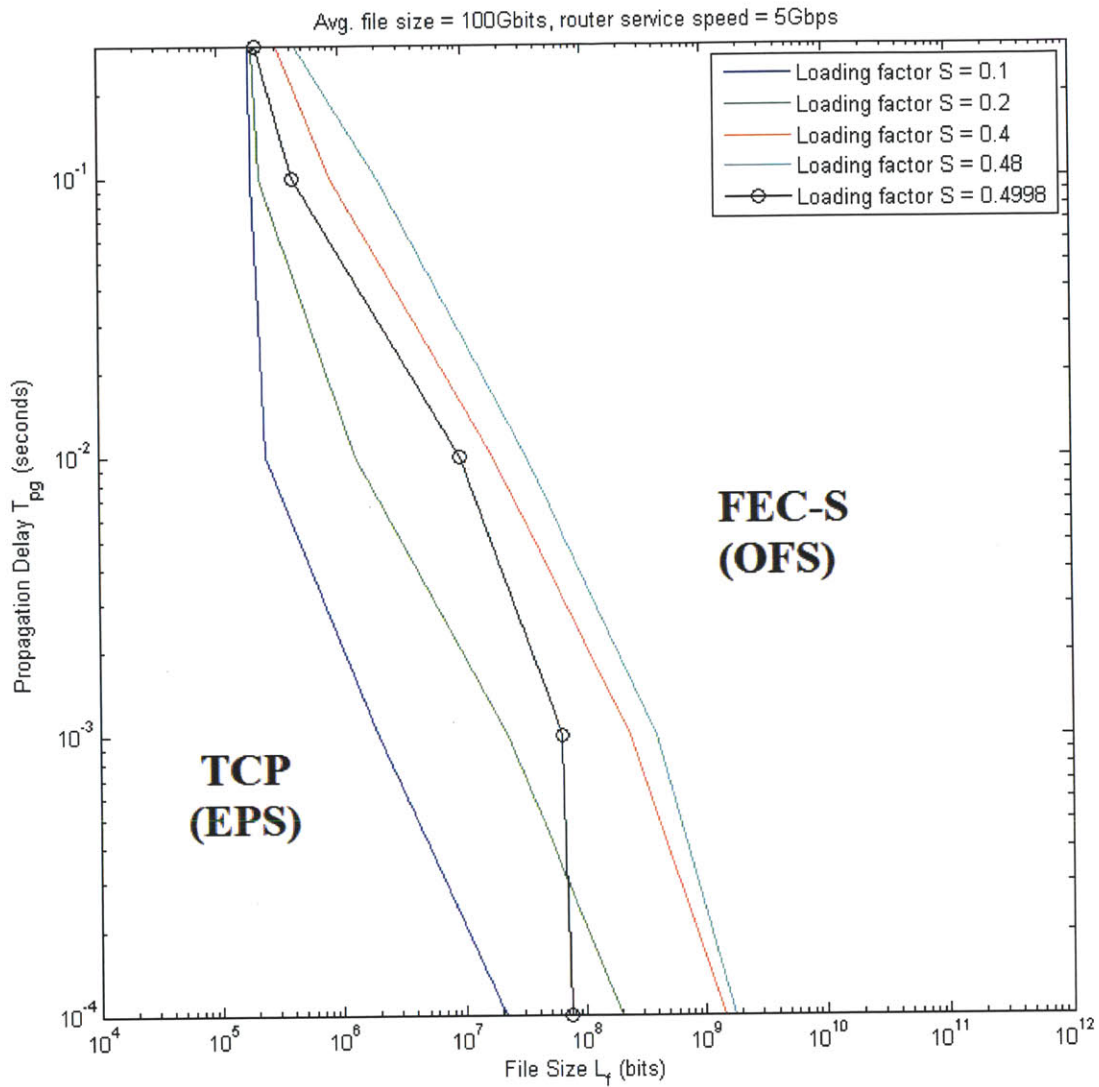


Figure 7.23 (b): Preference maps for EPS and OFS for different flow lengths L_f and propagation delays when $p_e = 10^{-8}$. The size of all other files is assumed to be 100Gbits except for the one being considered, which can be of any size. The line rates are assumed to be 10Gbps for both EPS and OFS, but the router service speed is limited to be 5Gbps. The four solid lines are the boundaries for TCP and FEC-S protocols with loading factors $S = 0.1, 0.2, 0.4$ and 0.48 respectively (which correspond to loading factors 0.2, 0.4, 0.8 and 0.96 for router service speed 2.5Gbps). For each loading factor, if the pair (L_f, T_{pg}) falls into the region to the right of the boundary line, then FEC-S (OFS) is preferred; otherwise TCP over EPS is preferred. $R_{OFS} = 10\text{Gbps}$, $D_{TCP} = 8T_{pg}^{EPS}$, $D_{pg}^{sch} = 2T_{pg}^{EPS}$.

