

2006

The effect of workload dependence in systems: Experimental evaluation, analytic models, and policy development

Qi Zhang

College of William & Mary - Arts & Sciences

Follow this and additional works at: <https://scholarworks.wm.edu/etd>



Part of the [Computer Sciences Commons](#)

Recommended Citation

Zhang, Qi, "The effect of workload dependence in systems: Experimental evaluation, analytic models, and policy development" (2006). *Dissertations, Theses, and Masters Projects*. Paper 1539623507.

<https://dx.doi.org/doi:10.21220/s2-n7eq-0f90>

This Dissertation is brought to you for free and open access by the Theses, Dissertations, & Master Projects at W&M ScholarWorks. It has been accepted for inclusion in Dissertations, Theses, and Masters Projects by an authorized administrator of W&M ScholarWorks. For more information, please contact scholarworks@wm.edu.

The Effect of Workload Dependence in Systems: Experimental Evaluation, Analytic Models, and Policy Development

A Dissertation

Presented to

The Faculty of the Department of Computer Science

The College of William and Mary in Virginia

In Partial Fulfillment

Of the Requirements for the Degree of

Doctor of Philosophy

by

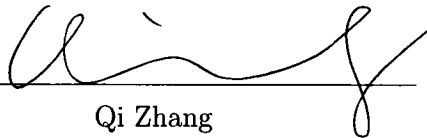
Qi Zhang

2006

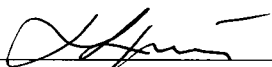
APPROVAL SHEET

This dissertation is submitted in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy



Qi Zhang

Approved, December 2006


Evgenia Smirni
Dissertation Advisor


Phil Kearns


Weizhen Mao


Peter Kemper


Ludmila Cherkasova
Hewlett-Packard Labs

To my parents and dear husband ...

Table of Contents

Acknowledgments	viii
List of Tables	ix
List of Figures	x
Abstract	xviii
1 Introduction	2
1.1 Contributions	5
1.1.1 Experimental Evaluation	5
1.1.2 Analytic Models	6
1.1.3 Policy Development	7
1.2 Organization	8
2 Background	10
2.1 Autocorrelation	10
2.2 Batch Markovian Arrival Processes (BMAPs)	13
2.2.1 Markovian Arrival Processes (MAPs)	15
2.2.2 Markovian-Modulated Poisson Processes (MMPPs)	16

2.2.3	Hyperexponential Processes	17
3	Autocorrelation in Closed Systems	19
3.1	Related Work	20
3.2	Closed Tandem Queues	23
3.2.1	A 2-tier System	23
3.3	Autocorrelation Propagation	25
3.4	Performance Effects	28
3.5	Autocorrelation Analysis in TPC-W	32
3.5.1	Experimental Environment	33
3.5.1.1	Capacity Planning	36
3.5.2	Autocorrelation Propagation in TPC-W	37
3.5.3	TPC-W Model	40
3.6	Taking Advantage of ACF	43
3.7	Summary	48
4	Bottleneck Analysis in TPC-W	50
4.1	Capacity Planning	52
4.2	System Implications of the Stationary Workloads	55
4.3	Transient Workload Propagation	57
4.4	System Implications of the Transient Workloads	65
4.5	Summary	71
5	Analytic Models in Multi-tiered Systems	73
5.1	Related Work	75
5.2	Background	76

5.2.1	M/G/1-type Processes	76
5.2.2	Example: ETAQA Representation of the $M^{[2]}/M/1$ Queue	81
5.3	ETAQA-based Approximation of the BMAP/MAP/1 Departure Process . .	82
5.3.1	Exact Departure Process of the BMAP/MAP/1 Queue	82
5.3.2	Truncating the Exact Departure Process	83
5.3.2.1	Special Case: Output Approximations for the MAP/MAP/1 Queue	85
5.3.2.2	Illustration for the $M^{[2]}/M/1$ Departure Process	86
5.3.3	Proof of the Marginal Distributions and ACF of the ME Departure Approximation	87
5.3.4	Experimental Results	98
5.4	Summary	109
6	Improved Truncation Models for the BMAP/MAP/1 Departure Process	110
6.1	MAP Approximation Output Models	112
6.2	Comparison of MAP Approximation and ME Approximation	120
6.2.1	Experimental Comparison	120
6.2.2	Complexity Issues	126
6.2.2.1	ME Output Model	127
6.2.2.2	MAP Output Model	128
6.2.2.3	Departure Lag Coefficients via the BMAP/GI/1 Approach .	129
6.3	Identifying a Truncation Level	131
6.3.1	Asymptotic Behavior of the ACF	132
6.3.2	The Role of Utilization	136

6.3.3	Remarks on the ME Approximation	138
6.4	Experimental Results	139
6.5	Improving Accuracy with Multiple Lumping	142
6.6	Summary	146
7	Policy Development for Handling Systems with Autocorrelated Flows	147
7.1	Autocorrelation Effects on Load Balancing Policies	151
7.1.1	Load Balancing Policies	152
7.1.2	Performance Analysis	153
7.2	Unbalancing Load to Improve Performance	157
7.2.1	S_EQAL: Static Policy	157
7.2.1.1	Arrival process with short-range dependence	159
7.2.1.2	Arrival process with long-range dependence	162
7.2.2	D_EQAL: On-line Policy	163
7.2.2.1	Performance of D_EQAL	165
7.3	Summary	168
8	Conclusions and Future Work	171
8.1	Future work	173
8.1.1	Analytic Direction	173
8.1.2	Application Direction	175
A	Equivalence of open/closed systems	177
	Bibliography	179
	Vita	187

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor, Dr. Evgenia Smirni, for her continuous support, guidance and inspiration over the years. I have benefited greatly from our close relationship. Her passion for high quality research has influenced me significantly. On a more personal note, her advice on balancing family and career has been and will continue to be a guide in my future endeavors. Without her patience and encouragement, I would not be the researcher I am today. Evgenia, thank you.

Special thanks go out to Dr. Alma Riska, who has been a great mentor and collaborator over the years. My work in multi-tiered systems started as an intern project under the guidance of Alma while I was visiting Seagate Research Laboratory in the fall of 2003. Her experience and knowledge greatly improved my understanding of many research problems. I would also like to thank Dr. Ludmila Cherkasova for her mentorship during my internship at HP Labs in the summer of 2006. Working closely with her on capacity planning problems has taught me many valuable research skills. I am especially grateful to Dr. Cherkasova for serving as the external member of my committee, for her comments and suggestions on this dissertation, and for her encouragement. I also thank Dr. Armin Heindl at the University of Erlangen-Nuremberg in Germany. He introduced to me the problem of departure processes that forms an important part of my dissertation.

I would like to thank Dr. Andreas Stathopoulos with whom I collaborated in several projects. His wide breadth of knowledge and rigorous research attitude impressed me deeply. I am grateful to Dr. Phil Kearns, Dr. Weizhen Mao, and Dr. Peter Kemper for serving on my dissertation committee, and providing me with many valuable suggestions. I also would like to thank Dr. Gianfranco Ciardo, Dr. Xiaodong Zhang, and Dr. Dimitrios Nikolopoulos for providing me with good suggestions throughout my research career. I am especially grateful to my teammate Ningfang Mi for many engaging discussions. Our collaborative efforts on different projects form an important component of this dissertation.

I would like to extend my thanks to all the faculty and staff in the Computer Science Department for providing me with a friendly and productive work environment. In particular, I would like to thank Vanessa Godwin for her kind assistance on many administrative issues enabling me to focus completely on my research work, and for not letting me forget any University deadlines. To my fellow graduate student colleagues, all of you have made my stay at William and Mary truly memorable.

To all my past teachers, I would like to thank them for their excellent teaching and advice during my formative years in China. Their dedication has given me a firm foundation for my further studies in the United States.

Finally, I would like to dedicate this dissertation to my dear father Quanxing Zhang, my mother Wenling Li, and my husband Xin Chen, for their unlimited love and understanding during my Ph.D period. My parents taught me the value of education and worked hard to provide me with the best of it. My brother Yi Zhang has been a constant source of joy and support. I am also fortunate to have Xin by my side during this journey, always standing by me, loving me, understanding me and encouraging me. Without them, I would never have gone this far.

List of Tables

3.1	Summary of the two scenarios.	25
3.2	Summary of the MMPPs used in the two scenarios.	26
3.3	Hardware components on the on-line bookstore implementation	33
3.4	Sizes of important tables	35
6.1	The second largest eigenvalue of $(-\mathbf{D}_{0,n}^{(\text{MAP})})^{-1}\mathbf{D}_{1,n}^{(\text{MAP})}$	134
6.2	The second largest eigenvalue of $(-\mathbf{D}_{0,n}^{(\text{ME})})^{-1}\mathbf{D}_{1,n}^{(\text{ME})}$	139

List of Figures

1.1	(a) Average response time, (b) average queue length, and (c) average slowdown as a function of system utilization when inter-arrivals are independent, weakly dependent, and strongly dependent.	3
2.1	ACF of inter-arrival and service times for disk level traces measured in enterprise systems and consumer electronics devices.	12
2.2	State transitions of MMPP(2).	16
2.3	State transitions of H_2	18
3.1	A closed system with M queues.	23
3.2	ACF of departures from Q_1 (arrivals to Q_2), departures from Q_2 (arrivals to Q_1) for both scenarios, and the ACF of the correlated service process. ACF of the MMPP service process in (a)/(c) is SRD and that in (b)/(d) is LRD.	27
3.3	Performance measures: (a) mean round trip time, (b) mean queue length, (c) mean utilization, and (d) mean throughput at each queue for <i>Scenario 1</i> . The bottleneck queue (Q_2) has autocorrelated service times.	29
3.4	Performance measures: (a) mean round trip time, (b) mean queue length, (c) mean utilization, and (d) mean throughput at each queue for <i>Scenario 2</i> . The non-bottleneck queue (Q_1) has autocorrelated service times.	30

3.5	CDFs of (a) response time at Q_1 , (b) response time at Q_2 , and (c) round trip time for <i>Scenario 1</i> with $MPL=25$. The bottleneck queue (Q_2) has autocorrelated service times.	32
3.6	CDFs of (a) response time at Q_1 , (b) response time at Q_2 , and (c) round trip time for <i>Scenario 2</i> with $MPL=25$. The non-bottleneck queue (Q_1) has autocorrelated service times.	32
3.7	E-commerce experimental environment.	33
3.8	(a) Throughput, (b) front-end CPU utilization, (c) database CPU utilization, and (d) database memory utilization for the browsing mix and database sizes of 10,000, 100,000, 500,000, and 1,000,000 items.	36
3.9	TPC-W experimental environment.	37
3.10	ACF at various points in the system. Experiments are done using the browsing mix, a database with 10,000 items, and (a) 128 EBs, (b) 384 EBs, and (c) 512 EBs.	38
3.11	Average performance measures with the browsing mix.	38
3.12	A queuing model of TPC-W.	41
3.13	Autocorrelation propagation in our queuing model parameterized using the measurements of Section 3.5.2 with MPL equal to (a) 128, (b) 384, and (c) 512.	43
3.14	Model prediction and experimental performance measures.	43
3.15	CCDFs of (a) round trip time, (b) response time of front server, (c) response time of database server using the model of Section 3.5 where the front server has ACF in its service process. In all experiments MPL is equal to 512. . . .	46

3.16	CCDFs of (a) round trip time, (b) response time of front server, (c) response time of database server when the database server has ACF in its service process. In all experiments MPL is equal to 512.	47
4.1	(a) Throughput, (b) front-end CPU utilization, (c) database CPU utilization, and (d) database memory utilization as a function of system load for the database with 500,000 items and various TPC-W mixes.	53
4.2	Query time distribution (CDF) for the browsing mix under various loads (left column) and under medium load for various TPC-W mixes (right column). .	54
4.3	Disk access pattern for the 1,000,000 items database. The system is under 90% new searches mix.	56
4.4	Throughputs and utilizations in multiple tiers under transient load of the browsing mix.	58
4.5	Arrival rates and system utilizations in multiple levels under transient work, for the ordering mix and a variant (I) and the browsing mix and a variant (II). 62	
4.6	Outstanding requests (queue lengths) at the disk and at the database during overload and overwork scenarios of (a) experiment two and (b) experiment four. 67	
4.7	Response time at the disk as a function of time under (a) normal scheduling, (b) overload-aware scheduling, and (c) the complementary cumulative distribution during the overload period of experiment two.	69
4.8	Response time at the disk as a function of time under (a) normal scheduling, (b) overload-aware scheduling, and (c) the complementary cumulative distribution during the overwork period of experiment four.	70

5.1	A queuing model of the experimental environment in Chapter 3 but with infinite buffers.	73
5.2	State-space partitioning of a M/G/1 solved (a) via the traditional matrix-analytic method, (b) via ETAQA, and (c) ETAQA that preserves the first n M/G/1 levels, $n \geq 1$	78
5.3	Dual tandem queues	98
5.4	The Markov chain that models an $M^{[2]}/M/1$ queue.	99
5.5	ACF of inter-arrival times of batches in the system (dashed curve) and of inter-arrival times of actual arrivals (solid curve).	100
5.6	Experimental results for example 1: ACF of departures from server 1/arrivals to server 2 (a–b), mean queue length at server 2 (c–d), queue length distribution at server 2 for different approximation levels (e–f), and ACF of departures from server 2 (g–h).	101
5.7	ACF of inter-arrival times of batches in the system (dashed curve) and of inter-arrival times of actual arrivals (solid curve).	104
5.8	Experimental results for example 2: ACF of departures from server 1/arrivals to server 2 (a–b), mean queue length at server 2 (c–d), queue length distribution at server 2 for different approximation levels (e–f), and ACF of departures from server 2 (g–h).	105
5.9	Experimental results for example 3: ACF of departures from server 1/arrivals to server 2 (a–b), mean queue length at server 2 (c–d), queue length distribution at server 2 for different approximation levels (e–f), and ACF of departures from server 2 (g–h).	108

6.1	Experimental results of MAP approximation for example 1: ACF of departures from server 1/arrivals to server 2 (a-b), mean queue length at server 2 (c-d).	122
6.2	Experimental results of MAP approximation for example 2: ACF of departures from server 1/arrivals to server 2 (a-b), mean queue length at server 2 (c-d).	124
6.3	Experimental results of MAP approximation for example 3: ACF of departures from server 1/arrivals to server 2 (a-b), mean queue length at server 2 (c-d).	126
6.4	Autocorrelation of the departures from a 30% utilized and an 80% utilized BMAP(3)/H ₂ /1 systems.	136
6.5	Relative ACF error under different truncation level n for system utilizations equal to 30% and 80%.	137
6.6	Probabilities of the lumped states for different truncation levels n under system utilizations 30% and 80%.	138
6.7	Experimental results of MAP approximation: mean queue length relative errors at server 2 of I. 30%-30% system, II. 30%-80% system, III. 80%-30% system and IV. 80%-80% system.	141
6.8	High level idea of lumping multiple levels.	142
6.9	High level idea of multiple lumping: consecutive C levels are lumped in single lumps.	143
6.10	Total number of block levels is 51: autocorrelation of the departures from server 1/arrivals to server 2.	143
6.11	Setting $[C_1, C_2, \dots, C_m]$ using a greedy algorithm.	144

6.12	Relative error of average queue length in server 2.	145
7.1	Model of a clustered server.	148
7.2	(a) ACF for the three arrival processes used in the simulation and (b) Average request size for every 10000 requests in the ten million sample space.	152
7.3	Performance metrics under four load balancing policies: (a) average response time, (b) average queue length, (c) average slowdown, and (d) average utilization.	154
7.4	ACF in inter-arrivals at each server, where the arriving requests at the front-end dispatcher have (a) no dependence, (b) short range dependence (SRD), and (c) long range dependence (LRD).	156
7.5	S_EQAL's high level idea to recalculate boundaries under autocorrelated inter-arrival times.	159
7.6	Setting the shifting percentages p_i for S_EQAL.	159
7.7	Average slowdown and average response time as a function of the corrective constant R under SRD inter-arrival times.	160
7.8	Per server performance measures: (a) average slowdown, (b) average response time, (c) average queue length and (d) average utilization as a function of the corrective constant R with SRD inter-arrival times. The order of bars for each policy reflect the server identity.	161
7.9	Average slowdown and average response time as a function of corrective constant R under LRD inter-arrivals times.	162

7.10	Per server performance metrics as a function of the corrective constant R under LRD traffic: (a) average slowdown, (b) average response time, (c) average queue length and (d) average utilization. The order of bars for each policy reflect the server identity.	163
7.11	D_EQAL: dynamically adjusting R	166
7.12	Performance of NOACF, SRD ad LRD traffic under average utilization 20%. The first two rows are average slowdown and average response time for the original ADAPTLLOAD, S_EQAL with various values of R , and D_EQAL. The third row is the corrective constant R as a function of time (measured in processed requests) for $C = 300K$	167
7.13	Performance of NOACF, SRD ad LRD traffic under average utilization 62%. The first two rows are average slowdown and average response time for the original ADAPTLLOAD, S_EQAL with various values of R , and D_EQAL. The third row is the corrective constant R as a function of time (measured in processed requests) for $C = 300K$	168
7.14	Performance of NOACF, SRD ad LRD traffic under average utilization 80%. The first two rows are average slowdown and average response time for the original ADAPTLLOAD, S_EQAL with various values of R , and D_EQAL. The third row is the corrective constant R as a function of time (measured in processed requests) for $C = 300K$	169
7.15	Corrective constant R , average slowdown, average normalized response time as a function of time (measured in processed requests) for $C = 300K$ under NOACF, SRD and LRD traffic. System utilization is 62%.	170

A.1	Illustration of the equivalence of an open system with finite buffer of size M	
	and a closed system with $MPL = M$	178

ABSTRACT

This dissertation presents an analysis of performance effects of burstiness (formalized by the autocorrelation function) in multi-tiered systems via a 3-pronged approach, i.e., experimental measurements, analytic models, and policy development. This analysis considers (a) systems with finite buffers (e.g., systems with admission control that effectively operate as closed systems) and (b) systems with infinite buffers (i.e., systems that operate as open systems).

For multi-tiered systems with a finite buffer size, experimental measurements show that if autocorrelation exists in any of the tiers in a multi-tiered system, then autocorrelation propagates to *all* tiers of the system. The presence of autocorrelated flows in all tiers significantly degrades performance. Workload characterization in a real experimental environment driven by the TPC-W benchmark confirms the existence of autocorrelated flows, which originate from the autocorrelated service process of one of the tiers. A simple model is devised that captures the observed behavior. The model is in excellent agreement with experimental measurements and captures the propagation of autocorrelation in the multi-tiered system as well as the resulting performance trends.

For systems with an infinite buffer size, this study focuses on analytic models by proposing and comparing two families of approximations for the departure process of a BMAP/MAP/1 queue that admits batch correlated flows, and whose service time process may be autocorrelated. One approximation is based on the ETAQA methodology for the solution of M/G/1-type processes and the other arises from lumpability rules. Formal proofs are provided: both approximations preserve the marginal distribution of the inter-departure times and their initial correlation structures.

This dissertation also demonstrates how the knowledge of autocorrelation can be used to effectively improve system performance, D_EQAL, a new load balancing policy for clusters with dependent arrivals is proposed. D_EQAL separates jobs to servers according to their sizes as traditional load balancing policies do, but this separation is biased by the effort to reduce performance loss due to autocorrelation in the streams of jobs that are directed to each server. As a result of this, not all servers are equally utilized (i.e., the load in the system becomes unbalanced) but performance benefits of this load unbalancing are significant.

The Effect of Workload Dependence in Systems: Experimental
Evaluation, Analytic Models, and Policy Development

Chapter 1

Introduction

Internet servers are developing into complex but central components in the information infrastructure, and are accessed by an ever-increasing and diversified user population. In such computing or networking systems, burstiness, as a form of temporal dependence within the workload, has been widely recognized as a significant factor affecting performance.

Prior work in the context of networking [22] has shown that burstiness in the arrival process in a single server system may result in user response times that are slower by several orders of magnitude when comparing to a system with independent arrivals. To appreciate this, we present performance measures of a single server queue with different degrees of dependence in its inter-arrival process. Three sets of experiments are presented here labeled as “independent”, “weakly dependent”, and “strongly dependent”, but all other stochastic characteristics of these inter-arrival processes, i.e., the mean and all higher moments, are identical. The service process in all experiments is exponentially distributed and appropriately scaled such that we examine performance under different utilization levels.

Figure 1.1 presents performance measures for this queuing model as a function of system utilization. The figure shows average response time (see Figure 1.1(a)) defined as the sum

of the request service time and its waiting time in the queue, average queue length (see Figure 1.1(b)) which is the total number of requests in the queue including the one in service, and average request slowdown (see Figure 1.1(c)) which is the ratio of the response time of a request to its service time. The effect of dependence on system performance is tremendous: the stronger the dependence, the worse the system performance, which can worsen by 3 orders of magnitude when comparing to the case with independent arrivals.¹

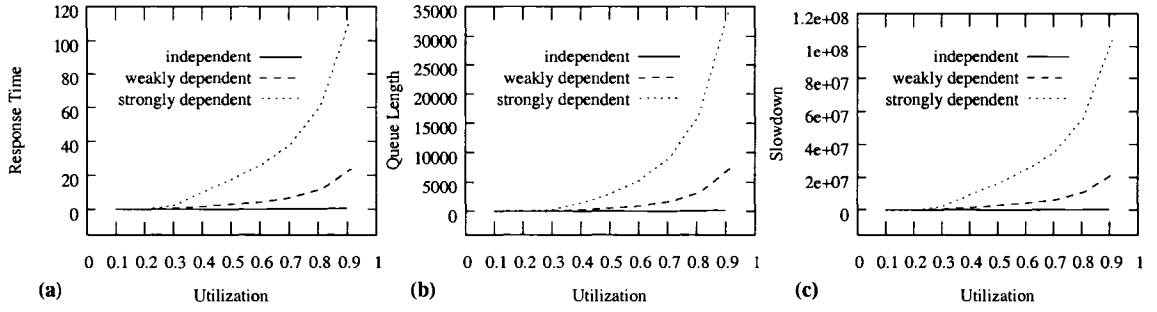


Figure 1.1: (a) Average response time, (b) average queue length, and (c) average slowdown as a function of system utilization when inter-arrivals are independent, weakly dependent, and strongly dependent.

These performance measures illustrated in Figure 1.1 serve as motivation for the work presented in this dissertation. We stress that none of prior work has considered the effect of dependent flows on multi-tiered systems. Multi-tiered architectures, such as the ones in e-commerce web sites that separate the database server from the web and application servers, are widely used today to improve performance. But they also introduce difficulty in identifying the source of burstiness (e.g., burstiness may not exist due to arrivals from a wide-area network), and hence make it hard for effective development of resource management policies that improve performance or provide service level provisioning. Dependence in the

¹Because of the linear scale used in the figure, the performance measures with independent arrivals appear flat. With independent arrivals for utilization equal to 0.9, queue length is equal to 152 as expected, but this number is dwarfed in comparison to the numbers under dependent flows.

arrival or service flows of one tier propagates into its lower tiers in an open system or *all* the tiers in a closed system.

The focus of this dissertation is on identifying dependent flows in multi-tiered systems and on understanding their performance implications. Such understanding is significant for the development of robust resource allocation policies that remain effective under variable workload conditions. Analytic performance models are usually excellent tools to quickly compare the performance of alternative system designs, but they are often limited due to restrictive assumptions [60], e.g., the inter-arrival and service processes must be independent and identically distributed. These assumptions do not apply in current bursty environments.

This dissertation provides a formalization of burstiness based on the autocorrelation function [44] that characterizes the temporal dependence structure in request flows. New analytic models are devised to capture the performance effects of autocorrelation in queueing systems for both open and closed systems. Based on these, development of new resource allocation policies is proposed to take advantage of temporal dependence in the workload arrivals and service demands in a variety of applications. To best address the above open problems, this dissertation uses a 3-pronged approach:

- *Experimental evaluation:* Obtain a better understanding of autocorrelation and its performance effects using an experimental system of an on-line bookstore using TPC-W, the current industry standard for e-commerce benchmarks. This work concentrates on characterizing the propagation of workload burstiness in a multi-tiered system. We also use an array of metrics ranging from average throughput and average utilization to response time distributions to better understand the behavior of the system and the triggering of different bottlenecks that may bring the system to overload.

- *Analytic models:* Devise two families of finite approximations for the departure process of a BMAP/MAP/1 queue. A BMAP/MAP/1 queue is a general queueing model of a system having input that is best described as dependent flows with batch arrivals, and where the service time is also dependent. It can be used as a building block to model a single tier in a multi-tiered system.
- *Policy development:* Use dependence information in flows (i.e., autocorrelation) to provide advice for resource management, such as capacity planning, admission control, scheduling and load balancing. The development of such policies makes an important step toward autonomic servers.

1.1 Contributions

The contributions of this dissertation are summarized as follows.

1.1.1 Experimental Evaluation

- We observe the existence of dependent flows and present how it propagates through the network on an e-commerce system that is built according to the TPC-W benchmark by measuring the autocorrelation of the arrival and departure processes of each tier. This characterization shows that autocorrelation in the service process can be the source of dependent flows in *all* tiers of this system, even to the ones that precede the source tier of autocorrelation.
- By quantitatively comparing the performance effects of the presence of autocorrelated flows in multi-tiered systems with the performance of the same system with independent flows, we show that end-to-end performance significantly deteriorates while

bottleneck devices are less utilized. Lower utilization levels in such systems falsely indicate that the system is able to sustain higher loads.

- We show that in contrast to systems where no burstiness is observed, the tails of the overall response time distributions do not necessarily reflect the time spent at the bottleneck tier, but instead are shaped by the response time tail at the tier that is the source of autocorrelation, irrespective of its utilization level.
- We show that both the transient excessive load in the system (i.e., higher than usual number of users, number of web sessions, or number of network flows) and the effects of transient excessive work (i.e., sudden increase in the demand of system resources by the current users of the system) could results in system overload. The further load/work propagates down the system hierarchy, i.e., the memory/disk, the higher the performance penalty, and the more difficult it is to recover performance effectively.

1.1.2 Analytic Models

- The traditional analytic techniques of closed systems, e.g., MVA models [53, 43], cease to apply in systems with dependent flows. We use a queueing model to model an e-commerce system based on the TPC-W benchmark by representing the service times of the dependent server with an autocorrelated process. The model and experimental results are in excellent agreement, despite the fact that no rigorous fitting method is used to fit the measured data of the service processes. Stochastic processes that capture autocorrelation show promise to effectively model complex systems via surprisingly simple queueing models.
- For open systems, the departure process is an effective way of solving the network of

queues. We present two families of finite approximations for the departure process of a BMAP/MAP/1 queue. One is derived via an exact aggregate solution technique called ETAQA applied to M/G/1-type Markov processes, and the second one is based on lumpability. The proposed approximations are indexed by a parameter n ($n > 1$), which determines the size of the output model as $n + 1$ block levels of the M/G/1-type process.

- We formally prove that these two output approximation models preserve exactly the marginal distribution of the true departure process and the lag correlations of the inter-departure times up to lag $(n - 2)$ with the ETAQA-based approximation and up to lag $(n - 1)$ with the lumpability-based approximation. A comparison of these two approximations is carefully done, focusing on both accuracy and time complexity. The asymptotic properties of the approximations are examined, which then aid in identifying the level n that maintains well the queueing behavior of the real departure process.
- For queues with high utilization and strong dependence in arrival/service process, we present multiple lumping as a way to use significantly smaller size models while maintaining nearly the same performance properties as the lumpability-based approximations of much larger size.

1.1.3 Policy Development

- We present results on how to take advantage of the knowledge of autocorrelation in flows for resource management via a preliminary study of admission control algorithms that consider dynamic system behavior. Instead of doing admission control at the

bottleneck tiers, this algorithm controls the flows in the tier that is the source of autocorrelation in the system, and achieves promising performance improvements. This dissertation also shows a first proof-of-concept of self-adaptive resource management at the lower tiers that can detect and handle overload. Such mechanisms can help to achieve graceful performance degradation and to improve system availability.

- This dissertation also presents a size-based load balancing policy in a cluster of servers with a single system image. The policy is called D_EQAL. It aims at reducing the performance degradation due to autocorrelation in each server by distributing the work guided by the correlation structure of the arrivals to each server. This policy does not assume any *a priori* knowledge of the job service time distribution nor any knowledge of the intensity of the dependence structure in the arrival streams. Using trace-driven simulation, we show that D_EQAL is an effective on-line policy: by monitoring performance measures it self-adjusts its parameters to transient workload conditions and achieves superior performance in comparison to other classic policies.

1.2 Organization

The dissertation is organized as follows: Chapter 2 presents an overview of basic concepts and terminology that are used in this dissertation. Chapter 3 uses a simple queuing network to quantify the performance effects of autocorrelation in closed systems. This chapter also presents an experimental study using the TPC-W benchmark, and shows how autocorrelation propagates across all the tiers in a multi-tiered system if it is found in the service process of one of the tiers. A simple queuing model that captures the benchmark's behavior is given. Chapter 4 further characterizes the e-commerce workload driven by TPC-W

benchmark, specially focusing on system overload. Chapter 5 presents ETAQA-based departure process models of BMAP/MAP/1 queues, and Chapter 6 presents lumpability-based departure process models. Comparison of these two models and asymptotic properties of the approximated departure process are also discussed in Chapter 6. Chapter 7 presents D_EQAL, a size-based load balancing policy to illustrate a case study of policy development for handling systems with autocorrelated flows. A summary of this dissertation and future work are outlined in Chapter 8.

Chapter 2

Background

This chapter presents an overview of basic concepts and terminology that are used in this dissertation. Section 2.1 gives a the definition of autocorrelation function, a metric used in this dissertation to quantitatively describe dependence in a flow. Section 2.2 gives an overview of the definitions and properties of the Batch Markovian Arrival Process (BMAP) and Markovian Arrival Process (MAP), a special case of BMAP. These processes can be used to model dependent flows and can be used as inputs to analytic models.

2.1 Autocorrelation

The autocorrelation function is used as a statistical measure of the relationship between a random variable and itself [9]. Consider a stationary time series of random variables $\{X_n\}$, where $n = 0, \dots, \infty$, in discrete time. The autocorrelation function $ACF(k)$ shows the value of the correlation coefficient for different time lags $k > 0$:

$$ACF(k) = ACF_{X_t, X_{t+k}} = \frac{E[(X_t - \mu)(X_{t+k} - \mu)]}{\delta^2},$$

where μ is the mean and δ^2 is the common variance of $\{X_n\}$. The argument k is called the lag and denotes the number of observations that separate X_t and X_{t+k} . The values of $ACF(k)$ may range from -1 to 1. The higher $|ACF(k)|$ is, the stronger the dependence between X_t and X_{t+k} . If $ACF(k) = 0$, then there is no dependence at lag k .

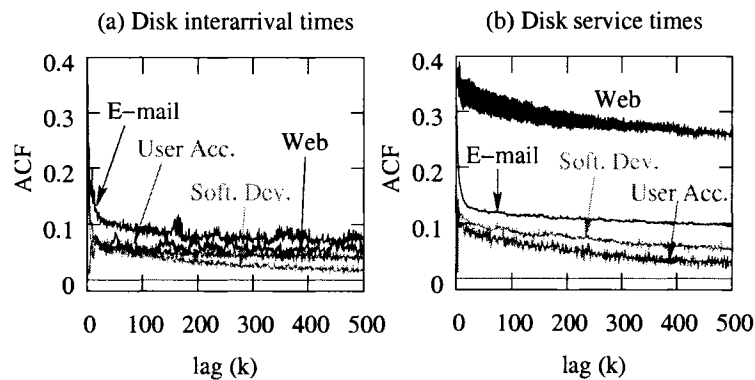
In most cases ACF approaches zero as k increases. The decay rate of the ACF distinguishes the time series as short-range dependent (SRD) or long-range dependent (LRD). The ACF essentially captures the “ordering” of random values in the time series. Intuitively, if there is no autocorrelation (i.e., ACF is zero, which implies there is independence in the stochastic process), a random variable is generated as follows: first a random number is drawn between 0 and 1 and then this random number is mapped into the distribution space via the inverse distribution function to obtain the random value. This way of sampling does not create any temporal locality, i.e., given the current drawn value, *any* value of the distribution space is equally likely to occur in the next drawing. In distributions that have correlation there is a temporal bias in this sampling, i.e., random variables are sampled within a certain range for some time before moving into another range. This creates temporal locality, yet overall all the values of the distribution space are sampled as dictated by the distribution function. High positive ACF values imply that in the time series a value of the random variable has a high probability to be followed by another variable of the same order of magnitude, while negative ACF values imply the inverse.

To show that the ACF measure is not artificial but instead can be found in real systems, we present small examples that give strong indication about the existence of autocorrelated flows in real systems¹. Case 1 in Figure 2.1 shows the ACF for a collection of enterprise storage systems, i.e., a web server, an E-mail server, a Software Development server, and

¹The data presented here are generously provided by Seagate Research.

a User Accounts server. Measurements are taken of the arrival process at the disk in an open-loop, finite buffer, enterprise server - user arrivals feed into the server (first tier) where they are modified by caches and other processing before being passed to the disk (second tier). Figure 2.1(a) shows the ACF of the arrival times at the disk (departure process from the server), while Figure 2.1(b) depicts the ACF of the service process at the disk.

Case 1: Enterprise Systems



Case 2: Consumer Electronics

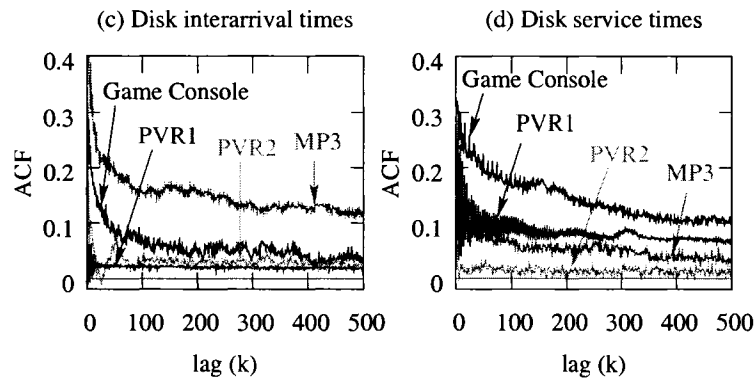


Figure 2.1: ACF of inter-arrival and service times for disk level traces measured in enterprise systems and consumer electronics devices.

Case 2 in Figure 2.1 shows the ACF for a collection of traces from consumer electronics devices, i.e, a Personal Video Recorder (PVR) in two different application scenarios, an MP3 player, and a Game console, which represents classical closed-loop systems such as the one we consider in Chapter 3. Since there is only a single user, the first tier server (caches and

application processing) passes requests directly to the disk (second tier) which feeds back to the request arrival process. Figure 2.1(c) shows the ACF of the arrival times at the disk (departure process from the server), which also differs markedly from the ACF of the service process at the disk in Figure 2.1(d) in some cases.

In both sets of measurements, arrivals and service times are correlated, with some cases of pronounced long-range dependence, i.e., ACF lines that decay slowly to zero, ACF is stronger pronounced in the service times of the Web server in Figure 2.1(b) and the Game console in Figure 2.1(d). These measurements show that autocorrelation exists at the disk tier for different workloads in large and small systems. Server processing - in particular the cache hierarchy and algorithms along the I/O path - determines how arrivals and service demands are shaped at this lowest tier.

2.2 Batch Markovian Arrival Processes (BMAPs)

A BMAP, as introduced by Lucantoni [48], is controlled by an ergodic Continuous-Time Markov Chain (CTMC) with finite state space $\{1, 2, \dots, m_{\text{BMAP}}\}$. In state i , the sojourn time of the process is exponentially distributed with mean rate λ_i . At the end of such a sojourn time, a batch of size k ($k \geq 1$) may occur with probability $p_{i,j}^{(k)}$, and the CTMC passes to state j ($1 \leq i, j \leq m_{\text{BMAP}}$). Alternatively, no customer arrives ("batch of size 0") with probability $p_{i,j}^{(0)}$, while the CTMC passes to state j ($j \neq i$). Naturally, we require that

$$\sum_{j=1, j \neq i}^{m_{\text{BMAP}}} p_{i,j}^{(0)} + \sum_{k=1}^{\infty} \sum_{j=1}^{m_{\text{BMAP}}} p_{i,j}^{(k)} = 1 \quad \text{for } 1 \leq i \leq m_{\text{BMAP}} \quad .$$

The corresponding transition rates $\lambda_i p_{i,j}^{(k)}$ may be grouped into the BMAP-matrices \mathbf{D}_k ($k = 0, 1, \dots$) according to $(\mathbf{D}_k)_{i,j} = \lambda_i p_{i,j}^{(k)}$ for $k = 0, 1, \dots$ with the exception that $(\mathbf{D}_0)_{i,i} = -\lambda_i$

in order to obtain a true CTMC generator $\mathbf{Q}_{\text{BMAP}} = \sum_{k=0}^{\infty} \mathbf{D}_k$. Consequently, matrix \mathbf{D}_k governs transitions that correspond to arrivals of batches of size k . All BMAP-matrices are of order $m_{\text{BMAP}} \times m_{\text{BMAP}}$, where

- \mathbf{D}_0 is a matrix with negative diagonal elements and nonnegative off-diagonal elements, and
- \mathbf{D}_k are nonnegative rate matrices ($k \geq 1$).

We require the infinitesimal generator \mathbf{Q}_{BMAP} to be irreducible and $\mathbf{Q}_{\text{BMAP}} \neq \mathbf{D}_0$ so that \mathbf{D}_0 is a nondegenerate, stable matrix, and as a consequence invertible.

Let $\boldsymbol{\pi}_{\text{BMAP}}$ be the stationary probability vector of the CTMC generator (i.e., $\boldsymbol{\pi}_{\text{BMAP}} \mathbf{Q}_{\text{BMAP}} = \mathbf{0}$, $\boldsymbol{\pi}_{\text{BMAP}} \mathbf{e} = 1$, where $\mathbf{0}$ and \mathbf{e} denote vectors of zeros and ones of the appropriate dimension).

Then, the mean rate of BMAP is computed as

$$\lambda_{\text{BMAP}} = \boldsymbol{\pi}_{\text{BMAP}} \sum_{k=1}^{\infty} k \mathbf{D}_k \mathbf{e} \quad . \quad (2.1)$$

Often, performance measures related to the inter-arrival times between *batches* are considered for BMAPs. The batch arrival rate and the squared coefficient of variation of the inter-batch arrival process X are given by

$$\lambda_{\text{batch}} = \boldsymbol{\pi}_{\text{BMAP}} (-\mathbf{D}_0) \mathbf{e} \quad , \quad (2.2)$$

$$c_{\text{batch}}^2 = \frac{E[X^2]}{(E[X])^2} - 1 = 2\lambda_{\text{batch}} \boldsymbol{\pi}_{\text{BMAP}} (-\mathbf{D}_0)^{-1} \mathbf{e} - 1 \quad . \quad (2.3)$$

The lag- k coefficients of autocorrelation ($k > 0$) of the (stationary) inter-batch arrival process are computed as [62] :

$$\begin{aligned} \text{ACF}(k) &= \frac{E[(X_0 - E[X])(X_k - E[X])]}{\text{Var}[X]} \\ &= \frac{\lambda_{\text{batch}} \boldsymbol{\pi}_{\text{BMAP}} ((-\mathbf{D}_0)^{-1} (\mathbf{Q}_{\text{BMAP}} - \mathbf{D}_0))^k (-\mathbf{D}_0)^{-1} \mathbf{e} - 1}{2\lambda_{\text{batch}} \boldsymbol{\pi}_{\text{BMAP}} (-\mathbf{D}_0)^{-1} \mathbf{e} - 1}, \end{aligned} \quad (2.4)$$

where X_0 and X_k denote two inter-batch times k lags apart.

2.2.1 Markovian Arrival Processes (MAPs)

A MAP is a special case of BMAP with the batch of size equal to one, i.e., $k = 1$. Correspondingly only two matrices \mathbf{D}_1 and \mathbf{D}_0 are used to describe a MAP, such that the matrix $\mathbf{Q}_{\text{MAP}} = \mathbf{D}_0 + \mathbf{D}_1$ is the irreducible infinitesimal generator of a CTMC, where \mathbf{D}_0 governs transitions that do not correspond to events, while \mathbf{D}_1 governs those transitions that do correspond to events.

