

TABLE OF CONTENTS

DEDICATION	iii
ACKNOWLEDGEMENTS	iv
LIST OF TABLES	vii
LIST OF FIGURES	viii
ABSTRACT	xi
I INTRODUCTION	1
1.1 Statement of the problem	2
1.2 Problems with existing algorithms	2
1.3 Response time components	3
1.4 Our approach	5
1.5 Organization of the thesis	6
II LITERATURE REVIEW	7
2.1 Web server request scheduling	7
2.1.1 Web sever scheduling implementation:	7
2.1.2 Web sever scheduling simulation studies:	8
2.1.3 Web sever scheduling theoretical studies:	9
2.2 Admission control and service differentiation	10
III IMPLEMENTATION OF SRRT	12
3.1 Differentiated services and traffic control in Linux	12
3.1.1 Default Linux qdisc	13
3.1.2 prio qdisc to implement SRRT	15
3.2 Modifications to Apache web server to implement SRRT	19
3.3 Using TCP protocol information to implement SRRT	21
3.3.1 Data transmission over TCP	21
3.3.2 TCP congestion control	22
3.3.3 TCP time-out and round trip time	24
3.3.4 Approximating the remaining response time for the TCP connection	25
3.3.5 The final SRRT algorithm	26

IV EXPERIMENT SETUP	28
4.1 Benchmarking web server	28
4.2 The bottleneck	28
4.2.1 Network connection	29
4.2.2 Client machines	30
4.2.3 The server	30
4.3 Experiment setup	30
4.4 Workload traffic generators	31
4.4.1 Performance metrics	31
4.4.2 Workload generators	32
4.4.3 SURGE: Workload generator for web traffic	32
4.5 Network emulation	35
4.6 Tuning parameters	36
4.6.1 Tuning Apache web server	36
4.6.2 Tuning the operating system at the server	37
V RESULTS AND ANALYSIS	39
5.1 Experiment run	39
5.2 Important comments	41
5.3 Main results	42
5.3.1 Results for 10Mbps link capacity	44
5.3.2 Results for 100Mbps link capacity	49
5.3.3 Starvation analysis	50
5.3.4 Sources of performance of SRRT over SRPT	51
VI CONCLUSION AND FUTURE WORK	53
REFERENCES	55
APPENDIX A — SHORTEST TOTAL ESTIMATED RESPONSE TIME (STERT)	60
APPENDIX B — APACHE SOURCE CODE MODIFICATIONS	62
B.2.2 Changes to http_core.c file	64