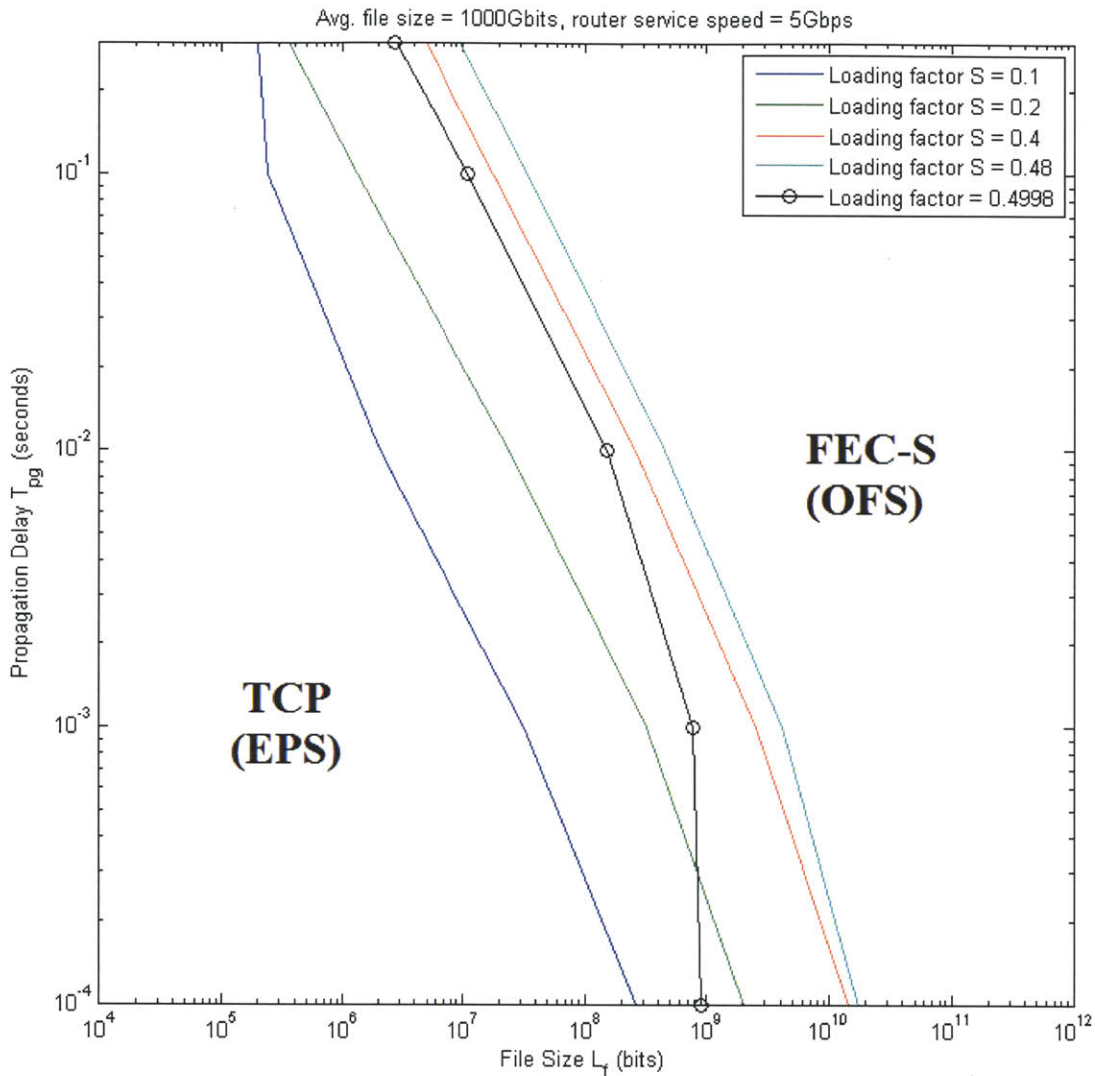


Figure 7.23 (c): Preference maps for EPS and OFS for different flow lengths L_f and propagation delays when $p_e = 10^{-8}$. The size of all other files is assumed to be **1000Gbits** except for the one being considered, which can be of any size. The line rates are assumed to be **10Gbps** for both EPS and OFS, but the router service speed is limited to be **5Gbps**. The four solid lines are the boundaries for TCP and FEC-S protocols with loading factors $S = 0.1, 0.2, 0.4$ and 0.48 respectively (which correspond to loading factors $0.2, 0.4, 0.8$ and 0.96 for router service speed 2.5Gbps). For each loading factor, if the pair (L_f, T_{pg}) falls into the region to the right of the boundary line, then FEC-S (OFS) is preferred; otherwise TCP over EPS is preferred. $R_{\text{OFS}} = 10\text{Gbps}$, $D_{\text{TCP}} = 8T_{\text{pg}}^{\text{EPS}}$, $D_{\text{pg}}^{\text{sch}} = 2T_{\text{pg}}^{\text{EPS}}$.

It can be seen from Fig. 7.21 (a)-(c) that for the same loading factor, a smaller propagation delay gives a larger cut-off file size. This is expected, as a smaller propagation delay gives a smaller RTT, resulting in a smaller delay for TCP which highly depends on RTT. For very large propagation delays, such as 100ms, the cut-off file size is on the order of $10^5 \sim 10^6$ bits. That is, for files with size greater than 10^6 bits, if the propagation delay is large (>100 ms), we prefer to use FEC-S (OFS). The reason for this result is that as propagation delay gets large, it takes longer time for TCP to ramp up speed and hence results in longer delay. On the other hand, FEC-S (OFS) is not affected much by this longer propagation delay. *This shows that OFS is preferred over EPS for large files and/or long propagation delays.*

When the propagation delay is kept constant, an increased loading factor from 0.05 to 0.24 results in an increased cut-off file size between TCP and FEC-S. This is expected as the effect of loading factor on queuing delay of TCP is not as large as that on queuing delay of FEC-S. When the loading factor is too close to 0.25 (e.g. 0.2499), the boundary line starts to shift to the left, implying a decreased cut-off file size since the EPS queuing delay is growing towards infinity. However, this is an artifact of our assumption that the buffer size is infinite. For finite buffers, packets will be dropped and the TCP window will decrease, which has the same effect of increasing delay but with different rate.

Similar conclusions can be drawn from Fig. 7.22 (a)-(c) when the router service speed is $R_{\text{router}} = 5\text{Gbps}$.

The preference maps with loading factor and file size as the axes are also shown in Fig. 7.23 (a)-(c) and Fig. 7.24 (a)-(c).

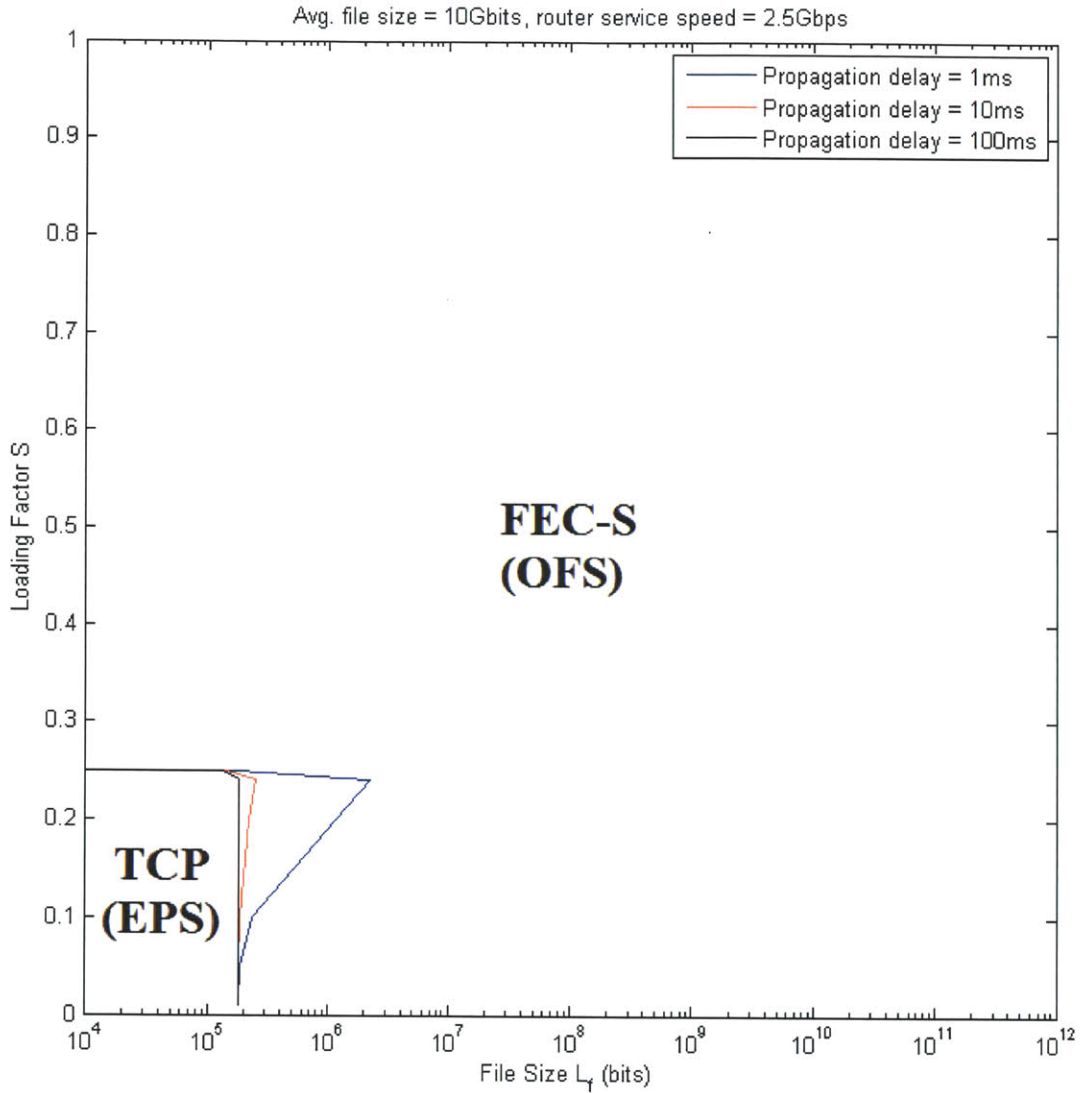


Figure 7.23 (a): Preference maps for EPS and OFS for different flow lengths L_f and loading factors S when $p_e = 10^{-8}$. The size of all other files is assumed to be 10Gbits except for the one being considered, which can be of any size. The line rates are assumed to be 10Gbps for both EPS and OFS, but the router service speed is limited to be 2.5Gbps. The three solid lines are the boundaries for TCP and FEC-S protocols with propagation delays 1ms, 10ms and 100ms, respectively. For each propagation delay, if the pair (L_f, S) falls into the region to the right of the boundary line, then FEC-S (OFS) is preferred; otherwise TCP over EPS is preferred. $R_{\text{OFS}} = 10\text{Gbps}$, $D_{\text{TCP}} = 8T_{\text{pg}}^{\text{EPS}}$, $D_{\text{pg}}^{\text{sch}} = 2T_{\text{pg}}^{\text{EPS}}$.