Note that different from a BMAP, a MAP can provide a variety of precise performance measures of inter-arrival times directly from its matrix representation. Let $\boldsymbol{\pi}_{\text{MAP}}$ be the stationary probability vector of the CTMC generator (i.e., $\boldsymbol{\pi}_{\text{MAP}} \mathbf{Q}_{\text{MAP}} = \mathbf{0}$, $\boldsymbol{\pi}_{\text{MAP}} \mathbf{e} = 1$). The mean rate and the squared coefficient of variation of the MAP with inter-event time X are given by

$$\lambda_{\text{MAP}} = \boldsymbol{\pi}_{\text{MAP}} \mathbf{D}_1 \mathbf{e}, \quad (2.5)$$

$$c_{\text{MAP}}^2 = \frac{E[X^2]}{(E[X])^2} - 1 = 2\lambda_{\text{MAP}} \boldsymbol{\pi}_{\text{MAP}} (-\mathbf{D}_0)^{-1} \mathbf{e} - 1. \quad (2.6)$$

The autocorrelation function of a stationary MAP, i.e., the lag- k coefficients of autocorrela-

tion ($k > 0$), is computed as :

$$\begin{aligned} \text{ACF}(k) &= \frac{E[(X_0 - E[X])(X_k - E[X])]}{\text{Var}[X]} \\ &= \frac{\lambda_{\text{MAP}} \boldsymbol{\pi}_{\text{MAP}} ((-\mathbf{D}_0)^{-1} \mathbf{D}_1)^k (-\mathbf{D}_0)^{-1} \mathbf{e} - 1}{2\lambda_{\text{MAP}} \boldsymbol{\pi}_{\text{MAP}} (-\mathbf{D}_0)^{-1} \mathbf{e} - 1}, \end{aligned} \quad (2.7)$$

where X_0 and X_k denote two inter-event times k lags apart.

2.2.2 Markovian-Modulated Poisson Processes (MMPPs)

For many experiments in the later chapters, we use a 2-state Markovian-Modulated Poisson Process (MMPP), a special case of the Markovian Arrival Process (MAP) [42], to model auto-correlated flows because it is analytically tractable. The matrix representation of MMPP(2) is denoted by the following equations:

$$\begin{aligned} \mathbf{D}_0^{\text{MMPP}(2)} &= \begin{bmatrix} -(l_1 + v_1) & v_1 \\ v_2 & -(l_2 + v_2) \end{bmatrix}, \\ \mathbf{D}_1^{\text{MMPP}(2)} &= \begin{bmatrix} l_1 & 0 \\ 0 & l_2 \end{bmatrix}, \end{aligned} \quad (2.8)$$

where l_1 and l_2 are the mean rates of the two Markovian states, and v_1 and v_2 describe the mean rates of transitions between these two states, as illustrated in Figure 2.2.

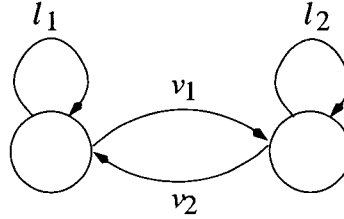


Figure 2.2: State transitions of MMPP(2).

MMPP models can be easily parameterized so that they show correlation. To match a pre-defined mean, squared coefficient of variation and $\text{ACF}(1)$ of a process, one may use Eqs.(2.5), (2.6), and (2.7). The two 2×2 matrices of the MMPP model, $\mathbf{D}_0^{\text{MMPP}(2)}$ and

$\mathbf{D}_1^{\text{MMPP}(2)}$, have four parameters, i.e., v_1 , v_2 , l_1 , and l_2 . Hence, this matching technique has one degree of freedom, and one can set l_1 as the free parameter to obtain a solution.

Other stochastic processes have been shown in the literature to capture burstiness and dependence, but their parameterization is not as easy for our purpose in this dissertation. By appropriate parameterization, one could create a MMPP that gives an independent process (no ACF) and a MMPP with dependence (ACF), while keeping the same mean and variance. This property is critical for examining the performance of systems with and without dependence because we can concentrate on the performance effects of ordering in two samples from the same distribution.

2.2.3 Hyperexponential Processes

Hyperexponential is another special case of the Markovian Arrival Process (MAP) [42]. A Hyperexponential is an independent process with high variance, which also makes it attractive in analytical models. The matrix representation of a two-order Hyperexponential process (H_2) is as the following equations:

$$\begin{aligned} \mathbf{D}_0^{H_2} &= \begin{bmatrix} -l_1 & 0 \\ 0 & -l_2 \end{bmatrix}, \\ \mathbf{D}_1^{H_2} &= \begin{bmatrix} l_1 p & l_1(1-p) \\ l_2 p & l_2(1-p) \end{bmatrix}, \end{aligned} \tag{2.9}$$

where l_1 and l_2 are the mean rates of services in the two Markovian states, and p describes the probability of re-entering the first Markovian state, as illustrated in Figure 2.3.

The H_2 models are easily parameterized as well. Similarly with that of MMPP(2), the moment matching approach can be used to get a H_2 matrix solution for a pre-defined mean and a squared coefficient of variation. Since its ACF is 0 for all lag values, one freedom variable is also needed here, which is often the probability p . In the following chapters H_2

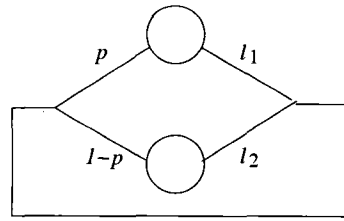


Figure 2.3: State transitions of H_2 .

processes are typically used to fit or model independent processes of high variability.

Chapter 3

Autocorrelation in Closed Systems

A system with finite buffers and/or admission control, which is indeed the case in a multi-tiered system as buffer sizes or the maximum number of simultaneous connections is restricted by software and/or hardware, behaves in essence like a closed system, i.e., a system with a closed loop structure [63]. Workload characterization studies of such systems usually examine the stochastic characteristics of arrivals to the system and wait/service times at various tiers aiming at bottleneck identification, diagnosing the conditions under which bottlenecks are triggered, and assisting the development of resource management policies to improve performance or provide service level provisioning [21, 38, 49, 18].

This chapter complements prior workload characterization studies of multi-tiered systems [8] by examining how burstiness in the arrival or service process of *any* of the tiers affects end-to-end performance. The burstiness is expressed by the *dependence structure* of the request flows across the various tiers of a closed loop, which is described and quantified via the *autocorrelation function* (ACF) as introduced in Chapter 2. In multi-tiered systems with a closed loop structure, if autocorrelation exists in the service process of *any* of the tiers, then it propagates across the *entire* loop in the closed system and is present in the

arrival stream of tiers (queues) that *precede* that tier (queue), unexpectedly affecting their performance as well as end-to-end performance.

The existence of autocorrelation flows in a multi-tiered system is confirmed via a case study based on the TPC-W benchmark. A simple queueing model of this system demonstrates that autocorrelated flows can originate from the stochastic behavior in the service processes of only one of the tiers and *captures* the system performance very well. This chapter also presents some discussion on how to take advantage of the knowledge of autocorrelation in flows for the development of admission control algorithms that consider dynamic system behavior.

This chapter is organized as follows. Section 3.1 overviews related work of workload characterization in multi-tiered systems. In Section 3.2 we describe a simple queueing network that is used to quantify the performance effects of autocorrelation in closed systems. Section 3.3 presents how autocorrelation propagates through the queueing network. Performance implications of the autocorrelation propagation are discussed in Section 3.4. In Section 3.5, we present an experimental study with the TPC-W benchmark that shows autocorrelation propagation a multi-tiered system and a simple queueing model that captures the benchmark's behavior. Section 3.6 presents an example of how one can use autocorrelation for system design. Finally, Section 3.7 summarizes the contributions of this chapter.

3.1 Related Work

Internet servers and services have evolved from centralized and information based only to distributed peer-to-peer, global storage, and grid-based services, personalized e-commerce, and sensor networks. Burstiness as expressed by self-similarity has been identified as a salient

characteristic of traffic in communication networks [45, 19, 67, 56, 20] that critically impacts their capacity and performance [45, 8]. It has also been shown to exist in computer systems including CPU utilization levels in a cluster of workstations [95], inter-arrival times at a large memory system with nonblocking caches [79], and file system activity [31]. Recent studies show that burstiness persists [92, 15], but emphasize that attention needs to be paid to the more complex and sophisticated nature of both Internet services and systems [6, 64, 66]. The latter is a direct result of the personalized nature of Internet-related services that need an array of resources, i.e., CPU, memory, and I/O, for serving requests. All the above works concur that burstiness results in unpredictability of system performance and argue for feedback-control frameworks to dynamically adapt resource allocation to changing service demands [18, 80].

Current workload characterization studies focus on bottleneck identification of multi-tiered systems [27]. Multi-tiered architectures that separate the database server from the web and application servers are used to improve performance, but introduce more points where bottlenecks may occur [5]. Furthermore, current Internet servers, such as e-commerce systems, have stricter requirements for quick response time, high security transactions, and persistent and reliable storage [89, 25, 47, 51] making the need of detailed workload knowledge even more pressing. Several recently published studies indicate that different e-commerce servers do share similar characteristics: arrivals are best characterized as bursty with high degrees of self-similarity, the most significant portion of requests are for dynamically generated objects, and the popularity distribution is Zipf-like [89, 50, 86, 6].

Data on actual e-commerce sites are difficult to obtain as they are subject to non-disclosure agreements. Consequently, one can only resort to synthetic workload generators to study such systems, the most prominent being the TPC-W benchmark for an e-commerce

site [83]. Studies based on TPC focus on bottleneck identification and find a variety of causes [27, 3, 90, 49]. Note that the TPC benchmarks implement stationary arrivals only, which gives a restricted system view: observing the system in steady state does not tell us how the systems behaves in transient overload.

All the workload characterization studies aim at guiding the development of admission control strategies that improve peak throughput [21, 38], prioritized scheduling at the database server to meet different service level agreements [49], and scheduling policies that minimize consistency overheads in clustered environments that support query caching and database replication [4]. However, none of the above characterization studies of multi-tiered systems has identified autocorrelation as an important system characteristic for performance.

Traditionally, models of multi-tiered systems focus on modeling the bottleneck tier [87, 70] or modeling all tiers using a single queue [38]. A closed-system model of a multi-tiered system that is based on Mean Value Analysis (MVA) and does consider all tiers has been proposed in [85]. In the method of layers, the system performance model is viewed as a sequence of layers, and each layer is solved separately [75, 78]. Aggregation of models of individual resource demands and interaction overheads for each tier in a multi-tiered system is also proposed to predict system throughput and response times [80]. The effect of autocorrelation in open systems has been examined in [22] where it was shown via simulation that long-range dependence in the arrival process of a single queue results in sharp performance degradation. Similar results are reported in [2] where the performance effects of short-range dependence versus long-range dependence in the arrival streams are examined.

In contrast to all of the above works, in this dissertation autocorrelation is proposed as a compact characterization of burstiness in multi-tiered systems. Our work further demonstrates that stochastic processes that capture autocorrelation can be used in surprisingly

simple models that can effectively capture performance trends of burstiness in complex systems.

3.2 Closed Tandem Queues

We use the simplest closed queuing system (see Figure 3.1) that resembles the topology of a multi-tiered application. The purpose of this analysis is twofold: (1) to observe how autocorrelation propagates through all tiers in the system and (2) to observe how autocorrelation affects system performance.

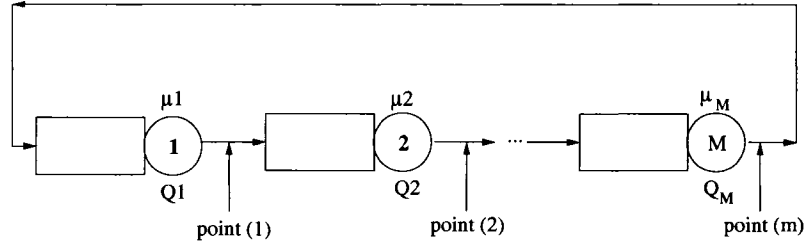


Figure 3.1: A closed system with M queues.

Autocorrelation in the arrival or service processes directly implies that the system is non-product-form [43], therefore one can only use simulation for its analysis. We stress that although we present performance results for closed systems, these results readily apply to open systems with admission control. For a sketch of the proof of equivalence of closed systems and open systems with admission control, see Appendix.

3.2.1 A 2-tier System

To establish a basic understanding of how autocorrelation affects system performance in a closed system, we consider the simplest case of the system depicted in Figure 3.1, i.e., a closed queuing network with two queues only, Q_1 and Q_2 , with mean service rates μ_1 and

μ_2 , respectively. We assume that a fixed number of jobs circulate in the queuing network. This number is also known as the *multiprogramming level* (MPL).

We assume that Q_2 is the bottleneck device and that Q_1 is twice as fast as Q_2 .¹ The source of any autocorrelation in the flows of this network is the service process of at least one of the queues. We use a 2-state Markovian-Modulated Poisson Process (MMPP), a special case of the Markovian Arrival Process (MAP) [42], as illustrated in Chapter 2, to model autocorrelated service times. The MMPP process allows to build two distinct sets of samples that share the same PDF but allow for different ordering in each set. Therefore, by appropriate parameterization, one could create an MMPP that gives an independent process (no ACF) and an MMPP with dependence (ACF), while keeping all moments identical, essentially maintaining the same PDF. We evaluate two scenarios.

Scenario 1: The service times of Q_1 are exponentially distributed with mean rate $\mu_1 = 2$, while the service times of Q_2 are drawn from a 2-state MMPP with mean service rate $\mu_2 = 1$ and squared coefficient of variation $SCV = 20$. The service process of the bottleneck device Q_2 has autocorrelation.

Scenario 2: The service time of Q_1 is autocorrelated and is drawn from a 2-state MMPP with $\mu_1 = 2$, $SCV = 20$. The service process of Q_2 is exponentially distributed with mean service rate $\mu_2 = 1$. Now, Q_1 , with ACF in its service process, is not the bottleneck.

In order to quantify the effect of autocorrelation on system performance, we also conduct experiments with different MMPPs for *Scenario 1* and *Scenario 2* such that we maintain the

¹Experiments with varying relative speed of the two devices yield qualitatively the same results as those reported here .

same mean, SCV, and higher moments in the service process, but we change its autocorrelation structure to be short-range dependent, long-range dependent, or independent (i.e., ACF equal to 0 in all lags). These experiments are labeled as SRD, LRD, or NOACF. We obtain a SRD process with ACF equal to 0.47 at lag=1 but decaying to 0 at lag=100, and a LRD process with ACF equal to 0.47 at lag=1 but flattening to 0.28 beyond lag=500. Table 3.1 summarizes the two scenarios. The matrix representations of these MMPPs are presented in Table 3.2. All simulations are done in a ten million sample space.

Table 3.1: Summary of the two scenarios.

	Q_1	Q_2 (Bottleneck)
<i>Scenario 1</i>	Exponential	MMPP (SRD, LRD or NOACF)
<i>Scenario 2</i>	MMPP (SRD, LRD or NOACF)	Exponential

3.3 Autocorrelation Propagation

First, we present how autocorrelation propagates through the queuing network by measuring the ACF of the departure process of queue Q_1 (i.e., at point 1 in Figure 3.1) and the ACF of the departure process from queue Q_2 (i.e., at point 2 in Figure 3.1). Note that the departure process of queue Q_1 becomes the arrival process to queue Q_2 . Similarly, the departure process of queue Q_2 becomes the arrival process to queue Q_1 .

Figure 3.2 illustrates the autocorrelation propagation for *Scenario 1* and *Scenario 2*, respectively, for $MPL = 25$. These results are representative of other MPL levels. The ACF of the 2-state MMPP that generates the service times at Q_2 and Q_1 for *Scenario 1*

Table 3.2: Summary of the MMPPs used in the two scenarios.

		D_0		D_1	
Scenario 1 (Q_2)	SRD	-12.048956	0.048956	12.000000	0.000000
		0.004079	-0.088048	0.000000	0.083969
	LRD	-12.000975	0.000975	12.000000	0.000000
		0.000081	-0.088001	0.000000	0.087920
	NOACF	-13.061404	1.061404	12.000000	0.000000
		0.096491	-0.096491	0.000000	0.000000
Scenario 2 (Q_1)	SRD	-12.041012	0.041012	12.000000	0.000000
		0.007571	-0.161417	0.000000	0.153846
	LRD	-12.000818	0.000818	12.000000	0.000000
		0.000150	-0.161292	0.000000	0.161142
	NOACF	-12.877193	0.877193	12.000000	0.000000
		0.175439	-0.175439	0.000000	0.000000

and *Scenario 2*, is also shown in the figure. In *Scenario 1* (see Figure 3.2(a)/(b)), ACF propagates through all tiers of the closed system with almost identical strength as the one at the service process of the bottleneck device, i.e., queue Q_2 , which injected autocorrelation into the system. In *Scenario 2* (see Figure 3.2(c)/(d)), autocorrelation propagates through the tiers, but with reduced strength compared to the autocorrelation of the service process that injected autocorrelation into the system (i.e., queue Q_1), which is not the bottleneck queue in the system.

The behavior observed in both experiments is explained by the general queuing theoretic observation that the departure process of a busy queue resembles its service process rather

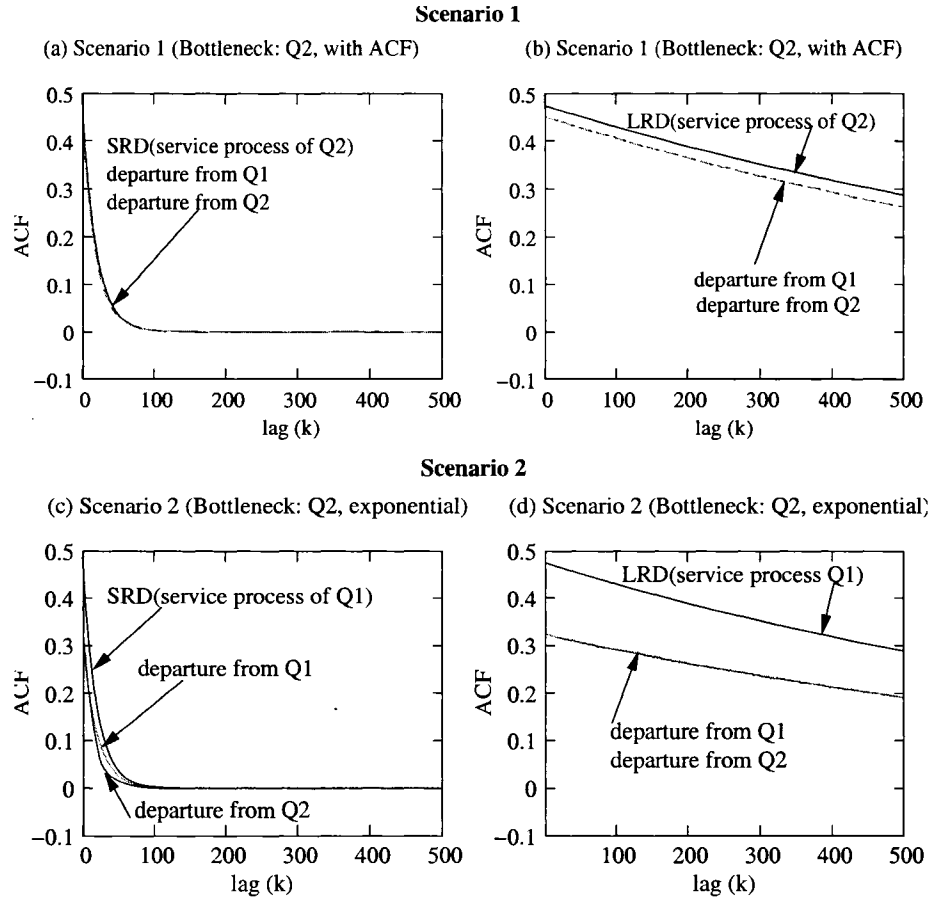


Figure 3.2: ACF of departures from Q_1 (arrivals to Q_2), departures from Q_2 (arrivals to Q_1) for both scenarios, and the ACF of the correlated service process. ACF of the MMPP service process in (a)/(c) is SRD and that in (b)/(d) is LRD.

than its arrival process. Instead, for a lightly loaded queue, its departure process resembles its arrival process [100]. Hence, for *Scenario 1*, the departure process from the heavily loaded queue Q_2 , and consecutively the arrival and departure processes of the lightly loaded queue Q_1 resemble the service process at Q_2 . Therefore, autocorrelation propagates with the same strength across all tiers. In *Scenario 2*, because the heavy loaded Q_2 has exponential service times, the autocorrelation injected in its arrivals from the service of queue Q_1 is reduced since departures from this server are spaced further apart, thanks to the exponential service times of Q_2 (see Figure 3.2(c)/(d)). Experiments with variable MPL levels show the same qualitative behavior for both of the above scenarios and for different autocorrelation

strengths. Our first observation is summarized as follows.

Observation 1 *Autocorrelation becomes present at all queues in a closed queuing network as long as it is exhibited in the service process of one queue.*

3.4 Performance Effects

We now turn to the effects of autocorrelation on system performance. We evaluate the mean response time (i.e., wait time plus service time), the mean queue length, and the mean utilization in each server. Furthermore, we also report on the mean round trip time (i.e., sum of all response times) that captures end-to-end system performance. In an effort to *quantify* the effect of SRD and LRD on system performance, we also conduct the same experiments as those described in *Scenario 1* and *Scenario 2* but with the MMPP process adjusted so that it does not have any autocorrelation but maintains the same mean, SCV, and higher moments.

Figure 3.3 shows performance under *Scenario 1* and Figure 3.4 shows performance under *Scenario 2*. The presence of autocorrelation (both SRD and LRD) in the closed system degrades the overall system performance – compare round trip times in Figure 3.3(a) and Figure 3.4(a). Looking closely into how round-trip time is distributed between the two servers, we notice that performance of the non-bottleneck queue significantly decreases as MPL increases. This is also reflected in Figures 3.3(b) and 3.4(b) that plot the average queue length in each server.

Although the overall performance effects on the non-bottleneck queue are the same in both scenarios, the reasons are different. In *Scenario 1*, performance degrades due to autocorrelated arrivals to the non-bottleneck queue – recall that there is no autocorrelation

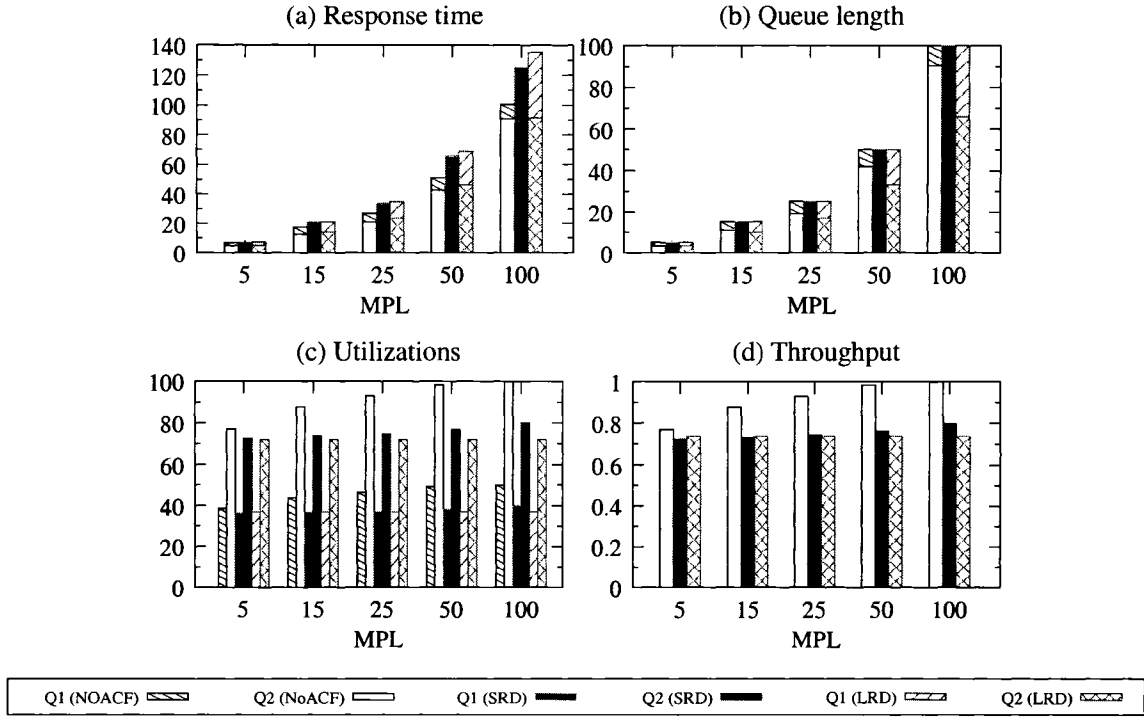


Figure 3.3: Performance measures: (a) mean round trip time, (b) mean queue length, (c) mean utilization, and (d) mean throughput at each queue for *Scenario 1*. The bottleneck queue (Q_2) has autocorrelated service times.

in its service process. In *Scenario 2*, autocorrelation in both arrival and service processes of the non-bottleneck queue degrades performance. Longer queues at the non-bottleneck server cause the mean queue length of the bottleneck queue to surprisingly decrease (see Figure 3.3(b) and Figure 3.4(b)) as requests spend now more time in Q_1 . This redistribution of requests in the system “balances” the load, as reflected in the per queue utilizations, which are now lower than when there is no autocorrelation, Comparing the systems with and without correlation, although the overall performance degrades in presence of correlation, the per queue utilizations decrease (see Figures 3.3(c) and 3.4(c)). The above performance trends persist and become slightly more pronounced in the LRD experiments, but are clearly not commensurate with the degree of dependence in the LRD process. This is due to the fixed MPL level, that in effect acts as an upper bound on the number of jobs that circulate

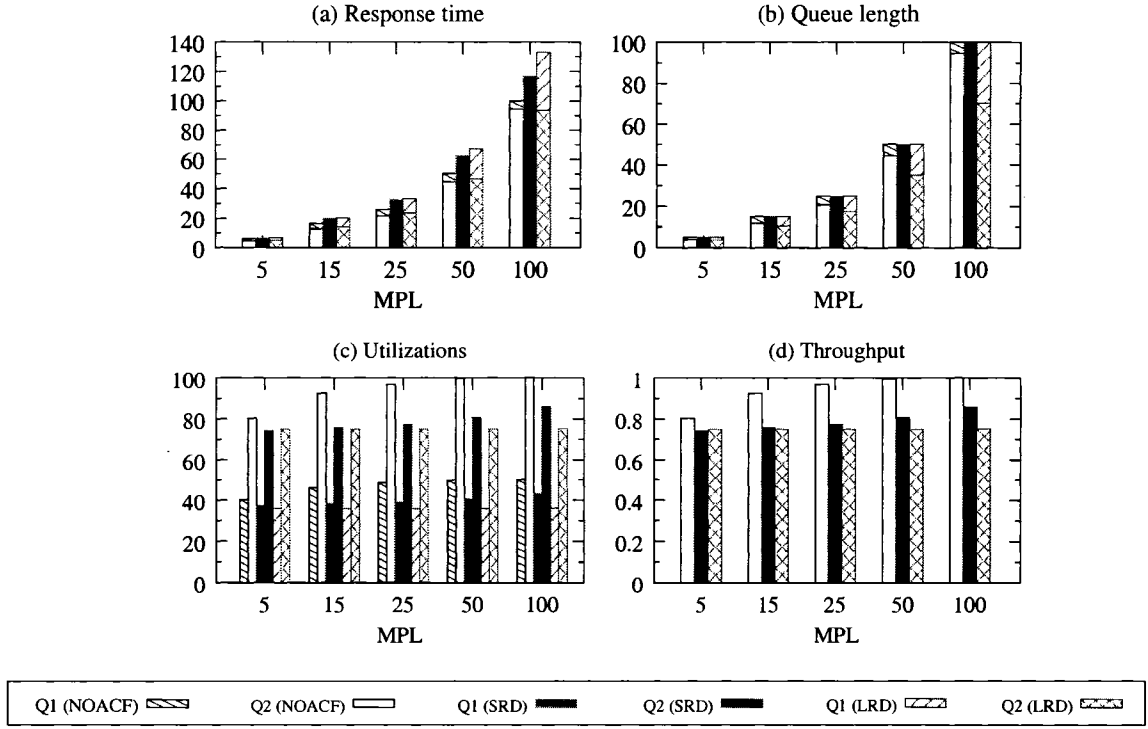


Figure 3.4: Performance measures: (a) mean round trip time, (b) mean queue length, (c) mean utilization, and (d) mean throughput at each queue for *Scenario 2*. The non-bottleneck queue (Q_1) has autocorrelated service times.

in the system at all times.

Figures 3.3(d) and 3.4(d) show the system throughput for the two scenarios as a function of MPL. Consistent with the utilization behavior, we see that the system with no autocorrelation reaches its maximum throughput at $MPL=100$, which is equal to the service rate of the bottleneck queue, and remains flat after that point. For the experiments with ACF, we see that throughput increases very slowly as MPL increases, consistent with the very slow growth of utilization at the bottleneck queue. Indeed, maximum throughput and the corresponding 99.5% utilization are reached with much higher $MPL=1500$ for the SRD experiments and with $MPL=70000$ for the LRD experiments! Therefore, throughput and utilization, metrics that are easily obtainable from measurements and are prevalently used to gauge system capacity, give a distorted view of the user-perceived performance. Our

observations are summarized as follows.

Observation 2 *Autocorrelated flows in a closed system degrade overall system performance, i.e., increase mean round trip time and decrease mean throughput. Counter-intuitively, they also decrease the anticipated utilization of each queue, including the anticipated utilization at the bottleneck device.*

Observation 3 *Autocorrelated flows in a closed system balance the load among all queues, i.e., decrease mean queue length and mean response time of the bottleneck queue and increase those of the non-bottleneck queue.*

These observations have an important effect on capacity planning. If autocorrelated flows exist in the system, then reduced utilization levels at a queue do not mean that the system can sustain more load.

To better understand where each job spends most of its time waiting, we plot in Figure 3.5 and Figure 3.6 the CDFs of response times (per queue and round-trip) for the LRD experiments.² For a substantial range of response times the performance of the system with LRD is better than that with no autocorrelation (see the cross-over points in CDFs in Figures 3.5(c) and 3.6(c) as well as cross-over points in CDFs that show per-server response times). However, response time tails at servers with ACF (Figure 3.5(b) and Figure 3.6(a)) dominate tails of round-trip times and significantly bias mean response times.

Observation 4 *In a closed system, the service process with autocorrelated structure (be it in the bottleneck queue or not) is the source of tails in the end-to-end response times and dominates average performance measures.*

²The CDF results of the SRD experiment are qualitatively the same.

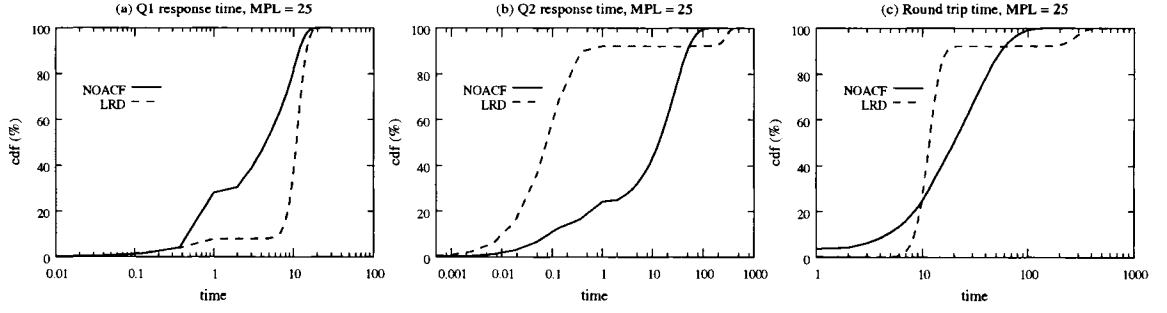


Figure 3.5: CDFs of (a) response time at Q_1 , (b) response time at Q_2 , and (c) round trip time for *Scenario 1* with MPL=25. The bottleneck queue (Q_2) has autocorrelated service times.

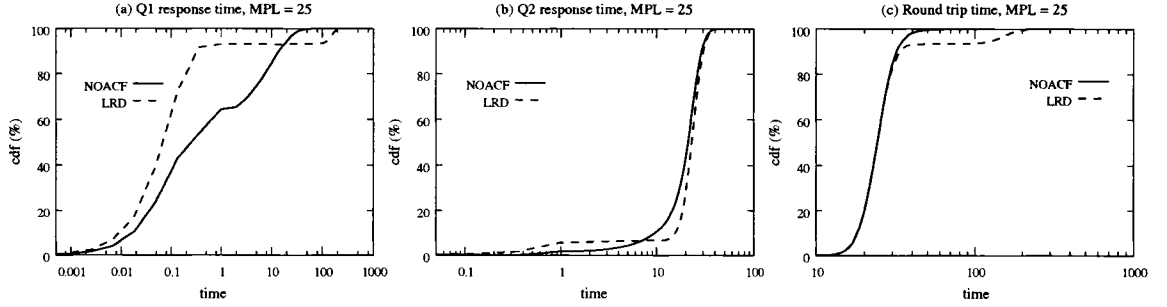


Figure 3.6: CDFs of (a) response time at Q_1 , (b) response time at Q_2 , and (c) round trip time for *Scenario 2* with MPL=25. The non-bottleneck queue (Q_1) has autocorrelated service times.

The immediate implication of the above observation is that capacity planning or admission control at a server with autocorrelation that aims at reducing the response time tail, may incur significant performance improvements. We will return to this point in Section 3.6.

3.5 Autocorrelation Analysis in TPC-W

In this section, we confirm the existence of autocorrelation flows in a multi-tiered system via a case study based on the TPC-W benchmark [83]. TPC-W is a widely used e-commerce benchmark that simulates a Business-to-Consumer (B2C) site [27].

3.5.1 Experimental Environment

We collect measurements of a multi-tiered e-commerce site implemented according to TPC-W, which simulates the operation of an on-line bookstore. A high-level overview of the experimental set-up is illustrated in Figure 3.7 and specifics of the software/hardware used are given in Table 3.3.

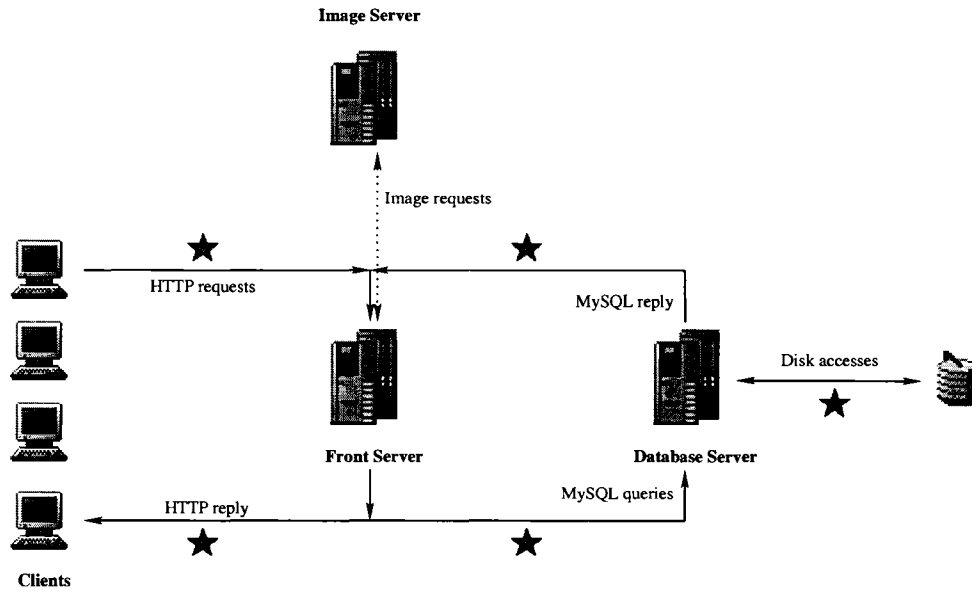


Figure 3.7: E-commerce experimental environment.

Table 3.3: Hardware components on the on-line bookstore implementation

	Processor	Memory	OS
Clients [68]	Pentium 4 / 2 GHz	256 MB	Redhat 9.0
Web Server - Apache2.0/Tomcat4.0 [82]	Pentium III / 1.3GHz	2 GB	Redhat 9.0
DB Server - MySQL4.0 [58]	Intel Xeon / 1.5 GHz	1GB / 768 MB	Redhat 9.0
Disk	SEAGATE: ST373453LC; SCSI; 73 GB; 15,000 rpm		

Because we focus on the activity across all tiers in the system, we collect measurements

at the front-end server (that hosts the web and application servers), the back-end database server, and the disk. In Figure 3.7, all measurement points are denoted with a star (*).

Trace data are collected via the following utilities:

- Arrivals at the front-end server are obtained by tracing the workload generation modules.
- CPU and memory activity at each server is measured via the *sysstat* Linux utility.
- Query activity at the database server is provided by MySQL logs.
- VMware [88] is used to run the database server in a Linux virtual machine hosted by the database server machine. This allows the physical SCSI disk to appear as a process in the database host. We use the *strace* Linux utility to trace all I/O activity.

The host of the database server has 1 GB of memory but the virtual machine uses only 768 MB. We also separate the image files at the front-end server in an effort to minimize their effect. Our experiments show that images, i.e., static content attached to each dynamically generated page, have negligible service times when compared to that of dynamic requests, thus their impact is not considered.

According to TCP-W specifications, the number of customers or emulated browsers (EBs) is kept constant throughout the experiment. For each EB, TCP-W statistically defines the user session length, the user think time, and the queries that are generated by the session. To better simulate the behavior of a real system, there is a time-out period (uniformly distributed between 5 and 15 minutes) that is associated with each EB. If a time-out occurs, then the session ends and a new session starts immediately. Four Pentium 4 machines are used to simulate the EBs. If there are n EBs in the system, each machine simulates $n/4$ EBs.

The database of the online store has 10 tables. One of the most important ones is the ITEM table which stores information on the items available for purchase. The database size is determined by the number of items and the number of customers. In our experiments, we found that the size of the ITEM table is critical for performance. Therefore, we present results on different databases that are distinguished by the size of the ITEM table. We run experiments in three databases: one with 10,000 items (small), one with 100,000 items (medium), one with 500,000 items (large), and one with 1,000,000 items (very large). Table 3.4 shows the size of the most important tables in the databases used in our experiments.

Table 3.4: Sizes of important tables

DB	ITEM	CUSTOMER	ORDER_LINE	Total
10 K	5.1 MB	362 MB	338 MB	1.5 GB
100 K	51 MB	362 MB	338 MB	1.5 GB
500 K	256 MB	362 MB	338 MB	1.9 GB
1 M	510 MB	362 MB	338 MB	2.1 GB

TPC-W defines 14 different Web interactions which are coarsely classified as either browsing or ordering. According to the weight of each type of activity in a given traffic mix, TPC-W defines 3 types of traffic mixes, namely, the *browsing mix* with 95% browsing and 5% ordering, the *shopping mix* with 80% browsing and 20% ordering, and the *ordering mix* with 50% browsing and 50% ordering.

3.5.1.1 Capacity Planning

The purpose of this section is to give an initial analysis of the system capacity and bottleneck that will be used in the analysis of Section 3.5.2. We will return to the detailed bottleneck analysis under various workloads using different stochastic metrics in Chapter 4.

Figure 3.8 illustrates system throughput (measured in interactions per second) at the front server, the front-end CPU utilization, and the database CPU and memory utilizations (i.e., the second tier) as a function of the number of emulated browsers in the system (i.e., system load). We do not report on the front-end memory utilization as it is always low. Results are presented for one TPC-W mix type, the browsing mix, and four database sizes. Overall, Figure 3.8 shows that it is the database server that becomes the bottleneck independently of the load in the system. In the following sections, we specifically focus on the browsing mix with a database of 10,000 items, where the system's throughput flattens out after the number of EBs greater than 256.

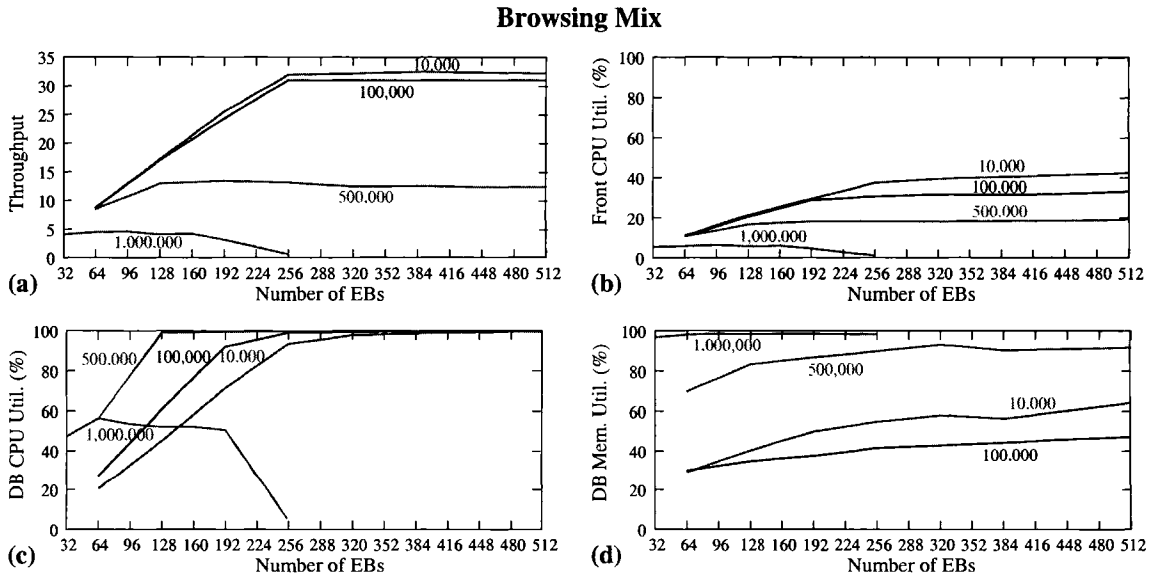


Figure 3.8: (a) Throughput, (b) front-end CPU utilization, (c) database CPU utilization, and (d) database memory utilization for the browsing mix and database sizes of 10,000, 100,000, 500,000, and 1,000,000 items.

