



The University of Sydney

**School of Electrical and Information Engineering**

**Master of Engineering (Research)**

**Fair Service for High-Concurrent Requests**

**Zhanwen Li**

## Acknowledgement

First and foremost, I am indebted to David Levy and Shiping Chen, my supervisors, for giving me the chance to take on this research project, and their unwavering support and guidance. I thank them so much for their extensive experience, and incredibly broad vision to this thesis work. They are courageous at working on very hard problems, and they also have amazing ways of cutting to the core of complex subjects and focusing on the important details.

I owe a great deal of thanks to Alex Ng from the Macquarie University, who offered critical analysis for my research. I am grateful to have his comments on my project.

I must also thank the anonymous referees from the Conferences of Middleware 2006, MoDDM 2006 and ASWEC2007, who thoroughly reviewed my papers and gave very insightful feedback.

Other thanks are due to many other people at the University of Sydney. In particular, David Peterson inspired me to undertake this project at the beginning stage. David Wang, Justin Tang, Guillaume, Donghe Nam, Jeff Zhou and others have been involved in many aspects of this research, providing countless hours of discussion and debate during this study.

Despite my efforts in producing this work, this thesis would be seriously challenged grammatically if it was not for the contribution of Jeremy Wu, Joshua Ho and Liheng Li, who kindly edited and proofread this entire document. Thank you so much.

The biggest thank you must go to my family for supporting me to pursue my wild dream. I have enjoyed this year so much. Thank you.

Zhanwen Li

# **Abstract**

Fair Service for High-Concurrent Requests

By

Zhanwen Li

Master of Engineering (Research)

University of Sydney

Associate Professor David Levy

This thesis presents a new approach to ensuring fair service for highly concurrent requests. Our design uses the advantages of staged event-driven architecture (SEDA) to support high-concurrent loadings and makes use of control theory to manage the system performance.

In order to guarantee the quality of service is fairly made to each request, based on SEDA, the control system for fairness is developed as a combination of a global control framework and a set of local self-tune stags. The global control framework is used to control the performance of the whole staged network at the top-level, aimed at coordinating the performance of the stages in the network. On the other hand, each self-tune stage under the control framework is built on the thread pool model, and will use automatic control theory to adjust its performance locally in order to meet the overall target performance. The automatic control system in each stage consists of an automatic modeling mechanism and a feedback module, which optimizes the controller parameters

in the system automatically and guarantees the quality of performance (service rate here) for the stage at runtime.

Based on mathematical proof and simulation results, our designs are implemented in a SEDA-based web server running in a dynamic loading environment. Results demonstrate that the performance of the new system in the real world is almost the same as the theoretical results. It demonstrates that the design is able to adaptively ensure the quality of service to the high-concurrent requests fairly. Compared to the original SEDA design, our design is an effective and handy approach to significantly enhancing the performance of SEDA in a variety of aspects, including fairer service, faster convergent speed, better robustness, higher accuracy and ease of deployment in various practical applications.

## Figure List

Figure 2-1 Simplified Request Processing Steps .....	13
Figure 2-2 Finite state machine for a simple HTTP server request .....	14
Figure 4-1 Global Control Framework .....	28
Figure 4-2 Self-Tune Stage Model .....	30
Figure 4-3 Selective Dropper .....	32
Figure 4-4 Feedback Control Model .....	36
Figure 4-5 Traditional Controlled System Model .....	38
Figure 4-6 Stage Thread Pool Model .....	38
Figure 4-7 Comparison of the estimated output and the actual output .....	41
Figure 4-8 Stage Feedback Control Model .....	42
Figure 4-9 PID control on the stage .....	43
Figure 4-10 P-Control Based Pre-Compensator Control System .....	47
Figure 4-11 The simulation result of P control on Stage .....	47
Figure 4-12 The process that achieves the optimal values to configure controller .....	49
Figure 4-13 Polar Coordinator .....	51
Figure 4-14 The simulation result of PI control on the stage .....	53
Figure 4-15 The simulation result of PD control on the stage .....	54
Figure 5-1 SEDA-based web server .....	55
Figure 5-2 The performance of P, PI and PD control .....	56
Figure 5-3 The performance of P control in simulation and practice .....	57
Figure 5-4 The performance of PI control in simulation and practice .....	57
Figure 5-5 The performance of PD control in simulation and practice .....	57
Figure 5-6 Comparison of the P-Control and Heuristic Control .....	58
Figure 5-7 Adjust the 90th Response Time On-The-Fly .....	60
Figure 5-8 The Fair Service under Dynamic Workload .....	61
Figure 5-9 The Difference of the Arrival Rate and Departure Rate (positive value means Arrival Rate is greater than Departure Rate) .....	63
Figure 5-10 Distribution of the Response Time .....	63
Figure 5-11 the Distribution of the Response Time .....	66
Figure 5-12 the 90th Response Time in Runtime .....	67
Figure 5-13 The Comparison of the Flow of Rate .....	68
Figure 5-14 Performance of the Stages under the Automatic Fairness Control Strategy .....	69
Figure 5-15 Performance of the Stages in the Original SEDA .....	69

## Table of Contents

Acknowledgement .....	2
Abstract .....	3
Figure List .....	5
Table of Contents .....	6
Chapter 1 Introduction .....	8
1.1 Problem Addressing .....	8
1.2 Dissertation Roadmap .....	10
Chapter 2 Related Work .....	12
2.1 Thread-based Concurrency Model .....	12
2.2 Event-Driven Concurrency Model .....	13
2.3 SEDA (Staged Event-Driven Architecture) .....	15
2.4 Admission Control on Queuing .....	16
2.5 Classical Control Approaches .....	17
2.6 Other Control Strategies .....	18
Chapter 3 Architecture for High-Concurrency: Staged Event-Driven Architecture (SEDA) .....	20
3.1 Introduction .....	20
3.2 Staged Event-Driven Architecture (SEDA) .....	20
3.2.1 Fundamental Unit of SEDA: STAGE .....	21
3.2.2 Staged Network .....	23
3.3 Performance Control on SEDA .....	25
Chapter 4 Quality-Guaranteed Fair Service on SEDA .....	26
4.1 Introduction .....	26
4.2 Control Framework for SEDA-based Fair Service .....	27
4.2.1 Global Control Framework .....	28
4.2.2 Self-Tune Stage Model .....	29
4.2.3 Early Selective Dropping .....	31
4.3 Control Algorithms for Fair Service .....	32
4.3.1 Mapping the Percentile Response Time to Throughput .....	33
4.3.2 Algorithm for Throughput Control .....	36
4.3.3 Automatic Modeling .....	36
4.3.4 Feedback Control System .....	41
Chapter 5 Tests, Evaluation and Analysis .....	55
5.1 Throughput Control and Evaluations .....	56
5.2 Controlling the 90 <sup>th</sup> Percentile Response Time .....	59
5.2.1 Follow the Dynamic 90 <sup>th</sup> Percentile Response Time Target .....	60
5.2.2 Maintain stable 90th percentile response Time .....	61
5.3 Compared with the Original SEDA in an Open-Loop Benchmark .....	64
Chapter 6 Conclusions and Future Work .....	71
6.1 Conclusions of the Thesis .....	71
6.2 Future Work .....	71
6.2.1 Better Control Strategies .....	72
6.2.2 Use in Large-Scale Systems .....	72
6.2.3 Deployment in Single Thread Pool System .....	72

References..... 73