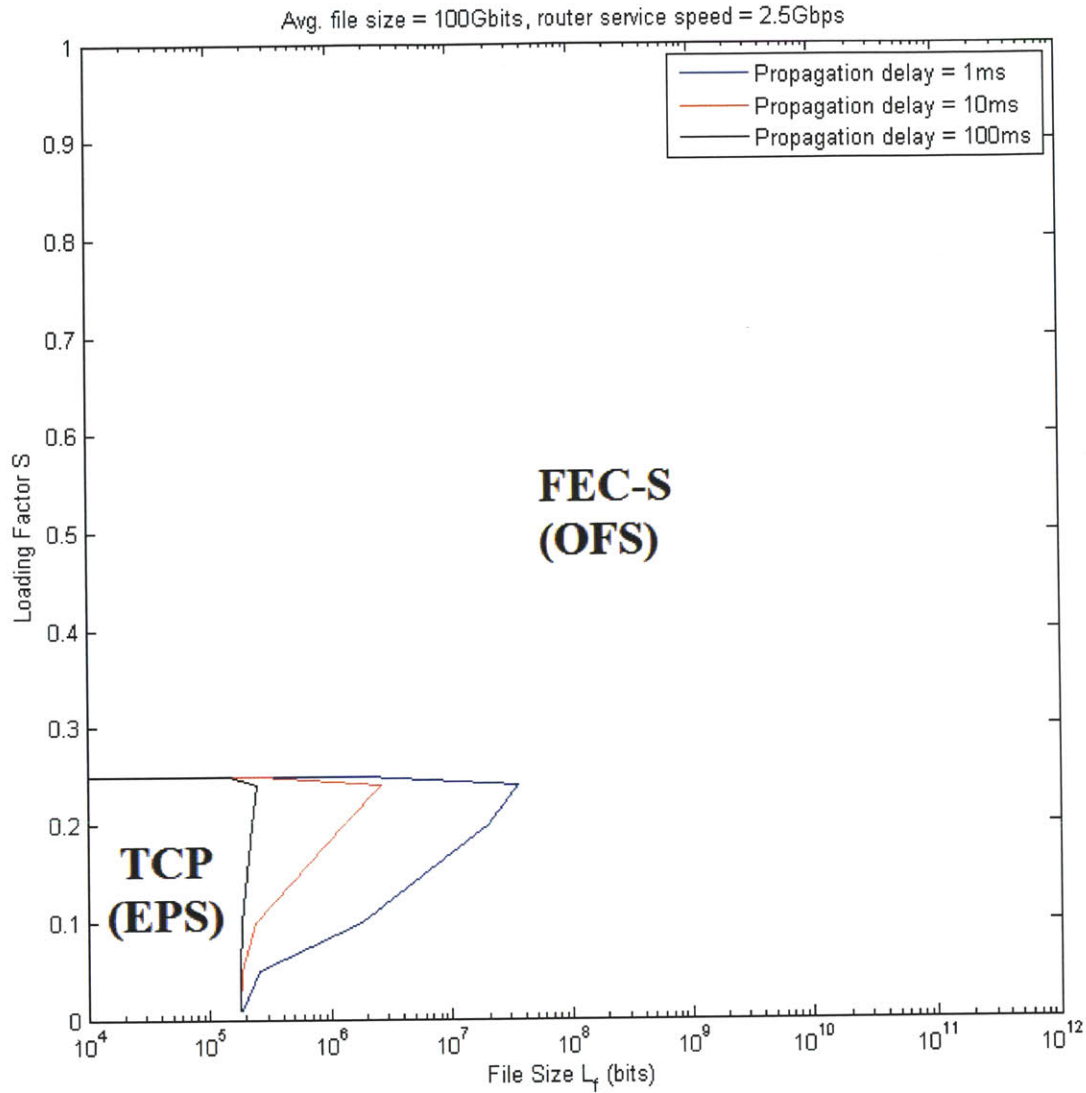


Figure 7.23 (b): Preference maps for EPS and OFS for different flow lengths L_f and loading factors S when $p_e = 10^{-8}$. The size of all other files is assumed to be **100Gbits** except for the one being considered, which can be of any size. The line rates are assumed to be 10Gbps for both EPS and OFS, but the router service speed is limited to be 2.5Gbps. The three solid lines are the boundaries for TCP and FEC-S protocols with propagation delays 1ms, 10ms and 100ms, respectively. For each propagation delay, if the pair (L_f, S) falls into the region to the right of the boundary line, then FEC-S (OFS) is preferred; otherwise TCP over EPS is preferred. $R_{\text{OFS}} = 10\text{Gbps}$, $D_{\text{TCP}} = 8T_{\text{pg}}^{\text{EPS}}$, $D_{\text{pg}}^{\text{sch}} = 2T_{\text{pg}}^{\text{EPS}}$.

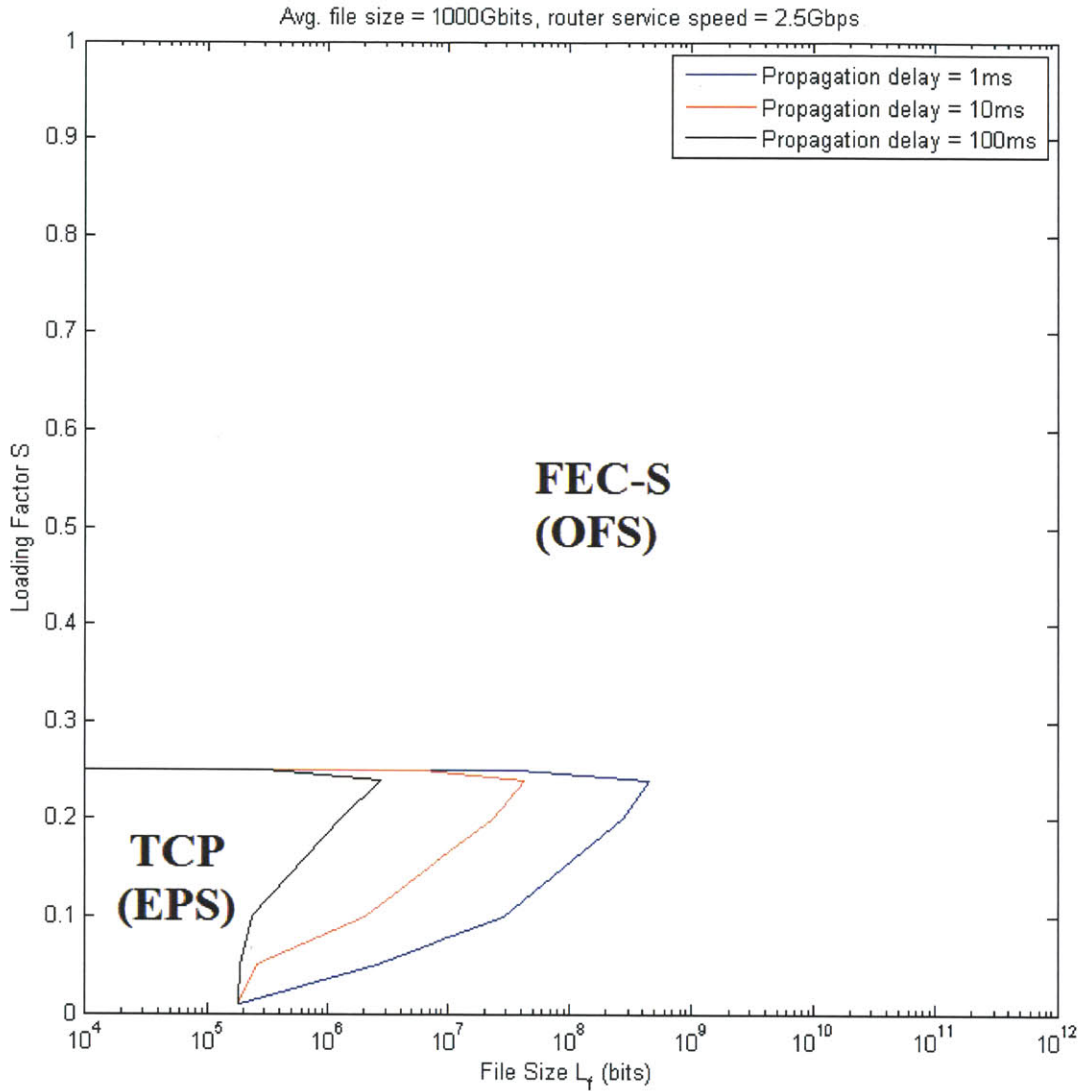


Figure 7.23 (c): Preference maps for EPS and OFS for different flow lengths L_f and loading factors S when $p_e = 10^{-8}$. The size of all other files is assumed to be 1000Gbits except for the one being considered, which can be of any size. The line rates are assumed to be 10Gbps for both EPS and OFS, but the router service speed is limited to be 2.5Gbps. The three solid lines are the boundaries for TCP and FEC-S protocols with propagation delays 1ms, 10ms and 100ms, respectively. For each propagation delay, if the pair (L_f, S) falls into the region to the right of the boundary line, then FEC-S (OFS) is preferred; otherwise TCP over EPS is preferred. $R_{\text{OFS}} = 10\text{Gbps}$, $D_{\text{TCP}} = 8T_{\text{pg}}^{\text{EPS}}$, $D_{\text{pg}}^{\text{sch}} = 2T_{\text{pg}}^{\text{EPS}}$.