3.5.2 Autocorrelation Propagation in TPC-W

Here, we report on the existence of autocorrelated flows in a multi-tiered system that is built according to the TPC-W specifications. Corresponding to Figure 3.7, Figure 3.9 also illustrates the flow of requests from the clients to the front server (which hosts the web and application servers) and the back-end database server. Data is collected at several points as

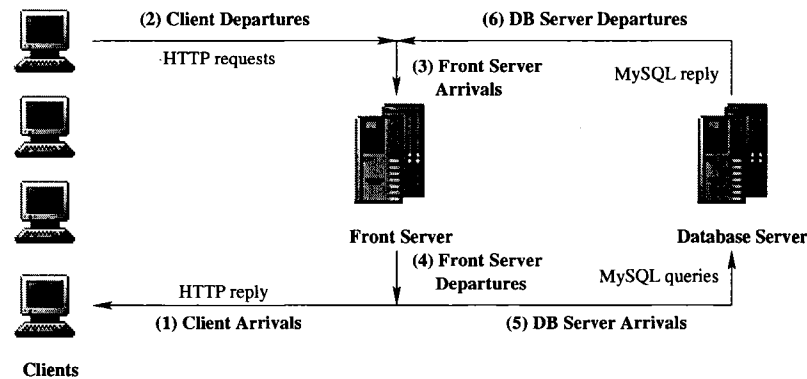


Figure 3.9: TPC-W experimental environment.

illustrated in Figure 3.9. Specifically, we record

- all responses sent from the front-end server to the client at point (1) labeled “client arrivals”, collected at the workload generation modules;
- all requests sent from the clients to the front-end server at point (2) labeled “client departures”, collected at the workload generation modules;
- all requests received by the front server (i.e., both client requests and database responses) at point (3) labeled “front server arrivals”, collected at the workload generation modules and MySQL logs;
- all responses sent from the front-end server (i.e., to both the clients and database queries) at point (4) labeled “front server departures”, collected at the workload generation modules and MySQL logs;

- all queries sent from the front server to the database server at point (5) labeled “DB server arrivals”, collected at the MySQL logs;
- all query results sent from the database to the front server at point (6) labeled “DB server departures”, collected at the MySQL logs.

Figure 3.10 shows the measured ACF at the various points indicated in Figure 3.9 for three browsing mix experiments and a database of 10,000 items for different numbers of concurrent EBs in the system (i.e., different workload intensities). Figure 3.11 plots the average queue lengths, average response times, and average CPU utilizations, at the clients, front-end, and database servers.

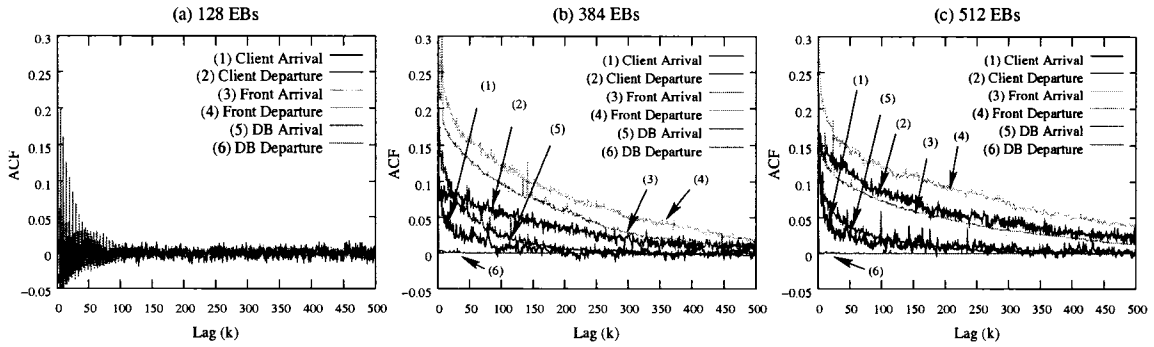


Figure 3.10: ACF at various points in the system. Experiments are done using the browsing mix, a database with 10,000 items, and (a) 128 EBs, (b) 384 EBs, and (c) 512 EBs.

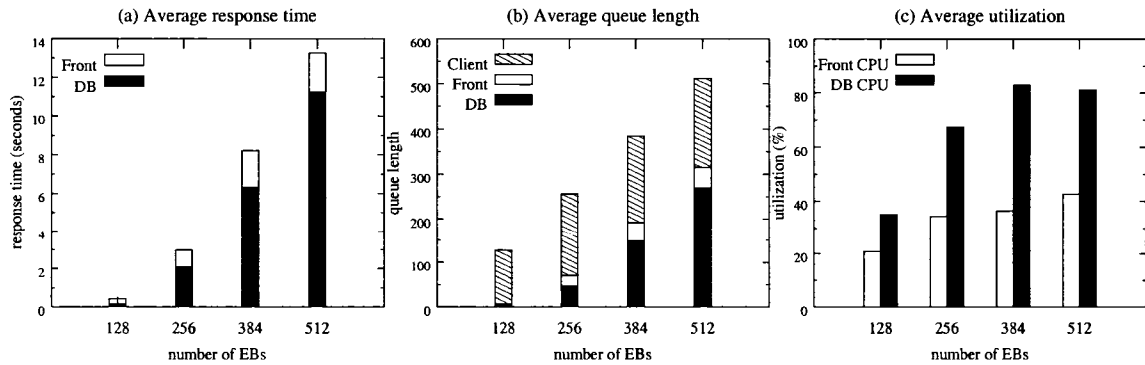


Figure 3.11: Average performance measures with the browsing mix.

With only 128 EBs, the system is lightly loaded, the front-end utilization is 20% and the database server utilization is 38%. The ACF in Figure 3.10(a) is very close to zero in almost all measurement points. The ACF at point (5), i.e., at the DB arrival process, is oscillating at low lags (from -0.05 to 0.2) and quickly decreases to nearly zero. The ACF of the database departure process, i.e., at point (6), follows the ACF of arrivals, consistently with discussion in Section 3.3, where we showed that under low load in a queuing system the arrival process rather than the service process determines the shape of the departure process. Looking carefully into the traces, we notice that the source of the correlated arrivals to the database comes from the JDBC drivers connecting Tomcat Java servlets and MySQL database server as one long query usually follows several small queries there.

As we increase the number of EBs in the system to 384, the system load increases to 38% utilization at the front-end and to 82% at the bottleneck DB server. This is a case of heavy load, where oscillating ACF values are not observed anymore. Figure 3.10(b) shows that autocorrelation is higher now in almost all measurement points, with the exception of points (1) and (6) which represent client arrivals and database departures, respectively. Experiments with 512 EBs capture very similar, although much stronger, trends on ACF propagation as depicted in Figure 3.10(c). Inspecting the ACFs at points (3) and (4) in Figures 3.10(b)-(c), we infer that the service process at the front-end server is correlated because ACF at its departure point is much higher than ACF in its arrivals. The autocorrelation measured at point (4) is the strongest among all measurement points, and becomes even stronger as load increases, see Figure 3.10(c). These ACF values suggest that there is no correlation in the DB service process as measurements at point (6) show a flat-to-zero ACF line – the service process in the DB “takes away” the correlation in the flow of arrivals.

Average performance values are presented in Figure 3.11. Despite the fact that queue lengths and response times increase fast as a function of EBs (see Figures 3.11(a) and (b)), utilization levels increase very slowly, consistently with the results presented in Section 3.4.

We have conducted several experiments using different TPC-W workload mixes (i.e., shopping and ordering) and larger database sizes (i.e., 100,000 and 1,000,000 items). These experiments can be summarized as follows. The amount of ACF that propagates through the system and measured at various points is different for the three TPC-W workload mixes. This is expected as each workload has different service demands. Autocorrelated flows are *not* always observed. In some experiments there is very little or no autocorrelation. For the cases that ACF is observed, we attribute its presence to autocorrelated service processes in the front and/or database servers because the workload generation at the EBs guarantees that there is no autocorrelation in the arrival process coming from the clients. In the following section, we present a simple model that captures the performance trends observed here and that confirms our conjecture about the existence of autocorrelation in the service process.

3.5.3 TPC-W Model

The model is illustrated in Figure 3.12. Queues Q_1 and Q_2 correspond to the front-end server and the back-end database server, respectively. Because the TPC-W benchmark is session-based, we use an *infinite-server queue* Q_0 with as many servers as the system's MPL to emulate client activity. The collected TPC-W trace data shows that each dynamic request at the application server generates several database requests. We capture this behavior by adding a feedback loop: with probability p a completed request from Q_1 is forwarded to queue Q_2 and with probability $1 - p$ it goes back to the client, in Q_0 . We also define the same six measurement points as in the real system of Figure 3.9.

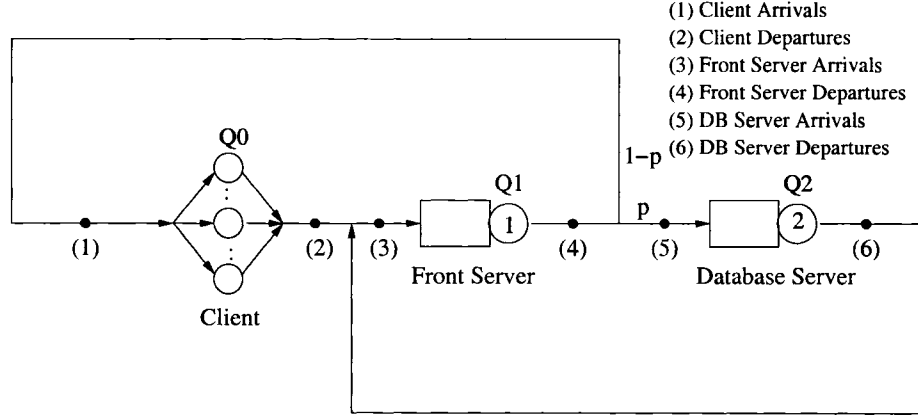


Figure 3.12: A queuing model of TPC-W.

This model is solved using simulation and is parameterized using measurements from a lightly loaded system, where there is virtually no queuing. Measuring the service process in each of the system tiers is not straightforward but in a lightly loaded system that guarantees nearly zero wait times, response times gives a good approximation of service times

- The *think time* in each server of Q_0 is exponentially distributed with mean 7sec, as specified by TPC-W.
- We use a 2-state MMPP to generate service times in the front-end server with mean $\mu_1 = 582.70$ and $SCV_1 = 20$. This MMPP has autocorrelation which is equal to 0.47 at lag 1 and decays to nearly zero at lag 300. The MMPP used here has the following MAP parameterization:

$$\begin{aligned} D_0^{(S)} &= \begin{bmatrix} -2001.004655 & 1.00465 \\ 0.384642 & -40.457034 \end{bmatrix}, \\ D_1^{(S)} &= \begin{bmatrix} 2000 & 0 \\ 0 & 40.072392 \end{bmatrix}. \end{aligned} \quad (3.1)$$

Note that we do not perform a rigorous fitting to capture the exact shape of autocorrelation in the service process, we simply match the first two moments of the empirical data and we adjust the MMPP parameters in order to induce autocorrelation.

- Service times at the database server are generated using a 2-stage hyperexponential with $\mu_2 = 224.34$ and $SCV_2 = 100$. This distribution in MAP presentation is as follows:

$$\begin{aligned} D_0^{(S)} &= \begin{bmatrix} -1000 & 0 \\ 0 & -3.461149 \end{bmatrix}, \\ D_1^{(S)} &= \begin{bmatrix} 987.991444 & 12.008556 \\ 3.419586 & 0.041563 \end{bmatrix}. \end{aligned} \quad (3.2)$$

- The probability p is set to 0.876, which is again obtained by measurements.

Figures 3.13(a)-(c) show the ACF propagation with MPL set to 128, 384, and 512, respectively. The queuing model captures well the autocorrelation trends observed in the TPC-W experiments, compared to Figure 3.10. Consistently with experimental results, the departure intervals from the front-end server at (4) and the arrival intervals at the database server at (5) have the highest autocorrelation. The slowest decaying ACF is at point (2), i.e., the departures from the clients. Note that the independent service process at the database server results in independent departures at (6), which also minimally affects arrivals to the clients at (1). Figure 3.13 verifies our speculation about the existence of autocorrelation in the service process of the front-end server. Existence of autocorrelation only at the service process of Q_1 (i.e., front-end server) causes the entire system to operate under almost independent flows when the load is low (Figure 3.13(a)) because the workload generation at the clients (which is driven by the exponential distribution) dominates the departure and arrival processes at all the queues. When the load is high (Figures 3.13(b)-(c)), the autocorrelated service process at Q_1 (i.e., front-end server) dominates the departure process at Q_1 and as showed in Section 3.3 propagates in the entire closed system.

Average performance measures from the model are presented in Figure 3.14 where performance measures of the TPC-W experiments are also presented for easy comparison. The dependent service process in Q_1 significantly affects system performance, especially round-

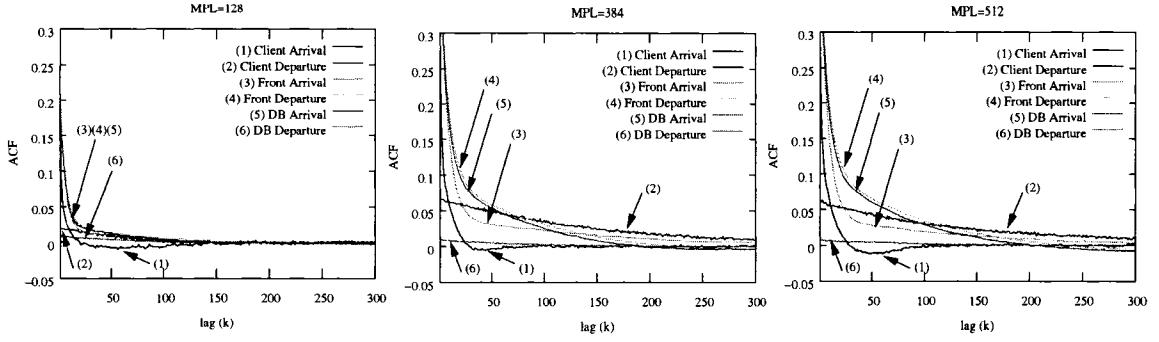


Figure 3.13: Autocorrelation propagation in our queuing model parameterized using the measurements of Section 3.5.2 with *MPL* equal to (a) 128, (b) 384, and (c) 512.

trip times. Model and experimental results are in excellent agreement, despite the fact that no rigorous fitting method was used to fit the experimentally collected service times and their autocorrelation of the front-end server to the 2-stage MMPP that was used in the model.

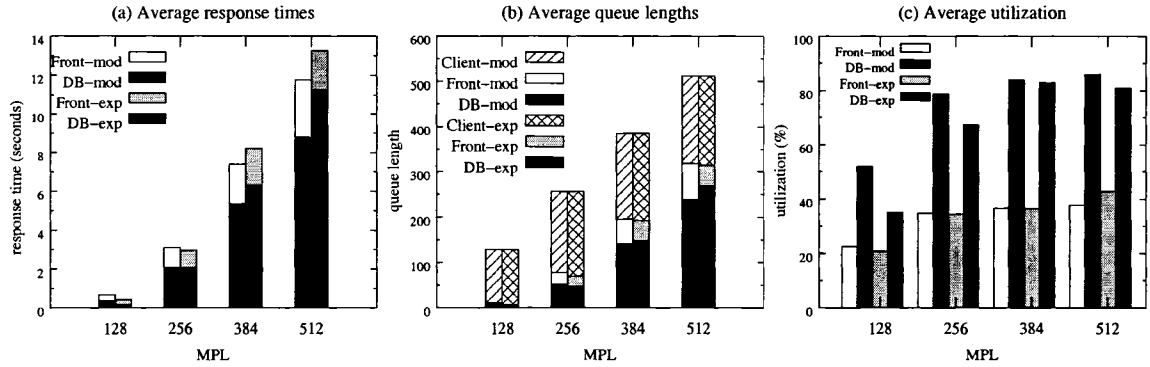


Figure 3.14: Model prediction and experimental performance measures.

3.6 Taking Advantage of ACF

After having established the importance of autocorrelated flows for the performance of multi-tiered systems, we now turn to how to use this information for effective system design. In general, capturing burstiness in the flows of complex systems can be used to implicitly model caching, context switching overhead, contention for memory or locks, while keeping the model surprisingly simple. Here, we present a case study that illustrates how ACF

can guide admission control. Naturally, a myriad of policies exist for admission control. Presenting an ideal admission control policy is outside the scope of this work. Instead, we focus on how to use knowledge of autocorrelated flows to improve policy development.

In Section 3.4 we show that the server with ACF in its service process is the one that most contributes to the response time tails. Based on this observation, we devise a simple admission control strategy that rejects the jobs which are highly probable to contribute to the long tail of round trip times. Identification of these jobs is based on the temporal locality of autocorrelated flows.

Assuming that we know a priori which is the tier that is the source of autocorrelation, we deploy admission control at that tier.³ Admission control is triggered when the queue length at the ACF tier reaches a pre-defined threshold Q_T of MPL. Upon each job completion, the current queue length is checked to see whether it exceeds threshold Q_T . If this is the case, then the request at the head of the waiting queue is dropped (i.e., directed back to Q_0 , the client queue), with probability weighed by the ACF value of the stream at $\text{lag}(1)$, provided that ACF has a positive value. Then, the next waiting request is also dropped with a probability weighed by the ACF value at $\text{lag}(2)$. The dropping of waiting requests stops when the queue length reaches Q_T or a job is admitted for service.⁴

We use the model in Section 3.5 to evaluate this admission control policy. The base line for the evaluation is the case with no admission control. For comparison, we also evaluate a policy with random dropping at the same tier. Similar to the ACF-guided policy, the

³Even if the autocorrelation function of the flow in the tier is not known a priori, it is possible to calculate it on-line using a modified version of Welford's one-pass algorithm to calculate the mean and variation of a sample [91].

⁴We experimented with different probability weights that gave us qualitatively similar performance. Here, we present results with weight equal to 1, i.e., we use directly the ACF function as the dropping probability.

random policy drops always from the head of the waiting queue with probability equal to the overall dropping rate of the ACF-guided admission control policy. This way, we maintain equal dropping rates in both admission control policies.

We first evaluate exactly the same setting as in Section 3.5, i.e., the front server has ACF in its service process that starts at 0.47 for lag 1 and decays to nearly zero beyond lag 300. MPL is set to 512. Q_T is defined as 60% of MPL because the front tier is not the bottleneck and it is expected to be less loaded than the DB tier. Consistently with experiments presented in Section 3.5, we assume that the service process at the DB tier is not correlated and that the DB is the bottleneck. The dropping rate for the ACF-guided policy is 8.2% and average round trip times become 8.93 seconds. The round trip times under the no-dropping and random dropping scenarios are 11.55 and 10.07 seconds, respectively. To focus on tail performance, Figure 3.15 illustrates the complementary cumulative distribution function (CCDF) of round trip times and of response times at the front server and the database server. The figure shows that ACF-guided policy improves the tail of the front server response times and respectively round trip times, given that the tails of response times at the DB server of the two admission control policies are almost identical.

We now use the model of Section 3.5 but assume that the bottleneck tier (i.e., the database server) has ACF in its service times. This is motivated by the disk ACF service times in Figure 2.1. Again MPL is 512 but Q_T is now set to 90% of the MPL, because the DB is the bottleneck and we expect most of the jobs to be stuck there. The ACF-guided admission control drops only 5.8% of the total requests and achieves an average round trip of 4.50 seconds. Round trip times with no-dropping and random dropping are 16.65 and 16.63 seconds, respectively. Both round trip times and database server response times significantly improve with the ACF-guided policy (see the CCDFs in Figures 3.16). With

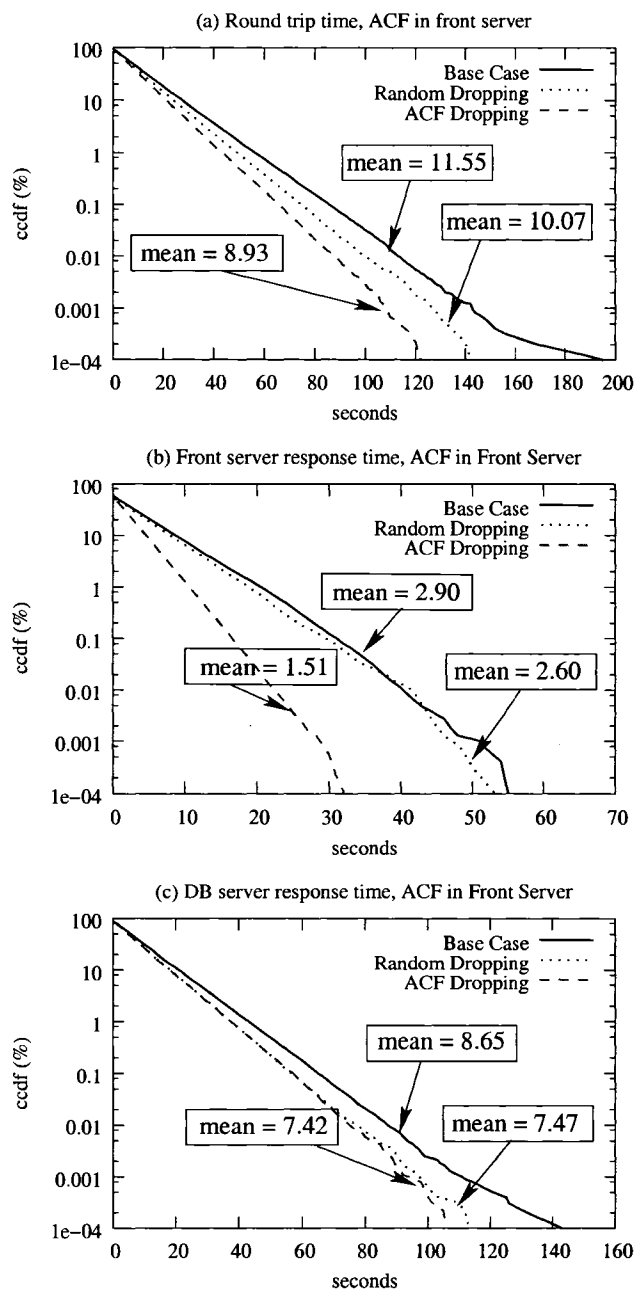


Figure 3.15: CCDFs of (a) round trip time, (b) response time of front server, (c) response time of database server using the model of Section 3.5 where the front server has ACF in its service process. In all experiments MPL is equal to 512.

random dropping, improvements are very small. Both experiments, although preliminary, confirm that selective dropping as guided by ACF can dramatically improve performance. By selectively dropping those requests that contribute most to ACF, the queue lengths in the

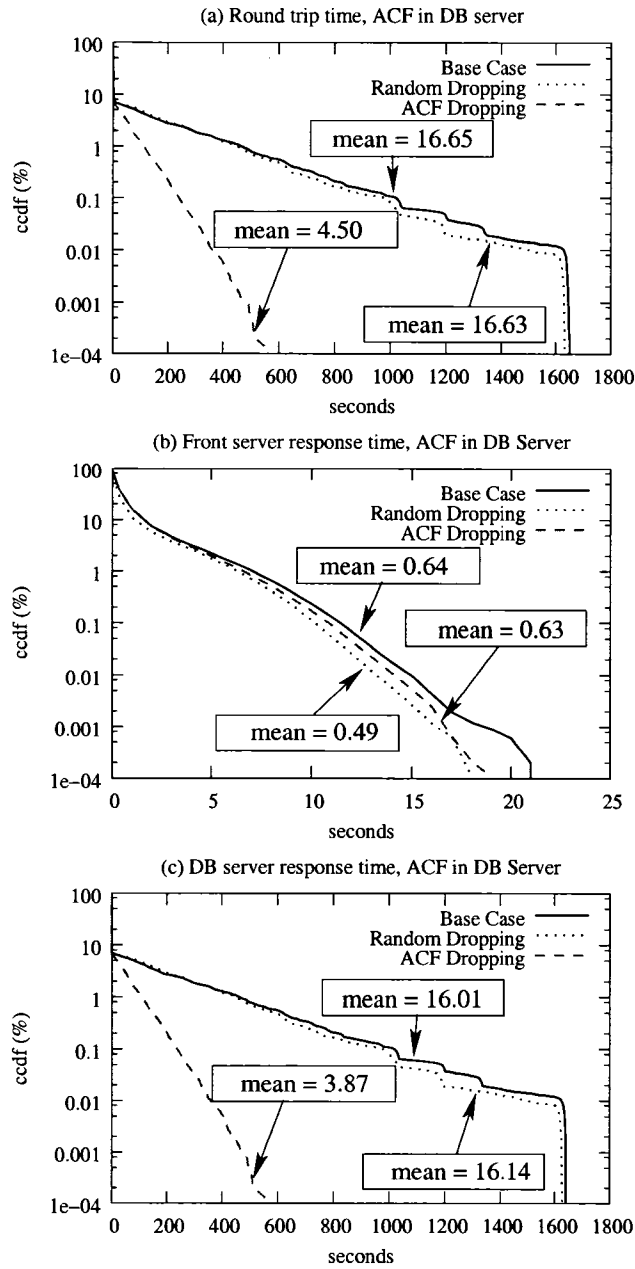


Figure 3.16: CCDFs of (a) round trip time, (b) response time of front server, (c) response time of database server when the database server has ACF in its service process. In all experiments MPL is equal to 512.

queue with autocorrelation significantly reduce, the ACF flows in the entire system weaken, and performance in every server improves.

Similar analysis can be done for capacity planning studies. There, the focus should be on

first identifying and then bolstering the server that is the source of autocorrelation, which is not necessarily the system bottleneck.

3.7 Summary

We presented a study that shows the presence of autocorrelated flows in a multi-tiered system with a closed-loop structure and their performance effects. Comparing the performance effects of the presence of autocorrelated flows in multi-tiered systems with the performance of the same system with independent flows, we show that end-to-end performance significantly deteriorates while bottleneck devices are less utilized, falsely indicating that the system is able to sustain higher load. Furthermore, we show that in contrast to systems where no burstiness is observed, the tails of the overall response time distributions do not necessarily reflect the time spent at the bottleneck tier, but instead are shaped by the response time tail at the tier that is the source of autocorrelation, irrespective of its utilization level. If autocorrelated flows are ignored, then throughput and utilization of specific devices - metrics often used in capacity planning and admission control - may give a distorted view of system load.

The measurements from a real system based the TPC-W benchmark demonstrate the existence of autocorrelation flows in a closed multi-tiered system. A queueing model is used to model this system by representing the service times of the guess-dependent server with an autocorrelated 2-state MMPP. Model and experimental results are in excellent agreement, despite the fact that no rigorous fitting method was used to fit the measured results, which proves that stochastic processes that capture autocorrelation may be used to effectively model complex systems via simple queueing models.

This chapter also presents a case study of admission control guided by the knowledge of the autocorrelation information. The case study illustrates that dependence in flows within the system is critical for effective admission control, thus should be the focus of the resource management strategies.

Chapter 4

Bottleneck Analysis in TPC-W

In addition to the significant impact of autocorrelation flows on a multi-tiered system, the mostly dynamically generated contents in contemporary servers also make it very challenging to understand the resource requirements of dynamic requests. It is possible that a request can cause a substantial portion of the database to be accessed even when just a few kilobytes of text are eventually sent back to the client. The “size” of a request is not a simple one-dimensional property, and thus is difficult to quantify a priori. For a dynamic request, the amount of data accessed from storage, the computational requirements for processing in order to generate the page content, and the amount of data sent back to the client are unrelated and unpredictable. Furthermore, wide disparities in the various resource requirements of different dynamic requests trigger multiple bottlenecks in the system. Resource allocation is further complicated by bursty user request rates that fluctuate dramatically even within short periods of time, resulting in systems that operate often under conditions of transient overload.

Effective system provisioning in such complex systems requires a detailed understanding of the system workload. While over provisioning helps Internet systems to operate under

light to medium load conditions, it does not preclude the fact that they are still susceptible to transient overload. Overload can be controlled via admission control mechanisms either at the front-end of the system (usually the network link) or at the application level, with the intention of stabilizing performance [15, 57], or via effective resource management that focuses on graceful performance degradation [92]. Performance degradation is by far more desirable than service unavailability. The latter, even if it occurs for a short time period only, can be detrimental for business profitability [52].

The purpose of this chapter is to present a detailed analysis of the resource demands in a typical e-commerce server under *steady load* and under *transient load*, to identify how the workload propagates through all system tiers, and to determine the conditions under which bottlenecks occur. We still use the typical configurations of an e-commerce site following TPC-W as in Chapter 3.5, which consists of a web and application server, a database server, and a storage system (see Figure 3.7).

We are particularly interested in how the workload's transient characteristics propagates through the system hierarchy and places resource demands on CPU, memory, and I/O devices. Our experiments show that the lower tiers of the system, i.e., the database server and the storage system, become the bottleneck while the resources at the front-end remain underutilized. The system capacity, directly related with the existence of bottlenecks in the system, depends not only on the load, but also on the characteristics of the system workload. Some workloads utilize system resources much faster than others, for the same number of users in the system. These workloads are more I/O bound than others.

This chapter is organized as follows. In Section 4.1 and 4.2 we present performance results that can be used for capacity planning, focusing on response time analysis for bottleneck identification. In Section 4.3 we show how overload develops and propagates in the system

under transient conditions that are distinguished by sudden changes in the customer arrival intensity and/or changes in the requested work. System implications of the presented analysis are presented in Section 4.4. Finally Section 4.5 summarizes the contributions of this chapter.

4.1 Capacity Planning

In a multi-tiered system, the load of each tier has an impact on the user-perceived performance. The personalized nature of the requests sent to e-commerce servers makes the second tier, i.e., the database server, the bottleneck [49]. Our experiments show that this bottleneck is triggered either by excessive *load*, as reflected by the number of simultaneous requests in the system, or by excessive *work*, as reflected by the amount of required system resources to service the requests.

Initially, we illustrate how load and work affect user perceived performance and consequently capacity planning. The system performance under different loads has been reported in Section 3.5.1.1 (see Figure 3.8). Results are presented for one TPC-W mix type, the browsing mix, and four database sizes.

Figure 4.1 illustrates the effect of different TPC-W mixes. In this case, we experiment with only one database, i.e., the one with 500K items, and report on the system throughput, front-end CPU utilization, and database CPU and memory utilizations for the ordering, shopping, and browsing mixes. Figure 4.1 indicates that mixes affect system performance and resources availability. Observe that different mixes determine different levels of the sustainable system load. Overall, Figures 3.8 and 4.1 show that it is the database server that becomes the bottleneck independently of the load or the work in the system. In the following, we focus on the database server performance and the characteristics of dynamic

page generation.

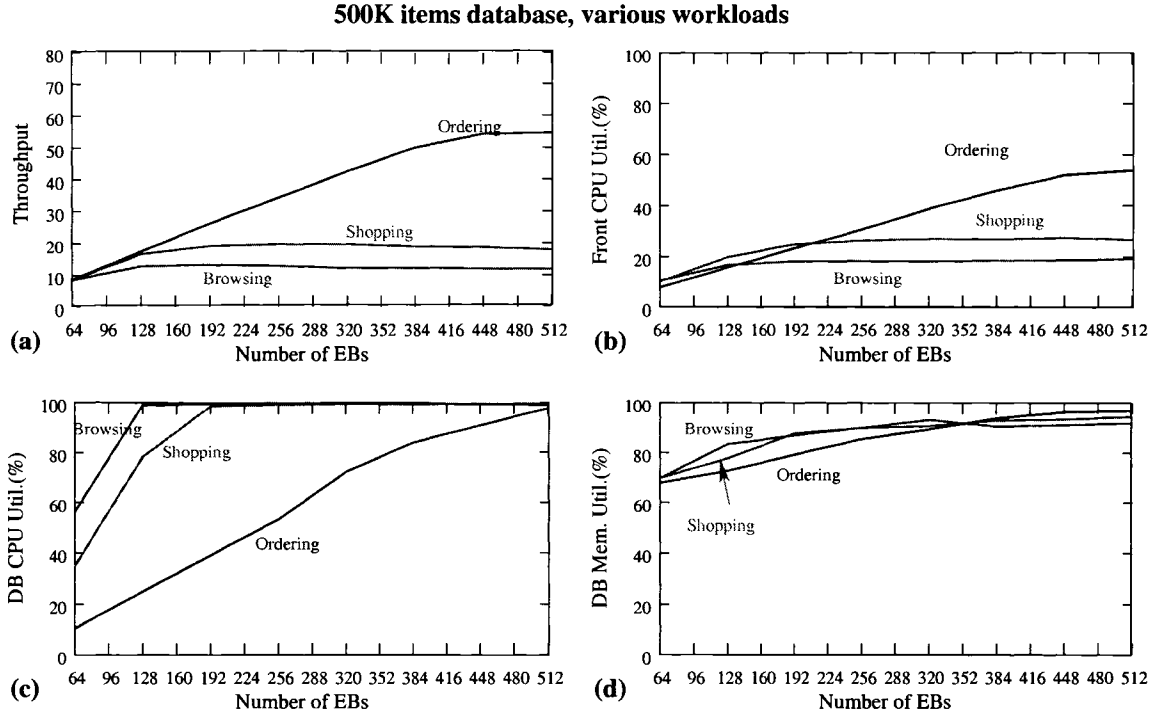


Figure 4.1: (a) Throughput, (b) front-end CPU utilization, (c) database CPU utilization, and (d) database memory utilization as a function of system load for the database with 500,000 items and various TPC-W mixes.

Figure 4.2 illustrates query time distributions, as a metric of the database server performance. Because we focus on understanding the system behavior under a variety of workloads, we introduce two additional traffic mixes that stress the system further by increasing the I/O traffic:

- *modified browsing mix*: the percentage of new product searches accounts for 90% of all requests and the remaining of the requests proportionally adjusted according to the browsing mix, while in the original browsing mix new product searches are only 11% of requests, and
- *modified ordering*: where we raise the portion of administration interactions of the ordering mix from originally 0.11% to 10%, and adjust the percentage of the rest of

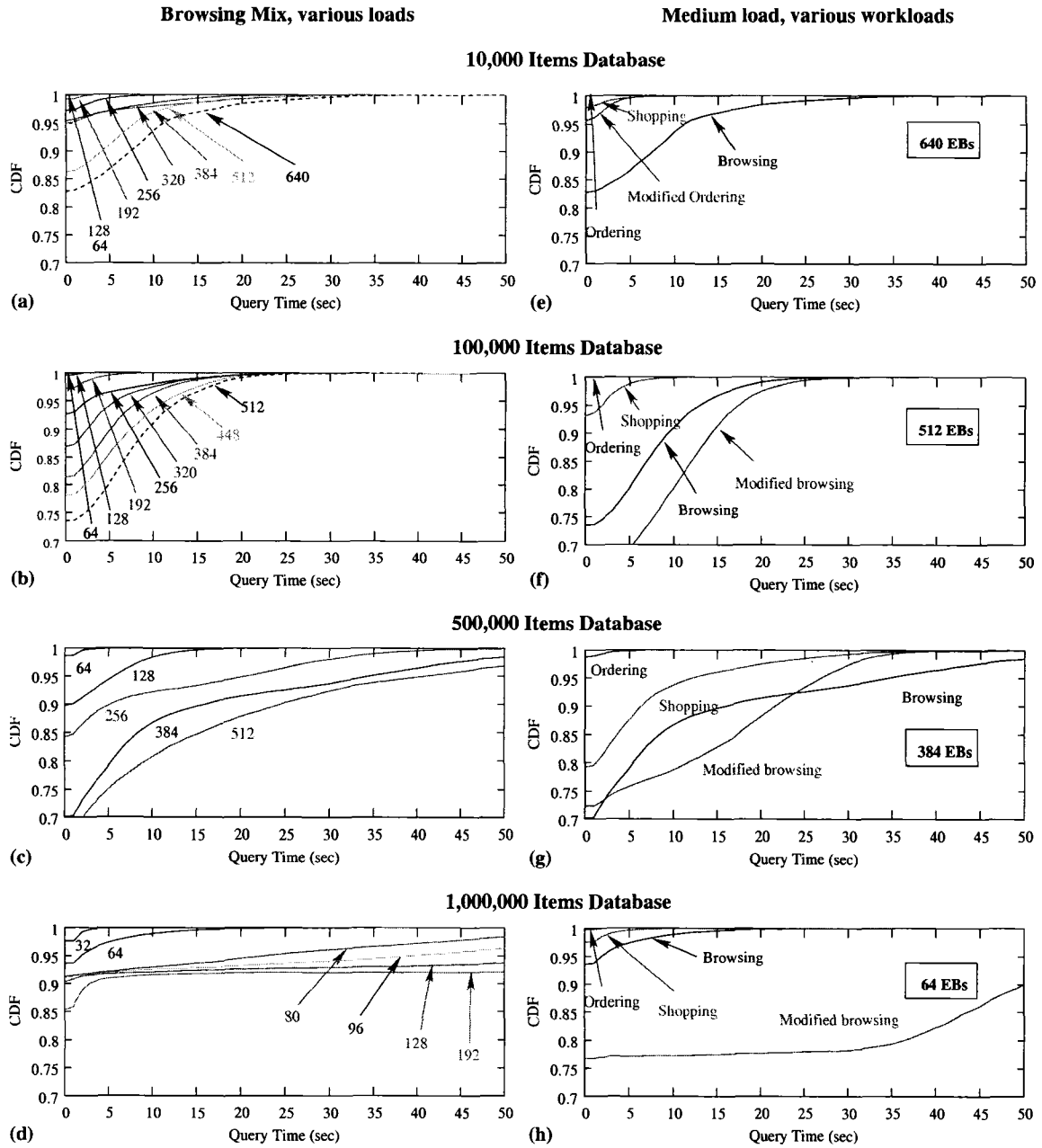


Figure 4.2: Query time distribution (CDF) for the browsing mix under various loads (left column) and under medium load for various TPC-W mixes (right column).

the requests in the ordering mix.

Since the database server is the bottleneck, this metric directly relates to the user perceived performance. We focus on the query time distribution for various database sizes, loads, and mixes. For each database size, which essentially determines the level of system

resources availability, higher load degrades database server performance. For the two large databases, where the system resources are limited, the impact of higher load is more apparent, resulting in distributions with considerably longer tails (see Figure 4.2(c)-(d)). For various TPC-W mixes (see the right column of Figure 4.2), the query time distribution for work-intensive workloads, such as the modified browsing mix, is quite different from that of less work-intensive workloads, such as shopping or ordering mix. We identify a workload as work-intensive when it requires more CPU, memory, and I/O to generate the response to a dynamic request.

4.2 System Implications of the Stationary Workloads

To better understand the behavior of the service process, we take a close look at the database behavior using the modified browsing mix (see Figure 4.3). The number of EBs is set to 32, which results in the significant arrival rate at the database disk (see Figure 4.3(a)). Note that in the same experiment but with the original browsing mix the arrival rate to the disk is at most 50. Figure 4.3(b) reports on the disk access pattern as a function of time. The entire disk is mapped on the y-axis which also marks the physical layout of each database table on the disk. The figure clearly shows that I/O accesses are bursty. Figure 4.3(c) reports on the memory utilization of the database server and shows that memory is periodically freed because of memory pressure. Intuitively, if there is memory pressure at the database, a memory miss suggests that another memory miss will soon occur, or if the disk is accessed then more accesses to the disk are to follow, which suggests that the service process is correlated. Such bulks of disk I/O operations result in heavier tail of the query-time distribution (see Figure 4.2(h)), which is attributed to the slower service rates at the disk and the FCFS-like

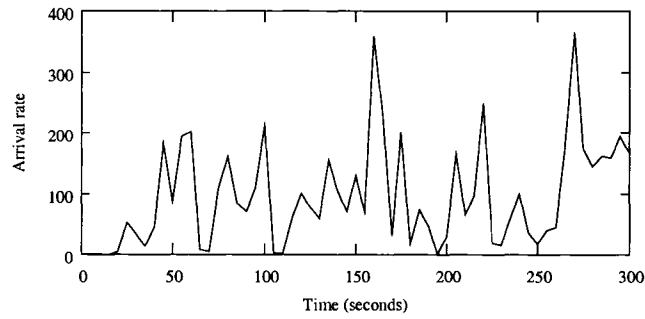
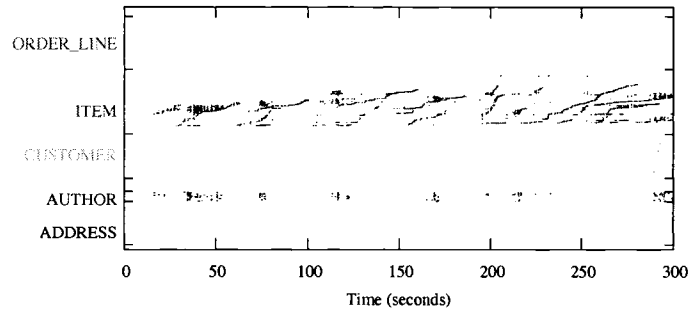
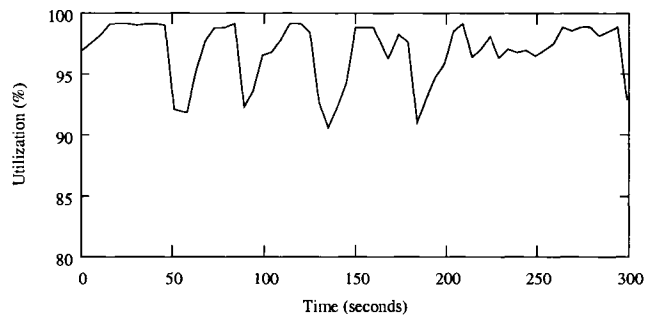
(a) Arrival rate in database disk**(b) Disk access positions****(d) Database server memory utilization**

Figure 4.3: Disk access pattern for the 1,000,000 items database. The system is under 90% new searches mix.

scheduling discipline. On the other hand, the burstiness of the disk accesses indicates that the system does not consistently suffer from memory misses. During the less I/O intensive period, CPU/memory performance, which employs PS scheduling discipline, dominates the response time.