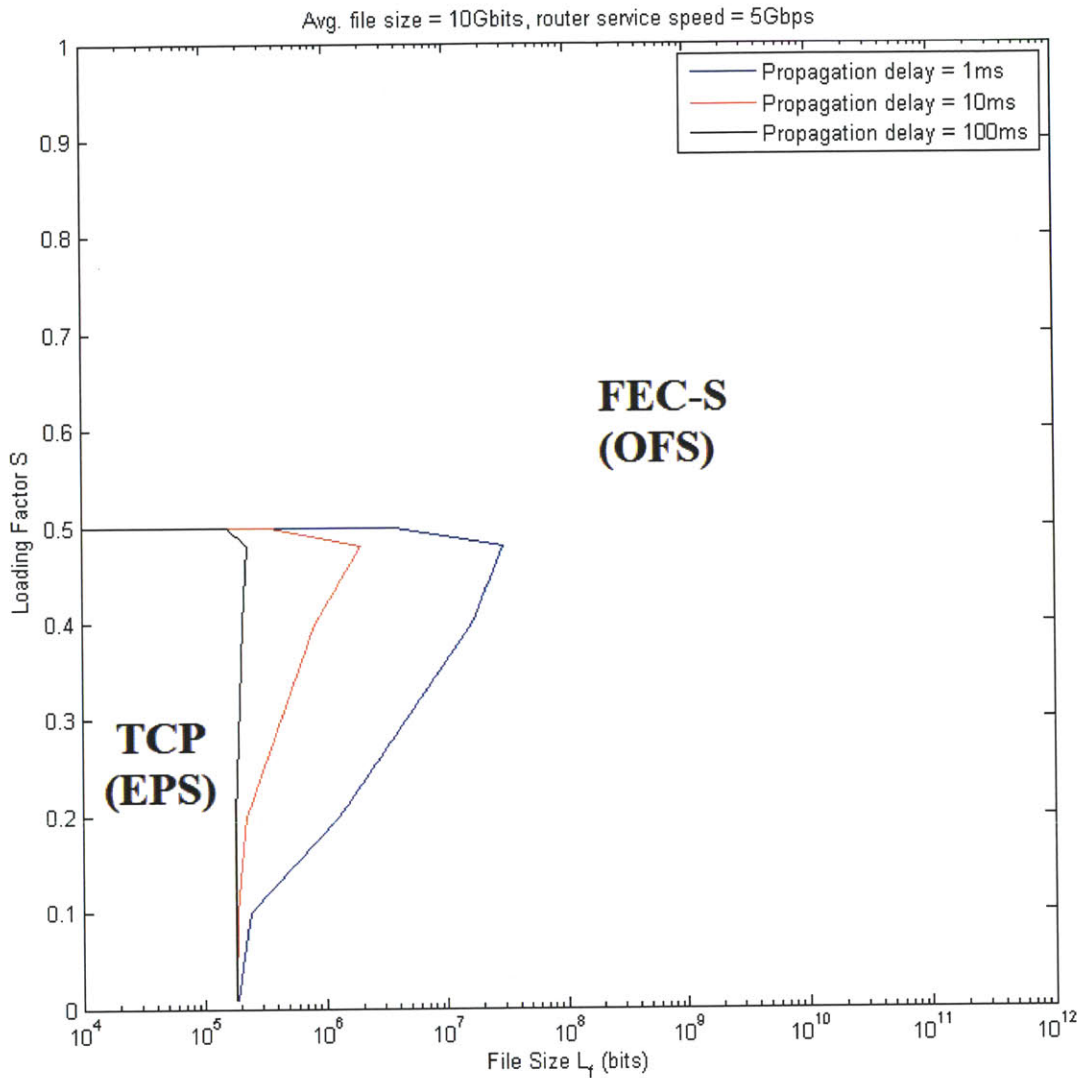


Figure 7.24 (a): Preference maps for EPS and OFS for different flow lengths L_f and loading factors S when $p_e = 10^{-8}$. The size of all other files is assumed to be 10Gbits except for the one being considered, which can be of any size. The line rates are assumed to be 10Gbps for both EPS and OFS, but the router service speed is limited to be 5Gbps. The three solid lines are the boundaries for TCP and FEC-S protocols with propagation delays 1ms, 10ms and 100ms, respectively. For each propagation delay, if the pair (L_f, S) falls into the region to the right of the boundary line, then FEC-S (OFS) is preferred; otherwise TCP over EPS is preferred. $R_{\text{OFS}} = 10\text{Gbps}$, $D_{\text{TCP}} = 8T_{\text{pg}}^{\text{EPS}}$, $D_{\text{pg}}^{\text{sch}} = 2T_{\text{pg}}^{\text{EPS}}$.

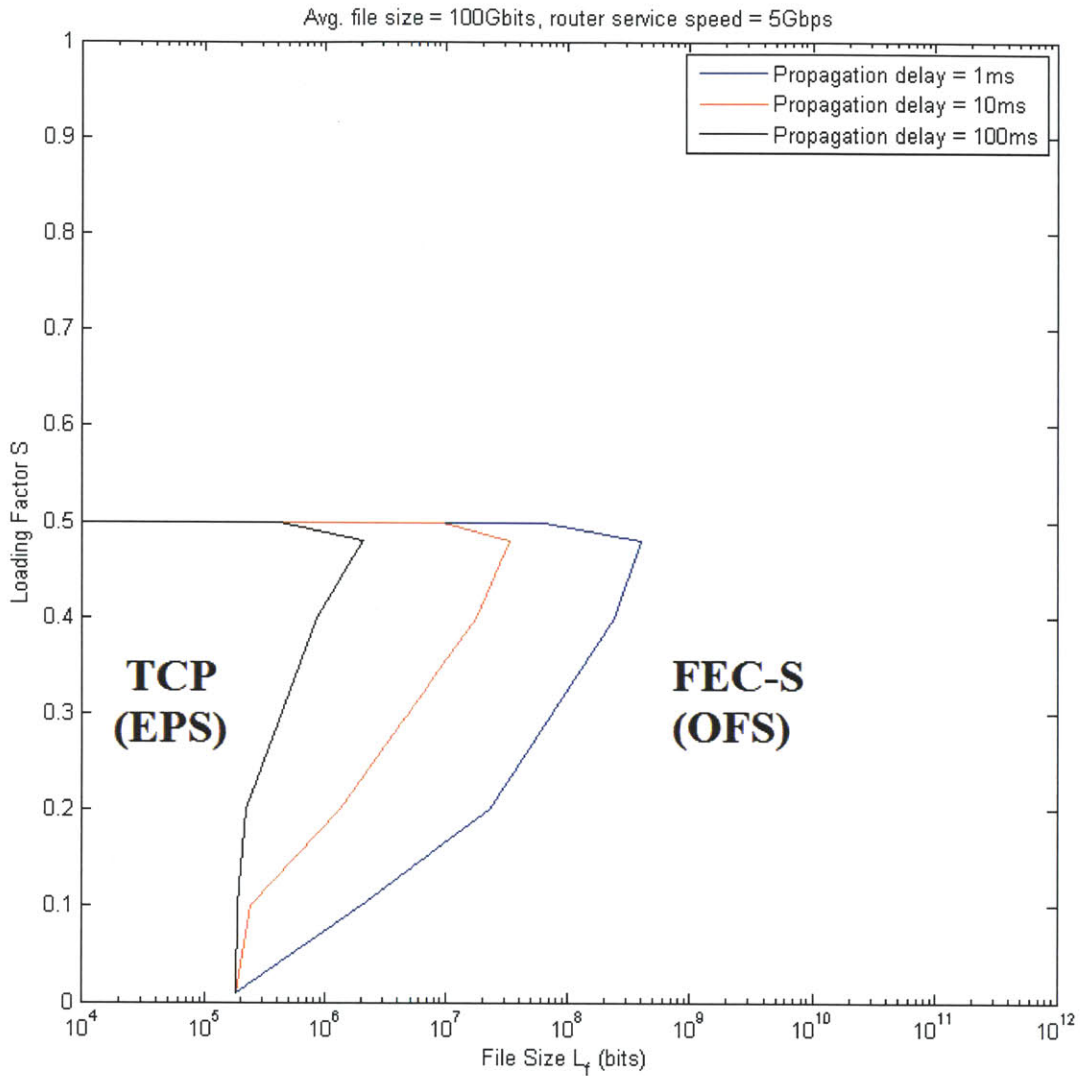


Figure 7.24 (b): Preference maps for EPS and OFS for different flow lengths L_f and loading factors S when $p_e = 10^{-8}$. The size of all other files is assumed to be 100Gbits except for the one being considered, which can be of any size. The line rates are assumed to be 10Gbps for both EPS and OFS, but the router service speed is limited to be 5Gbps. The three solid lines are the boundaries for TCP and FEC-S protocols with propagation delays 1ms, 10ms and 100ms, respectively. For each propagation delay, if the pair (L_f, S) falls into the region to the right of the boundary line, then FEC-S (OFS) is preferred; otherwise TCP over EPS is preferred. $R_{\text{OFS}} = 10\text{Gbps}$, $D_{\text{TCP}} = 8T_{\text{pg}}^{\text{EPS}}$, $D_{\text{pg}}^{\text{sch}} = 2T_{\text{pg}}^{\text{EPS}}$.

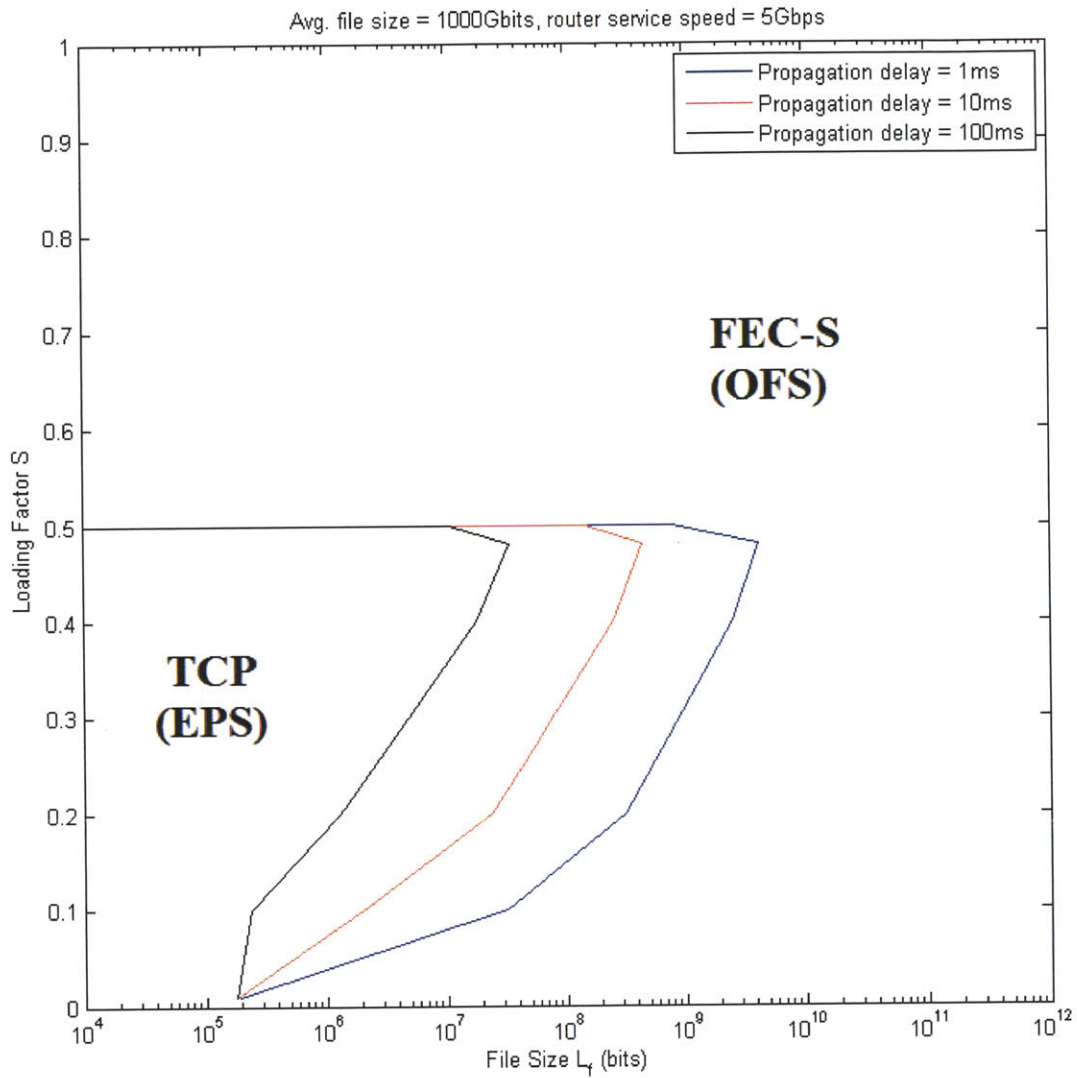


Figure 7.24 (c): Preference maps for EPS and OFS for different flow lengths L_f and loading factors S when $p_e = 10^{-8}$. The size of all other files is assumed to be 1000Gbits except for the one being considered, which can be of any size. The line rates are assumed to be 10Gbps for both EPS and OFS, but the router service speed is limited to be 5Gbps. The three solid lines are the boundaries for TCP and FEC-S protocols with propagation delays 1ms, 10ms and 100ms, respectively. For each propagation delay, if the pair (L_f, S) falls into the region to the right of the boundary line, then FEC-S (OFS) is preferred; otherwise TCP over EPS is preferred. $R_{\text{OFS}} = 10\text{Gbps}$, $D_{\text{TCP}} = 8T_{\text{pg}}^{\text{EPS}}$, $D_{\text{pg}}^{\text{sch}} = 2T_{\text{pg}}^{\text{EPS}}$.

From Fig. 7.23 (a)-(c) we can see that when we increase the loading factor in a range close to 0.25 (corresponding to an effective loading factor of close to 1), the cut-off file size increases, and starts to decrease again when the loading factor gets very close to 0.25. At loading factor of 0.25, basically FEC-S (OFS) is better than TCP, as in theory the router is fully loaded and can incur a very long queuing delay to switch the packet across the router. For any loading factor larger than 0.25, we always prefer to use OFS because EPS results in an infinite delay and OFS has a finite delay.

Also, similar conclusions can be drawn from Fig. 7.24 (a)-(c) when the router service speed is $R_{\text{router}} = 5\text{Gbps}$.

7.3 Summary of Chapter 7

In this chapter, we first compared the various OFS protocols proposed in Chapter 3-6, and found out that FEC-S (OFS) can give the best performance in terms of minimized delay if the file is transmitted via OFS. We then compared FEC-S (OFS) and TCP and drew the preference maps based on the following parameters: file size, BER, propagation delay and loading factor. Comparison results show that OFS is preferred than EPS when the files are large and/or when the propagation delay is large and/or when the loading factor is large. Preference maps in Fig. 7.21-7.24 can serve as guidance for protocol choice in practice given different file sizes, propagation delays and loading factors.

In all the cases we solved in this thesis we assumed that the routers have infinite memory and no congestion packet drop. In practice, routers do have finite memory and congestion packet drop, and preference maps in Fig. 7.23-7.24 will change. With finite buffers, TCP will experience window closing more often and hence longer delay compared to the case with infinite buffers, especially when the loading factor is large. Therefore, TCP over EPS with congestion packet loss will have worse performance in terms of delay than the EPS

performance considered in this thesis. The boundary lines in Fig. 7.23-7.24 will shift to the left, and FEC-S (OFS) will then have a larger preference region than before. Fig. 7.25 below shows an illustration of what would happen for finite buffers.