4.3 Transient Workload Propagation

After having understood what the system bottlenecks are under steady state conditions, we now turn to transient analysis as systems in the real world are rarely subject to stability in their workloads. For the specific case of the e-commerce server, a popular new product or a seasonal sale could lead to sudden increases in arrival intensity. Different arrival patterns may be experienced during day or night times. Furthermore, the pattern of user requests, i.e., the information they request may be substantially different from the expected one, affecting the demands put on system resources. First, we concentrate on how ephemeral changes in the arrival intensity (i.e., load) propagate through the system tiers and affect system performance. Then, we focus on how changes in the type of requests (i.e., work) affect the bottleneck resource and how these patterns are inherited into lower system levels. For all experiments, we present activity in all three tiers across time. All experiments are run for 60 minutes. Results of the first 20 minutes are ignored to mask out warm up effects.

Experiment One (10K Database, Transient Load): The first experiment uses the browsing mix of TPC-W but changes the number of active browsers in a controlled manner. For the experiment that uses the small database (see left column of Figure 4.4), for the first 600 seconds the number of browsers is set to 64, for the next 300 seconds it is set to 384 to induce a short-lived overload condition, and for the remaining 25 minutes of the experiment the number of EBs is reset to 64. Figures 4.4(I.a)–(I.c) show the intensity of arrivals across time at the front-end server, database server, and the database disk, respectively. Note the flux of arrivals from one server to the other, as well as the time where significant increases/decreases are shown. The transient load causes a severe increase in the arrival intensity at all tiers except the disk, where short arrival bursts are detected over time with

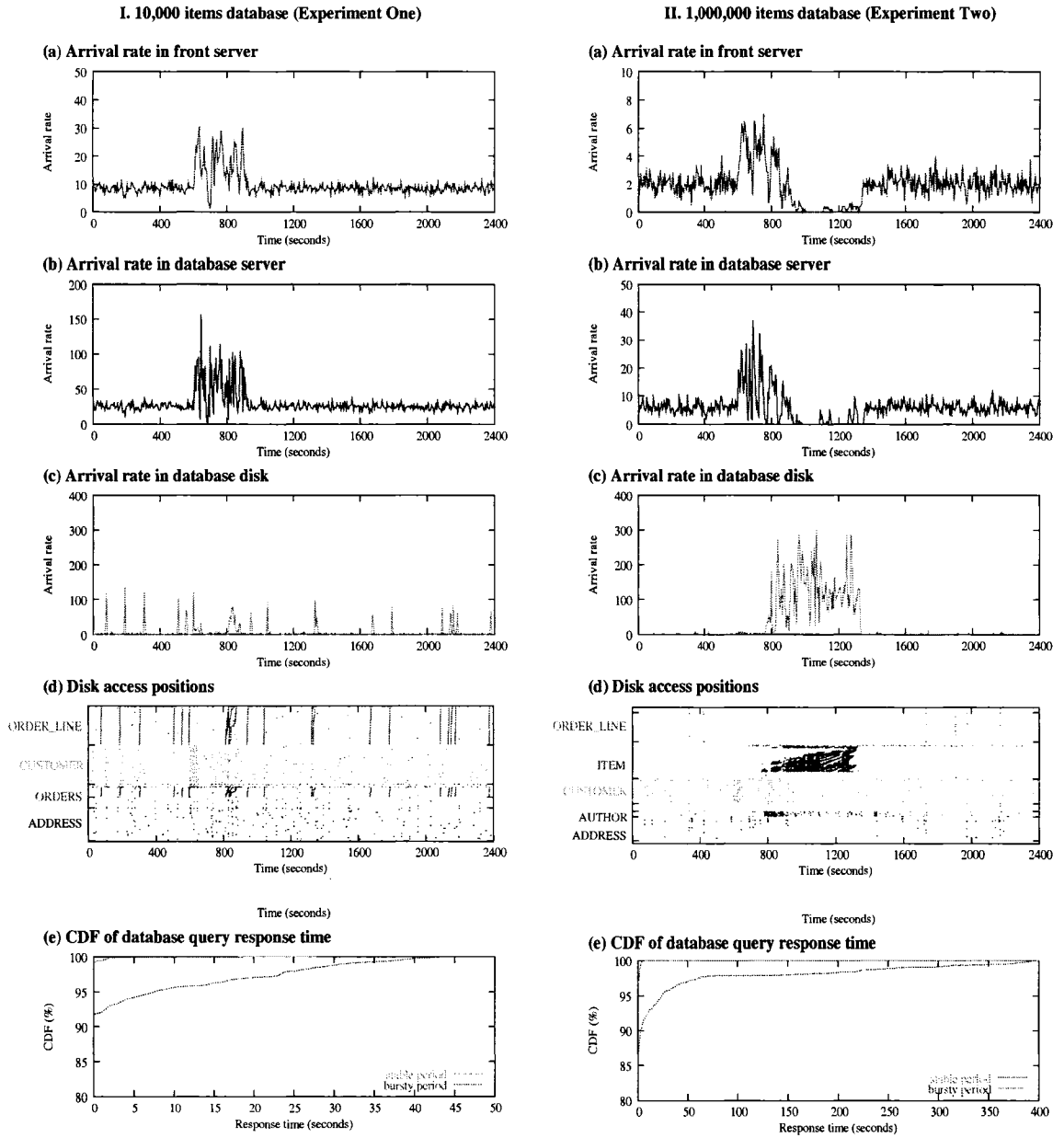


Figure 4.4: Throughputs and utilizations in multiple tiers under transient load of the browsing mix.

an exception toward the end of the overload period (after the 800th second) where the arrival burst is sustained longer.

Figure 4.4(I.d) reports on the disk access pattern as a function of time. The entire disk is mapped on the y-axis which marks the physical layout of each database table on the disk.

Vertical lines in this figure indicate disk sequential accesses that correspond to an entire table being scanned. Clearly, there is higher intensity of disk activity during the period of the arrival burst, which makes disk transfers slower, especially after the 800th second. Note that entire table accesses are not shown as vertical lines anymore, indicating that these accesses take now longer. The effect of the bursty arrivals on the query response time are reported in Figure 4.4(I.e). As expected, during the overload period database queries take longer to complete which negatively affects user perceived performance.

Note that since TPC-W simulates a closed system, the arrival rate to the first server is an equivalent indicator of system throughput. During the overload condition both arrival intensity and system throughput increase (Figures 4.4(I.a) and 4.4(I.b)). This overload propagates from the front-end web server down to the database server, but not to the disk. System performance is affected (Figure 4.4(I.e)), but not to the point of affecting system throughput. We conclude that the system sustains its performance during the short-lived overload period.

Experiment Two (1M Database, Transient Load): The right column of Figure 4.4 reports on system performance at various levels for the large database and the browsing mix. Recall that for the experiments with the large database, memory may become the bottleneck as the ITEM table is 512 MB and the available memory only 768 MB. In this experiment, for the first 600 seconds the number of browsers is set to 16, for the next 300 seconds it is set to 96 to cause a short-lived overload condition, and for the remaining 25 minutes is reset to 16. Note that the arrival intensity propagates from one tier to the next, reaching the database disk finally. Yet, there is a significant drop in the arrival rate/throughput at the front server after the 900th second despite the fact that the bursty period ended and the number of EBs is restored to 16 (see Figure 4.4(II.a)). This drop persists for the next 400

seconds and it is reflected on the database server also (see Figure 4.4(II.b)).

The opposite is observed during the same time period at the disk as in Figure 4.4(II.c). High arrival rates at the disk imply the presence of severe queueing at the storage system, resulting in slower response times. Because the TPC-W simulates a closed system (i.e., the number of EBs in the system remains the same during the experiment), the majority of the requests are accumulated in the slowest tier, resulting in a significant drop in the rate of request completions, which reduces the throughput and consequently arrival rates to the front and database servers (see Figures 4.4(II.a)–(II.b)). We emphasize that the drop of the throughput at the front-end and at the database server is as drastic as to imply conditions of service unavailability for the period of time that the storage system suffers from overload.

Figure 4.4(II.d) sheds light to this behavior. After the 800th second the system operates under conditions of severe overload. The system recovers after the 1300th second, well after the burst ends and the number of browsers is reset to 16. This overload period at the disk is the result of memory thrashing at the database server, causing repetitive, long sequential scans to the ITEM table. Under normal conditions the ITEM table is fully stored in memory. In this experiment, the sequential scans of the ITEM table are represented as *almost* horizontal lines in Figure 4.4(II.d) indicating that it takes a long time to complete some of the database queries. This is also reflected in the large gap between the query response time distributions during normal and overload conditions in Figure 4.4(II.e). Approximately 5% of all queries have a response time of more than 1 minute, which indicates service unavailability.

Concluding on experiments one and two, we stress that it is important for performance that overload propagates up to the database server CPU only (experiment one, Figure 4.4(I)) rather than further down to the database disk (experiment two, Figure 4.4(II)). In experiment one, overload does not affect system throughput and the system recovers as soon as the bursty

period ends. In experiment two system throughput drops to the point of service unavailability and the recovery process is much longer than the bursty period itself. Overload propagation down to the storage system slows down the entire system operation, and significantly affects system availability. The deeper overload propagates down the system hierarchy, the more severe the performance degradation and the slower the system recovery.

The results of the first two transient load experiments highlight a case that can be managed via admission control at the front-end. However, it is possible that overload happens when the number of arrivals remains the same at the upper system tiers: all that is needed is to change the type of work requested by the customers. For this new set of experiments that simulate transient work in the system, we fix the number of customers but we change the nature of their work for 300 seconds (from the 600th until the 900th second).

Experiment Three (10K Database, Transient Work): We report results for the small database using the ordering mix with 640 EBs on the left column in Figure 4.5. Recall that under the ordering mix the front-end rather than the database server operates close to its capacity and 640 EBs do not saturate system resources in the lower tiers. In this experiment, we keep the number of EBs steady but from the 600th to the 900th seconds of the experiment, we change the work done by the browsers as follows: we increase the percentage of ORDER_DISPLAY requests from 0.22% to 30% and we proportionally adjust the percentage of the rest of the requests in the ordering mix. ORDER_DISPLAY requests search in the CUSTOMER, ADDRESS, and ORDER_LINE tables to generate reports on all orders placed by a single customer and on related best-selling items for each order. Note that we do not introduce any new query in the TPC-W workload mix, we only modify the weight that each of the pre-defined transactions have to simulate an unusual workload mix in the system

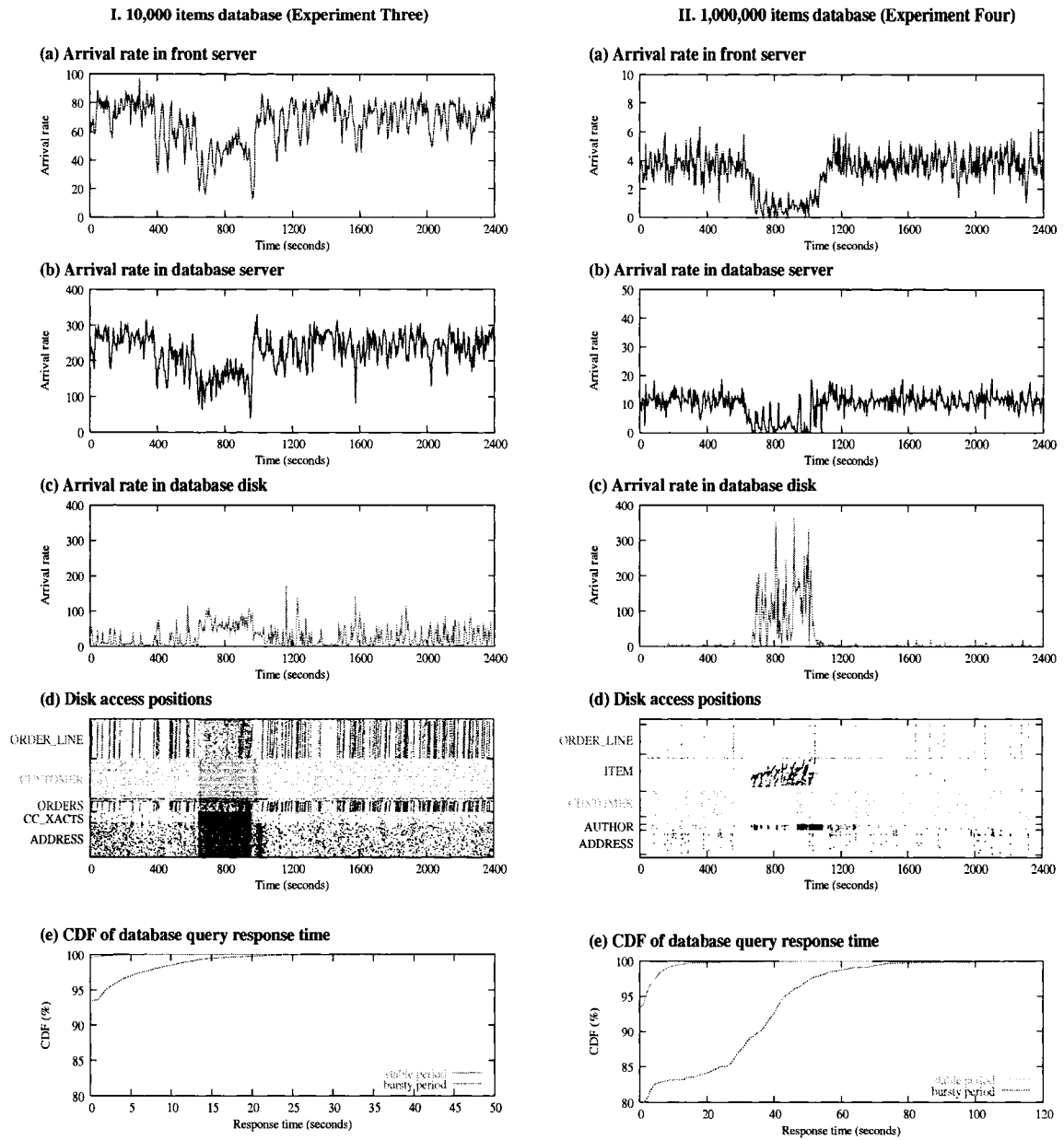


Figure 4.5: Arrival rates and system utilizations in multiple levels under transient work, for the ordering mix and a variant (I) and the browsing mix and a variant (II).

Figures 4.5(I.c)–(I.d) confirm that there is significantly higher disk activity in multiple tables, in contrast to the disk activity of the browsing mix in experiment one (see Figure 4.4(I.d)), indicating that the database working set is substantially larger under the ordering mix and its variant than for the browsing mix. Recall that the size of the entire small

database is approximately 1.5 GB, suggesting that a large working set would not fit into the database server memory of 768 GB. This is the reason that the disk becomes the bottleneck with the ordering mix variant. As a result, arrival rates (and consequently system throughput) at the front-end and database servers significantly drop from the 600th until the 950th second (see Figures 4.5(I.a) and 4.5(I.b)), suggesting that the system operates in overload. System throughput is reduced during the period of bursty work, but does not become zero as in experiment two. Overwork in experiment three, similarly to the overload in experiment one, impacts negatively the user-perceived performance (see Figure 4.5(I.e)) but does not drive the system to unavailability. Comparing results of experiments one and three, one can see that while load did not propagate down to the lowest tier, work did propagate. This indicates a critical difference between overload and overwork: work can propagate further down to the database disk and can make the system susceptible to overload independently of the available system resources.

Experiment Four (1M Database, Transient Work): The last experiment reports on performance data using the large database and the browsing mix and is illustrated on the right column of Figure 4.5. Now the number of browsers is set to 32 throughout the experiment. In the TPC-W browsing mix, the percentage of requests that ask for new products is 11%. For 5 minutes (from the 600th till the 900th second), 90% of requests are for new products, with the remaining of the requests proportionally adjusted. After the 900th second, the system operates under the default browsing mix again. This change drastically increases the arrival rate to the database disk, as the ITEM and AUTHOR tables need to be accessed (see Figures 4.5(II.c)-(II.d)). These disk accesses significantly increase the average query response time during the sensitive period (see Figure 4.5(II.e)). During the bursty work period, throughput at the front-end web server reduces to zero for nearly 200

seconds beyond the end of the bursty period at the 900th second (see Figures 4.5(II.a)). This behavior, similar to experiment two, indicates system unavailability during the transient overwork period.

Recall that the system with the large database has less available resources than the system with the small database as its working set is larger but the available system memory is the same. Under both transient load (experiment two) and transient work (experiment four), the system becomes unavailable. For the experiments with the small database (experiments one and two) the system resources are well provisioned, as a result during overload or overwork performance drops but the system quickly recovers and does not become unavailable.

In contrast to the transient load which may be detected by simply monitoring the arrival rate at the system front-end, transient work may be detected by observing slower response times, despite the fact that the number of users in the system (or the number of connections) remains relatively unchanged. Yet, we show that even for the small database where steady state analysis (see Section 4.1) shows that memory is hardly the bottleneck, scenarios that make the system suffer from severe overload are not hard to devise.

To summarize, in this section we present experiments using the TCP-W that do show that transient overload as:

- load-related, where the number of simultaneous customers may force the system to operate beyond its capacity, and to
- work-related, where the number of simultaneous customers remains the same but the changes in the work done by them brings the system to the brink of thrashing.

If overload due to transient load or transient work propagates down to the lowest tier, i.e., the database disk, the effect is reflected in the system hierarchy up to the highest tier, as

dips in the arrival rate/ throughput in the front end indicate that the system shows signs of service unavailability.

Effective system provisioning is difficult, as changes in the work done may quickly drive system resources to saturation, and system recovery from these states may become very slow. In the next section we elaborate on methods to speed up system recovery, focusing on the resource allocation policies at the storage system.

4.4 System Implications of the Transient Workloads

The previous sections show via measurements how the system workload propagates down the tiers of an e-commerce site. The goal is to understand the conditions under which certain tiers of the system become the bottleneck and negatively affect service availability and user-perceived performance. The straightforward conclusion is that too much load at the front-end generates too much load at the lower levels of the system, increasing the average request service time and causing the service to become unavailable. The experiments also show that excessive work, especially in the lower tiers, may also cause response times to become so slow that can considerably degrade user-perceived performance. By propagating down the system hierarchy, excessive work critically affects system performance and, similarly to excessive load, might bring system availability to a halt as depicted in Figures 4.4(II.a)–(II.b) and 4.5(II.a)–(II.b.) Consequently, effective handling of such overload conditions, e.g., via *work-shedding* of some form, becomes as important as any *load-shedding* technique.

The straightforward way to avoid overloading is admission control, i.e., rejecting service to new customers at the system front-end and/or interrupt service to existing users. Note that from the perspective of the service provider, service interruption to existing users bears

more penalty than rejection of service to new users. As shown in Section 4.3, excessive work might result even from the same set of users, which leaves service interruption at the front-end as the main alternative to sustain service availability during the transient overwork period.

Complementary to front-end admission control is the development of work-shedding policies at the lower tiers of the system that adapt their configuration parameters according to the current resource demands. In fact, it is possible for system resources to achieve much of the benefits of work-shedding by taking advantage of local information and by understanding their own behavior.

Figure 4.6 illustrates the number of outstanding requests (i.e., the queue length) at the database server and the storage system during the overload and overwork scenarios in experiments two and four of the previous section. In the overload case (see Figure 4.6(a)), queue lengths of both resources follow the same pattern, while for the overwork case (see Figure 4.6(b)) spikes in the queue length at the disk and the database are complementary to each other across time.

This further shows that lower tiers may have an advantage in work-shedding over the front-end. The front-end cannot easily distinguish the *source* of slow response time: is it because of increased arrival intensity (overload), or is it because of a slower service process (overwork)? Furthermore, the front-end cannot make the best decisions about which requests to drop, as it cannot accurately assess which are expensive and which are not. As a result, it may end up dropping requests randomly, leaving a large fraction unserved [92] instead of a small fraction of well-chosen requests. For requests that have already reached deep in the system hierarchy and cannot be easily rejected, simple work reordering, i.e., preferential scheduling in the same spirit as in [34], may result in faster system recovery from overload.

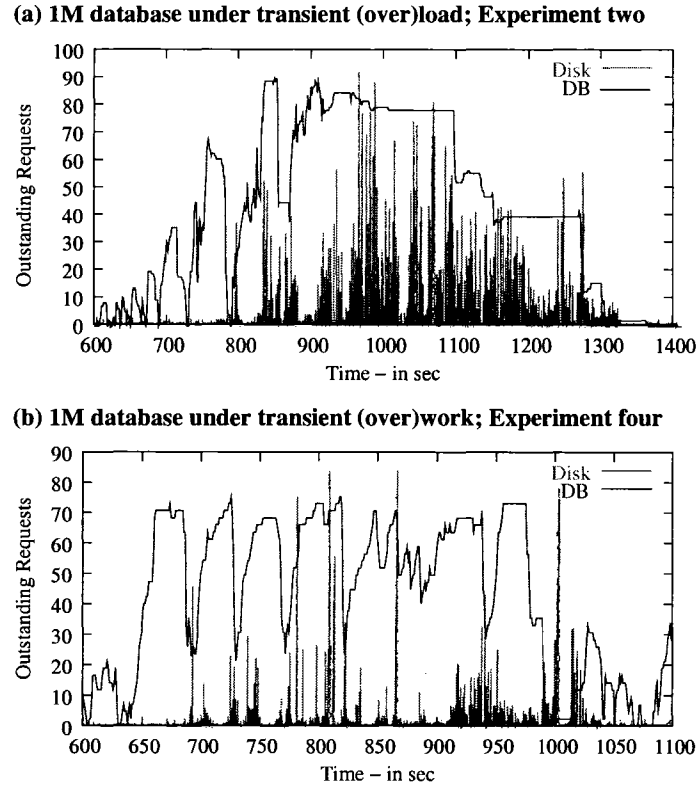


Figure 4.6: Outstanding requests (queue lengths) at the disk and at the database during overload and overwork scenarios of (a) experiment two and (b) experiment four.

For example, the system resource may choose to postpone certain actions in order to achieve graceful degradation in system performance and avoid service unavailability. This can be an effective strategy especially if the overload (or overwork) condition is only temporary. In the following part, we give a proof-of-concept that it is possible for system resources to make independent decisions toward effective overload/overwork handling.

Case study: handling overload/overwork at the storage system

Now, we focus only at the lowest level in the system, i.e., the storage system, and propose a technique to handle disk overload/overwork. The goal is to adapt disk operation such that during overload, or overwork, graceful performance degradation is achieved. In order to handle overload at the disk where service rejection is not straightforward, we prioritize the

work such that the tail of disk response time, i.e., the worst case, is contained.

Note that a single request for a database table gets transformed to several, sometimes hundreds of, I/O requests. If these I/O requests are for consecutive blocks of the disk media, they are considered to be a sequential stream. A fully random I/O workload instead consists of requests uniformly distributed over the disk media. In an e-commerce system, usually, the disk workload consists of a mix of sequential streams and random requests, as Figures 4.4(I.d), 4.4(II.d), 4.5(I.d), and 4.5(II.d) illustrate. By characterizing the I/O workload as a set of streams, one can identify long running streams and short running streams. This characterization becomes very useful in times of overload where it may be better to prioritize service of short-running streams and postpone service of the long-running ones, in the same spirit as the Shortest Job First (SJF) family of policies. We implemented this prioritization of streams into the Shortest Positioning Time First (SPTF) policy [96], which is widely implemented in disk drives today. We detect overload/overwork at the disk by monitoring its queue length. Once queue length reaches a predefined threshold, SPTF serves short-running streams and postpones the long-running ones for the future. The postponed requests are served after the transient overload period ends or a predetermined time interval has elapsed. By viewing the I/O workload as streams of requests, decisions at the disk level can affect only a small number of database requests.

We evaluate this disk overload-aware policy via trace driven simulation. DiskSim [26] is used as the disk-level simulator, which is driven by traces from the testbed described in Section 3.5, specifically traces from experiments two and four (see Figure 4.4(II) and Figure 4.5(II)). Here, we concentrate on handling overload at the disk level only. A comprehensive evaluation of the performance implications of this disk scheduling policy in the entire system is the subject of future work.

Figures 4.7 and 4.8 present the response times of all disk requests during the overload and overwork periods in experiments two and four, respectively. Results are presented as

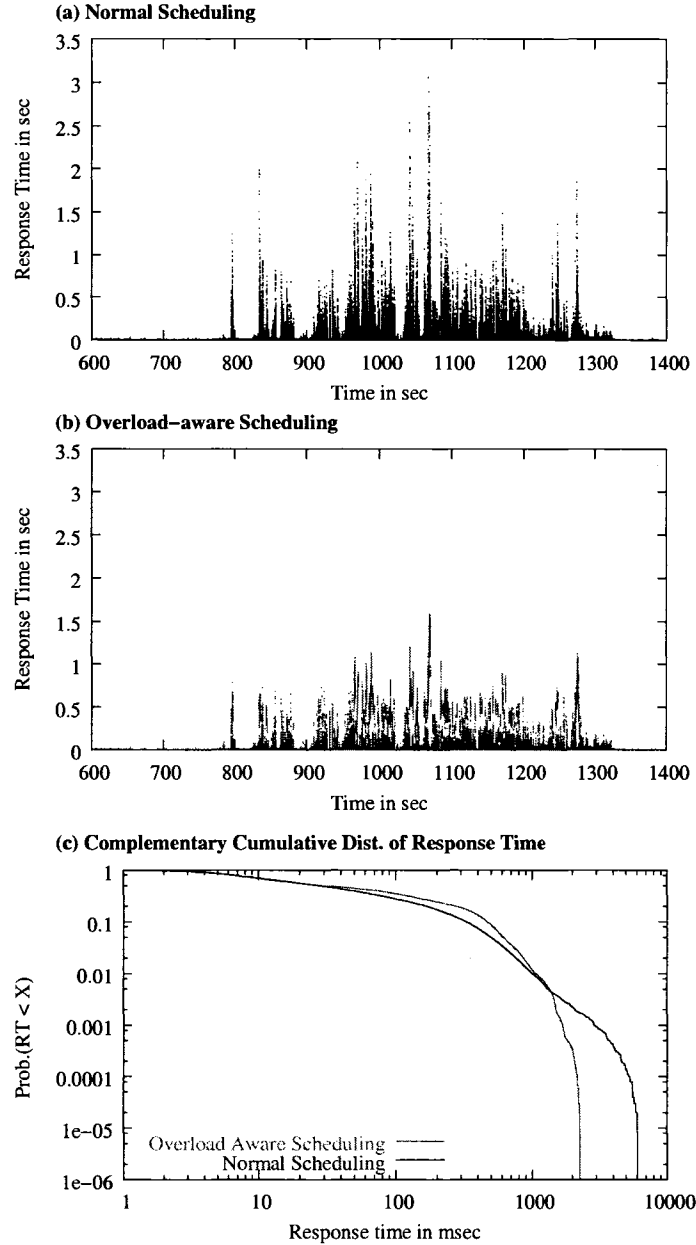


Figure 4.7: Response time at the disk as a function of time under (a) normal scheduling, (b) overload-aware scheduling, and (c) the complementary cumulative distribution during the overload period of experiment two.

a function of time using the normal SPTF scheduling policy and the overload-aware SPTF.

Observe that for both experiments, disk response times using the overload-aware scheduling

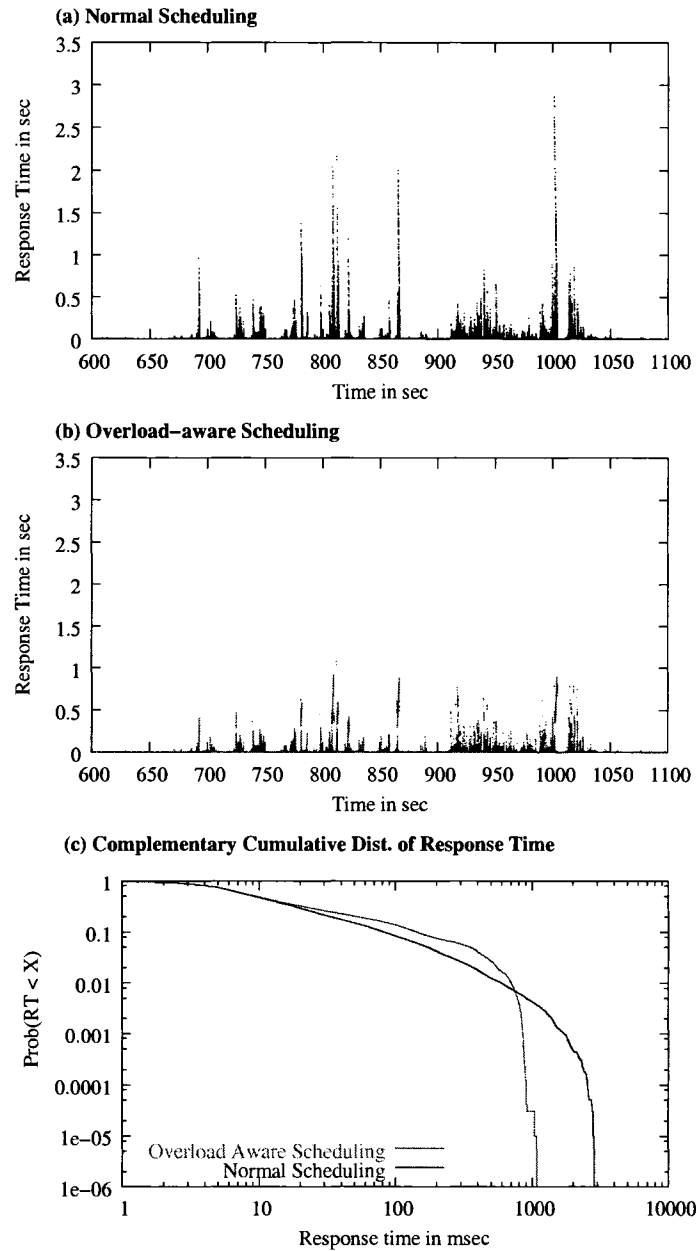


Figure 4.8: Response time at the disk as a function of time under (a) normal scheduling, (b) overload-aware scheduling, and (c) the complementary cumulative distribution during the overwork period of experiment four.

algorithm are diminished to less than half. This fact is emphasized also in the complementary distribution functions of the response times for both policies (see Figures 4.7(c) and 4.8(c)), which show that there is a clear reduction in the tail of the response time distribution when overload-aware scheduling is used. This reduction can greatly help in avoiding the time

during which the system operates on a “red” zone, i.e., when the system is so slow that it is unavailable. By modifying the disk scheduling discipline, we reduced the worst response times at the disk by a factor of two, clearly improving the responsiveness of the lowest tier and improving on the overall system availability.

4.5 Summary

In this chapter, we present a detailed workload characterization study via experimental measurements in a 3-tier e-commerce system built according to the TPC-W specifications to study the system performance under overloads and evaluate how workload propagates through all levels of the system hierarchy. By measuring resource utilization through all 3 tiers of the system, i.e., at the front end web server, at the database server, and at the database storage system, we show that it is the lower tiers i.e., the database server and the disk, that suffer most from such overload/overwork conditions. More specifically, we summarize our observations as follows:

- The time spent at the database server (including the disk) is the one that dominates user-perceived performance. The workloads which utilize mostly the database server memory and less the disk, can sustain more load, while the workloads that are more I/O oriented sustain less load as a result of a slower service process and FCFS-like service discipline at the disk.
- Both the transient excessive load in the system (i.e., an higher than usual number of users, number of web sessions, or number of network flows) and the effects of transient excessive work (i.e., sudden increase in the demand of system resources by the current users of the system) could results in system overloading. The further overload/overwork

propagates down the system hierarchy, i.e., the memory/disk, the higher performance penalty, and the more difficult it is to recover performance effectively.

- Complementary to front-end admission control mechanisms, effective resource management at the various devices can significantly aid system performance. We have showed a first proof-of-concept that self-adaptive resource management at the lower tiers that can detect and handle overload and overwork cases can help in graceful performance degradation and in avoiding system unavailability.

Chapter 5

Analytic Models in Multi-tiered Systems

The experiments in previous chapters indicate the existence of autocorrelation in closed systems. The impact of autocorrelation in an open multi-tiered system is different from that in a closed system, first because the correlated tier can only affect its descending tiers and second because there is no upper bound of system performance due to the unlimited customer population in an open system. In this chapter we focus on analytic models of multi-tiered systems with no restriction in their buffer sizes, i.e., open systems. An abstraction of such systems is illustrated in Figure 5.1. It resembles the structure of tandem queues, i.e., queues whose departure process becomes the arrival process of the next queue.

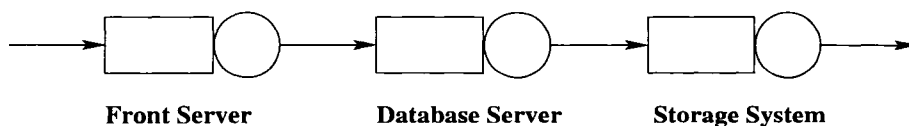


Figure 5.1: A queuing model of the experimental environment in Chapter 3 but with infinite buffers.

Developing analytic techniques for solving networks of queues as the one depicted above is very challenging. A queue-by-queue analysis of such networks [35] often is the only ana-

lytic alternative to simulation, especially when queues exhibit autocorrelations in the arrival and/or service processes. The persistent presence of rare events in simulations of such systems require several replications of a tremendously large sample space in the order of hundreds of millions to reach results within accepted confidence intervals, even for the simplest case of a single queue, necessitating analytic models. Modeling the departure process of a queue accurately is essential to queue-by-queue analysis, as the departure process may serve as arrival process to downstream queues.

It has been extensively shown that batch arrivals and long-range dependence are characteristics of Internet systems [40, 22], while there are clear indications that dependence exists in the service process of systems as well [71]. The presence of (positive) autocorrelation in the arrival and/or service process has detrimental effect on performance [22]. Therefore, it is of critical importance for departure processes to capture as accurately as possible autocorrelation in the departure flows of a queue, else analytic models may fail to capture performance degradation due to dependence in flows.

The models presented in this chapter are as general as possible and do consider correlation in their arrival and/or service processes. Customers (or packets) in such systems may arrive in batches, significantly impacting queueing behavior. As introduced in Chapter 2, correlated flows with batches can be represented by the Batch Markovian Arrival Process (BMAP) [48]. Correlated service processes may be described by MAPs. Data fitting to BMAPs, MAPs, and their subclasses is subject of several recent works [37, 10, 11, 74].

In this chapter we present a family of finite approximations for the departure process of a BMAP/MAP/1 queue, which is derived via an exact aggregate solution technique called ETAQA [72] applied to M/G/1-type Markov processes. This chapter is organized as follows. In Section 5.1 we give an overview of related work. We present techniques for the solution

of M/G/1-type Markov processes in Section 5.2. In Section 5.3, we construct the family of finite matrix-exponential (ME) matrix representations, from which characteristics of the departure process are computed and formally proved that the approximation preserves the marginal distribution and certain autocorrelation of the exact departure process. Section 5.4 gives a summary of this chapter.

5.1 Related Work

Characteristics of departure processes of BMAP/GI/1 queues are studied in [24], i.e., queues with general, but uncorrelated service times and batches. Algorithms and explicit formulas to compute various measures, including the moments and covariances of the inter-departure times, are developed for different types of queues, including queues with finite buffers and vacation servers. The methods presented in this dissertation could be applied to servers with vacation (at the expense of increased QBD block dimensions), but are not designed for finite queues. In contrast to [24], we can treat correlated service times and not only deliver a set of output characteristics, but also an approximate output model that matches these characteristics exactly.

Different approximation models of departure processes, which also capture the inter-departure distribution and the first lag coefficients of correlation of the departure process, have been proposed for single-server queues (e.g., [30, 77, 41]) – all of them based on arrival and service processes in matrix notation. Kumaran et al. [41] suggest a model for the departure process of an ME/ME/1 queue. The notation used to decide matrix-exponential (ME) processes strongly resembles that of a MAP. It is more general than MAP but lacks MAP’s stochastic interpretation. The approximation models proposed by Green [29, 30]

work for MAP/PH/1 queues (i.e., when there is no correlation among service times), and the proofs of the preserved properties of the output models, such as marginal distribution and correlation coefficients can be found in [29]. Among the alternative approaches in the literature, the family of models proposed in [77] is the most general, it guarantees that the output approximations are MAPs, but it only applies to MAP/MAP/1 queues. However to the best of our knowledge, none of the others' work considers batch arrival processes in such traffic-based decomposition techniques. Additionally, this dissertation presents thorough analytic proofs of how well the approximation matches the lag correlations as a function of the size of the approximation model.

5.2 Background

In this section, we recall the definitions and properties of M/G/1-type Markov processes and cite a theorem on the aggregate solution of such Markov processes upon which the analysis of the departure process is based.

5.2.1 M/G/1-type Processes

A BMAP/MAP/1 queue defines an M/G/1-type Markov process. The infinitesimal generator \mathbf{Q}_∞ of such a CTMC¹ has an upper block Hessenberg form

$$\mathbf{Q}_\infty = \begin{bmatrix} \hat{\mathbf{L}} & \mathbf{F}^{(1)} & \mathbf{F}^{(2)} & \mathbf{F}^{(3)} & \mathbf{F}^{(4)} & \dots \\ \mathbf{B} & \mathbf{L} & \mathbf{F}^{(1)} & \mathbf{F}^{(2)} & \mathbf{F}^{(3)} & \dots \\ \mathbf{0} & \mathbf{B} & \mathbf{L} & \mathbf{F}^{(1)} & \mathbf{F}^{(2)} & \dots \\ \mathbf{0} & \mathbf{0} & \mathbf{B} & \mathbf{L} & \mathbf{F}^{(1)} & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}, \quad (5.1)$$

¹Note that although we restrict the presentation to continuous-time queues, the presented techniques can be directly adapted to discrete-time queues.

where the state space is partitioned into levels, i.e., $\mathcal{S}^{(j)} = \{s_1^{(j)}, \dots, s_m^{(j)}\}$, for $j \geq 0$ and $m \geq 1$. Intuitively, $\mathcal{S}^{(0)}$ represents the state configuration when the queue is empty. The states that account for the state of the system when the queue is nonempty (with j customers) correspond to sets $\mathcal{S}^{(j)}$, for $j \geq 1$, and the interaction of successive sets has a “repetitive” structure (see Figure 5.2(a)). In Eq. (5.1), The letters “L”, “F” and “B” are used according to “local”, “forward” and “backward” transition rates, respectively, in relation to a set of states $\mathcal{S}^{(j)}$ for $j \geq 0$. For BMAP/MAP/1 queues, the block matrices are defined as follows using Kronecker notation:

$$\begin{aligned}\hat{\mathbf{L}} &= \mathbf{D}_0^{(A)} \otimes \mathbf{I}_S \\ \mathbf{L} &= \mathbf{D}_0^{(A)} \oplus \mathbf{D}_0^{(S)} = \mathbf{D}_0^{(A)} \otimes \mathbf{I}_S + \mathbf{I}_A \otimes \mathbf{D}_0^{(S)} \\ \mathbf{B} &= \mathbf{I}_A \otimes \mathbf{D}_1^{(S)} \\ \mathbf{F}^{(i)} &= \mathbf{D}_i^{(A)} \otimes \mathbf{I}_S \quad \text{for } i \geq 1 \quad ,\end{aligned}$$

where the matrices $\mathbf{D}_i^{(A)}$ ($i \geq 0$) describe the BMAP of the arrival process of order m_A and $\mathbf{D}_0^{(S)}$ and $\mathbf{D}_1^{(S)}$ describe the MAP of the service process of order m_S . All matrices \mathbf{B} , $\mathbf{F}^{(i)}$, \mathbf{L} and $\hat{\mathbf{L}}$ are square $(m \times m)$ -matrices, where $m = m_A m_S$. For general M/G/1-type processes, the set $\mathcal{S}^{(0)}$ might differ in cardinality from m , but for presentation simplicity we need not consider this here.

Let $\boldsymbol{\pi}^{(j)}$ for $j \geq 0$ be the stationary probability vectors (of dimension m) for states in $\mathcal{S}^{(j)}$. For the computation of the stationary probability vector

$$\boldsymbol{\pi}_\infty = \begin{bmatrix} \boldsymbol{\pi}^{(0)} & \boldsymbol{\pi}^{(1)} & \dots \end{bmatrix} \quad , \quad (5.2)$$

defined by $\boldsymbol{\pi}_\infty \mathbf{Q}_\infty = \mathbf{0}$ and $\boldsymbol{\pi}_\infty \mathbf{e} = 1$, matrix-analytic methods have been proposed [61].

Commonly, the sub-vectors $\boldsymbol{\pi}^{(j)}$ are determined using Ramaswami’s recursive formula [69],

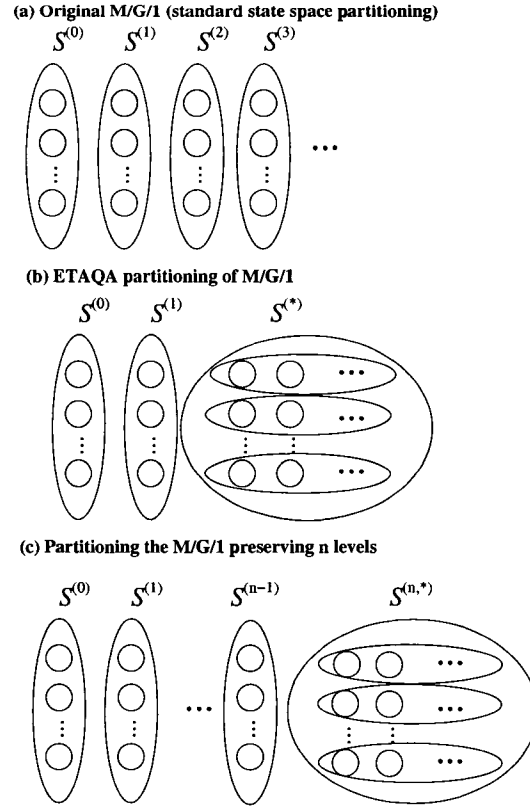


Figure 5.2: State-space partitioning of a M/G/1 solved (a) via the traditional matrix-analytic method, (b) via ETAQA, and (c) ETAQA that preserves the first n M/G/1 levels, $n \geq 1$.

which is based on matrix \mathbf{G} , the key element to matrix-analytic methods and solution of

$$\mathbf{B} + \mathbf{L}\mathbf{G} + \sum_{i=1}^{\infty} \mathbf{F}^{(i)}\mathbf{G}^{i+1} = \mathbf{0} . \quad (5.3)$$

Matrix \mathbf{G} has an important probabilistic interpretation: an entry (l, k) in \mathbf{G} expresses the conditional probability of the process first entering $S^{(i-1)}$ through state k , given that it starts from state l of $S^{(i)}$ [61, page 81]. Iterative algorithms are used to calculate \mathbf{G} , with the cyclic reduction algorithm being the most efficient [42].

To formulate Ramaswami's formula, we define the matrices

$$\mathbf{S}^{(j)} = \sum_{i=j}^{\infty} \mathbf{F}^{(i)}\mathbf{G}^{i-j} \quad \text{for } j \geq 0 , \quad (5.4)$$

where we additionally set $\mathbf{F}^{(0)} \equiv \mathbf{L}$. Note that Eq. (5.3) then takes the form $\mathbf{B} + \mathbf{S}^{(0)}\mathbf{G} = \mathbf{0}$.

Ramaswami's formula defines the following recursive relation:

$$\boldsymbol{\pi}^{(j)} = - \left(\sum_{i=0}^{j-1} \boldsymbol{\pi}^{(i)} \mathbf{S}^{(j-i)} \right) (\mathbf{S}^{(0)})^{-1} \quad \text{for all } j \geq 1 \quad . \quad (5.5)$$

Before applying Eq. (5.5) to iteratively compute $\boldsymbol{\pi}^{(j)}$ for $j \geq 1$, we first have to solve the following system of m linear equations to obtain vector $\boldsymbol{\pi}^{(0)}$: where the last column in the matrix corresponds to normalization, which replaces any one of the other equations.

In [72], ETAQA was proposed as a methodology for the exact analysis of M/G/1-type Markov processes. Originally, ETAQA truncates these infinite Markov processes on level $n = 2$ in such a way that the stationary level distributions $\boldsymbol{\pi}^{(0)}$ and $\boldsymbol{\pi}^{(1)}$ are preserved (see Figure 5.2(b)). However, it is easily seen from [72] that aggregation can occur for *any* level $n \geq 2$ (and in fact, also for $n = 1$ with a structure as in Eq. (5.1)), as illustrated in Figure 5.2(c). The main theorem for the solution of M/G/1-type processes can then be restated as follows:

Theorem 5.1 [ETAQA] *Given an ergodic CTMC with infinitesimal generator \mathbf{Q}_∞ (see Eq. (5.1)) and with stationary probability vector $\boldsymbol{\pi}_\infty$ (see Eq. (5.2)), the system of linear equations (parameterized with n)*

$$\boldsymbol{\pi}_n \hat{\mathbf{Q}}_n = \mathbf{0} \quad ,$$

where $\hat{\mathbf{Q}}_n \in \mathbf{R}^{(n+1)m \times (n+1)m}$ is defined by

$$\hat{\mathbf{Q}}_n = \begin{bmatrix} \hat{\mathbf{L}} & \mathbf{F}^{(1)} & \mathbf{F}^{(2)} & \dots & \mathbf{F}^{(n-2)} & \mathbf{F}^{(n-1)} - \sum_{i=n+1}^{\infty} \mathbf{S}^{(i)} \mathbf{G} & \sum_{i=n}^{\infty} \mathbf{F}^{(i)} + \sum_{i=n+1}^{\infty} \mathbf{S}^{(i)} \mathbf{G} \\ \mathbf{B} & \mathbf{L} & \mathbf{F}^{(1)} & \dots & \mathbf{F}^{(n-3)} & \mathbf{F}^{(n-2)} - \sum_{i=n}^{\infty} \mathbf{S}^{(i)} \mathbf{G} & \sum_{i=n-1}^{\infty} \mathbf{F}^{(i)} + \sum_{i=n}^{\infty} \mathbf{S}^{(i)} \mathbf{G} \\ \mathbf{0} & \mathbf{B} & \mathbf{L} & \ddots & \vdots & \mathbf{F}^{(n-3)} - \sum_{i=n-1}^{\infty} \mathbf{S}^{(i)} \mathbf{G} & \sum_{i=n-2}^{\infty} \mathbf{F}^{(i)} + \sum_{i=n-1}^{\infty} \mathbf{S}^{(i)} \mathbf{G} \\ \mathbf{0} & \mathbf{0} & \ddots & \ddots & \mathbf{F}^{(1)} & \vdots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \mathbf{L} & \mathbf{F}^{(1)} - \sum_{i=3}^{\infty} \mathbf{S}^{(i)} \mathbf{G} & \sum_{i=2}^{\infty} \mathbf{F}^{(i)} + \sum_{i=3}^{\infty} \mathbf{S}^{(i)} \mathbf{G} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{B} & \mathbf{L} - \sum_{i=2}^{\infty} \mathbf{S}^{(i)} \mathbf{G} & \sum_{i=1}^{\infty} \mathbf{F}^{(i)} + \sum_{i=2}^{\infty} \mathbf{S}^{(i)} \mathbf{G} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{B} - \sum_{i=1}^{\infty} \mathbf{S}^{(i)} \mathbf{G} & \sum_{i=1}^{\infty} \mathbf{F}^{(i)} + \mathbf{L} + \sum_{i=1}^{\infty} \mathbf{S}^{(i)} \mathbf{G} \end{bmatrix} \quad (5.6)$$

admits a unique solution

$$\boldsymbol{\pi}_n = [\boldsymbol{\pi}^{(0)} \quad \boldsymbol{\pi}^{(1)} \quad \dots \quad \boldsymbol{\pi}^{(n-1)} \quad \boldsymbol{\pi}_n^{\infty}] \quad ,$$

where $\boldsymbol{\pi}_n^{\infty} = \sum_{i=n}^{\infty} \boldsymbol{\pi}^{(i)}$, given that we discard one column (any) and replace it with a column of 1's due to the normalization condition, i.e., $\boldsymbol{\pi}_n \mathbf{e} = 1$.

Proof: The proof follows directly from the main theorem in [72]. ■

$\hat{\mathbf{Q}}_n$ is not necessarily an infinitesimal generator, since non-diagonal numbers might be negative due to the subtractions in Eq. (5.6). However, from $\hat{\mathbf{Q}}_n$, the initial sequence of (invariant) stationary probability vectors $\boldsymbol{\pi}^{(j)}$ ($j = 0, 1, \dots, n-1$) and $\boldsymbol{\pi}^{(n,*)}$ may be derived similarly as for Markov chains. The case $n = 1$ with two block levels only (namely 0 and 1) may also be included. However, we will see that this particular case (unlike $n > 1$) does not prove favorable for the desired output approximations of BMAP/MAP/1 queues.

5.2.2 Example: ETAQA Representation of the $M^{[2]}/M/1$ Queue

For illustrative purposes, we present here the aggregate ETAQA representation of Theorem 5.1 for the simplest queue with arrival batches of 1 and 2, namely an $M^{[2]}/M/1$ system. With the settings

$$\begin{aligned} \mathbf{F}^{(1)} &= [\lambda_1] & \mathbf{F}^{(2)} &= [\lambda_2] & \mathbf{F}^{(i)} &= [0] \quad \text{if } i \geq 3 \\ \mathbf{B} &= [\mu] & \mathbf{L} &= [-(\lambda_0 + \mu)] & \hat{\mathbf{L}} &= [-\lambda_0] = [-(\lambda_1 + \lambda_2)] \\ \mathbf{S}^{(1)} &= [\lambda_1 + \lambda_2] & \mathbf{S}^{(2)} &= [\lambda_2] & \mathbf{S}^{(i)} &= [0] \quad \text{if } i = 0, 3, 4, \dots, \end{aligned}$$

we obtain the “matrix” $\mathbf{G} = [1]$, where obviously all block matrices are of dimension 1 ($= m_A m_S$). The resulting ETAQA representation

$$\hat{\mathbf{Q}}_n = \begin{bmatrix} -\lambda_0 & \lambda_1 & \lambda_2 & 0 & 0 & \cdots & 0 \\ \mu & -(\lambda_0 + \mu) & \lambda_1 & \lambda_2 & 0 & \cdots & 0 \\ 0 & \mu & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \lambda_1 & \lambda_2 & 0 \\ 0 & 0 & 0 & \ddots & -(\lambda_0 + \mu) & \lambda_1 & \lambda_2 \\ 0 & 0 & 0 & \cdots & \mu & -(\lambda_0 + \lambda_2 + \mu) & \lambda_0 + \lambda_2 \\ 0 & 0 & 0 & \cdots & 0 & \mu - \lambda_0 - \lambda_2 & -\mu + \lambda_0 + \lambda_2 \end{bmatrix} \quad (5.7)$$

defines a true infinitesimal generator, if $\mu > \lambda_0 + \lambda_2 = \lambda_1 + 2\lambda_2$, which corresponds to the stability condition $\frac{1}{E[S]} > \lambda_{\text{BMAP}} = \boldsymbol{\pi}_{\text{BMAP}} \sum_{k=1}^{\infty} k \mathbf{D}_k \mathbf{e}$, assuming that the mean service time of the queue is $E[S] = \frac{1}{\mu}$.

5.3 ETAQA-based Approximation of the BMAP/MAP/1 Departure Process

5.3.1 Exact Departure Process of the BMAP/MAP/1 Queue

Starting from the infinitesimal generator \mathbf{Q}_∞ (see Eq. (5.1)), we give the exact departure process of a BMAP/MAP/1 queue as a MAP of infinite order. By “filtration” (see [29]), i.e., by collecting in matrix $\mathbf{D}_{1,\infty}$ “backward” transitions of \mathbf{Q}_∞ that correspond to departures, we arrive at the following MAP representation:

$$\mathbf{D}_{0,\infty} = \begin{bmatrix} \hat{\mathbf{L}} & \mathbf{F}^{(1)} & \mathbf{F}^{(2)} & \mathbf{F}^{(3)} & \mathbf{F}^{(4)} & \dots \\ \mathbf{0} & \mathbf{L} & \mathbf{F}^{(1)} & \mathbf{F}^{(2)} & \mathbf{F}^{(3)} & \dots \\ \mathbf{0} & \mathbf{0} & \mathbf{L} & \mathbf{F}^{(1)} & \mathbf{F}^{(2)} & \dots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{L} & \mathbf{F} & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}, \quad \mathbf{D}_{1,\infty} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots \\ \mathbf{B} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots \\ \mathbf{0} & \mathbf{B} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots \\ \mathbf{0} & \mathbf{0} & \mathbf{B} & \mathbf{0} & \mathbf{0} & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix}. \quad (5.8)$$

Applying the MAP definitions, the mean rate of this exact departure process is:

$$\lambda_\infty = \boldsymbol{\pi}_\infty \mathbf{D}_{1,\infty} \mathbf{e}. \quad (5.9)$$

The SCV of the exact departure process is

$$c_\infty^2 = 2\lambda_\infty \boldsymbol{\pi}_\infty (-\mathbf{D}_{0,\infty})^{-1} \mathbf{e} - 1, \quad (5.10)$$

and the associated autocorrelation function

$$\text{ACF}_\infty(k) = \frac{\lambda_\infty \boldsymbol{\pi}_\infty ((-\mathbf{D}_{0,\infty})^{-1} \mathbf{D}_{1,\infty})^k (-\mathbf{D}_{0,\infty})^{-1} \mathbf{e} - 1}{c_\infty^2}. \quad (5.11)$$

The infinite order of the above MAP is impractical for further processing. In the next subsection, we present a finite representation, from which several performance measures of the exact departure process can be computed.

5.3.2 Truncating the Exact Departure Process

One obvious way to obtain a tractable (approximate) representation of the BMAP/MAP/1 departure process is to truncate the infinite representation in Eq. (5.8). For arrival processes without batches, as for the MAP/MAP/1 queue, This has been done in different ways (e.g., see [77]). Then, it suffices to adjust the last block row (chosen at an arbitrary block level n , $n \geq 1$) to obtain a representation that preserves the marginal distribution and the coefficients of correlation up to the first $n - 1$ lags.

The applicability of ETAQA to M/G/1-type Markov processes allows to obtain an appropriate truncation for the BMAP/MAP/1 departure process. With similar “filtration” as for the exact departure process in Section 5.3.1, one can easily construct the following matrix representations from the ETAQA matrix:

$$D_{0,n}^{(ME)} = \begin{bmatrix} \hat{L} & F^{(1)} & F^{(2)} & \dots & F^{(n-2)} & F^{(n-1)} - \sum_{i=n+1}^{\infty} S^{(i)}G & \sum_{i=n}^{\infty} F^{(i)} + \sum_{i=n+1}^{\infty} S^{(i)}G \\ 0 & L & F^{(1)} & \dots & F^{(n-3)} & F^{(n-2)} - \sum_{i=n}^{\infty} S^{(i)}G & \sum_{i=n-1}^{\infty} F^{(i)} + \sum_{i=n}^{\infty} S^{(i)}G \\ 0 & 0 & L & \ddots & \vdots & F^{(n-3)} - \sum_{i=n-1}^{\infty} S^{(i)}G & \sum_{i=n-2}^{\infty} F^{(i)} + \sum_{i=n-1}^{\infty} S^{(i)}G \\ \vdots & \vdots & \ddots & \ddots & F^{(1)} & \vdots & \vdots \\ 0 & 0 & 0 & \ddots & L & F^{(1)} - \sum_{i=3}^{\infty} S^{(i)}G & \sum_{i=2}^{\infty} F^{(i)} + \sum_{i=3}^{\infty} S^{(i)}G \\ 0 & 0 & 0 & \dots & 0 & L - \sum_{i=2}^{\infty} S^{(i)}G & \sum_{i=1}^{\infty} F^{(i)} + \sum_{i=2}^{\infty} S^{(i)}G \\ 0 & 0 & 0 & \dots & 0 & 0 & \sum_{i=1}^{\infty} F^{(i)} + L \end{bmatrix} \quad (5.12)$$

$$\mathbf{D}_{1,n}^{(\text{ME})} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{B} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{B} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \ddots & \mathbf{B} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{B} - \sum_{i=1}^{\infty} \mathbf{S}^{(i)} \mathbf{G} & \sum_{i=1}^{\infty} \mathbf{S}^{(i)} \mathbf{G} \end{bmatrix}. \quad (5.13)$$

Index n ($n > 1$) indicates that the dimensions of matrices $\mathbf{D}_{0,n}^{(\text{ME})}$ and $\mathbf{D}_{1,n}^{(\text{ME})}$ may be chosen flexibly. The order of the truncated representation is $(n+1)m = (n+1)m_S m_A$. Furthermore, the block elements of $\mathbf{D}_{0,n}^{(\text{ME})}$ and $\mathbf{D}_{1,n}^{(\text{ME})}$ are given directly in terms of the arrival and service process representations and the fundamental-period matrix \mathbf{G} .

The notation $\mathbf{D}_{0,n}^{(\text{ME})}/\mathbf{D}_{1,n}^{(\text{ME})}$ resembles that of a MAP, and indeed moments of the marginal distribution and coefficients of correlation (of the true departure process) are computed correspondingly (e.g., Eqs. (2.5), (2.6), (2.7)). However, the subtractions in the next-to-last columns of both matrices may violate the non-negativity constraint imposed on off-diagonal elements of $\mathbf{D}_{0,n}^{(\text{ME})}$ and $\mathbf{D}_{1,n}^{(\text{ME})}$. Still, we have $(\mathbf{D}_{0,n}^{(\text{ME})} + \mathbf{D}_{1,n}^{(\text{ME})})\mathbf{1} = \mathbf{0}$. In fact, representation (5.12)/(5.13) defines a matrix-exponential (ME) process [41]. Such correlated sequences of matrix exponentials are generalizations of MAPs used in linear-algebraic queueing theory [46, 55]. ME matrices can be used analogously to the corresponding MAP matrices in computational procedures for queueing systems, which do not rely on this probabilistic interpretation. Thus, we may also use the ME representation (5.12)/(5.13) as an approximate output model of the BMAP/MAP/1 queue. The corresponding notations for the mean rate,

the squared coefficient of variation and the autocorrelation function of the above ME are:

$$\begin{aligned}\lambda_n^{\text{ME}} &= \pi_n \mathbf{D}_{1,n}^{(\text{ME})} \mathbf{e} \quad , \\ (c_n^{\text{ME}})^2 &= 2\lambda_n^{\text{ME}} \pi_n (-\mathbf{D}_{0,n}^{(\text{ME})})^{-1} \mathbf{e} - 1 \quad , \\ \text{ACF}_n^{\text{ME}}(k) &= \frac{\lambda_n^{\text{ME}} \pi_n ((-\mathbf{D}_{0,n}^{(\text{ME})})^{-1} \mathbf{D}_{1,n}^{(\text{ME})})^k (-\mathbf{D}_{0,n}^{(\text{ME})})^{-1} \mathbf{e} - 1}{(c_n^{\text{ME}})^2} \quad ,\end{aligned}$$

where $\pi_n = [\pi^{(0)} \pi^{(1)} \pi^{(2)} \dots \pi^{(n-1)} \pi_n^\infty]$ as defined in Theorem 5.1.

As the last two columns of $\mathbf{D}_{0,n}^{(\text{ME})}/\mathbf{D}_{1,n}^{(\text{ME})}$ in Eq. (5.12)/(5.13) are now different from the exact departure process due to the aggregation, only the first $(n-2)$ coefficients of correlation can be preserved for an n th-level truncation. However the marginal distribution remains invariant for $n \geq 2$ because Theorem 5.1 guarantees that the stationary distribution of the embedded Markov chain of the true departure process (i.e., of the original M/G/1-type process) is maintained by ETAQA technique. The above mentioned properties of the exact departure process are formally proved in Section 5.3.3.

5.3.2.1 Special Case: Output Approximations for the MAP/MAP/1 Queue

The ETAQA methodology for the MAP/MAP/1 departure process has been first proposed in [36]. Here, we customize the results of the previous section to the MAP/MAP/1 queue, i.e., when the queue accepts batches of size 1 only. Note that the subtractions in the next-to-last column of the ME representation (5.12)/(5.13) disappear, which causes one more coefficient of correlation to be matched (i.e., $n-1$ instead of $n-2$). The matrices $\mathbf{D}_{0,n}^{(\text{ME})}$ and

$\mathbf{D}_{1,n}^{(\text{ME})}$ for the MAP/MAP/1 case are:

$$\mathbf{D}_{0,n}^{(\text{ME})} = \begin{bmatrix} \hat{\mathbf{L}} & \mathbf{F} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{L} & \mathbf{F} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \ddots & \mathbf{0} & \mathbf{L} & \mathbf{F} \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{L} + \mathbf{F} \end{bmatrix}, \quad \mathbf{D}_{1,n}^{(\text{ME})} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{B} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \mathbf{B} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B} - \mathbf{F}\mathbf{G} & \mathbf{F}\mathbf{G} \end{bmatrix}. \quad (5.14)$$

The block matrix $\mathbf{F}\mathbf{G}$ in Eq. (5.14) now fully captures the flow backward *within* the aggregate state encompassing original levels n to ∞ , while $\mathbf{B} - \mathbf{F}\mathbf{G}$ corresponds to the flow that actually leads from the aggregate state to level $n - 1$ (see [36] for a detailed treatment).

5.3.2.2 Illustration for the $M^{[2]}/M/1$ Departure Process

Specializing our output process results to the stable $M^{[2]}/M/1$ queue described in Section 5.2.2, we obtain from Eqs. (5.12) and (5.13) the following output MAP approximation:

$$\mathbf{D}_{0,n}^{(\text{ME})} = \begin{bmatrix} -\lambda_0 & \lambda_1 & \lambda_2 & 0 & 0 & \cdots & 0 \\ 0 & -(\lambda_0 + \mu) & \lambda_1 & \lambda_2 & 0 & \cdots & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \lambda_1 & \lambda_2 & 0 \\ 0 & 0 & 0 & \ddots & -(\lambda_0 + \mu) & \lambda_1 & \lambda_2 \\ 0 & 0 & 0 & \cdots & 0 & -(\lambda_0 + \lambda_2 + \mu) & \lambda_0 + \lambda_2 \\ 0 & 0 & 0 & \cdots & 0 & 0 & -\mu \end{bmatrix},$$

$$\mathbf{D}_{1,n}^{(\text{ME})} = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ \mu & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & \mu & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \ddots & 0 & 0 & 0 \\ 0 & 0 & 0 & \ddots & \mu & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & \mu - \lambda_0 - \lambda_2 & \lambda_0 + \lambda_2 \end{bmatrix}.$$

5.3.3 Proof of the Marginal Distributions and ACF of the ME Departure Approximation

Theorem 5.2 contains the proof that the marginal distributions of the true and the approximated departure processes are identical. As a prerequisite of this proof, we show $\lambda_n^{\text{ME}} = \lambda_\infty$ for all $n \geq 1$:

$$\begin{aligned}
 \lambda_n^{\text{ME}} &= \boldsymbol{\pi}_n \mathbf{D}_{1,n}^{(\text{ME})} \mathbf{e} \\
 &= \left(\sum_{i=1}^{n-1} \boldsymbol{\pi}^{(i)} \mathbf{B} + \boldsymbol{\pi}_n^\infty (\mathbf{B} - \sum_{i=1}^{\infty} \mathbf{S}^{(i)} \mathbf{G}) + \boldsymbol{\pi}_n^\infty \sum_{i=1}^{\infty} \mathbf{S}^{(i)} \mathbf{G} \right) \mathbf{e} \\
 &= \left(\sum_{i=1}^{n-1} \boldsymbol{\pi}^{(i)} \mathbf{B} + \boldsymbol{\pi}_n^\infty \mathbf{B} \right) \mathbf{e} = \left(\sum_{i=1}^{\infty} \boldsymbol{\pi}^{(i)} \mathbf{B} \right) \mathbf{e} \\
 &= \boldsymbol{\pi}_\infty \mathbf{D}_{1,\infty} \mathbf{e} = \lambda_\infty \quad .
 \end{aligned}$$

Theorem 5.2 *The complete inter-departure time distribution of the true departure process is preserved by the ME output approximation.*

Proof: For both the infinite and truncated output ME representations (5.8) and (5.12)/(5.13), respectively, the inter-departure time can be seen as a composition of

a service time (whose transient phases are described by $\mathbf{L} + \sum_{i=1}^{\infty} \mathbf{F}^{(i)}$) when the respective ME enters a level greater than 0 and

a convolution of an idle period (described by $\hat{\mathbf{L}}$) and a service time when the respective ME enters level 0.

Note that in the bottom row of the ME presentation (5.12)/(5.13) the service-terminating rates also sum up to matrix \mathbf{B} , since

$$(\mathbf{B} - \sum_{i=1}^{\infty} \mathbf{S}^{(i)} \mathbf{G}) + \sum_{i=1}^{\infty} \mathbf{S}^{(i)} \mathbf{G} = \mathbf{B}.$$

Let the vectors $\mathbf{x}_{I,\infty}/\mathbf{x}_{B,\infty}$ (of block dimension m) be the stationary distributions that the BMAP/MAP/1 queue is empty/nonempty (or idle/busy) immediately after a departure. With $\mathbf{x}_{I,n}^{ME}/\mathbf{x}_{B,n}^{ME}$, we denote the respective counterparts for the lumped Markov chain in Eq. (5.6). In PH-type notation, the outlined composition of the true inter-departure time distribution can be expressed by the initial phase distribution $\boldsymbol{\alpha}$ and the transient rate matrix \mathbf{T} as follows:

$$\begin{aligned}\boldsymbol{\alpha} &= \begin{bmatrix} \mathbf{x}_{I,\infty} & \mathbf{x}_{B,\infty} \end{bmatrix} = \begin{bmatrix} \frac{1}{\lambda} \boldsymbol{\pi}^{(1)} \mathbf{B} & \frac{1}{\lambda} \left(\sum_{i=2}^{\infty} \boldsymbol{\pi}^{(i)} \right) \mathbf{B} \end{bmatrix} \\ \mathbf{T} &= \begin{bmatrix} \widehat{\mathbf{L}} & \sum_{i=1}^{\infty} \mathbf{F}^{(i)} \\ \mathbf{0} & \mathbf{L} + \sum_{i=1}^{\infty} \mathbf{F}^{(i)} \end{bmatrix} .\end{aligned}$$

As mentioned above, matrix \mathbf{T} remains the same for the lumped model. Thus, the invariance of the inter-departure time distribution is proved, if we show that $\mathbf{x}_{I,\infty} = \mathbf{x}_{I,n}^{ME}$ and $\mathbf{x}_{B,\infty} = \mathbf{x}_{B,n}^{ME}$. Note that we have already shown that $\lambda_n^{ME} = \lambda_{\infty} \equiv \lambda$ for $n \geq 1$.

For $n > 1$, we obtain

$$\begin{aligned}\mathbf{x}_{I,n}^{ME} &= \frac{1}{\lambda} \boldsymbol{\pi}^{(1)} \mathbf{B} = \mathbf{x}_{I,\infty} \\ \mathbf{x}_{B,n}^{ME} &= \frac{1}{\lambda} \left[\left(\sum_{i=2}^{n-1} \boldsymbol{\pi}^{(i)} \right) \mathbf{B} + \boldsymbol{\pi}_n^{\infty} (\mathbf{B} - \sum_{i=1}^{\infty} \mathbf{S}^{(i)} \mathbf{G}) + \boldsymbol{\pi}_n^{\infty} \sum_{i=1}^{\infty} \mathbf{S}^{(i)} \mathbf{G} \right] = \frac{1}{\lambda} \left(\sum_{i=2}^{\infty} \boldsymbol{\pi}^{(i)} \right) \mathbf{B} \\ &= \mathbf{x}_{B,\infty} .\end{aligned}$$

For $n = 1$, we obtain

$$\begin{aligned}\mathbf{x}_{I,1}^{ME} &= \frac{1}{\lambda} \boldsymbol{\pi}_1^{\infty} (\mathbf{B} - \mathbf{F} \mathbf{G}) = \frac{1}{\lambda} \left(\sum_{i=1}^{\infty} \boldsymbol{\pi}^{(i)} \right) (\mathbf{I} - \mathbf{R}) \mathbf{B} = \frac{1}{\lambda} \boldsymbol{\pi}^{(1)} \mathbf{B} = \mathbf{x}_{I,\infty} \\ \mathbf{x}_{B,1}^{ME} &= \frac{1}{\lambda} \boldsymbol{\pi}_1^{\infty} \mathbf{F} \mathbf{G} = \frac{1}{\lambda} \left(\sum_{i=1}^{\infty} \boldsymbol{\pi}^{(i)} \right) \mathbf{R} \mathbf{B} = \frac{1}{\lambda} \left(\sum_{i=2}^{\infty} \boldsymbol{\pi}^{(i)} \right) \mathbf{B} = \mathbf{x}_{B,\infty} .\end{aligned}$$

This concludes the identity proof for the inter-departure time distribution. ■

The above theorem implies $c_\infty^2 = c_n^2$ for all $n \geq 1$ and identity for all higher moments. In the remainder of this section, we prove that representation (5.12)/(5.13), indexed by n , preserves the first $n - 2$ coefficients of correlation coefficients, i.e.,

$$\text{ACF}_n^{\text{ME}}(k) = \text{ACF}_\infty(k), \text{ for all } n \geq 3 \text{ and } 1 \leq k \leq n - 2.$$

Before stating this theorem (see Theorem 5.3), we first introduce three auxiliary lemmas.

Lemma 5.1 *Matrix $(-\mathbf{D}_{0,\infty})^{-1}$ can be rewritten as:*

$$(-\mathbf{D}_{0,\infty})^{-1} = \begin{bmatrix} \hat{\mathbf{P}}_0 & \hat{\mathbf{P}}_1 & \hat{\mathbf{P}}_2 & \hat{\mathbf{P}}_3 & \cdots \\ \mathbf{0} & \mathbf{P}_0 & \mathbf{P}_1 & \mathbf{P}_2 & \cdots \\ \mathbf{0} & \mathbf{0} & \mathbf{P}_0 & \mathbf{P}_1 & \cdots \\ \vdots & \ddots & \ddots & \ddots & \ddots \end{bmatrix}, \quad (5.15)$$

where

$$\begin{aligned} \hat{\mathbf{P}}_0 &= (-\hat{\mathbf{L}})^{-1}, \\ \hat{\mathbf{P}}_i &= \sum_{j=1}^i \hat{\mathbf{P}}_{i-j} \mathbf{F}^{(j)} (-\mathbf{L})^{-1}, \quad \text{for } i \geq 1, \\ \mathbf{P}_0 &= (-\mathbf{L})^{-1}, \\ \mathbf{P}_i &= \sum_{j=1}^i \mathbf{P}_{i-j} \mathbf{F}^{(j)} (-\mathbf{L})^{-1}, \quad \text{for } i \geq 1. \end{aligned}$$

Proof: Eq. (5.15) is directly obtained by the closed-form formula of the inverse of an upper diagonal matrix. ■

Lemma 5.2 Matrix $(-\mathbf{D}_{0,n}^{(ME)})^{-1}$ can be represented as

$$(-\mathbf{D}_{0,n}^{(ME)})^{-1} = \begin{bmatrix} \hat{\mathbf{P}}_0 & \hat{\mathbf{P}}_1 & \hat{\mathbf{P}}_2 & \cdots & \hat{\mathbf{P}}_{n-2} & \hat{\mathbf{U}}_{n-1} & \hat{\mathbf{V}}_n \\ \mathbf{0} & \mathbf{P}_0 & \mathbf{P}_1 & \cdots & \mathbf{P}_{n-3} & \mathbf{U}_{n-2} & \mathbf{V}_{n-1} \\ \mathbf{0} & \mathbf{0} & \mathbf{P}_0 & \cdots & \mathbf{P}_{n-4} & \mathbf{U}_{n-3} & \mathbf{V}_{n-2} \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{P}_0 & \mathbf{U}_1 & \mathbf{V}_2 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{U}_0 & \mathbf{V}_1 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{V}_0 \end{bmatrix}, \quad (5.16)$$

where $\hat{\mathbf{P}}_i$ and \mathbf{P}_i , $i \geq 0$, are defined in Lemma 5.1, and

$$\begin{aligned} \hat{\mathbf{U}}_{n-1} &= \sum_{j=1}^{n-1} \hat{\mathbf{P}}_{n-1-j} (\mathbf{F}^{(j)} - \sum_{l=j+2}^{\infty} \mathbf{S}^{(l)} \mathbf{G}) (-(\mathbf{L} - \sum_{l=2}^{\infty} \mathbf{S}^{(l)} \mathbf{G}))^{-1}, \\ \mathbf{U}_i &= \sum_{j=1}^i \mathbf{P}_{i-j} (\mathbf{F}^{(j)} - \sum_{l=j+2}^{\infty} \mathbf{S}^{(l)} \mathbf{G}) (-(\mathbf{L} - \sum_{l=2}^{\infty} \mathbf{S}^{(l)} \mathbf{G}))^{-1}, \quad \text{for } 1 \leq i \leq n-2, \\ \mathbf{U}_0 &= -(\mathbf{L} - \sum_{l=2}^{\infty} \mathbf{S}^{(l)} \mathbf{G})^{-1}, \\ \hat{\mathbf{V}}_n &= \sum_{j=n-1}^{\infty} \hat{\mathbf{P}}_j - \hat{\mathbf{U}}_{n-1}, \\ \mathbf{V}_i &= \sum_{j=i-1}^{\infty} \mathbf{P}_j - \mathbf{U}_{i-1}, \quad \text{for } 1 \leq i \leq n-1, \\ \mathbf{V}_0 &= \sum_{j=0}^{\infty} \mathbf{P}_j. \end{aligned}$$

Proof: By the closed-form formula of the inverse of an upper diagonal matrix, we can directly get

$$(-\mathbf{D}_{0,n}^{(ME)})^{-1} = \begin{bmatrix} \hat{\mathbf{P}}_0 & \hat{\mathbf{P}}_1 & \hat{\mathbf{P}}_2 & \cdots & \hat{\mathbf{P}}_{n-2} & \hat{\mathbf{U}}_{n-1} & \hat{\mathbf{V}}'_n \\ \mathbf{0} & \mathbf{P}_0 & \mathbf{P}_1 & \cdots & \mathbf{P}_{n-3} & \mathbf{U}_{n-2} & \mathbf{V}'_{n-1} \\ \mathbf{0} & \mathbf{0} & \mathbf{P}_0 & \cdots & \mathbf{P}_{n-4} & \mathbf{U}_{n-3} & \mathbf{V}'_{n-2} \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{P}_0 & \mathbf{U}_1 & \mathbf{V}'_2 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{U}_0 & \mathbf{V}'_1 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{V}'_0 \end{bmatrix}, \quad (5.17)$$

where

$$\begin{aligned}
\widehat{\mathbf{V}}'_n &= \sum_{j=2}^n \widehat{\mathbf{P}}_{n-j} \left(\sum_{l=j}^{\infty} \mathbf{F}^{(l)} + \sum_{l=j+1}^{\infty} \mathbf{S}^{(l)} \mathbf{G} \right) \left(-(\mathbf{L} + \sum_{l=1}^{\infty} \mathbf{F}^{(l)}) \right)^{-1} \\
&\quad + \widehat{\mathbf{U}}_{n-1} \left(\sum_{l=1}^{\infty} \mathbf{F}^{(l)} + \sum_{l=2}^{\infty} \mathbf{S}^{(l)} \mathbf{G} \right) \left(-(\mathbf{L} + \sum_{l=1}^{\infty} \mathbf{F}^{(l)}) \right)^{-1}, \\
\mathbf{V}'_i &= \sum_{j=2}^i \mathbf{P}_{i-j} \left(\sum_{l=j}^{\infty} \mathbf{F}^{(l)} + \sum_{l=j+1}^{\infty} \mathbf{S}^{(l)} \mathbf{G} \right) \left(-(\mathbf{L} + \sum_{l=1}^{\infty} \mathbf{F}^{(l)}) \right)^{-1} \\
&\quad + \mathbf{U}_{i-1} \left(\sum_{l=1}^{\infty} \mathbf{F}^{(l)} + \sum_{l=2}^{\infty} \mathbf{S}^{(l)} \mathbf{G} \right) \left(-(\mathbf{L} + \sum_{l=1}^{\infty} \mathbf{F}^{(l)}) \right)^{-1}, \quad \text{for } 2 \leq i \leq n-1, \\
\mathbf{V}'_1 &= \mathbf{U}_0 \left(\sum_{l=1}^{\infty} \mathbf{F}^{(l)} + \sum_{l=2}^{\infty} \mathbf{S}^{(l)} \mathbf{G} \right) \left(-(\mathbf{L} + \sum_{l=1}^{\infty} \mathbf{F}^{(l)}) \right)^{-1}, \quad \text{for } 1 \leq l \leq n-2, \\
\mathbf{V}'_0 &= -(\mathbf{L} + \sum_{l=1}^{\infty} \mathbf{F}^{(l)})^{-1}.
\end{aligned}$$

To prove that $\widehat{\mathbf{V}}'_n = \widehat{\mathbf{V}}_n$ and $\mathbf{V}'_i = \mathbf{V}_i$ for all $0 \leq i \leq n-1$, we only need to prove that

$$\begin{cases} \sum_{j=0}^{i-2} \mathbf{P}_j + \mathbf{U}_{i-1} + \mathbf{V}'_i = \sum_{j=0}^{\infty} \mathbf{P}_j, & \text{for all } 0 \leq i \leq n-1 \\ \sum_{j=0}^{n-2} \widehat{\mathbf{P}}_j + \widehat{\mathbf{U}}_{n-1} + \widehat{\mathbf{V}}'_n = \sum_{j=0}^{\infty} \widehat{\mathbf{P}}_j. \end{cases} \quad (5.18)$$

It is easy to show that

$$\sum_{j=0}^{\infty} \mathbf{P}_j = -(\mathbf{L} + \sum_{l=1}^{\infty} \mathbf{F}^{(l)})^{-1}, \quad \sum_{j=0}^{\infty} \widehat{\mathbf{P}}_j = (-\widehat{\mathbf{L}})^{-1}(-\mathbf{L}) - (\mathbf{L} + \sum_{l=1}^{\infty} \mathbf{F}^{(l)})^{-1}.$$

Then we prove Eq. (5.18) step by step.