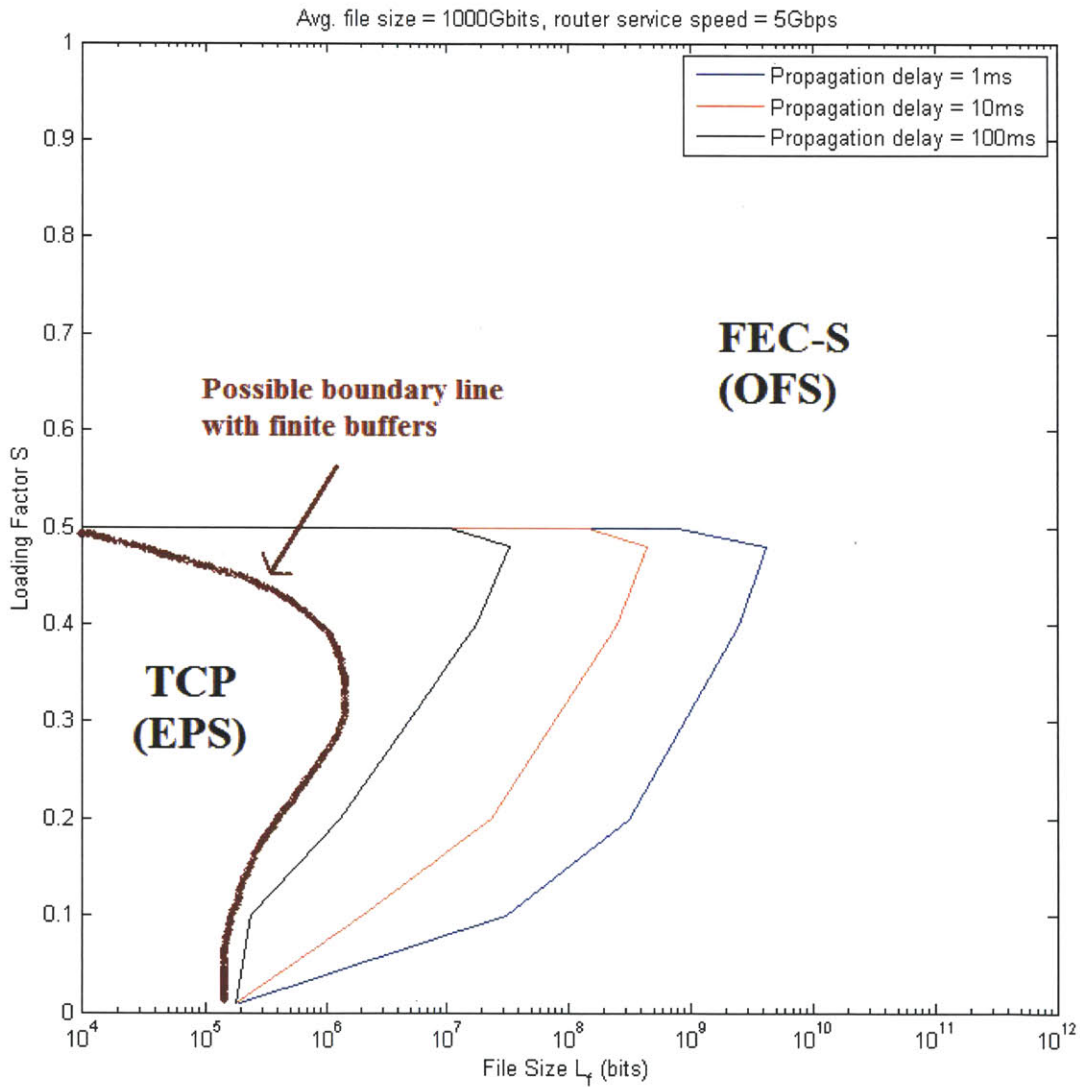


Figure 7.25: Preference map as in Fig. 7.24(c) except for the addition of a possible boundary line when the buffers have finite memory. The regions where TCP (EPS) is optimum become smaller than the case with infinite buffers.

As for the adaptive choice between OFS and EPS for retransmissions, depending on the various parameters at the time of retransmissions, we may choose whether to use OFS or EPS based on preference maps in Section 7.2.

Chapter 8

Discussion of Results in the Larger Context of the Transport Layer Problems

With the continuous rapid increase of bandwidth demand, OFS has been proposed for future optical networks to serve large transactions in a cost-effective manner, by means of an all-optical data plane employing end-to-end lightpaths. Transport Layer protocols of this new architecture need to be developed to realize the three functions comparable to that of TCP over EPS: congestion control, rate matching between the sender and receiver, and ensuring end-to-end data transfer reliability. OFS congestion control is taken care of by wavelength channel reservation and flow scheduling. Rate matching between the end users can be done with an agreed constant rate between the end users over the entire flow duration (this is possible because in OFS a wavelength channel, once reserved, is dedicated to a particular

flow). Discussion on the issue of end-to-end data transfer reliability is a focus of this work. Nevertheless, we put several constraints on our analysis and the results are useful in that they can serve as guidance for the design of a new Transport Layer to a certain extent.

Optical signals transmitted over the optical fiber suffer from attenuation, have noise added to them from optical amplifiers (such as EDFA), and sustain a variety of other impairments, such as dispersion and nonlinearity (e.g. crosstalk). At the receiver, the transmitted optical data must be converted back to electrical signals, and recovered with an acceptable bit error rate. In general optical channel effects contain two types of errors: independent and identically distributed (IID) errors and burst errors. IID errors can be caused by the thermal noise, shot noise, and amplified spontaneous emission noise [22]. Burst errors can be caused by possible power undershoot or overshoot during EDFA transients when wavelength channels are added or dropped while others are being used [22]. In this work, we only considered the case of IID errors due to time limitations. We can interleave the data before transmissions so that the burst errors may appear to be independent at the receiver, though this may not be the optimum strategy. This converts the case of burst errors to the case of IID errors. Future work should be done to carefully analyze the case of burst errors, including causes of burst errors, frequency of burst occurrences and duration of the bursts.

For baseline IP over EPS, packet loss can be caused by packet errors or congestion at the routers or both. To simplify our discussions, in our work we assumed infinite buffers at the routers so that no packet is dropped due to router congestion, and packet loss only happens when there are errors in the packets. If the packet drop effect due to congestion was included, TCP window closing would occur during congestion, leading to a smaller average window size and a longer expected delay. This corresponds to a longer delay for the same file size than that in the case of infinite buffers. As a result, the boundary lines in the preference maps would probably shift to the left, meaning that OFS is preferred than TCP over EPS even for smaller files.

Furthermore, it should be noticed that we assumed "deterministic" flow sizes for the case of OFS. That is, for all other flows other than the one being considered, we assume they are of the same size. This assumption allows for great simplification to estimate the queuing delay, but still provides valuable insight into the problem. Nevertheless, the queuing delay would depend on the flow size distributions which can vary depending on the scenarios. For more detailed and accurate analysis of the queuing delay, we should take into consideration random flow size distributions in future work.

Another issue we did not have time to address is the efficiency of channel usage. For EPS, we do not tend to make the fiber links fully loaded or nearly fully loaded because of the use of suboptimum switching algorithm used at the routers and the congestion and infinite delay that may occur at high loads – the router may only be 20% loaded for most applications that have time deadlines (e.g. Voice over IP, stock exchange information, etc). For OFS networks, a wavelength channel is dedicated to at most one session at any given time and can be nearly fully loaded (e.g. 80% with still acceptable queuing delay). Such loading effect was studied in Chapter 7 in some level of details but not fully examined.

In the process of segmentation, there are usually some more processing efforts needed at the end users to segment the original flow into smaller blocks, to encode each block at the sender and to decode them at the receiver. In general, the more blocks the flow is divided into, the more processing efforts are required. For proper discussions of such processing cost in case of segmentation, modeling of the processing at the sender, and at the receiver should be done in the future.

Based on all the discussions above, it should be noted that this work provides some insight into the problem of Transport Layer protocols under some assumptions that may be simplified from real-world situations. Nevertheless, the main purpose of this thesis is to demonstrate and discuss possible OFS protocols and compare OFS with EPS. The preference maps based on the bit error rate, file size, propagation delay, and loading factor show the

corresponding regions where FEC-S (OFS) is preferred over TCP over EPS under those assumptions. Under different assumptions, the preference maps may be different, but results of the thesis still provide valuable insight and references to the protocol design. When the file size is small (e.g. several mega bytes), the total transmission time can be very small even with EPS and the difference between EPS and OFS is not significant. However, when the file size gets large, the advantages of OFS become more and more obvious, as the preference maps show. This suggests that OFS can be a good candidate for future networks to serve large transactions in a delay-efficient manner.

The inferior delay of the EPS transfer mode here is mostly due to the TCP windowing function. If a new Transport Layer protocol can be created that takes care of congestion control without windowing, EPS will have much improved delay and can be as good as OFS. In fact the protocols described are applicable to circuit services with guaranteed rates even if the circuits are a mix of optics and electronics and not OFS.