(1)

$$\mathbf{V}'_0 = -(\mathbf{L} + \sum_{l=1}^{\infty} \mathbf{F}^{(l)})^{-1} = \sum_{j=0}^{\infty} \mathbf{P}_j.$$

(2)

$$\begin{aligned}
\mathbf{U}_0 + \mathbf{V}'_1 &= (-\mathbf{L} - \sum_{l=2}^{\infty} \mathbf{S}^{(l)} \mathbf{G})^{-1} + \\
&\quad (-\mathbf{L} - \sum_{l=2}^{\infty} \mathbf{S}^{(l)} \mathbf{G})^{-1} \left(\sum_{l=1}^{\infty} \mathbf{F}^{(l)} + \sum_{l=2}^{\infty} \mathbf{S}^{(l)} \mathbf{G} \right) (-\mathbf{L} + \sum_{l=1}^{\infty} \mathbf{F}^{(l)})^{-1} \\
&= (-\mathbf{L} - \sum_{l=2}^{\infty} \mathbf{S}^{(l)} \mathbf{G})^{-1} (-\mathbf{L} + \sum_{l=1}^{\infty} \mathbf{F}^{(l)}) (-\mathbf{L} + \sum_{l=1}^{\infty} \mathbf{F}^{(l)})^{-1} \\
&\quad + (-\mathbf{L} - \sum_{l=2}^{\infty} \mathbf{S}^{(l)} \mathbf{G})^{-1} \left(\sum_{l=1}^{\infty} \mathbf{F}^{(l)} + \sum_{l=2}^{\infty} \mathbf{S}^{(l)} \mathbf{G} \right) (-\mathbf{L} + \sum_{l=1}^{\infty} \mathbf{F}^{(l)})^{-1} \\
&= (-\mathbf{L} - \sum_{l=2}^{\infty} \mathbf{S}^{(l)} \mathbf{G})^{-1} (-\mathbf{L} - \sum_{l=2}^{\infty} \mathbf{S}^{(l)} \mathbf{G}) (-\mathbf{L} + \sum_{l=1}^{\infty} \mathbf{F}^{(l)})^{-1} \\
&= -(\mathbf{L} + \sum_{l=1}^{\infty} \mathbf{F}^{(l)})^{-1} = \sum_{j=0}^{\infty} \mathbf{P}_j.
\end{aligned}$$

(3) For any $0 \leq k \leq n-3$,

$$\begin{aligned}
\mathbf{U}_{k+1} + \mathbf{V}'_{k+2} &= \mathbf{U}_{k+1} + \sum_{j=2}^{k+2} \mathbf{P}_{k+2-j} \left(\sum_{l=j}^{\infty} \mathbf{F}^{(l)} + \sum_{l=j+1}^{\infty} \mathbf{S}^{(l)} \mathbf{G} \right) (-\mathbf{L} + \sum_{l=1}^{\infty} \mathbf{F}^{(l)})^{-1} \\
&\quad + \mathbf{U}_{k+1} \left(\sum_{l=1}^{\infty} \mathbf{F}^{(l)} + \sum_{l=2}^{\infty} \mathbf{S}^{(l)} \mathbf{G} \right) (-\mathbf{L} + \sum_{l=1}^{\infty} \mathbf{F}^{(l)})^{-1} \\
&= \sum_{j=1}^{k+1} \mathbf{P}_{k+1-j} (\mathbf{F}^{(j)} - \sum_{l=j+2}^{\infty} \mathbf{S}^{(l)} \mathbf{G}) (-\mathbf{L} + \sum_{l=1}^{\infty} \mathbf{F}^{(l)})^{-1} \\
&\quad + \sum_{j=2}^{k+2} \mathbf{P}_{k+2-j} \left(\sum_{l=j}^{\infty} \mathbf{F}^{(l)} + \sum_{l=j+1}^{\infty} \mathbf{S}^{(l)} \mathbf{G} \right) (-\mathbf{L} + \sum_{l=1}^{\infty} \mathbf{F}^{(l)})^{-1} \\
&= \sum_{j=1}^{k+1} \mathbf{P}_{k+1-j} \left(\sum_{l=j}^{\infty} \mathbf{F}^{(l)} \right) (-\mathbf{L} + \sum_{l=1}^{\infty} \mathbf{F}^{(l)})^{-1}.
\end{aligned}$$

Then

$$\begin{aligned}
\sum_{j=0}^k \mathbf{P}_j + \mathbf{U}_{k+1} + \mathbf{V}_{k+2} &= \sum_{j=1}^{k+1} \mathbf{P}_{k+1-j} + \sum_{j=1}^{k+1} \mathbf{P}_{k+1-j} \left(\sum_{l=j}^{\infty} \mathbf{F}^{(l)} \right) \left(-(\mathbf{L} + \sum_{l=1}^{\infty} \mathbf{F}^{(l)}) \right)^{-1} \\
&= \sum_{j=2}^{k+1} \mathbf{P}_{k+1-j} + \sum_{j=2}^{k+1} \mathbf{P}_{k+1-j} \left(\sum_{l=j}^{\infty} \mathbf{F}^{(l)} \right) \left(-(\mathbf{L} + \sum_{l=1}^{\infty} \mathbf{F}^{(l)}) \right)^{-1} \\
&\quad + \mathbf{P}_k + \mathbf{P}_k \left(\sum_{l=1}^{\infty} \mathbf{F}^{(l)} \right) \left(-(\mathbf{L} + \sum_{l=1}^{\infty} \mathbf{F}^{(l)}) \right)^{-1} \\
&= \sum_{j=2}^{k+1} \mathbf{P}_{k+1-j} + \sum_{j=2}^{k+1} \mathbf{P}_{k+1-j} \left(\sum_{l=j}^{\infty} \mathbf{F}^{(l)} \right) \left(-(\mathbf{L} + \sum_{l=1}^{\infty} \mathbf{F}^{(l)}) \right)^{-1} \\
&\quad + \sum_{j=1}^k \mathbf{P}_{k-j} \mathbf{F}^{(j)} \left(-(\mathbf{L} + \sum_{l=1}^{\infty} \mathbf{F}^{(l)}) \right)^{-1} \\
&= \sum_{j=1}^k \mathbf{P}_{k-j} + \sum_{j=1}^k \mathbf{P}_{k-j} \left(\sum_{l=j}^{\infty} \mathbf{F}^{(l)} \right) \left(-(\mathbf{L} + \sum_{l=1}^{\infty} \mathbf{F}^{(l)}) \right)^{-1} \\
&= \dots \\
&= \mathbf{P}_0 + \mathbf{P}_0 \left(\sum_{l=1}^{\infty} \mathbf{F}^{(l)} \right) \left(-(\mathbf{L} + \sum_{l=1}^{\infty} \mathbf{F}^{(l)}) \right)^{-1} \\
&= -(\mathbf{L} + \sum_{l=1}^{\infty} \mathbf{F}^{(l)})^{-1} = \sum_{j=0}^{\infty} \mathbf{P}_j.
\end{aligned}$$

(4) With the similar way as in (3), we can prove that

$$\begin{aligned}
\sum_{j=0}^{n-2} \hat{\mathbf{P}}_j + \hat{\mathbf{U}}_{n-1} + \hat{\mathbf{V}}'_n &= \hat{\mathbf{P}}_0 + \hat{\mathbf{P}}_0 \left(\sum_{l=1}^{\infty} \mathbf{F}^{(l)} \right) \left(-(\mathbf{L} + \sum_{l=1}^{\infty} \mathbf{F}^{(l)}) \right)^{-1} \\
&= (-\hat{\mathbf{L}})^{-1} (-\mathbf{L}) \left(-(\mathbf{L} + \sum_{i=1}^{\infty} \mathbf{F}^{(i)}) \right)^{-1} = \sum_{j=0}^{\infty} \hat{\mathbf{P}}_j.
\end{aligned}$$

Summing (1)-(4), Eq. (5.18) is true, and Lemma 5.2 is proved. ■

Based on Lemma 5.1 and Lemma 5.2, one can easily show that

$$\begin{cases} (-\mathbf{D}_{0,\infty})^{-1} \mathbf{e} = [\mathbf{X}_0 \mathbf{e}, \mathbf{X}_1 \mathbf{e}, \mathbf{X}_1 \mathbf{e}, \dots]^T \\ (-\mathbf{D}_{0,n}^{(ME)})^{-1} \mathbf{e} = [\mathbf{X}_0 \mathbf{e}, \mathbf{X}_1 \mathbf{e}, \dots, \mathbf{X}_1 \mathbf{e}]^T \end{cases}, \quad (5.19)$$

where $n \geq 1$, $\mathbf{X}_0 = (-\hat{\mathbf{L}})^{-1}(-\mathbf{L})(-\mathbf{L} + \sum_{i=1}^{\infty} \mathbf{F}^{(i)})^{-1}$, and $\mathbf{X}_1 = -(\mathbf{L} + \sum_{i=1}^{\infty} \mathbf{F}^{(i)})^{-1}$. For more general cases, the following lemma is introduced.

Lemma 5.3 *Matrix $((-\mathbf{D}_{0,\infty})^{-1}\mathbf{D}_{1,\infty})^k(-\mathbf{D}_{0,\infty})^{-1}\mathbf{e}$, for $0 \leq k \leq n-2$, has the form*

$$[\mathbf{Z}_0^{(k)}\mathbf{e}, \mathbf{Z}_1^{(k)}\mathbf{e}, \mathbf{Z}_2^{(k)}\mathbf{e}, \dots, \mathbf{Z}_{k+1}^{(k)}\mathbf{e}, \mathbf{Z}_{k+1}^{(k)}\mathbf{e}, \dots]^T,$$

and the matrix $((-\mathbf{D}_{0,n}^{(ME)})^{-1}\mathbf{D}_{1,n}^{(ME)})^k(-\mathbf{D}_{0,n}^{(ME)})^{-1}\mathbf{e}$ has the form

$$[\mathbf{Z}_0^{(k)}\mathbf{e}, \mathbf{Z}_1^{(k)}\mathbf{e}, \mathbf{Z}_2^{(k)}\mathbf{e}, \dots, \mathbf{Z}_k^{(k)}\mathbf{e}, \mathbf{Z}_{k+1}^{(k)}\mathbf{e}, \dots, \mathbf{Z}_{k+1}^{(k)}\mathbf{e}]^T,$$

where

$$\mathbf{Z}_0^{(0)} = \mathbf{X}_0, \quad \mathbf{Z}_1^{(0)} = \mathbf{X}_1,$$

and for all $i \geq 1$,

$$\begin{aligned} \mathbf{Z}_0^{(i)} &= \sum_{l=1}^i \mathbf{P}_l \mathbf{B} \mathbf{Z}_{l-1}^{(i-1)} + \sum_{l=i+1}^{\infty} \hat{\mathbf{P}}_l \mathbf{B} \mathbf{Z}_i^{(i-1)}, \\ \mathbf{Z}_j^{(i)} &= \sum_{l=0}^{i-j} \mathbf{P}_l \mathbf{B} \mathbf{Z}_{l-1}^{(i-1)} + \sum_{l=i+1-j}^{\infty} \mathbf{P}_l \mathbf{B} \mathbf{Z}_i^{(i-1)}, \quad 1 \leq j \leq i+1. \end{aligned}$$

Proof: We prove this Lemma using induction.

(1) From Eq. (5.19), we know the statement is true for $k = 0$, where $\mathbf{Z}_0^{(0)} = \mathbf{X}_0$ and $\mathbf{Z}_1^{(0)} = \mathbf{X}_1$.

(2) Assume that for all $0 \leq i \leq k$, where $0 \leq k \leq n-3$, the statement is true, i.e.,

$$((-\mathbf{D}_{0,\infty})^{-1}\mathbf{D}_{1,\infty})^i(-\mathbf{D}_{0,\infty})^{-1}\mathbf{e} = [\mathbf{Z}_0^{(i)}\mathbf{e}, \mathbf{Z}_1^{(i)}\mathbf{e}, \mathbf{Z}_2^{(i)}\mathbf{e}, \dots, \mathbf{Z}_i^{(i)}\mathbf{e}, \mathbf{Z}_{i+1}^{(i)}\mathbf{e}, \mathbf{Z}_{i+1}^{(i)}\mathbf{e}, \dots]^T,$$

$$((-\mathbf{D}_{0,n}^{(ME)})^{-1}\mathbf{D}_{1,n}^{(ME)})^i(-\mathbf{D}_{0,n}^{(ME)})^{-1}\mathbf{e} = [\mathbf{Z}_0^{(i)}\mathbf{e}, \mathbf{Z}_1^{(i)}\mathbf{e}, \mathbf{Z}_2^{(i)}\mathbf{e}, \dots, \mathbf{Z}_i^{(i)}\mathbf{e}, \mathbf{Z}_{i+1}^{(i)}\mathbf{e}, \dots, \mathbf{Z}_{i+1}^{(i)}\mathbf{e}]^T.$$

Then the following will prove that the statement is also true for case $i = k + 1$.

$$\begin{aligned}
& ((-\mathbf{D}_{0,\infty})^{-1}\mathbf{D}_{1,\infty})^{k+1}(-\mathbf{D}_{0,\infty})^{-1}\mathbf{e} \\
&= \begin{bmatrix} \hat{\mathbf{P}}_1\mathbf{B} & \hat{\mathbf{P}}_2\mathbf{B} & \hat{\mathbf{P}}_3\mathbf{B} & \hat{\mathbf{P}}_4\mathbf{B} & \cdots \\ \mathbf{P}_0\mathbf{B} & \mathbf{P}_1\mathbf{B} & \mathbf{P}_2\mathbf{B} & \mathbf{P}_3\mathbf{B} & \ddots \\ 0 & \mathbf{P}_0\mathbf{B} & \mathbf{P}_1\mathbf{B} & \mathbf{P}_2\mathbf{B} & \ddots \\ 0 & 0 & \mathbf{P}_0\mathbf{B} & \mathbf{P}_1\mathbf{B} & \ddots \\ \vdots & \ddots & \ddots & \ddots & \ddots \end{bmatrix} \begin{bmatrix} \mathbf{Z}_0^{(k)}\mathbf{e} \\ \mathbf{Z}_1^{(k)}\mathbf{e} \\ \mathbf{Z}_2^{(k)}\mathbf{e} \\ \vdots \\ \mathbf{Z}_{k+1}^{(k)}\mathbf{e} \\ \mathbf{Z}_{k+1}^{(k)}\mathbf{e} \\ \vdots \end{bmatrix} \\
&= \begin{bmatrix} \sum_{i=1}^{k+1} \hat{\mathbf{P}}_i \mathbf{B} \mathbf{Z}_{i-1}^{(k)} \mathbf{e} + \left(\sum_{i=k+2}^{\infty} \hat{\mathbf{P}}_i \right) \mathbf{B} \mathbf{Z}_{k+1}^{(k)} \mathbf{e} \\ \sum_{i=0}^k \mathbf{P}_i \mathbf{B} \mathbf{Z}_i^{(k)} \mathbf{e} + \left(\sum_{i=k+1}^{\infty} \mathbf{P}_i \right) \mathbf{B} \mathbf{Z}_{k+1}^{(k)} \mathbf{e} \\ \sum_{i=0}^{k-1} \mathbf{P}_i \mathbf{B} \mathbf{Z}_{i+1}^{(k)} \mathbf{e} + \left(\sum_{i=k}^{\infty} \mathbf{P}_i \right) \mathbf{B} \mathbf{Z}_{k+1}^{(k)} \mathbf{e} \\ \vdots \\ \mathbf{P}_0 \mathbf{B} \mathbf{Z}_k^{(k)} \mathbf{e} + \left(\sum_{i=1}^{\infty} \mathbf{P}_i \right) \mathbf{B} \mathbf{Z}_{k+1}^{(k)} \mathbf{e} \\ \left(\sum_{i=0}^{\infty} \mathbf{P}_i \right) \mathbf{B} \mathbf{Z}_{k+1}^{(k)} \mathbf{e} \\ \left(\sum_{i=0}^{\infty} \mathbf{P}_i \right) \mathbf{B} \mathbf{Z}_{k+1}^{(k)} \mathbf{e} \\ \vdots \end{bmatrix} = \begin{bmatrix} \mathbf{Z}_0^{(k+1)}\mathbf{e} \\ \mathbf{Z}_1^{(k+1)}\mathbf{e} \\ \mathbf{Z}_2^{(k+1)}\mathbf{e} \\ \vdots \\ \mathbf{Z}_{k+1}^{(k+1)}\mathbf{e} \\ \mathbf{Z}_{k+2}^{(k+1)}\mathbf{e} \\ \mathbf{Z}_{k+2}^{(k+1)}\mathbf{e} \\ \vdots \end{bmatrix} .
\end{aligned}$$

For $0 \leq k \leq n-3$,

$$\begin{aligned}
& ((-\mathbf{D}_{0,n}^{(\text{ME})})^{-1} \mathbf{D}_{1,n}^{(\text{ME})})^{(k+1)} (-\mathbf{D}_{0,n}^{(\text{ME})})^{-1} \mathbf{e} \\
&= \begin{bmatrix} \hat{\mathbf{P}}_1 \mathbf{B} & \hat{\mathbf{P}}_2 \mathbf{B} & \cdots & \hat{\mathbf{P}}_{n-2} \mathbf{B} & \hat{\mathbf{U}}_{n-1} \mathbf{B} & \hat{\mathbf{V}}_n (\mathbf{B} - \sum_{i=n}^{\infty} \mathbf{S}^{(i)} \mathbf{G}) & \hat{\mathbf{V}}_n \sum_{i=n}^{\infty} \mathbf{S}^{(i)} \mathbf{G} \\ \mathbf{P}_0 \mathbf{B} & \mathbf{P}_1 \mathbf{B} & \cdots & \mathbf{P}_{n-3} \mathbf{B} & \mathbf{U}_{n-2} \mathbf{B} & \mathbf{V}_{n-1} (\mathbf{B} - \sum_{i=n}^{\infty} \mathbf{S}^{(i)} \mathbf{G}) & \mathbf{V}_{n-1} \sum_{i=n}^{\infty} \mathbf{S}^{(i)} \mathbf{G} \\ 0 & \mathbf{P}_0 \mathbf{B} & \cdots & \mathbf{P}_{n-4} \mathbf{B} & \mathbf{U}_{n-3} \mathbf{B} & \mathbf{V}_{n-2} (\mathbf{B} - \sum_{i=n}^{\infty} \mathbf{S}^{(i)} \mathbf{G}) & \mathbf{V}_{n-2} \sum_{i=n}^{\infty} \mathbf{S}^{(i)} \mathbf{G} \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \ddots & \mathbf{P}_0 \mathbf{B} & \mathbf{U}_1 \mathbf{B} & \mathbf{V}_2 (\mathbf{B} - \sum_{i=n}^{\infty} \mathbf{S}^{(i)} \mathbf{G}) & \mathbf{V}_2 \sum_{i=n}^{\infty} \mathbf{S}^{(i)} \mathbf{G} \\ 0 & 0 & \cdots & 0 & \mathbf{U}_0 \mathbf{B} & \mathbf{V}_1 (\mathbf{B} - \sum_{i=n}^{\infty} \mathbf{S}^{(i)} \mathbf{G}) & \mathbf{V}_1 \sum_{i=n}^{\infty} \mathbf{S}^{(i)} \mathbf{G} \\ 0 & 0 & \cdots & 0 & 0 & \mathbf{V}_0 (\mathbf{B} - \sum_{i=n}^{\infty} \mathbf{S}^{(i)} \mathbf{G}) & \mathbf{V}_0 \sum_{i=n}^{\infty} \mathbf{S}^{(i)} \mathbf{G} \end{bmatrix} \\
& \begin{bmatrix} \mathbf{Z}_0^{(k)} \mathbf{e} \\ \mathbf{Z}_1^{(k)} \mathbf{e} \\ \mathbf{Z}_2^{(k)} \mathbf{e} \\ \vdots \\ \mathbf{Z}_k^{(k)} \mathbf{e} \\ \mathbf{Z}_{k+1}^{(k)} \mathbf{e} \\ \vdots \\ \mathbf{Z}_{k+1}^{(k)} \mathbf{e} \end{bmatrix} \\
&= \begin{bmatrix} \sum_{i=1}^{k+1} \hat{\mathbf{P}}_i \mathbf{B} \mathbf{Z}_{i-1}^{(k)} \mathbf{e} + \left(\sum_{i=k+2}^{n-2} \hat{\mathbf{P}}_i + \hat{\mathbf{U}}_{n-1} + \hat{\mathbf{V}}_n \right) \mathbf{B} \mathbf{Z}_{k+1}^{(k)} \mathbf{e} \\ \sum_{i=0}^k \mathbf{P}_i \mathbf{B} \mathbf{Z}_i^{(k)} \mathbf{e} + \left(\sum_{i=k+1}^{n-3} \mathbf{P}_i + \mathbf{U}_{n-2} + \mathbf{V}_{n-1} \right) \mathbf{B} \mathbf{Z}_{k+1}^{(k)} \mathbf{e} \\ \sum_{i=0}^{k-1} \mathbf{P}_i \mathbf{B} \mathbf{Z}_{i+1}^{(k)} \mathbf{e} + \left(\sum_{i=k}^{n-4} \mathbf{P}_i + \mathbf{U}_{n-3} + \mathbf{V}_{n-2} \right) \mathbf{B} \mathbf{Z}_{k+1}^{(k)} \mathbf{e} \\ \vdots \\ \mathbf{P}_0 \mathbf{B} \mathbf{Z}_k^{(k)} \mathbf{e} + \left(\sum_{i=1}^{n-k-3} \mathbf{P}_i + \mathbf{U}_{n-k-2} + \mathbf{V}_{n-k-1} \right) \mathbf{B} \mathbf{Z}_{k+1}^{(k)} \mathbf{e} \\ \left(\sum_{i=0}^{n-k-4} \mathbf{P}_i + \mathbf{U}_{n-k-3} + \mathbf{V}_{n-k-2} \right) \mathbf{B} \mathbf{Z}_{k+1}^{(k)} \mathbf{e} \\ \vdots \\ (\mathbf{P}_0 + \mathbf{U}_1 + \mathbf{V}_2) \mathbf{B} \mathbf{Z}_{k+1}^{(k)} \mathbf{e} \\ (\mathbf{U}_0 + \mathbf{V}_1) \mathbf{B} \mathbf{Z}_{k+1}^{(k)} \mathbf{e} \\ \mathbf{V}_0 \mathbf{B} \mathbf{Z}_{k+1}^{(k)} \mathbf{e} \end{bmatrix} = \begin{bmatrix} \mathbf{Z}_0^{(k+1)} \mathbf{e} \\ \mathbf{Z}_1^{(k+1)} \mathbf{e} \\ \mathbf{Z}_2^{(k+1)} \mathbf{e} \\ \vdots \\ \mathbf{Z}_{k+2}^{(k+1)} \mathbf{e} \\ \mathbf{Z}_{k+2}^{(k+1)} \mathbf{e} \\ \vdots \\ \mathbf{Z}_{k+2}^{(k+1)} \mathbf{e} \end{bmatrix}
\end{aligned}$$

Therefore the statement is also true for $i = k + 1$.

Summarizing the proof of induction, the statement of Lemma 5.3 is true for all $0 \leq k \leq n - 2$. ■

Theorem 5.3 $ACF_\infty(k) = ACF_n^{\text{ME}}(k)$ for all $n \geq 3$ and $1 \leq k \leq n - 2$.

Proof: According to Lemma. 5.3, we have that

$$\begin{aligned}
 \text{COV}_\infty(k) &= \lambda_\infty \pi((-D_{0,\infty})^{-1} D_{1,\infty})^k (-D_{0,\infty})^{-1} \mathbf{e} - 1 \\
 &= \lambda_\infty \left(\sum_{i=0}^k \pi^{(i)} \mathbf{Z}_i^{(k)} \mathbf{e} + \sum_{i=k+1}^{\infty} \pi^{(i)} \mathbf{Z}_{k+1}^{(k)} \mathbf{e} \right) - 1. \\
 \text{COV}_n^{\text{ME}}(k) &= \lambda_n \pi_n((-D_{0,n}^{(\text{ME})})^{-1} D_{1,n}^{(\text{ME})})^k (-D_{0,n}^{(\text{ME})})^{-1} \mathbf{e} - 1 \\
 &= \lambda_n (\pi^{(0)} \mathbf{Z}_0 \mathbf{e} + \dots + \pi^{(k)} \mathbf{Z}_k^{(k)} \mathbf{e} + \pi^{(k+1)} \mathbf{Z}_{k+1}^{(k)} \mathbf{e} + \dots \\
 &\quad + \pi^{(n-1)} \mathbf{Z}_{k+1}^{(k)} \mathbf{e} + \pi_n^\infty \mathbf{Z}_{k+1}^{(k)} \mathbf{e}) - 1 \\
 &= \lambda_n \left(\sum_{i=0}^k \pi^{(i)} \mathbf{Z}_i^{(k)} + \sum_{i=k+1}^{n-1} \pi^{(i)} \mathbf{Z}_{k+1}^{(k)} \mathbf{e} + \pi_n^\infty \mathbf{Z}_{k+1}^{(k)} \mathbf{e} \right) - 1 \\
 &= \text{COV}_\infty(k).
 \end{aligned}$$

Then

$$ACF_\infty(k) = \frac{\text{COV}_\infty(k)}{c_\infty^2} = \frac{\text{COV}_n^{\text{ME}}(k)}{(c_n^{\text{ME}})^2} = ACF_n^{\text{ME}}(k).$$
■

Remark 1 The lemma/theorem in this section can be trivially adjusted for the departure process approximations from a MAP/MAP/1 queue. The proof is the same as above given that $\mathbf{F}^{(i)}$ is equal to $\mathbf{0}$ for all $i \geq 2$, and is omitted here.

5.3.4 Experimental Results

In this section, we present a set of experimental results that show the effectiveness of the ETAQA departure approximation under different systems and utilizations. The purpose of the experiments is to illustrate that a level- n approximation of the departure process captures the exact lag coefficients up to $n - 2$ for $n \geq 3$. For all experiments, we use a dual tandem queue (see Figure 5.3) and consider performance measures under two utilization levels (30% and 80%) for both servers. We first show the autocorrelation function (ACF) of the arrival

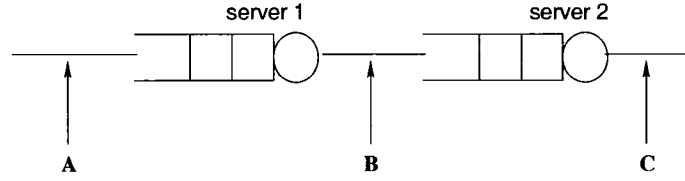


Figure 5.3: Dual tandem queues

process to the tandem queue (i.e., at point “A” in Figure 5.3) and the ACF of the departure process of the first queue (at point “B”) for different approximation level n . For the BMAP at point “A”, both the ACF of the inter-batch arrival process (see Eq. (2.7)) and the ACF, which does not ignore the zero inter-arrival times (as obtained by simulation) are given. In traffic-based decomposition, the approximation of the departure process from server 1 becomes the arrival process to the second queue. To appreciate the quality of the departure process approximation, we also illustrate the average queue length and its distribution in server 2 for different level n . Finally, in an effort to show how correlation propagates in the system, ACF of the departure process from the second server is shown, i.e., at point “C” in Figure 5.3. All analytic results are obtained via MAMSolver, a matrix-analytic methods tool [73]. To assess the quality of the approximations, simulation results are also presented. The simulation space is 100M requests. Each simulation is run 10 times with 10 different random number generator seeds. The reported small 99% confidence intervals indicate the

high accuracy of the simulations. The figures only plot the mean of the summary measures of the replications without confidence intervals to increase the readability of the graphs.

Example 1: $M^{[2]}/M/1 \rightarrow \text{Erlang-2}/1$

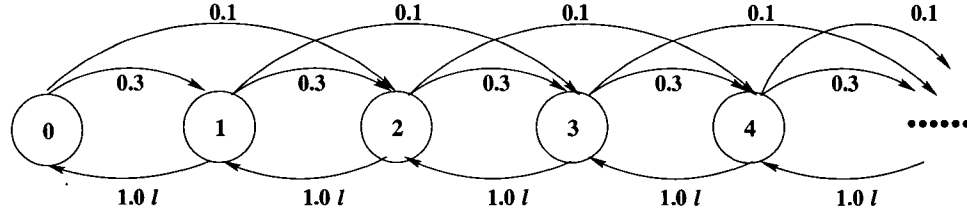


Figure 5.4: The Markov chain that models an $M^{[2]}/M/1$ queue.

In the first example, we use a simple dual tandem queue $M^{[2]}/M/1 \rightarrow \text{Erlang-2}/1$. Figure 5.4 illustrates the Markov chain that models the first queue ($M^{[2]}/M/1$), with values as assumed in the experiment. The $M^{[2]}$ arrival process is a BMAP of order 1:

$$\mathbf{D}_0^{(A)} = [-0.4] \quad , \quad \mathbf{D}_1^{(A)} = [0.3] \quad , \quad \mathbf{D}_2^{(A)} = [0.1] \quad .$$

This $M^{[2]}$ process has a mean arrival rate of 0.5 and a squared coefficient of variation (c^2) equal to 1.5. Its two ACFs, i.e., taking into account of zero inter-arrival times and ignoring zero inter-arrival times (simulation vs. analytic, respectively), are given in Figure 5.5.

The service process in the first queue is an exponential distribution with mean rate equal to $1.0l$, where l is a scaling coefficient equal to $\frac{5}{3}$ or $\frac{5}{8}$ resulting in a system lightly loaded (i.e., with 30% utilization) or highly loaded (i.e., with 80% utilization). The independent Erlang-2 services in the second queue are given in the following MAP notation with mean service rate equal to l and c^2 equal to 0.5:

$$\mathbf{D}_0^{(S_2)} = \begin{bmatrix} -2 & 2 \\ 0 & -2 \end{bmatrix} l \quad , \quad \mathbf{D}_1^{(S_2)} = \begin{bmatrix} 0 & 0 \\ 2 & 0 \end{bmatrix} l. \quad (5.20)$$

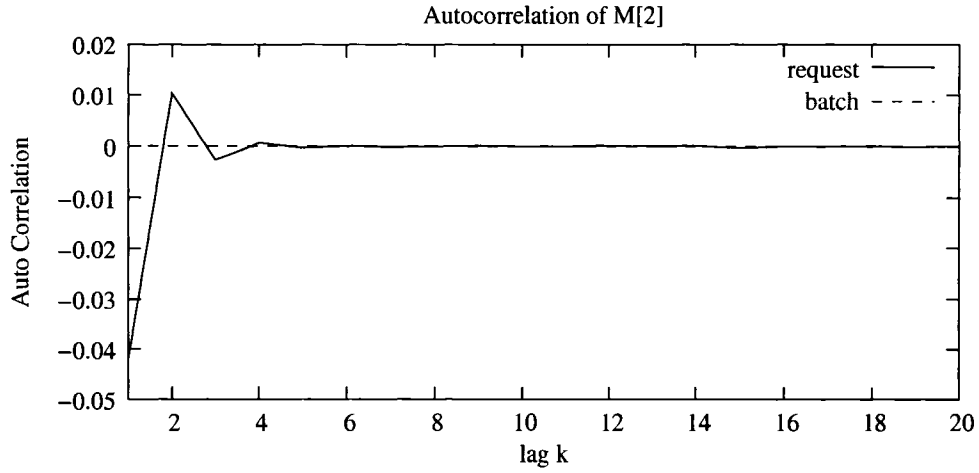


Figure 5.5: ACF of inter-arrival times of batches in the system (dashed curve) and of inter-arrival times of actual arrivals (solid curve).

Figure 5.6 gives the analytic and simulation results of this network. Figures 5.6(a) and 5.6(b) plot the ACF of the departure processes from server 1 (which are also the arrival processes to server 2) for several truncation levels (as given by parameter n) under 30% and 80% utilizations. Note that the generic form of these output approximations for the $M^{[2]}/M/1$ system is presented in Section 5.3.2.2 and represents MAP. To avoid overloading the graphs, we only plot the ACF for representative values of n . As expected, the approximation with $n = 3$ is rather poor as it only captures the lag-1 coefficient of correlation (which is negative for low load and positive for high load). Case $n = 5$ captures the first 3 coefficients and diverts after that point. Consistently, the ACFs of experiments $n = 10$ and $n = 50$ capture the correlations up to lag $k = 8$ and $k = 48$, respectively. For instance, under 30% utilization, the correlation coefficient of lag $k = 8$ is 0.00038 with the truncation models ($n = 10$ and $n = 50$) and 0.00038 ± 0.000083 for the simulation. In light load (Figure 5.6(a)), $n = 5$ appears sufficient for a good approximation. As load increases (see Figure 5.6(b)), more levels prior to truncation are needed to achieve a comparable quality of approximation. The inset graph in Figure 5.6(b) provides a better look of how close the ACFs of various

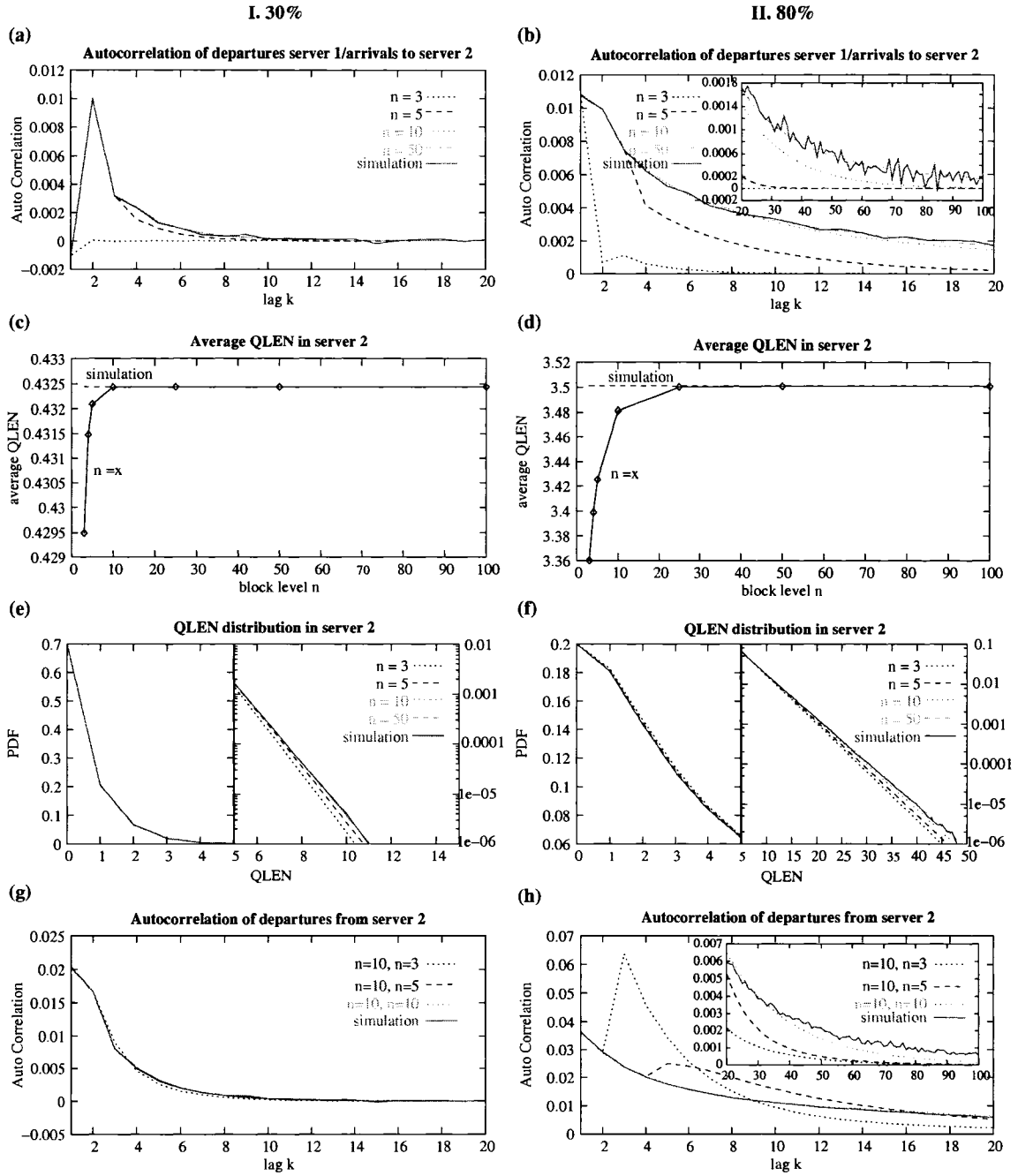


Figure 5.6: Experimental results for example 1: ACF of departures from server 1/arrivals to server 2 (a–b), mean queue length at server 2 (c–d), queue length distribution at server 2 for different approximation levels (e–f), and ACF of departures from server 2 (g–h).

departure approximations match simulation results for lags greater than 20 (such a graph is not provided for Figure 5.6(a) since all approximations only insignificantly deviate from

0 for $k \geq 10$). Also note that the higher utilization slows down the decay of the departure ACF for the same arrival process thus intensifying the correlation structure. For lower loads, the ACF of the departure process (Figure 5.6(a)) and the arrival process (Figure 5.5) bear a stronger similarity.

Figures 5.6(c) and 5.6(d) show the average queue length in server 2 as a function of the truncation level. Under 30% utilization, the approximation with $n = 5$ approaches the simulation closely (relative error of 0.08%), while $n \geq 10$ gives virtually exact results. For example, the average queue length is 0.4324 ± 0.000078 for simulation and 0.43244 for $n = 10$. Under 80% utilization, the approximations with $n \geq 25$ have a relative error less than 0.055% (the average queue lengths are 3.5020 ± 0.0018 for simulation and 3.5001 for $n = 25$). Figures 5.6(e) and 5.6(f) present the queue length distributions. Up to queue length equal to 5 we use linear scale for the y-axis. Beyond 5, we use logarithmic scale as this allows us to better distinguish the tail of the distributions for different truncation levels. In both figures, results for $n = 50$ match simulation results. Figures 5.6(e) and 5.6(f) offer the same conclusions as Figures 5.6(c) and 5.6(d): systems with higher load need higher truncation levels to meet the same accuracy requirements.

Figures 5.6(g) and 5.6(h) give the ACF of the departure process from server 2 (i.e., point “C” in Figure 5.3). We plot the simulation curve and analytic curves with approximation parameters equal to $n = 10$ for server 1 and $n = 3, 5, 10$ for server 2. The notation $n = x, n = y$ on the graph legend means that the approximation level for server 1 is equal to x and for server 2 equal to y . Since $n = 10$ for server 1 is good enough for both cases, the approximation of the departure process from server 2 may provide good results. In Figure 5.6(g), approximations with $n = 3$ at the second queue are in good agreement with simulation. For higher utilization, Figure 5.6(h) exhibits a less regular behavior.

We also note that at point “B” (see Figure 5.3), the marginal distribution is preserved for any approximation. Depending on the utilization, the c^2 of the departure process at point “B” is 1.35 (for 30%) and 1.1 (for 80%). While we conserve the flow also at point “C”, the level- n approximation of the internal traffic at “B” distorts the marginal distribution of the output approximation at the second server. At point “C”, the “ $n = 10, n = 10$ ” approximation yields the values of c^2 as 1.2513 (for 30%) and 0.7223 (for 80%).

Example 2: BMAP(3)/H₂/1 → Erlang-2/1

Here we study another dual tandem queue with a more complicated arrival process. The following BMAP of order 3 admits finite batches with sizes of up to 5. Note that $\mathbf{D}_{i+1}^{(A)} = \frac{1}{2}\mathbf{D}_i^{(A)}, 1 \leq i \leq 4$.

$$\begin{aligned} \mathbf{D}_0^{(A)} &= \begin{bmatrix} -0.290083 & 0.003728 & 0.000000 \\ 0.004349 & -0.014549 & 0.000621 \\ 0.000000 & 0.001243 & -1.207105 \end{bmatrix} \\ \mathbf{D}_1^{(A)} &= \begin{bmatrix} 0.005625 & 0.000000 & 0.142171 \\ 0.000000 & 0.004773 & 0.000170 \\ 0.619824 & 0.001364 & 0.001193 \end{bmatrix} \\ \mathbf{D}_2^{(A)} &= \begin{bmatrix} 0.002813 & 0.000000 & 0.071085 \\ 0.000000 & 0.002387 & 0.000085 \\ 0.309912 & 0.000682 & 0.000596 \end{bmatrix} \\ \mathbf{D}_3^{(A)} &= \begin{bmatrix} 0.001406 & 0.000000 & 0.035543 \\ 0.000000 & 0.001193 & 0.000043 \\ 0.154956 & 0.000341 & 0.000298 \end{bmatrix} \\ \mathbf{D}_4^{(A)} &= \begin{bmatrix} 0.000703 & 0.000000 & 0.017771 \\ 0.000000 & 0.000597 & 0.000021 \\ 0.077478 & 0.000170 & 0.000149 \end{bmatrix} \\ \mathbf{D}_5^{(A)} &= \begin{bmatrix} 0.000352 & 0.000000 & 0.008886 \\ 0.000000 & 0.000298 & 0.000011 \\ 0.038739 & 0.000085 & 0.000075 \end{bmatrix} . \end{aligned}$$

This BMAP(3) has mean rate 0.5000 and c^2 30.2335. Figure 5.7 gives the ACF of the inter-batch times as provided by (2.7) and the simulated ACF, which considers the zero inter-

arrival times of the arrival process. Figure 5.7 illustrates the noticeable difference between these correlation structures, especially the jagged shape of the analytic ACF in Figure 5.7.

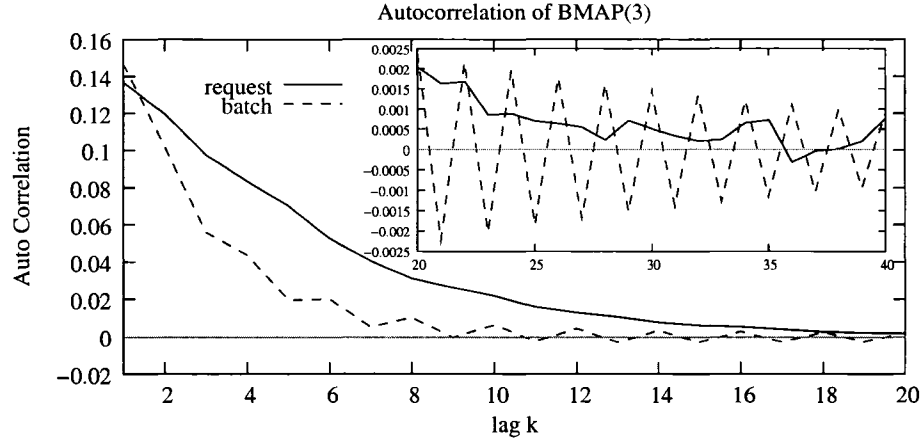


Figure 5.7: ACF of inter-arrival times of batches in the system (dashed curve) and of inter-arrival times of actual arrivals (solid curve).

The service in the first server is a two-stage hyper-exponential distribution H_2 , which we again give in MAP notation:

$$\mathbf{D}_0^{(S_1)} = \begin{bmatrix} -10 & 0 \\ 0 & -0.52632 \end{bmatrix} l, \quad \mathbf{D}_1^{(S_1)} = \begin{bmatrix} 5 & 5 \\ 0.26316 & 0.26316 \end{bmatrix} l. \quad (5.21)$$

This H_2 process has (a controllable) mean rate of l and (a fixed) c^2 of 2.6197. The Erlang-2 service at the second node is the same as in the first example (see Eq. (5.20)).

Figures 5.8(a) and 5.8(b) illustrate the autocorrelation of the departure process from server 1 for the two server utilization levels 30% and 80%. Again, approximations with $n = x$ (here ME processes and not MAPs) capture the lag correlations up to $k = x - 2$. It is interesting to observe how erratic the correlation structure of the output model may behave beyond $k = n - 2$, especially for high utilizations. Often, dips occur at $k = n$, which shrink for increasing n . The deviation between the analysis and the simulation result at lag $k = n$ is 0.4033 with $n = 3$ and 0.0912 with $n = 50$, suggesting that a larger number of levels is now required for high-quality approximations.

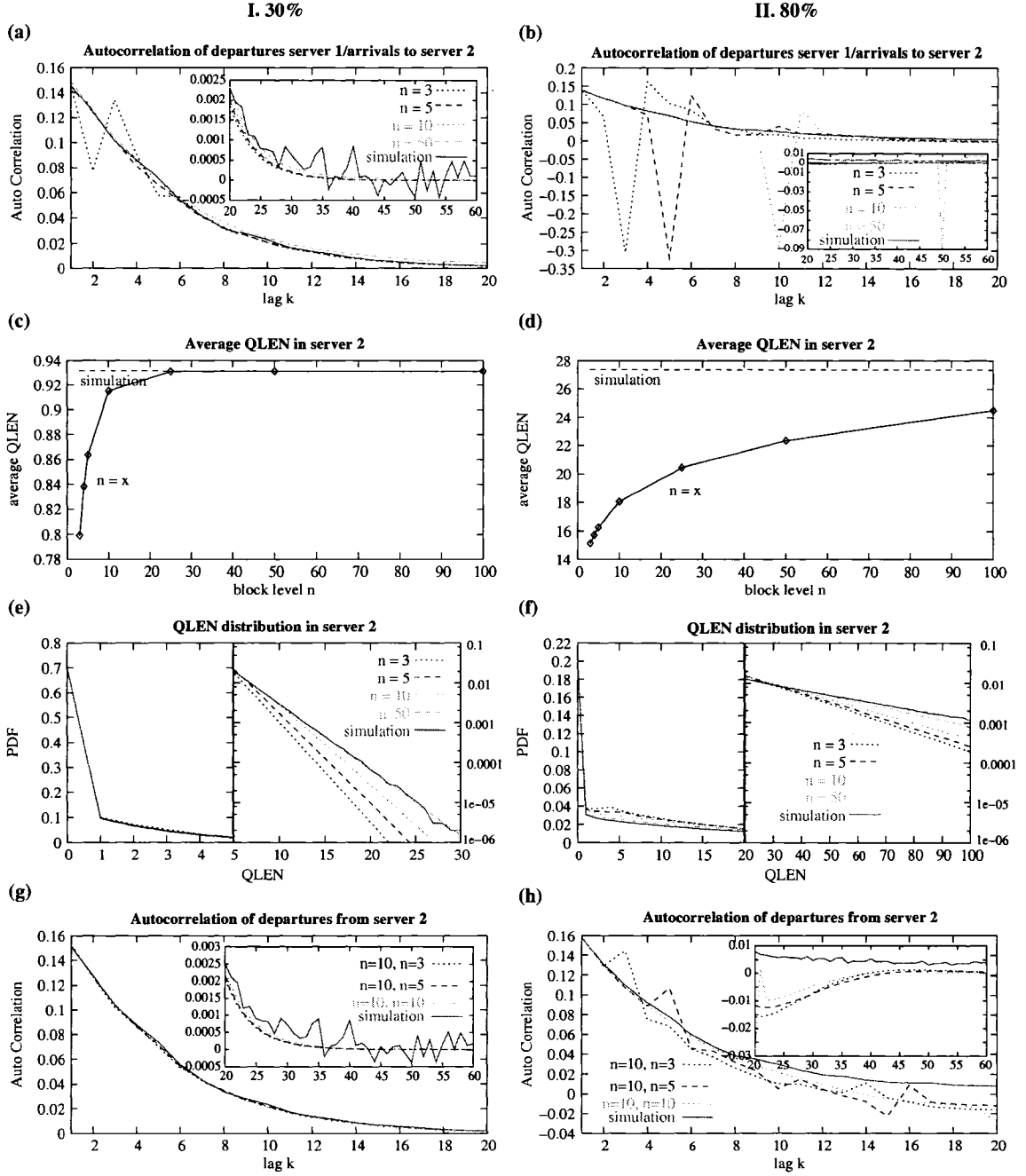


Figure 5.8: Experimental results for example 2: ACF of departures from server 1/arrivals to server 2 (a–b), mean queue length at server 2 (c–d), queue length distribution at server 2 for different approximation levels (e–f), and ACF of departures from server 2 (g–h).

Average queue lengths are displayed in Figures 5.8(c) and 5.8(d) and confirm the above observation. Here, $n = 25$ yields an accurate average queue length in the lightly loaded system with relative error of 0.05% (0.9314 ± 0.00079 for simulation and 0.9310 for $n = 25$).

Again, the output models tend to underestimate the average queue length. In Figure 5.8(d) the approximated average queue length still has a 10% relative error even when $n = 100$. Figures 5.8(e) and 5.8(f) show the queue length distribution in server 2. Comparing them with Figures 5.6(e) and 5.6(f), one can easily observe that high autocorrelation and c^2 (29.3905 for 30% utilization, and 14.8456 for 80% utilization) in the arrivals to server 2 increase the queue length significantly. Note that the x -axis in Figure 5.8(f) is up to 100, which still corresponds to a non-negligible probability value.

To plot the autocorrelation of the departure process from server 2, we use a truncation level $n = 10$ for the first server, and truncations equal to 3, 5, and 10 for the second server (see Figures 5.8(g) and 5.8(h)). Under 30% utilization, even with $n = 3$ in the second server, the ACF can be captured well in the approximation. Under 80% utilization, the approximate ACF for $n = 10$ rather closely follows the shape of the simulated ACF curve (see Figure 5.8(b)).

Example 3: BMAP(3)/MAP(2)/1 \rightarrow Erlang-2/1

To evaluate the importance of correlation in the service process (with different loads), we use the same scheme as in example 2, but substitute the renewal H_2 service in server 1 with a correlated MAP(2), which describes alternating exponential service times:

$$\mathbf{D}_0^{(S_1)} = \begin{bmatrix} -10 & 0 \\ 0 & -0.52632 \end{bmatrix} l, \quad \mathbf{D}_1^{(S_1)} = \begin{bmatrix} 0 & 10 \\ 0.52632 & 0 \end{bmatrix} l. \quad (5.22)$$

Note that this MAP(2) has the same marginal distribution as H_2 in example 2 (see Eq. (5.21)). Thus, any difference in departure process characteristics should stem from the observed correlation in the service process. This strong (but alternating) correlation oscillates between the values -0.3 and 0.3 (for the coefficients of correlation).

Figure 5.9(a) shows the autocorrelation of departures from server 1 under 30% utilization. Clearly, this ACF is dominated by the arrival process, while the service autocorrelation is reflected to some extent by the lightly oscillatory curves (note the jag in Figure 5.9(a), especially in the tail as shown in the inset figure). Figure 5.9(c) and (e) give the average queue length and queue length distribution of server 2. Observe that the oscillating autocorrelation introduced to the system by the service of queue 1 decreases queueing in the second node (the average queue length for simulation is 0.8705 ± 0.00096 as compared with 0.9314 ± 0.00079 in the previous example for this load). Figure 5.9(g) gives the ACF of departures from server 2 and illustrates that the Erlang-2 service process in server 2 smoothes the jagged behavior of the arrivals from this server.

Under heavy load, the influence of the service process is significantly more prominent, as illustrated in Figure 5.9(b). The autocorrelation of departures from server 1 drops from 0.14 in Figure 5.8(b) to 0.1 for lag $k = 1$, with pronounced subsequent oscillations. Due to the nature of the approximation (which as before are ME processes), adding a level to a small n causes inverted oscillations in the ACF for lag $k \geq n - 1$ (observe the approximation results for $n = 3, 4$ and 5). With increasing truncation levels, this behavior is attenuated and the analytic curve converges to the simulation result (note how the curve of $n = 10$ is closer to simulation than $n = 4$). Again, under heavy load, we need more levels to capture the departure process from server 1. According to Figure 5.9(d), the average queue length in server 2 of the approximation with $n = 100$ has an 11% relative error when compared with that of simulation (the numbers are 21.66 for $n = 100$ and 24.15 ± 0.21 for simulation).

Finally Figure 5.9(h) gives the autocorrelation of departures from server 2 when the approximation level at point “B” is 10. The Erlang-2 service process in server 2 increases the ACF for lag $k = 1$ and smoothes the oscillation. As we observe in Figure 5.9(b), $n = 10$

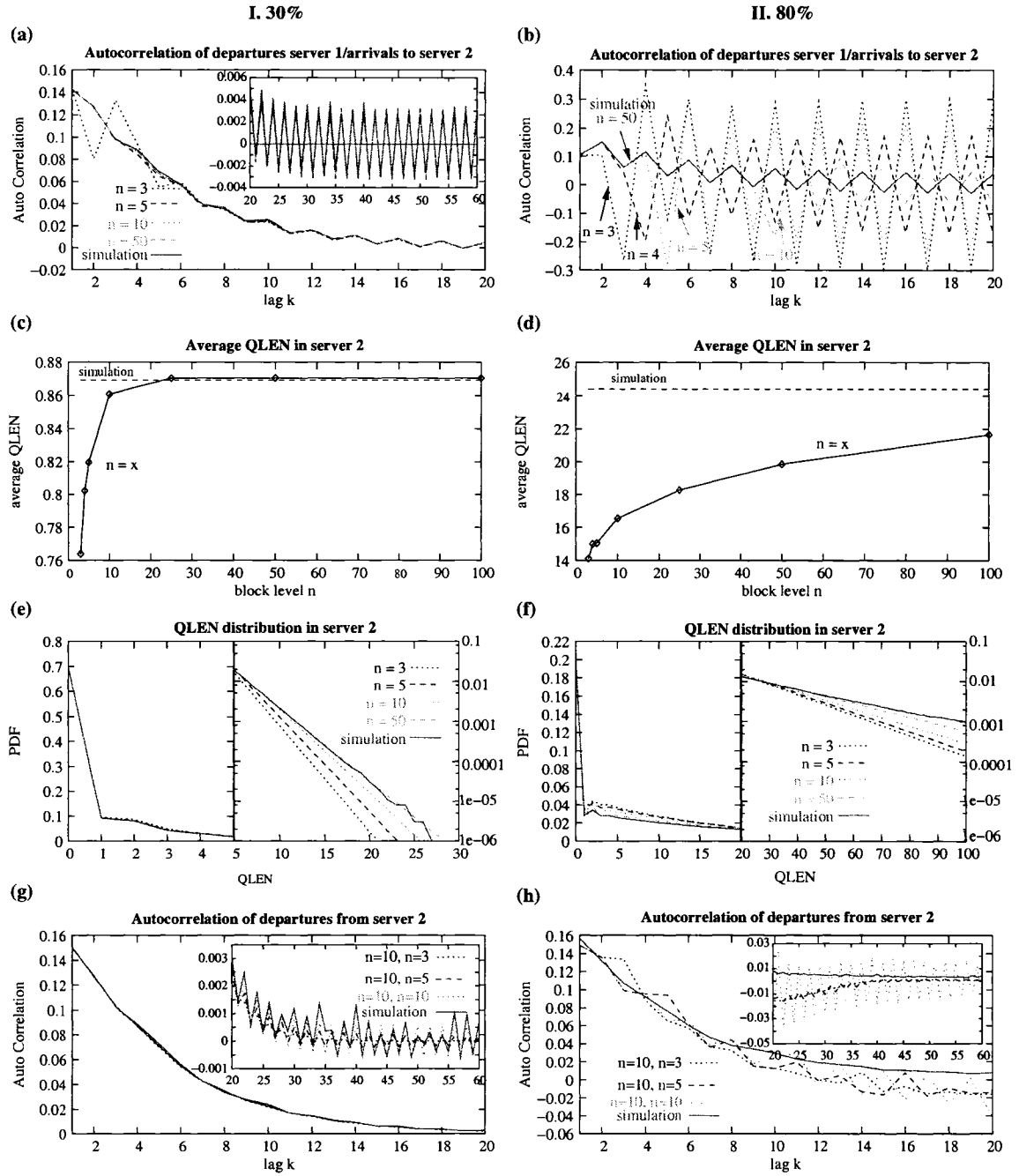


Figure 5.9: Experimental results for example 3: ACF of departures from server 1/arrivals to server 2 (a–b), mean queue length at server 2 (c–d), queue length distribution at server 2 for different approximation levels (e–f), and ACF of departures from server 2 (g–h).

does not capture well the departure process from the first server, which contributes to the differences between simulation and analytic curves in Figure 5.9(h).

5.4 Summary

In this chapter, a family of approximation models for the departure process of a BMAP/MAP/1 queue arising from ETAQA is presented. This family of approximations are indexed by a parameter n , which determines the size of the output model as $n + 1$ block levels of the M/G/1-type process. The approximations lend themselves to further use in network decomposition.

We formally proved that this approximation model can preserve the marginal distribution and the autocorrelation function up to lag $(n - 2)$ of the exact departure process of a BMAP/MAP/1 queue. This model is also shown to be efficient by experimental results, especially under light loaded systems. However, it is hard to achieve an accurate approximation for the system existing long-range dependence. Moreover, the output traffic descriptors formally belong to the class of matrix-exponential (ME) processes, which lack the physical interpretability of the rate matrices. In the next chapter, we address these drawbacks by presenting an alternative output approximation that is a MAP.

Chapter 6

Improved Truncation Models for the BMAP/MAP/1 Departure Process

In this chapter, we present an alternative family of analytic approximation models of the departure process of BMAP/MAP/1 queues that results in a MAP output process. The approximations have a form of a finite MAP and are developed based on the original (infinite) MAP departure process using lumpability arguments that are similar to the flow arguments presented in [77] for MAP/MAP/1 queues. The family of MAP approximation models that are proposed here can be broadly classified in two categories. The first category includes models that preserve the original first n levels of the infinite departure process and lump the remaining levels starting from level n into a single level. (Level numbering starts with 0.) This category preserves exactly the first $n - 1$ lag correlations of the true inter-departure process (i.e., one more accurate lag than the ME approximation in Chapter 5). Here, $n + 1$ is the size of the output model in terms of block levels. The second category includes models that do not focus on preserving the first $n - 1$ lag correlations exactly but instead aims at reducing significantly the size of the models while maintaining nearly the same performance

properties as models of much larger size in the first category. We provide proofs that both categories preserve exactly the marginal distribution of the original departure process. We also provide proofs that the first category matches exactly the first $n - 1$ lag coefficients of correlation.

Deciding the truncation level n of the output model is no trivial. As that also discussed in Chapter 5, the higher n is, the more lag coefficients of correlation of the true departure process are matched and the time series properties of the true departure process are better preserved. But using large n 's is not always possible, as this output model serves as input to downward queues. It is of paramount importance to *minimize* its size to avoid the problem of state space explosion to allow queue-by-queue analysis. To this end, we present a study that identifies the conditions that require an exact representation of higher lags and consequently identify the size of the output model. We use the probability mass of the lumped level as an indicator of the *ideal* size of the MAP departure process. Additionally, for cases that the asymptotic analysis suggests a prohibitively large output model, we propose ways to reduce it via alternative ways to further lump the state space of the output MAP without significant loss in the time series properties of the MAP and consequently in performance accuracy. Extensive experimentation illustrates the effectiveness of the proposed methods.

This chapter is organized as follows. Section 6.1 constructs the new lumpability-based approximation of the departure process that represents a MAP. Proofs that this representation matches exactly the lag coefficients of correlation up to a predefined lag are also given. Section 6.2 compares the performance of MAP approximation with that of ME approximation via experimental results. Complexity issues are also discussed here. Section 6.3 further studies the asymptotic behavior of the approximation and provides a methodology to identify the appropriate truncation level, which is validated by numerical examples in Section 6.4.

The second category of MAP approximations is given in Section 6.5. Section 6.6 summarizes this chapter.

6.1 MAP Approximation Output Models

To aid in the presentation, we first define a diagonal operator $\text{diag}(\cdot)$. Given the n -dimensional vector $\mathbf{x} = [x_0 \ x_1 \ x_2 \ \cdots \ x_n]$, $\text{diag}(\mathbf{x})$ is the n -dimensional matrix whose diagonal entries are the elements of vector \mathbf{x} and whose other entries are zero. Obviously, $\mathbf{x}(\text{diag}(\mathbf{x}))^{-1} = [1 \ 1 \ 1 \ \cdots \ 1]$ and $\mathbf{x}\text{diag}(\mathbf{y})(\text{diag}(\mathbf{x}))^{-1} = \mathbf{x}(\text{diag}(\mathbf{x}))^{-1}\text{diag}(\mathbf{y}) = \mathbf{y}$, where \mathbf{y} is a vector of same dimension as vector \mathbf{x} .

Applying basic lumpability rules (or flow arguments as in [77]), we can construct a finite Markov chain from the infinite original one \mathbf{Q}_∞ of the BMAP/MAP/1 process in Eq. (5.1). Both chains are identical up to the $(n-1)$ th level, whereas the finite chain “lumps” all states from the n th level onward. Correspondingly, the infinitesimal generator $\mathbf{Q}_n \in \mathbf{R}^{(n+1)m \times (n+1)m}$

of this new process is expressed as follows:

$$Q_n = \begin{bmatrix} \hat{L} & F^{(1)} & F^{(2)} & \dots & F^{(n-2)} & F^{(n-1)} & \sum_{i=n}^{\infty} F^{(i)} \\ B & L & F^{(1)} & \dots & F^{(n-3)} & F^{(n-2)} & \sum_{i=n-1}^{\infty} F^{(i)} \\ 0 & B & L & \ddots & \vdots & F^{(n-3)} & \sum_{i=n-2}^{\infty} F^{(i)} \\ \vdots & \vdots & \ddots & \ddots & F^{(1)} & \vdots & \vdots \\ 0 & 0 & 0 & \ddots & L & F^{(1)} & \sum_{i=2}^{\infty} F^{(i)} \\ 0 & 0 & 0 & \dots & B & L & \sum_{i=1}^{\infty} F^{(i)} \\ 0 & 0 & 0 & \dots & 0 & \text{diag}(\pi^{(n)}) (\text{diag}(\pi_n^\infty))^{-1} B & \text{diag}(\pi_{n+1}^\infty) \text{diag}(\pi_n^\infty)^{-1} B + \\ & & & & & & L + \sum_{i=1}^{\infty} F^{(i)} \end{bmatrix} \quad (6.1)$$

This representation requires knowledge of vectors $\pi^{(n)}$, $\pi_n^\infty = \sum_{i=n}^{\infty} \pi^{(i)}$ and $\pi_{n+1}^\infty = \sum_{i=n+1}^{\infty} \pi^{(i)}$, which can be computed by solving the ETAQA system Eq. (5.6) with the parameter $n + 1$. Note that $\pi^{(n)} + \pi_{n+1}^\infty = \pi_n^\infty$. The flow out of the lumped state is the original downward flow B weighted by $\text{diag}(\pi^{(n)}) (\text{diag}(\pi_n^\infty))^{-1}$. These weights are the probability ratios, with which the original chain is in states of level n . Note that in the original chain level $n - 1$ can only be reached from level n . One easily verifies that Eq. (6.1) represents a generator matrix and thus defines a true stochastic process. Therefore, filtration yields the following

MAP representation of the departure process:

$$\mathbf{D}_{0,n}^{(\text{MAP})} = \begin{bmatrix} \hat{\mathbf{L}} & \mathbf{F}^{(1)} & \mathbf{F}^{(2)} & \dots & \mathbf{F}^{(n-2)} & \mathbf{F}^{(n-1)} & \sum_{i=n}^{\infty} \mathbf{F}^{(i)} \\ \mathbf{0} & \mathbf{L} & \mathbf{F}^{(1)} & \dots & \mathbf{F}^{(n-3)} & \mathbf{F}^{(n-2)} & \sum_{i=n-1}^{\infty} \mathbf{F}^{(i)} \\ \mathbf{0} & \mathbf{0} & \mathbf{L} & \ddots & \vdots & \mathbf{F}^{(n-3)} & \sum_{i=n-2}^{\infty} \mathbf{F}^{(i)} \\ \vdots & \vdots & \ddots & \ddots & \mathbf{F}^{(1)} & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \ddots & \mathbf{L} & \mathbf{F}^{(1)} & \sum_{i=2}^{\infty} \mathbf{F}^{(i)} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{L} & \sum_{i=1}^{\infty} \mathbf{F}^{(i)} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{L} + \sum_{i=1}^{\infty} \mathbf{F}^{(i)} \end{bmatrix}, \quad (6.2)$$

$$\mathbf{D}_{1,n}^{(\text{MAP})} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{B} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{B} & \dots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \vdots & \ddots & \ddots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots & \mathbf{B} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \text{diag}(\boldsymbol{\pi}^{(n)}) (\text{diag}(\boldsymbol{\pi}_n^{\infty}))^{-1} \mathbf{B} & \text{diag}(\boldsymbol{\pi}_{n+1}^{\infty}) (\text{diag}(\boldsymbol{\pi}_n^{\infty}))^{-1} \mathbf{B} \end{bmatrix}. \quad (6.3)$$

According to Eqs. (2.5), (2.6) and (2.7) in Chapter 2, the mean rate, the squared coefficient of variation and the autocorrelation function of the above MAP are denoted as follows:

$$\begin{aligned} \lambda_n^{\text{MAP}} &= \boldsymbol{\pi}_n \mathbf{D}_{1,n}^{(\text{MAP})} \mathbf{e}, \\ (c_n^{\text{MAP}})^2 &= 2\lambda_n^{\text{MAP}} \boldsymbol{\pi}_n (-\mathbf{D}_{0,n}^{(\text{MAP})})^{-1} \mathbf{e} - 1, \\ \text{ACF}_n^{\text{MAP}}(k) &= \frac{\lambda_n \boldsymbol{\pi}_n ((-\mathbf{D}_{0,n}^{(\text{MAP})})^{-1} \mathbf{D}_{1,n}^{(\text{MAP})})^k (-\mathbf{D}_{0,n}^{(\text{MAP})})^{-1} \mathbf{e} - 1}{(c_n^{\text{MAP}})^2}, \end{aligned}$$

where $\boldsymbol{\pi}_n = [\boldsymbol{\pi}^{(0)} \boldsymbol{\pi}^{(1)} \boldsymbol{\pi}^{(2)} \dots \boldsymbol{\pi}^{(n-1)} \boldsymbol{\pi}_n^{\infty}]$ as defined in Theorem 5.1.

This truncated MAP process with level n can preserve the marginal distributions of the exact departure process and matches the autocorrelation function up to lag $(n-1)$. As a

prerequisite of the proof, we show $\lambda_n^{\text{MAP}} = \lambda_\infty$ for all $n \geq 1$:

$$\begin{aligned}
& \lambda_n^{\text{MAP}} \\
&= \pi_n \mathbf{D}_{1,n}^{(\text{MAP})} \mathbf{e} \\
&= \left(\sum_{i=1}^{n-1} \pi^{(i)} \mathbf{B} + \pi_n^\infty \text{diag}(\pi^{(n)}) (\text{diag}(\pi_n^\infty))^{-1} \mathbf{B} + \pi_n^\infty \text{diag}(\pi_{n+1}^\infty) (\text{diag}(\pi_n^\infty))^{-1} \mathbf{B} \right) \mathbf{e} \\
&= \left(\sum_{i=1}^{n-1} \pi^{(i)} \mathbf{B} + (\pi^{(n)} + \pi_{n+1}^\infty) \mathbf{B} \right) \mathbf{e} = \left(\sum_{i=1}^{\infty} \pi^{(i)} \mathbf{B} \right) \mathbf{e} \\
&= \pi_\infty \mathbf{D}_{1,\infty} \mathbf{e} = \lambda_\infty \quad .
\end{aligned}$$

Theorem 6.1 *The complete inter-departure time distribution of the true departure process is preserved by the MAP output approximation.*

Proof: With the similar proof of Theorem 5.2, we prove that the complete inter-departure time distribution is preserved by the MAP output approximation (6.2)/(6.3).

In the bottom row of Eq. (6.2)/(6.3) the service-terminating rates sum up to matrix \mathbf{B} , since

$$\begin{aligned}
& \text{diag}(\pi^{(n)}) (\text{diag}(\pi_n^\infty))^{-1} \mathbf{B} + \text{diag}(\pi_{n+1}^\infty) (\text{diag}(\pi_n^\infty))^{-1} \mathbf{B} \\
&= (\text{diag}(\pi^{(n)}) + \text{diag}(\pi_{n+1}^\infty)) (\text{diag}(\pi_n^\infty))^{-1} \mathbf{B} \\
&= \text{diag}(\pi_n^\infty) (\text{diag}(\pi_n^\infty))^{-1} \mathbf{B} = \mathbf{B} \quad .
\end{aligned}$$

Let vectors $\mathbf{x}_{I,\infty}^{\text{MAP}} / \mathbf{x}_{B,\infty}^{\text{MAP}}$ (of block dimension m) be the stationary distributions that the respective lumped Markov chain of the BMAP/MAP/1 queue (see Eq. (6.1)) is empty/nonempty (or idle/busy) immediately after a departure. Thus, the invariance of the inter-departure time distribution is proved, if we show that $\mathbf{x}_{I,\infty} = \mathbf{x}_{I,n}^{\text{MAP}}$ and $\mathbf{x}_{B,\infty} = \mathbf{x}_{B,n}^{\text{MAP}}$.

For $n > 1$, we obtain

$$\begin{aligned}
 \mathbf{x}_{I,n}^{MAP} &= \frac{1}{\lambda} \boldsymbol{\pi}^{(1)} \mathbf{B} = \mathbf{x}_{I,\infty} \\
 \mathbf{x}_{B,n}^{MAP} &= \frac{1}{\lambda} \left[\left(\sum_{i=2}^{n-1} \boldsymbol{\pi}^{(i)} \right) \mathbf{B} + \boldsymbol{\pi}_n^\infty \text{diag}(\boldsymbol{\pi}^{(n)}) (\text{diag}(\boldsymbol{\pi}_n^\infty))^{-1} \mathbf{B} + \boldsymbol{\pi}_n^\infty \text{diag}(\boldsymbol{\pi}_{n+1}^\infty) (\text{diag}(\boldsymbol{\pi}_n^\infty))^{-1} \mathbf{B} \right] \\
 &= \frac{1}{\lambda} \left(\sum_{i=2}^{\infty} \boldsymbol{\pi}^{(i)} \right) \mathbf{B} = \mathbf{x}_{B,\infty} \quad .
 \end{aligned}$$

For $n = 1$, we obtain

$$\begin{aligned}
 \mathbf{x}_{I,1}^{MAP} &= \frac{1}{\lambda} \boldsymbol{\pi}_1^\infty \text{diag}(\boldsymbol{\pi}^{(1)}) (\text{diag}(\boldsymbol{\pi}_1^\infty))^{-1} \mathbf{B} \\
 &= \frac{1}{\lambda} \boldsymbol{\pi}^{(1)} \mathbf{B} = \mathbf{x}_{I,\infty} \\
 \mathbf{x}_{B,1}^{MAP} &= \frac{1}{\lambda} \boldsymbol{\pi}_1^\infty \text{diag}(\boldsymbol{\pi}_2^\infty) (\text{diag}(\boldsymbol{\pi}_1^\infty))^{-1} \mathbf{B} \\
 &= \frac{1}{\lambda} \boldsymbol{\pi}_2^\infty \mathbf{B} = \frac{1}{\lambda} \left(\sum_{i=2}^{\infty} \boldsymbol{\pi}^{(i)} \right) \mathbf{B} = \mathbf{x}_{B,\infty} \quad .
 \end{aligned}$$

This concludes the identity proof for the inter-departure time distribution. ■

Theorem 6.2 $ACF_\infty(k) = ACF_n^{MAP}(k)$ for all $n \geq 2$ and $1 \leq k \leq n - 2$.

Proof: The proof is similar with that of Theorem 5.3 in Section 5.3.3 and is omitted here for the sake of brevity. ■

Remark 2 For the ME departure approximation of a BMAP/MAP/1 queue, the last two columns of $\mathbf{D}_{0,n}^{(ME)}$ are adjusted, but here for the MAP departure approximation, only the last one column is adjusted. As a result the inverse matrix $(-\mathbf{D}_{0,n}^{(MAP)})^{-1}$ only has the last column different from $(-\mathbf{D}_{0,\infty})^{-1}$, as shown in Lemma 6.1. Therefore in Theorem 6.2, ACF_n^{MAP} matches ACF_∞ up to lag $(n - 1)$ instead of $(n - 2)$ for all $n \geq 2$.

Lemma 6.1 Matrix $(-\mathbf{D}_{0,n}^{(MAP)})^{-1}$ can be represented as

$$(-\mathbf{D}_{0,n}^{(MAP)})^{-1} = \begin{bmatrix} \widehat{\mathbf{P}}_0 & \widehat{\mathbf{P}}_1 & \widehat{\mathbf{P}}_2 & \cdots & \widehat{\mathbf{P}}_{n-2} & \widehat{\mathbf{P}}_{n-1} & \widehat{\mathbf{W}}_n \\ 0 & \mathbf{P}_0 & \mathbf{P}_1 & \cdots & \mathbf{P}_{n-3} & \mathbf{P}_{n-2} & \mathbf{W}_{n-1} \\ 0 & 0 & \mathbf{P}_0 & \cdots & \mathbf{P}_{n-4} & \mathbf{P}_{n-3} & \mathbf{W}_{n-2} \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \ddots & \mathbf{P}_0 & \mathbf{P}_1 & \mathbf{W}_2 \\ 0 & 0 & 0 & \ddots & 0 & \mathbf{P}_0 & \mathbf{W}_1 \\ 0 & 0 & 0 & \cdots & 0 & 0 & \mathbf{W}_0 \end{bmatrix}, \quad (6.4)$$

where $\widehat{\mathbf{P}}_i$ and \mathbf{P}_i , $i \geq 0$, are defined in Lemma 5.1, and

$$\widehat{\mathbf{W}}_i = \sum_{j=i}^{\infty} \widehat{\mathbf{P}}_j, \quad \mathbf{W}_i = \sum_{j=i}^{\infty} \mathbf{P}_j.$$

Proof:

From Lemma 5.1, we get

$$\begin{aligned} & (-\mathbf{D}_{0,\infty})(-\mathbf{D}_{0,\infty})^{-1} \\ &= - \begin{bmatrix} \widehat{\mathbf{L}}\widehat{\mathbf{P}}_0 & \widehat{\mathbf{L}}\widehat{\mathbf{P}}_1 + \sum_{i=1}^1 \mathbf{F}^{(i)}\mathbf{P}_{1-i} & \widehat{\mathbf{L}}\widehat{\mathbf{P}}_2 + \sum_{i=1}^2 \mathbf{F}^{(i)}\mathbf{P}_{2-i} & \widehat{\mathbf{L}}\widehat{\mathbf{P}}_3 + \sum_{i=1}^3 \mathbf{F}^{(i)}\mathbf{P}_{3-i} & \cdots \\ 0 & \mathbf{L}\mathbf{P}_0 & \mathbf{L}\mathbf{P}_1 + \sum_{i=1}^1 \mathbf{F}^{(i)}\mathbf{P}_{1-i} & \mathbf{L}\mathbf{P}_2 + \sum_{i=1}^2 \mathbf{F}^{(i)}\mathbf{P}_{2-i} & \cdots \\ 0 & 0 & \mathbf{L}\mathbf{P}_0 & \mathbf{L}\mathbf{P}_1 + \sum_{i=1}^1 \mathbf{F}^{(i)}\mathbf{P}_{1-i} & \cdots \\ \vdots & \ddots & \ddots & \ddots & \ddots \end{bmatrix} \\ &= \mathbf{I}, \end{aligned}$$

Where \mathbf{I} is an identity matrix with appropriate dimension. So that

$$\widehat{\mathbf{L}}\widehat{\mathbf{P}}_0 = -\mathbf{I}, \quad \mathbf{L}\mathbf{P}_0 = -\mathbf{I}, \quad \widehat{\mathbf{L}}\widehat{\mathbf{P}}_i + \sum_{j=1}^i \mathbf{F}^{(j)}\mathbf{P}_{i-j} = \mathbf{0}, \quad \mathbf{L}\mathbf{P}_i + \sum_{j=1}^i \mathbf{F}^{(j)}\mathbf{P}_{i-j} = \mathbf{0} \quad \text{for all } i \geq 1. \quad (6.5)$$

Let us define matrix \mathbf{A} as

$$\mathbf{A} = \begin{bmatrix} \widehat{\mathbf{P}}_0 & \widehat{\mathbf{P}}_1 & \widehat{\mathbf{P}}_2 & \cdots & \widehat{\mathbf{P}}_{n-2} & \widehat{\mathbf{P}}_{n-1} & \widehat{\mathbf{W}}_n \\ \mathbf{0} & \mathbf{P}_0 & \mathbf{P}_1 & \cdots & \mathbf{P}_{n-3} & \mathbf{P}_{n-2} & \mathbf{W}_{n-1} \\ \mathbf{0} & \mathbf{0} & \mathbf{P}_0 & \cdots & \mathbf{P}_{n-4} & \mathbf{P}_{n-3} & \mathbf{W}_{n-2} \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \ddots & \mathbf{P}_0 & \mathbf{P}_1 & \mathbf{W}_2 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} & \mathbf{P}_0 & \mathbf{W}_1 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{0} & \mathbf{W}_0 \end{bmatrix}.$$

Then

$$\begin{aligned}
 & (-\mathbf{D}_{0,n}^{(\text{MAP})})\mathbf{A} \\
 &= - \begin{bmatrix} \widehat{\mathbf{L}}\widehat{\mathbf{P}}_0 & \widehat{\mathbf{L}}\widehat{\mathbf{P}}_1 + \mathbf{F}^{(1)}\mathbf{P}_0 & \cdots & \widehat{\mathbf{L}}\widehat{\mathbf{P}}_{n-1} + \sum_{i=1}^{n-1} \mathbf{F}^{(i)}\mathbf{P}_{n-1-i} & \widehat{\mathbf{L}}\widehat{\mathbf{W}}_n + \sum_{i=1}^{n-1} \mathbf{F}^{(i)}\mathbf{W}_{n-i} + \sum_{i=n}^{\infty} \mathbf{F}^{(i)}\mathbf{W}_0 \\ \\ 0 & \mathbf{L}\mathbf{P}_0 & \ddots & \mathbf{L}\mathbf{P}_{n-2} + \sum_{i=1}^{n-2} \mathbf{F}^{(i)}\mathbf{P}_{n-2-i} & \mathbf{L}\mathbf{W}_{n-1} + \sum_{i=1}^{n-2} \mathbf{F}^{(i)}\mathbf{W}_{n-1-i} + \sum_{i=n-1}^{\infty} \mathbf{F}^{(i)}\mathbf{W}_0 \\ \\ 0 & 0 & \ddots & \mathbf{L}\mathbf{P}_{n-3} + \sum_{i=1}^{n-3} \mathbf{F}^{(i)}\mathbf{P}_{n-3-i} & \mathbf{L}\mathbf{W}_{n-2} + \sum_{i=1}^{n-3} \mathbf{F}^{(i)}\mathbf{W}_{n-2-i} + \sum_{i=n-2}^{\infty} \mathbf{F}^{(i)}\mathbf{W}_0 \\ \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ \\ 0 & 0 & \ddots & \mathbf{L}\mathbf{P}_0 & \mathbf{L}\mathbf{W}_1 + \sum_{i=1}^{\infty} \mathbf{F}^{(i)}\mathbf{W}_0 \\ \\ 0 & 0 & \cdots & 0 & \left(\mathbf{L} + \sum_{i=1}^{\infty} \mathbf{F}^{(i)} \right) \mathbf{W}_0 \end{bmatrix}.
 \end{aligned}$$

From above, we see that $(-\mathbf{D}_{0,n}^{(\text{MAP})})\mathbf{A} = \mathbf{I}$ for the all columns except for the last one. To prove

that this identity also holds for the last column, we first note that the closed-form formula of the inverse of an upper diagonal matrix implies $\mathbf{W}_0 \equiv \sum_{i=0}^{\infty} \mathbf{P}_i = -(\mathbf{L} + \sum_{i=1}^{\infty} \mathbf{F}^{(i)})^{-1}$. This shows $-(\mathbf{L} + \sum_{i=1}^{\infty} \mathbf{F}^{(i)}) \mathbf{W}_0 = \mathbf{I}$. The first element in the last column of $(-\mathbf{D}_{0,n}^{(\text{MAP})})\mathbf{A}$ can be manipulated to

$$\begin{aligned}
& \widehat{\mathbf{L}}\widehat{\mathbf{W}}_n + \sum_{i=1}^{n-1} \mathbf{F}^{(i)}\mathbf{W}_{n-i} + \sum_{i=n}^{\infty} \mathbf{F}^{(i)}\mathbf{W}_0 \\
&= \widehat{\mathbf{L}} \sum_{j=n}^{\infty} \widehat{\mathbf{P}}_j + \sum_{i=1}^{n-1} \mathbf{F}^{(i)} \sum_{j=n-i}^{\infty} \mathbf{P}_j + \sum_{i=n}^{\infty} \mathbf{F}^{(i)} \sum_{j=0}^{\infty} \mathbf{P}_j \\
&= \sum_{j=n}^{\infty} \widehat{\mathbf{L}}\widehat{\mathbf{P}}_j + \sum_{i=1}^{n-1} \mathbf{F}^{(i)} \sum_{j=n}^{\infty} \mathbf{P}_{j-i} + \sum_{i=n}^{\infty} \mathbf{F}^{(i)} \sum_{j=n}^{\infty} \mathbf{P}_{j-n} \\
&= \sum_{j=n}^{\infty} \widehat{\mathbf{L}}\widehat{\mathbf{P}}_j + \sum_{j=n}^{\infty} \sum_{i=1}^{n-1} \mathbf{F}^{(i)} \mathbf{P}_{j-i} + \sum_{j=n}^{\infty} \sum_{i=n}^j \mathbf{F}^{(i)} \mathbf{P}_{j-i} \\
&= \sum_{j=n}^{\infty} \left(\widehat{\mathbf{L}}\widehat{\mathbf{P}}_j + \sum_{i=1}^j \mathbf{F}^{(i)} \mathbf{P}_{j-i} \right) = \sum_{j=n}^{\infty} \mathbf{0} = \mathbf{0} .
\end{aligned}$$

The last line is due to Eq. (6.5). In the same way, we get $\mathbf{L}\mathbf{W}_k + \sum_{i=1}^{k-1} \mathbf{F}^{(i)}\mathbf{W}_{k-i} + \sum_{i=k}^{\infty} \mathbf{F}^{(i)}\mathbf{W}_0 = \mathbf{0}$ for $k = 1, \dots, n-1$ so that indeed $(-\mathbf{D}_{0,n}^{(\text{MAP})})\mathbf{A} = \mathbf{I}$ and thus $(-\mathbf{D}_{0,n}^{(\text{MAP})})^{-1} = \mathbf{A}$, which completes the proof. \blacksquare

6.2 Comparison of MAP Approximation and ME Approximation

6.2.1 Experimental Comparison

The experiments of this section show the effectiveness of the MAP output models under different systems and utilizations and compare its performance to the results of ME representation. The very same systems are investigated in Section 5.3.4 with ME output approx-

imations (5.12)/(5.13) only. Since we have proved that the marginal distribution and the first lags of the autocorrelation structure of the inter-departure times are preserved by both families of approximations, we focus on the behavior of the correlation structure beyond the invariance threshold and the performance impact of the approximations of the inter-queue process on the downstream node. In each experiment, we show the autocorrelation function (ACF) of the departure process from server 1 (i.e., at point “B” in Figure 5.3) and the mean queue length (QLEN) at server 2 for selected truncation levels n .