Chapter 9

Conclusion

9.1 Summary of Contributions

In Chapter 1, we presented the necessity of deploying OFS for future communications with the exponentially growing bandwidth demands and stated the problem of end-to-end data transfer reliability for OFS.

We then modeled TCP using Markov chains in Chapter 2 and discussed the delays of linear, exponential and standard TCP over EPS with zero and non-zero BERs. When the BER and congestion level are zero, the TCP window size can increase up to the maximum value (e.g. 128 for the standard TCP) after several round trips (e.g. 71 for standard TCP), and the total delay is approximately proportional to the RTT, which includes the processing delay,

transmission delay, propagation delay and queuing delay (we discussed these delays separately in Chapter 7 when comparing EPS with OFS). When the BER is not zero, the average window size decreases with increasing BER, and it takes more round trip times to transmit the file and hence incurs a longer delay. TCP does not work well for large transactions due to its window build up delays whether the transport is OFS or reserved mixed optical and electronic circuits.

From Chapter 3 to Chapter 6, we discussed four types of OFS protocols that may be used to ensure end-to-end data transfer reliability: error detection by backward comparison, error detection code and no segmentation, error detection code with segmentation and forward error correction with segmentation. We started from the simplest protocol EDBC that determines whether the original data flow has errors in it by sending back the flow to the sender via the backward path and comparing with the original file. Although simple, this protocol incurs extra queuing delay and requires two paths (one forward and one backward path) that are very precious resources to complete the tasks. We then discussed the protocol with an error detection code and no segmentation (i.e. EDC-NS) in Chapter 4. EDC-NS can indeed give smaller delays than EDBC can, and requires shorter scheduling and queuing delays than EDBC. However, when the file size becomes so large that the product of BER and flow size becomes close to 1, the flows may need to be retransmitted for several times, incurring longer delays. Therefore, it may be necessary to segment the flow into smaller blocks before transmission and only retransmit those blocks that are erroneous. We then discussed in Chapter 5 and 6 two ways of doing so: EDC-S without and FEC-S with FEC. EDC-S protocol is a natural extension of the EDC-NS protocol with the flexibility of segmenting the original flow into smaller blocks. Instead of retransmitting the whole flow that is erroneous in EDC-NS, we only need to retransmit those erroneous blocks in EDC-S. This will save quite an amount of transmission time, and also reduce the number of retransmissions due to a smaller probability of error for small blocks. The use of error detection code without additional forward error correction code does not work too well

when the BER presented by the DLC is high. We may need to use FEC to reduce the BER for each block. In Chapter 6, we discussed FEC-S (OFS) protocol as a promising protocol to reduce the probability of errors and hence retransmissions and total delay. The error reduction is at the cost of adding redundancy and extra decoding delay. With proper choice of block size given a certain flow size, the delay can be minimized. The minimum delay is found when the block size is between 10^4 bits and $\frac{L_f}{100}$ bits when $L_f \geq 10^6$ bits, almost independent of the decoding delay coefficient.

We compared the various OFS protocols proposed in Chapter 3-6, and found out that FEC-S (OFS) can give the best performance in terms of minimized delay if the file is transmitted via OFS. We then compared FEC-S (OFS) and TCP and drew the preference maps based on the following parameters: file size, BER, propagation delay and loading factor. Comparison results show that OFS is preferred than EPS when the files are large and/or when the propagation delay is large and/or when the loading factor is large.

9.2 Future Work and Challenges

In Chapter 8, we discussed the results in the larger context of the Transport Layer problems, which include some future research directions. Due to time limitations, we only focused our attention on the delay metric when comparing performances of different protocols. Other metrics such as network resource usage, processing cost and data efficiency may also be used to compare the performances and can possibly lead to different preference maps.

Appendix A

Derivation of Results in Chapter 3

A.1 Derivation of Expected Total Number of Transmissions $E[N_t]$

Let N_t be a random variable that denotes the total number of transmissions (including retransmissions) required to make all blocks error-free. Let the segmented block length be L_b bits, and the number of blocks for a flow of length L_f bits be $n_b = \left\lceil \frac{L_f}{L_b} \right\rceil \approx \frac{L_f}{L_b}$, when $L_f \gg L_b$.

With IID bit error rate p_e , the block error rate $p_{e,b}$ can be expressed by

$$p_{e,b} = 1 - (1 - p_e)^{L_b}$$

Let E be a random variable that denotes the number of erroneous blocks. $P(E = k|n)$ is the probability of having k blocks in error given a total number of n blocks. Then

$$P(E = k|n) = P(k \text{ out of } n \text{ blocks are in error}) = \binom{n}{k} p_{e,b}^k (1 - p_{e,b})^{n-k}$$

With a total number of n_b blocks, the probability of transmitting only once (i.e. without retransmissions) is

$$P(N_t = 1|n_b) = P(K = 0|n_b) = (1 - p_{e,b})^{n_b}$$

With a total number of n_b blocks, the probability of transmitting exactly twice is

$$\begin{aligned} &P(N_t = 2|n_b) \\ &= \sum_{i=1}^{n_b} P(E = i|n_b)P(E = 0|i) \\ &= \sum_{i=1}^{n_b} \binom{n_b}{i} p_{e,b}^i (1 - p_{e,b})^{n_b-i} (1 - p_{e,b})^i \\ &= \sum_{i=1}^{n_b} \binom{n_b}{i} p_{e,b}^i (1 - p_{e,b})^{n_b} \\ &= (1 - p_{e,b})^{n_b} \left(\sum_{i=0}^{n_b} \binom{n_b}{i} p_{e,b}^i - 1 \right) \\ &= (1 - p_{e,b})^{n_b} [(1 + p_{e,b})^{n_b} - 1] \end{aligned}$$

The last equation follows because

$$(1 + p_{e,b})^{n_b} = \sum_{i=0}^{n_b} \binom{n_b}{i} p_{e,b}^i$$

Similarly, we can show that the probability of transmitting exactly three times is

$$\begin{aligned}
& P(N_t = 3|n_b) \\
&= \sum_{i=1}^{n_b} P(E = i|n_b) P(N_t = 2|i) \\
&= \sum_{i=1}^{n_b} \binom{n_b}{i} p_{e,b}^i (1 - p_{e,b})^{n_b-i} (1 - p_{e,b})^i [(1 + p_{e,b})^i - 1] \\
&= \sum_{i=1}^{n_b} \binom{n_b}{i} p_{e,b}^i (1 - p_{e,b})^{n_b} [(1 + p_{e,b})^i - 1] \\
&= (1 - p_{e,b})^{n_b} \sum_{i=1}^{n_b} \binom{n_b}{i} p_{e,b}^i [(1 + p_{e,b})^i - 1] \\
&= (1 - p_{e,b})^{n_b} \left[\sum_{i=1}^{n_b} \binom{n_b}{i} p_{e,b}^i (1 + p_{e,b})^i - \sum_{i=1}^{n_b} \binom{n_b}{i} p_{e,b}^i \right] \\
&= (1 - p_{e,b})^{n_b} \left[\left(\sum_{i=0}^{n_b} \binom{n_b}{i} p_{e,b}^i (1 + p_{e,b})^i - 1 \right) - \left(\sum_{i=0}^{n_b} \binom{n_b}{i} p_{e,b}^i - 1 \right) \right] \\
&= (1 - p_{e,b})^{n_b} \left[(1 + p_{e,b}(1 + p_{e,b}))^{n_b} - (1 + p_{e,b})^{n_b} \right] \\
&= (1 - p_{e,b})^{n_b} [(1 + p_{e,b} + p_{e,b}^2)^{n_b} - (1 + p_{e,b})^{n_b}]
\end{aligned}$$

We next use induction to show that for any $k \geq 2$

$$\begin{aligned}
& P(N_t = k|n_b) \\
&= (1 - p_{e,b})^{n_b} [(1 + p_{e,b} + p_{e,b}^2 + \cdots + p_{e,b}^{k-1})^{n_b} - (1 + p_{e,b} + p_{e,b}^2 + \cdots + p_{e,b}^{k-2})^{n_b}] \\
&= (1 - p_{e,b})^{n_b} \left[\left(\sum_{i=0}^{k-1} p_{e,b}^i \right)^{n_b} - \left(\sum_{i=0}^{k-2} p_{e,b}^i \right)^{n_b} \right]
\end{aligned}$$

The above expression holds for $k = 2$ and 3 . We assume for $k \geq 3$

$$P(N_t = k|n_b) = (1 - p_{e,b})^{n_b} \left[\left(\sum_{i=0}^{k-1} p_{e,b}^i \right)^{n_b} - \left(\sum_{i=0}^{k-2} p_{e,b}^i \right)^{n_b} \right]$$

Then we only need to show that for any $k \geq 3$, the following holds

$$P(N_t = k + 1|n_b) = (1 - p_{e,b})^{n_b} \left[\left(\sum_{i=0}^k p_{e,b}^i \right)^{n_b} - \left(\sum_{i=0}^{k-1} p_{e,b}^i \right)^{n_b} \right]$$

In fact, we have

$$\begin{aligned} & P(N_t = k + 1|n_b) \\ &= \sum_{i=1}^{n_b} P(E = i|n_b)P(N_t = k|i) \\ &= \sum_{i=1}^{n_b} \binom{n_b}{i} p_{e,b}^i (1 - p_{e,b})^{n_b-i} (1 - p_{e,b})^i \left[\left(\sum_{j=0}^{k-1} p_{e,b}^j \right)^i - \left(\sum_{j=0}^{k-2} p_{e,b}^j \right)^i \right] \\ &= (1 - p_{e,b})^{n_b} \sum_{i=1}^{n_b} \binom{n_b}{i} p_{e,b}^i \left[\left(\sum_{j=0}^{k-1} p_{e,b}^j \right)^i - \left(\sum_{j=0}^{k-2} p_{e,b}^j \right)^i \right] \\ &= (1 - p_{e,b})^{n_b} \sum_{i=1}^{n_b} \binom{n_b}{i} \left[\left(p_{e,b} \sum_{j=0}^{k-1} p_{e,b}^j \right)^i - \left(p_{e,b} \sum_{j=0}^{k-2} p_{e,b}^j \right)^i \right] \\ &= (1 - p_{e,b})^{n_b} \sum_{i=1}^{n_b} \binom{n_b}{i} \left[\left(\sum_{j=1}^k p_{e,b}^j \right)^i - \left(\sum_{j=1}^{k-1} p_{e,b}^j \right)^i \right] \\ &= (1 - p_{e,b})^{n_b} \left[\left(1 + \sum_{j=1}^k p_{e,b}^j \right)^{n_b} - \left(1 + \sum_{j=1}^{k-1} p_{e,b}^j \right)^{n_b} \right] \end{aligned}$$

$$= (1 - p_{e,b})^{n_b} \left[\left(\sum_{j=0}^k p_{e,b}^j \right)^{n_b} - \left(\sum_{j=0}^{k-1} p_{e,b}^j \right)^{n_b} \right]$$

Proved.

Bibliography

- [1] G. E. Weichenberg, "Design and Analysis of Optical Flow Switched Networks," Ph.D. Dissertation, Massachusetts Institute of Technology, 2009.

- [2] M. E. Crovella and A. Bestavros, "Self-similarity in world wide web traffic: evidence and possible causes," *IEEE/ACM Trans. Netw.*, vol. 5, no. 6, pp. 835–846, 1997.

- [3] M. S. Taqqu, W. Willinger, and R. Sherman, "Proof of a fundamental result in self-similar traffic modeling," *Comp. Commun. Rev.*, vol. 27, pp. 5–23, 1997.

- [4] W. Willinger, M. S. Taqqu, R. Sherman, and D. V. Wilson, "Self similarity through high variability: statistical analysis of Ethernet LAN traffic at the source level," *IEEE/ACM Trans. Netw.*, vol. 5, no. 1, pp. 71–86, 1997.

- [5] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, "On the self-similar nature of Ethernet traffic (extended version)," *IEEE/ACM Trans. Netw.*, vol. 2, no. 1, pp. 1–15, 1994.

- [6] V. Paxson and S. Floyd, "Wide-area traffic: the failure of Poisson modeling," *IEEE/ACM Trans. Netw.*, vol. 3, no. 3, pp. 226–244, 1995.
- [7] V. W. S. Chan, G. Weichenberg, and M. Médard, "Optical flow switching," in *3rd Int. Conf. on Broadband Communications, Networks and Systems, 2006. BROADNETS 2006*, San Jose, CA, Oct. 1–5, 2006, pp. 1–8.
- [8] G. Weichenberg, V. W. S. Chan, and M. Medard, "Design and analysis of optical flow-switched networks," *Optical Communications and Networking, IEEE/OSA Journal of*, vol. 1, no. 3, pp. B81–B97, August 2009.
- [9] G. Weichenberg, V. W. S. Chan, E.A. Swanson, and M. Medard, "Throughput-cost analysis of optical flow switching," in *Optical Fiber Communication - includes post deadline papers, 2009. OFC 2009. Conference on*, March 2009, pp. 1–3.
- [10] G. Weichenberg, V. W. S. Chan, and M. Medard, "Performance analysis of optical flow switching," in *Communications, 2009. ICC '09. IEEE International Conference on*, June 2009, pp.1–6.
- [11] G. Weichenberg, V. W. S. Chan, and M. Medard, "On the capacity of optical networks: A framework for comparing different transport architectures," in *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, April 2006, pp. 1–13.
- [12] G. Weichenberg, V. W. S. Chan, and M. Medard, "Cost-efficient optical network architectures," in *Optical Communications, 2006. ECOC 2006. European Conference on*, Sept. 2006, pp. 1–2.

- [13] G. Weichenberg, V. W. S. Chan, and M. Medard, "Opn08-06: On the throughput-cost tradeoff of multi-tiered optical network architectures," in Global Telecommunications Conference, 2006. GLOBECOM '06. IEEE, 27 2006-Dec. 1 2006, pp. 1–6.
- [14] G. Weichenberg, V.W.S. Chan, and M. Medard, "Access network design for optical flow switching," in Global Telecommunications Conference, 2007. GLOBECOM '07. IEEE, Nov. 2007, pp. 2390–2395.
- [15] G. Weichenberg, V. W. S. Chan, and Muriel Medard, "Design and analysis of optical flow-switched networks", J Opt. Commun. Netw. Vol. 1, no.3, pp. B81-97, Aug. 2009.
- [16] V. W. S. Chan, "Guest editorial optical communications and networking series," Selected Areas in Communications, IEEE Journal on, vol. 23, no. 8, pp. 1441–1443, Aug. 2005.
- [17] V. W. S. Chan, "Optical network architecture from the point of view of the end user and cost," Selected Areas in Communications, IEEE Journal on, vol. 24, no. 12, pp. 1–2, Dec. 2006.
- [18] J. Zhang, "Soft Handoff in MC-CDMA Cellular Networks Supporting Multimedia Services," Ph.D. Dissertation, University of Waterloo, 2004.
- [19] V. W. S. Chan, "Optical flow switching: A new green transport mechanism for fiber networks," in Transparent Optical Networks, 2009. ICTON '09. 11th International Conference on, 28 2009-July 2 2009, pp. 1–1.18
- [20] B. Ganguly, "Implementation and modeling of a scheduled Optical Flow Switching (OFS) network", Ph.D. Dissertation, Massachusetts Institute of Technology, 2008.

- [21] V. W. S. Chan, lecture notes, 6.267 Heterogeneous Networks, MIT, Dec. 2010.
- [22] R. Ramaswami, K. N. Sivarajan, and G. H. Sasaki. *Optical Networks: A Practical Perspective (Third Edition)*. USA: Elsevier Inc., 2010.
- [23] D. Bertsekas, and R. G. Gallager. *Data Networks (Second Edition)*. USA: Prentice-Hall, Inc., 1992.
- [24] R. G. Gallager, Low Density Parity Check Codes, Monograph, M.I.T. Press, 1963.
- [25] R. G., Gallager, Information Theory and Reliable Communication, John Wiley and Sons, New York, 1968.
- [26] T. V. Ramabadran, and S. S. Gaitonde, A tutorial on CRC computations [Online]. Available: <http://www.ee.bilkent.edu.tr/~ee538/crc.pdf>
- [27] E. J. Lee, "Free-Space Optical Networks - Fade and Interference Mitigation and Network Congestion Control," Ph.D. Dissertation, Massachusetts Institute of Technology, 2010.
- [28] --, *Cyclic Redundancy Checks* [Online]. Available: <http://www.mathpages.com/home/kmath458.htm>