Example 1: $M^{[2]}/M/1 \rightarrow \text{Erlang-2}/1$

The first example represents the dual tandem queue $M^{[2]}/M/1 \rightarrow \text{Erlang-2}/1$. The $M^{[2]}$ arrival process is a BMAP of order/dimension 1 with rates -0.3 and -0.1 for batch arrivals of size 1 and 2, respectively. This $M^{[2]}$ process has a mean arrival rate of 0.5 and a squared coefficient of variation (c^2) equal to 1.5. Its inter-batch ACF equals zero, while the ACF, which takes into account the “zero inter-arrival times”, has a negative first coefficient of around -0.04 and a positive second coefficient of around 0.01 (see Figure 5.5). The service processes are an exponential distribution (c^2 of 1) at the first queue and Erlang-2 distribution (c^2 of 0.5) at the second one. The rates of the service processes of the two nodes are scaled simultaneously in order to achieve light system load (30% utilization) and high system load (80% utilization) across both nodes. These functions are depicted in Section 5.3.4.

Figure 6.1 gives analytic and simulation results of this network. Figures 6.1(a) and 6.1(b) plot the ACFs of the departure processes from server 1 (which are also the arrival processes to server 2) for several truncation levels of the MAP output model (6.2)/(6.3) under 30% and 80% utilizations. The chosen values of the truncation parameter n are the same as in the corresponding example in Section 5.3.4 for the ME output representation (5.12)/(5.13). The

inset graph in Figure 6.1(b) provides a better look of how close the ACFs of the departure approximations match simulation results for lags greater than 40. Comparing low (a) and high (b) load situations, we observe the stronger long-term correlations in the departure process for the system in high load.

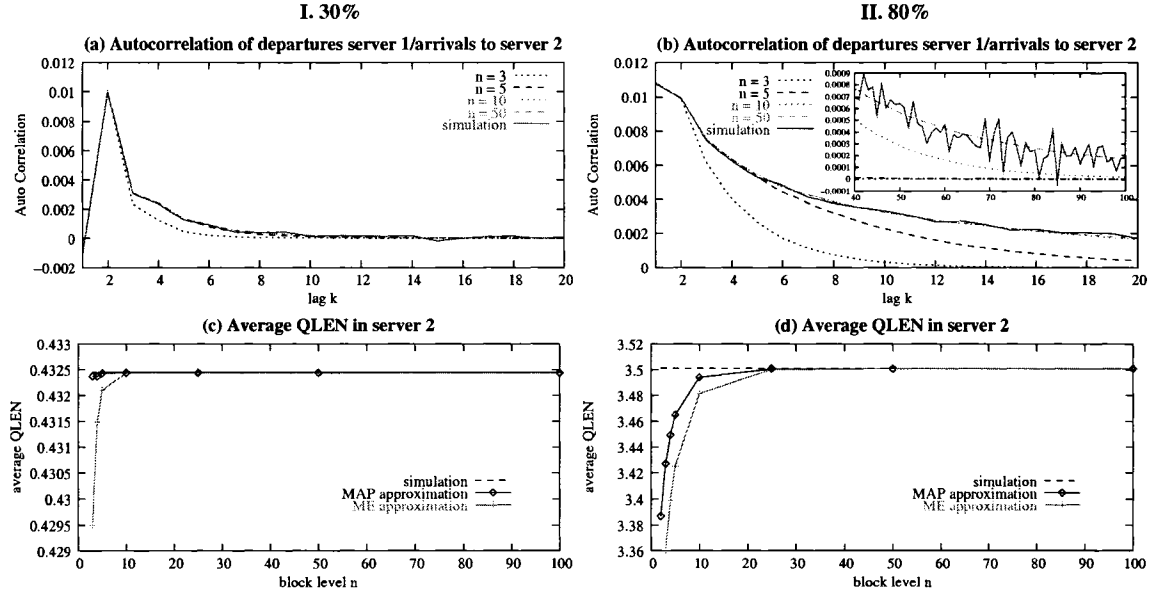


Figure 6.1: Experimental results of MAP approximation for example 1: ACF of departures from server 1/arrivals to server 2 (a–b), mean queue length at server 2 (c–d).

As proven, the ACF of the MAP output model with parameter n matches exactly the first $(n - 1)$ lag coefficients, e.g., cases $n = 3$ and $n = 5$ capture the correlation up to lag 2 and 4, respectively. The MAP approximation not only matches one more coefficient than the ME representation, but also the tail of its ACF deviates less from simulation results. Given that both approximations preserve the marginal distribution of the original departure process, we now explore how matching one more lag affects performance results for server 2. Figures 6.1(c) and 6.1(d) plot the average queue length (QLEN) at server 2 as a function of the truncation level n of the departure approximation from server 1. Results for both the MAP and the ME output model are shown for comparison. Both approximations

generally underestimate the mean queue length. For the MAP output model under light load, truncation $n = 3$ already gives a relative error of only -0.015% compared with simulation, and $n \geq 5$ yields exactly average queue lengths. The ME approximation results in virtually exact results only when $n \geq 10$. Under 80% utilization, both approximations have higher errors: for case $n = 10$, the relative error of -0.6% with the ME approximation is reduced to -0.2% with the MAP approximation. In this example, MAP output models with very small n appear sufficient for good approximations of the downstream mean queue length, where slightly larger n are required for the ME output model to achieve the same accuracy.

Example 2: BMAP(3)/H₂/1 → Erlang-2/1

The second dual tandem queue has a more complicated external BMAP, which is of order 3 and admits finite batches with sizes of up to 5. Its mean rate is 0.5 and its c^2 30.2335. Both inter-batch and inter-arrival ACFs start around 0.14 (positive lag-1 coefficient) and decay to negligible values (i.e., less than 0.0025 in absolute terms) within the first 20 lags (see Figure 5.7). Also, the service process at the first server is changed with respect to the first example, namely to a two-stage hyper-exponential distribution H₂ with rate ratio of 5.2632 and c^2 of 2.6197. Again, in this experiment service rates are adjusted to obtain the desired utilization levels at both servers. More details on this BMAP and the H₂ service are found in Section 5.3.4.

In Figure 6.2, we show analogous plots as for example 1. The ACFs of the departure process from server 1 in Figures 6.2(a) and 6.2(b) computed with the MAP output model can again be compared with the corresponding figures in Section 5.3.4 for the ME output representation. Especially for high loads, the level- n ME representation suffered erratic dips for the lag- n coefficient of correlation with significant deviations (see Figure 5.8(b)). These

dips disappear with the MAP output model, which makes the overall ACF approximation smoother and accounts for an improved tail behavior.

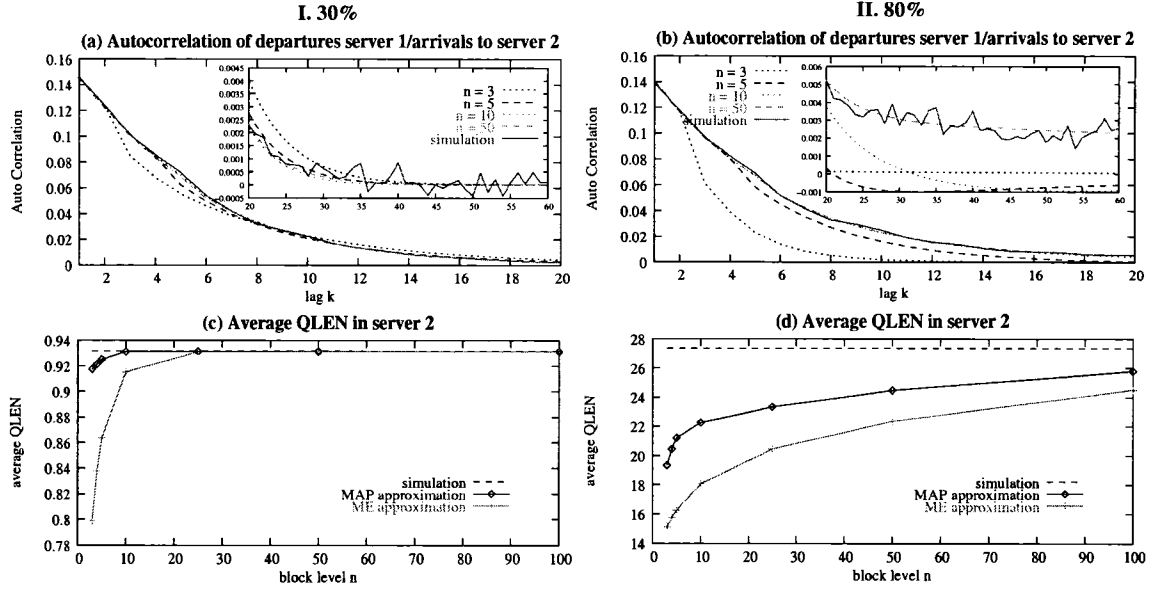


Figure 6.2: Experimental results of MAP approximation for example 2: ACF of departures from server 1/arrivals to server 2 (a–b), mean queue length at server 2 (c–d).

Since additionally the lag $(n - 1)$ correlation coefficient is matched exactly, level- n approximations with MAPs are noticeably more accurate than their ME counterparts, also with respect to the mean queue lengths at server 2 (see Figures 6.2(c) and 6.2(d)). This is especially true under low load (c): for $n = 3$, the MAP approximation only yields a relative error of -1.5% and an exactly average QLEN with $n = 10$. Under high load (d) with $n = 100$, the MAP approximation reduces the relative error to -5.7% from -11% with the ME approximation. From Figure 6.2(d) we see for both families of approximations that mean queue lengths only slowly converge to the simulated value in high load. In both cases, it requires more than 100 levels ($n > 100$) to achieve fair approximations to the mean queue length.

Example 3: BMAP(3)/MAP(2)/1 \rightarrow Erlang-2/1

This dual tandem queue differs from the one in the previous section only in the corre-

lation structure of the service process at server 1. The exponential phases of the two-stage hyper-exponential distribution H_2 are not chosen with equal probabilities, but alternate with each service. This defines a MAP service process of order 2, which has the same marginal distribution H_2 , but a non-zero ACF, which oscillates between -0.3 and 0.3 . More details on this MAP are found in Section 5.3.4.

The impact of the introduced service correlations at server 1 as compared to example 2 can be studied in Figure 6.3. Figures 6.3(a) and 6.3(b) clearly demonstrate how the service oscillations become more and more visible in the ACF of the departure process from server 1 with increasing utilization. Observe also that this oscillating autocorrelation decreases queueing in the second node as compared with the previous example. Besides the features mentioned before, another qualitative difference between the MAP and the ME output models can be identified: in the discussion of the corresponding example in Section 5.3.4, the ME output model gave rise to out-of-sync oscillations in the ACF of the output approximation for specific truncation levels ((see Figure 5.9(b))). For the MAP output model, Figure 6.3(b) shows that all oscillations are in the same phase independent of the value of n . Quantitatively, the ACF of the MAP output model also outperforms the ACF of the ME approximation of the same order. For example, the maximal absolute deviation of the ACFs from the simulated ACF occurs in both cases for lag 3 with level $n = 3$ and takes the value 0.04 in the MAP case and 0.2 in the ME case.

Figure 6.3(c) illustrates that the MAP approximation with small values of n can provide accurate average queue lengths in the second queue under 30% utilization. Mean queue lengths in high load are not as easily approximated. Under 80% utilization (see Figure 6.3(d)), the relative error in case $n = 100$ is still around -8% for the MAP approximation, reduced from -11% for the ME approximation. Generally, the approximation

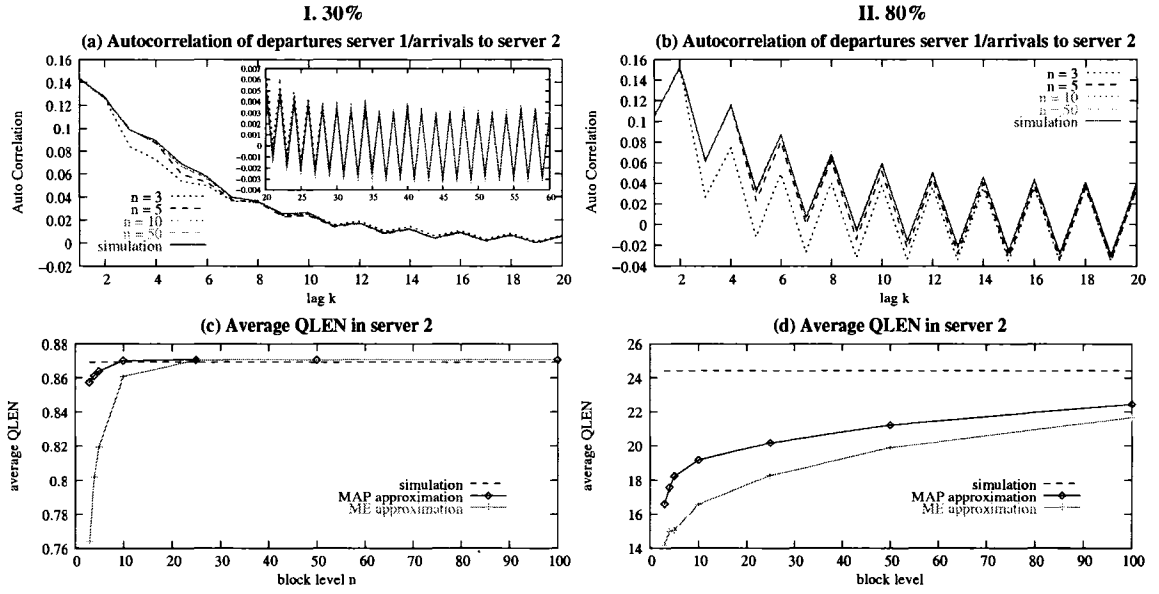


Figure 6.3: Experimental results of MAP approximation for example 3: ACF of departures from server 1/arrivals to server 2 (a–b), mean queue length at server 2 (c–d).

behavior for the downstream QLEN is similar to the previous example (see Figure 6.2), except that the accuracy gain of the MAP output model is even more mitigated in high loads.

6.2.2 Complexity Issues

We first compare the computational effort related to the ME representation (5.12)/(5.13) and the MAP output model (6.2)/(6.3). For this discussion, we assume that the lag k autocorrelation of the inter-departure times of a BMAP/MAP/1 queue with true batches needs to be computed exactly. The autocorrelation function is simply the numerator in equation (2.7). At the end, we also outline the time complexity using the BMAP/GI/1 approach in [24] to compute the lag k autocorrelation for a BMAP/PH/1 system of identical dimensions. In all three cases, ACFs of lag $i < k$ come at essentially no extra cost in the course of lag k computations.

6.2.2.1 ME Output Model

For both ME and MAP representations, the level dimension of the involved block matrices is $m = m_A m_S$. However, with true batches, the ME representation to approximate the departure process requires one more level for the exact lag k autocorrelation computation. The truncation parameter of representation (5.12)/(5.13) must be chosen as $n = k + 2$ so that the ME output model assumes the total order of $m_{\text{ME}} = (k + 3)m = (k + 3)m_A m_S$.

The time complexity in constructing the ME representation (5.12)/(5.13) is dominated by computing matrix \mathbf{G} of dimension m (see Eq. (5.3)). This matrix is often sparse and can be efficiently computed by matrix-analytic techniques [42, 59] with complexity $O(m^3)$. The series, which appear in (5.12)/(5.13), are usually finite sums due to batches of limited size. In any case, the summations of (5.4) are efficiently computed via backward recursions $\mathbf{S}^{(j)} = \mathbf{F}^{(j)} + \mathbf{S}^{(j+1)}\mathbf{G}$ for $j = b_{\text{max}} - 1, \dots, 1$, where b_{max} denotes the maximal batch size. Note that at this point, without any further matrix-matrix multiplications, the complete ME output model is at hand.

In order to compute the lag k autocorrelation (according to the numerator in (2.7)), one has to deal with vectors and matrices of dimension $m_{\text{ME}} = (k + 3)m_A m_S$. Both obtaining the inverse of $\mathbf{D}_{0,k+2}^{(\text{ME})}$ and the ETAQA stationary solution $\boldsymbol{\pi}_{k+2}$ in Eq. (5.6), which becomes $\boldsymbol{\pi}_{\text{MAP}}$ in Eq. (2.7), are rather expensive operations of worst-case complexity $O(m_{\text{ME}}^3) = O(((k + 3)m)^3) = O(k^3 m^3)$. Note, however, that the M/G/1-type structure of involved matrices and their sparsity allows efficient implementations to lower the complexity significantly (i.e., k^2 instead of k^3 and $(m \times \# [\text{non-zero entries in sum of all block matrices in } \mathbf{Q}_\infty \text{ plus } \mathbf{G}])$ instead of m^3 , see [72]). Explicit expressions for $(\mathbf{D}_{0,k+2}^{(\text{ME})})^{-1}$, similar to the ones given in Lemma 5.1, are found in [99]. Finally, with $(k + 2)$ additional vector-matrix and one more

matrix-matrix multiplication, the lag k autocorrelation is obtained.

6.2.2.2 MAP Output Model

The main advantage of the MAP output model (6.2)/(6.3) with respect to efficiency consists in that it requires one block level less, i.e., the truncation parameter can be chosen as $n = k + 1$ and the model dimension is $m_{\text{MAP}} = (k + 2)m = (k + 2)m_A m_S$. Further use of such a model in network decomposition and the computation of the lag k autocorrelation profit from this fact which has to be paid for by a slightly more expensive construction of the MAP model. At a first glance at the model representation (6.2)/(6.3), this construction even seems simpler: matrices $\mathbf{S}^{(j)}$ and related series expressions do not occur, neither does matrix \mathbf{G} . Still, exactly the same block matrices are needed as in the ME case, since vectors $\boldsymbol{\pi}^{(k+1)}$ and $\boldsymbol{\pi}_{k+2}^\infty$ have to be computed from the ETAQA matrix $\hat{\mathbf{Q}}_{k+2}$ from (5.6). Note that matrix $\hat{\mathbf{Q}}_{k+2}$ has the same dimensions as the ME output model of Section 6.2.2.1. The identical system of linear equations has to be solved as for $\boldsymbol{\pi}_{k+2}$ above, and this is exactly the overhead in the *construction* of the MAP output model¹.

When computing the lag k autocorrelation with Eq. (6.2)/(6.3) for $n = k + 1$, the “overhead computation” addressed before will be reused in an efficient implementation to extract the stationary solution $\boldsymbol{\pi}_{\text{MAP}}$ of the MAP for Eq. (2.7). Considering this, the MAP approach actually outperforms the ME approach by the difference of dealing with vectors and matrices of dimension m_{MAP} instead of m_{ME} in the following situations:

- when inverting matrix $\mathbf{D}_{0,k+1}^{(\text{MAP})}$ instead of $\mathbf{D}_{0,k+2}^{(\text{ME})}$,
- for $(k + 2)$ vector-matrix multiplications, and

¹Here, we ignore multiplications/inversions of diag-matrices in (6.2)/(6.3), which boil down to scalar-matrix multiplications.

- for 1 matrix-matrix multiplication.

Formally, while the construction of the MAP output model has complexity $O((m_{\text{MAP}} + m)^3) = O(((k + 3)m)^3) = O(k^3 m^3)$ (ETAQA solution), the additional effort for the lag k autocorrelation amounts to $O(m_{\text{MAP}}^3) = O(((k + 2)m)^3) = O(k^3 m^3)$. Again, exploiting sparsity and the M/G/1-type structures yields similar gains as pointed out in the ME case. Overall, a complexity of $O(k^2 m \times \# [\text{non-zero entries in sum of all block matrices in } \mathbf{Q}_\infty \text{ plus } \mathbf{G}])$ may be achieved.

Generally, one not only constructs an output model, but also further processes it – for computing performance characteristics or for employing it in downstream queue analyses. Especially in the latter case, where the order of the output model usually enters the calculations multiplicatively, the MAP output model is clearly advantageous. This superiority (due due lower order) vanishes without proper arrival batches, e.g., for the MAP/MAP/1 queue, where the ME representation might be preferred. Recall that all suggested output models also preserve the marginal distribution of the original departure process.

6.2.2.3 Departure Lag Coefficients via the BMAP/GI/1 Approach

Although our approach mainly aims at providing output models, it may still be worthwhile to compare the efficiency with another methodology that might as well be used to compute the lag k autocorrelation for BMAP/PH/1 queues (but does not deliver an output model). This methodology by Ferng/Chang [24] is based on the BMAP/GI/1 framework and as such does not admit correlated service processes. Ferng and Chang mostly deal with vectors and matrices of the BMAP order m_A , as they arise from matrix-analytic techniques for M/G/1-type queues embedded at the departure epochs. Clearly, the algorithms of Ferng and Chang benefit from the lower dimensions of the matrices involved in the computations. However, a

detailed analysis of the computation of the lag k autocorrelation, which presupposes the first k sub-vectors of the queue length distribution at departures (see [48] for their computation), shows that

- at least $3k + 1$ series with matrix-matrix multiplications (dependent on uniformization coefficients),
- $\frac{1}{6}k^3 + \frac{7}{2}k^2 + \frac{13}{3}k$ vector-matrix multiplications and
- 4 matrix inversions

constitute a high computational requirement. Especially, the first item may easily exceed the effort for computing matrix \mathbf{G} as needed for the ME and MAP output models, despite the lower dimension of m_A compared to $m_A m_S$ of \mathbf{G} . Due to the more compact notation and also due to the diverse series computations, the sparsity of the arrival and service processes cannot be as fully exploited as in the approaches via the ME and MAP output models.

Overall, a precise performance comparison is difficult and would depend very much on actual batch sizes, sparsity of service and arrival processes, series truncation rules, and actual implementation. In the light of the fact that our approaches to departure process characterizations are conceptually simpler and easier to implement, we claim that they are to be preferred when computing lag k coefficients for low and moderate values of k . This is especially valid, if a low-order PH (or MAP) service is specified. For example, with exponential service times, any performance advantage due to lower block dimensions disappears for the approach in [24]. On the contrary, for large values k (and non-MAP service), the algorithm by Ferng and Chang is expected to outperform the techniques presented here to compute the lag k departure autocorrelations.

6.3 Identifying a Truncation Level

We have proved that the autocorrelation function of the real departure process is *exactly* preserved up to lag $(n - 1)$ in a finite system with $n + 1$ block levels. For lags larger than $(n - 1)$ the autocorrelation is an approximation to the exact one. Naturally, the higher the value of n , the better the approximation accuracy. Large values of n , however, result in large output models and commensurate increase in the associated computational cost (see Section 6.2.2).

Here we concentrate on identifying the minimum size of the truncation level n that yields a sufficiently small approximation error from the true departure process so that the final error in the target performance metrics is negligible. Besides small computational costs, smaller n also offers better scalability of the analytic model. From Eq. (6.2)/(6.3), $\mathbf{D}_{0,n}^{(\text{MAP})}$ and $\mathbf{D}_{1,n}^{(\text{MAP})}$ are square matrices of size $(m_A m_s(n + 1) \times m_A m_s(n + 1))$, where m_A and m_S are the order of the arrival and service processes, respectively. Obviously, large n may result in state-space explosion in downward queues, making network decomposition infeasible.

An Example: BMAP(3)/H₂/1 departure process

We first use the same example of the BMAP(3)/H₂/1 system as in the previous sections that will help us characterize the behavior of the ACF of the departure process, and thus choose an appropriate truncation level n . By scaling the rates of the service process we control the utilization level of the queue. We consider two cases: a low system load (30% utilization) and a high system load (80% utilization).

6.3.1 Asymptotic Behavior of the ACF

In the formula for $\text{ACF}_n^{\text{MAP}}(k)$, the matrix $(-\mathbf{D}_{0,n}^{(\text{MAP})})^{-1}\mathbf{D}_{1,n}^{(\text{MAP})}$ is stochastic. Therefore, when raised to the power of k , the inner products it defines should decay geometrically according to its second largest eigenvalue, $l_{2,n}$. This is stated in the following theorem. Without loss of generality, we assume that the matrix is diagonalizable to avoid a complicated proof. The same analysis can be performed using the Jordan canonical form and the appropriate spectral projectors [76].

Theorem 6.3 *The autocorrelation of the approximated departure MAP decays geometrically with k , with rate equal to the second largest eigenvalue $l_{2,n}$ of $(-\mathbf{D}_{0,n}^{(\text{MAP})})^{-1}\mathbf{D}_{1,n}^{(\text{MAP})}$.*

Proof: From the proof of the marginal distributions in Section 6.1, we have that $\lambda_n^{\text{MAP}} = \lambda_\infty$ and $(c_n^{\text{MAP}})^2 = c_\infty^2$ for all truncation levels $n \geq 2$.

We know that the matrix $\mathbf{G}_n = (-\mathbf{D}_{0,n}^{(\text{MAP})})^{-1}\mathbf{D}_{1,n}^{(\text{MAP})}$ is a stochastic matrix, i.e., $\mathbf{G}_n \cdot \mathbf{e} = \mathbf{e}$, where \mathbf{e} is a column vector of all ones. Let $l_{i,n}$ denote its i th largest in magnitude eigenvalue, $x_{i,n}$ denote the i th right eigenvector, and $y_{i,n}^T$ denote the i th left eigenvector, for $1 \leq i \leq m$, where m is the dimension of \mathbf{G}_n . Then, $l_{1,n} = 1$, $x_{1,n} = \mathbf{e}$, and

$$\mathbf{G}_n = \sum_{i=1}^m l_{i,n} x_{i,n} y_{i,n}^T.$$

Then we have

$$\begin{aligned}
 \text{ACF}_n^{\text{MAP}}(k) &= \\
 & \frac{\lambda_n^{\text{MAP}} \pi_n \left(\sum_{i=1}^m l_{i,n}^k x_{i,n} y_{i,n}^T \right) (-\mathbf{D}_{0,n}^{(\text{MAP})})^{-1} \mathbf{e} - 1}{(c_n^{\text{MAP}})^2} = \\
 & \frac{\lambda_n^{\text{MAP}} \pi_n (\mathbf{e} y_{1,n}^T + \sum_{i=2}^m l_{i,n}^k x_{i,n} y_{i,n}^T) (-\mathbf{D}_{0,n}^{(\text{MAP})})^{-1} \mathbf{e} - 1}{(c_n^{\text{MAP}})^2} = \\
 & \frac{\lambda_n^{\text{MAP}} \pi_n \mathbf{e} y_{1,n}^T (-\mathbf{D}_{0,n}^{(\text{MAP})})^{-1} \mathbf{e} + \lambda_n^{\text{MAP}} \pi_n \left(\sum_{i=2}^m l_{i,n}^k x_{i,n} y_{i,n}^T \right) (-\mathbf{D}_{0,n}^{(\text{MAP})})^{-1} \mathbf{e} - 1}{(c_n^{\text{MAP}})^2}.
 \end{aligned}$$

Since $\pi_n \mathbf{e} = 1$, it suffices to prove that:

$$\lambda_n^{\text{MAP}} y_{1,n}^T (-\mathbf{D}_{0,n}^{(\text{MAP})})^{-1} \mathbf{e} - 1 = 0.$$

Since $\mathbf{Q}_n = \mathbf{D}_{0,n}^{(\text{MAP})} + \mathbf{D}_{1,n}^{(\text{MAP})}$ is the infinitesimal generator with $\pi_n \mathbf{Q}_n = \mathbf{0}$, then,

$$\begin{aligned}
 y_{1,n}^T &= y_{1,n}^T \mathbf{G}_n = y_{1,n}^T (-\mathbf{D}_{0,n}^{(\text{MAP})})^{-1} (\mathbf{Q}_n - \mathbf{D}_{0,n}^{(\text{MAP})}) \\
 &= y_{1,n}^T (-\mathbf{D}_{0,n}^{(\text{MAP})})^{-1} \mathbf{Q}_n + y_{1,n}^T.
 \end{aligned}$$

Therefore $y_{1,n}^T (-\mathbf{D}_{0,n}^{(\text{MAP})})^{-1} \mathbf{Q}_n = \mathbf{0}$, so that $y_{1,n}^T (-\mathbf{D}_{0,n}^{(\text{MAP})})^{-1} = C \cdot \pi_n$, where C is a constant.

Letting $y_{1,n}^T$ be normalized as $y_{1,n}^T \mathbf{e} = 1$, then

$$\begin{aligned}
 y_{1,n}^T \mathbf{e} &= y_{1,n}^T (-\mathbf{D}_{0,n}^{(\text{MAP})})^{-1} (-\mathbf{D}_{0,n}^{(\text{MAP})}) \mathbf{e} \\
 &= C \cdot \pi_n (-\mathbf{D}_{0,n}) \mathbf{e} = C \cdot \lambda_n^{\text{MAP}} = 1.
 \end{aligned}$$

Therefore $C = 1/\lambda_n^{\text{MAP}}$, and

$$\lambda_n^{\text{MAP}} y_{1,n}^T (-\mathbf{D}_{0,n}^{(\text{MAP})})^{-1} \mathbf{e} - 1 = \lambda_n^{\text{MAP}} \cdot \frac{1}{\lambda_n^{\text{MAP}}} \cdot \pi_n \mathbf{e} - 1 = 0.$$



When the eigenvalue for some block level is very close to the second-largest eigenvalue of the matrix \mathbf{G}_∞ for the true $\text{ACF}_\infty(k)$, that block level should capture very well the asymptotic behavior of $\text{ACF}_\infty(k)$. To identify this level n , we keep increasing n and compute the second eigenvalue $l_{2,n}$ of $(-\mathbf{D}_{0,n}^{(\text{MAP})})^{-1}\mathbf{D}_{1,n}^{(\text{MAP})}$ until two successive eigenvalues do not differ much. Then, we concentrate on the asymptotic decay of ACF, and search for the lag numbers k_∞ and k_n where the two ACFs have the same value, i.e., $\text{ACF}_\infty(k_\infty) \approx (l_{2,\infty})^{k_\infty} = (l_{2,n})^{k_n} \approx \text{ACF}_n^{\text{MAP}}(k_n)$, differ by less than, say, 5%. Then, $l_{2,n} = l_{2,\infty}^{k_\infty/k_n} = l_{2,\infty}^{1.05}$, which gives an acceptable n .

Table 6.1: The second largest eigenvalue of $(-\mathbf{D}_{0,n}^{(\text{MAP})})^{-1}\mathbf{D}_{1,n}^{(\text{MAP})}$.

30% Util		80% Util	
n	2nd Eigenvalue	n	2nd Eigenvalue
3	0.840610	3	0.978824
4	0.833388	4	0.978698
5	0.823615	5	0.978562
10	0.862645	10	0.977784
25	0.935070	25	0.978720
50	0.951438	50	0.986322
100	0.956885	100	0.993959
200	0.958519	200	0.997739
400	0.958972	400	0.999000
600	0.959060	600	0.999311
2000	0.959125	2000	0.999614

Let us illustrate this with the BMAP(3)/H₂/1 example. Table 6.1 gives the second largest eigenvalue of $(-\mathbf{D}_{0,n}^{(\text{MAP})})^{-1}\mathbf{D}_{1,n}^{(\text{MAP})}$ for this queue under the two utilization levels of 30% and 80%. Figure 6.4 displays the autocorrelation tails of the approximation with different truncation levels n under 30% and 80% utilizations. Note the asymptotically geometric decay of the autocorrelation with k . Figure 6.5 plots the relative error of the approximate ACFs for different block levels n . These are computed by integrating the absolute error and scaling it by the ACF area: $\sum_{k \geq n} |\text{ACF}_\infty(k) - \text{ACF}_n^{\text{MAP}}(k)| / \sum_{k \geq 1} \text{ACF}_\infty(k)$.

For the 30% utilization level, $l_{2,\infty} \approx 0.959125$, and $l_{2,n} = 0.959125^{1.05} = 0.957126$, which is obtained for n around 100. Because of the fast decay, however, for $n = 50$, $l_{2,n} = 0.951438$, the overall difference is only about 16%. For this case, we expect block levels $n = 50$ –100 to provide good ACF approximations. This is confirmed in Figure 6.4 where the ACF tails of all approximations with $n \geq 50$ are almost indistinguishable, with negligible relative error (see Figure 6.5).

For 80% utilization, $l_{2,n} = 0.999614^{1.05} = 0.999595$, which means that n greater than 1000, and possibly closer to 2000 is needed to have less than 5% difference with the same ACF. One would think that there is little hope to analyze this system, as $n = 2000$ would result in $\mathbf{D}_{0,n}^{(\text{MAP})}$ and $\mathbf{D}_{1,n}^{(\text{MAP})}$ with dimension 12006×12006 for the output MAP. However, Figure 6.4 shows that ACFs from block levels $n = 400$ and $n = 600$ capture the true ACF trend relatively well. Figure 6.5 quantifies this, measuring a 1% relative ACF error for $n = 600$.

Our results show that the asymptotic behavior for large lags does not capture well the transient effects for smaller lags, which, as we show later, turn out to be important in a downstream queue. Obviously, utilization plays an important role. Note that large lags k imply dependence of transition times between states that are k hops apart. If the probability of such an event is extremely low, capturing the appropriate $\text{ACF}(k)$ may not be as

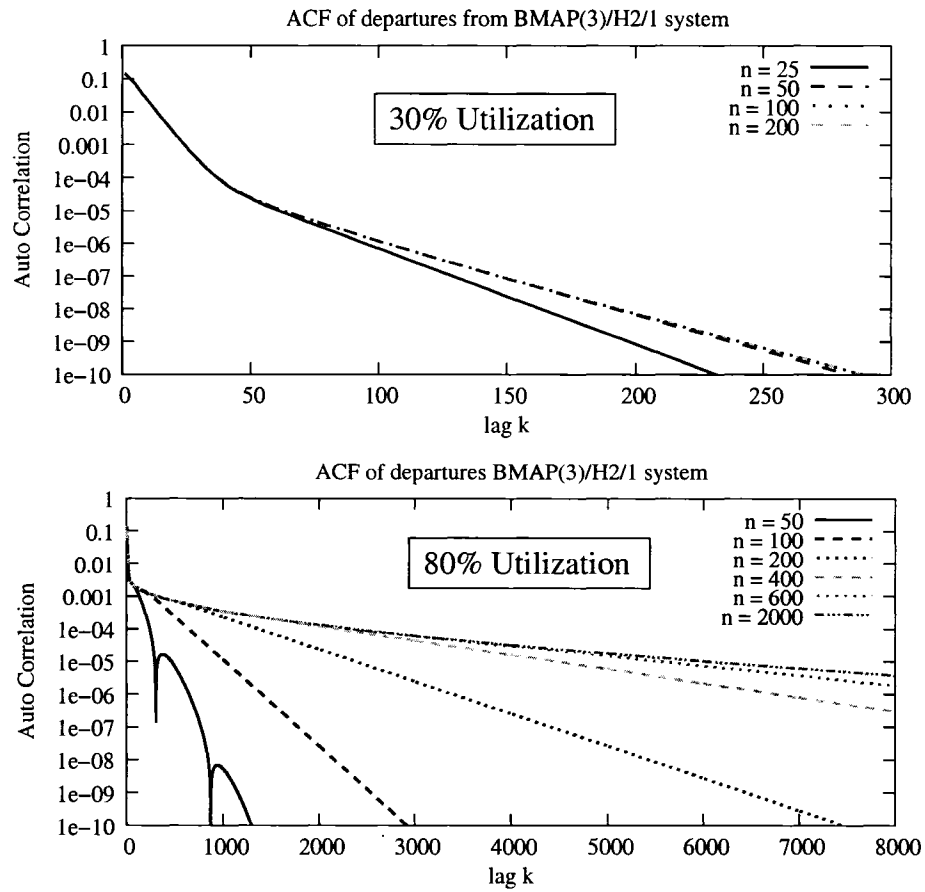


Figure 6.4: Autocorrelation of the departures from a 30% utilized and an 80% utilized BMAP(3)/H₂/1 systems.

important.

6.3.2 The Role of Utilization

The proof of Theorem 6.2 shows that it is the last two block columns in $\mathbf{D}_{0,n}^{(\text{MAP})}$ and $\mathbf{D}_{1,n}^{(\text{MAP})}$ (see Eq. (6.2)/(6.3)) of the approximate model that introduce the error. These two block columns correspond to the lumped level and its interaction with the rest of the Markov chain. Intuitively, all information about dependencies between the levels lumped into a *single* big level is overlooked. One can easily prove that if the stationary probability of the lumped state is zero, then the introduced error due to lumping reduces to zero as well. It is reasonable,

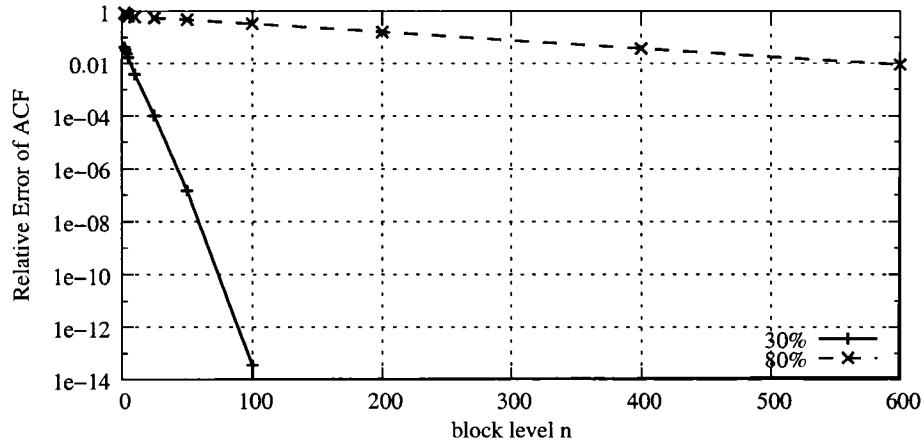


Figure 6.5: Relative ACF error under different truncation level n for system utilizations equal to 30% and 80%.

therefore, to expect very small approximation errors if the above probability is non-zero but sufficiently small. This is the case when the queue has a very low utilization. This suggests an approach for identifying the block level as the smallest n such that $\pi_n^\infty \mathbf{e} < \varepsilon$, where ε is a predefined small threshold.

Figure 6.6 gives the probabilities of the lumped states under different truncation levels n , for the two utilization levels of our example queue BMAP(3)/H₂/1. Under 30% utilization, π_n^∞ is less than $1e-6$ for all $n \geq 54$, and less than $1e-10$ for all $n \geq 92$. This is in agreement with our previous asymptotic analysis, but more surprisingly, the decay of probabilities approximates well the relative error in the ACF approximation as shown in Figure 6.5.

Turning into the system with 80% utilization, we see that π_n^∞ is still larger than $1e-6$ even for $n = 2000$. This also agrees with our asymptotic analysis. Also surprising here is the level of agreement between the decay of probabilities and the relative error in the ACF of Figure 6.5. For example, for $n = 600$ the relative ACF error is 0.01 while $\pi_{600}^\infty \mathbf{e} = 0.01$! It is possible, therefore, that an approximate departure process with two digits of accuracy ($n = 600$) produces a sufficiently accurate input to the second queue, and a sufficiently accurate metric such as mean queue length.

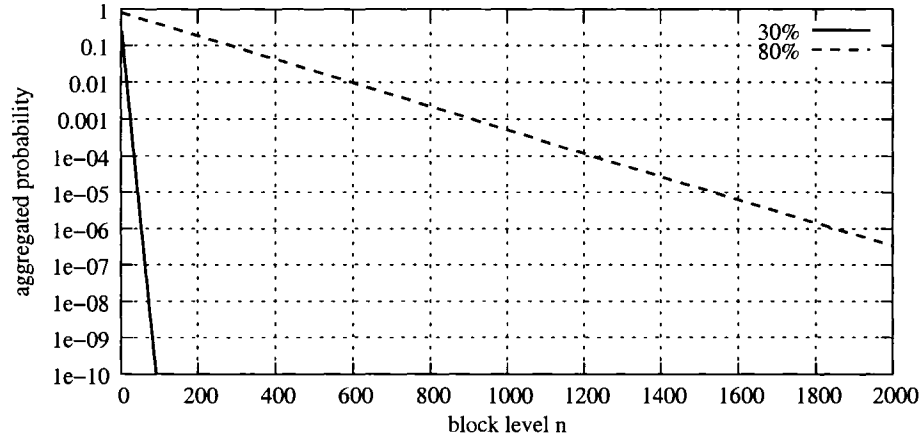


Figure 6.6: Probabilities of the lumped states for different truncation levels n under system utilizations 30% and 80%.

More importantly, the accuracy of the final metric depends on the utilization of both the first and second tier systems. As we show later, if the second tier system has very low utilization as in Figure 6.7, even a block level $n = 15$ for the first queue with 80% utilization provides excellent approximations to the queue length. We further elaborate on this in Section 6.4.

6.3.3 Remarks on the ME Approximation

Although ME output model $\mathbf{D}_{0,n}^{(ME)}/\mathbf{D}_{1,n}^{(ME)}$ in Chapter 5 lacks of physical interpretability as the MAP model, its preserves the same matrix properties. The above discussion of the asymptotic autocorrelation behavior is valid for the ME output as well.

Table 6.2 gives the second largest eigenvalue of $(-\mathbf{D}_{0,n}^{(ME)})^{-1}\mathbf{D}_{1,n}^{(ME)}$ for the BMAP(3)/H₂/1 example under the 30% and 80% utilizations. These values also confirms that the ME output model captures autocorrelation of the exact departure process worse than the MAP output model with compared to the second largest eigenvalues of $(-\mathbf{D}_{0,n}^{(MAP)})^{-1}\mathbf{D}_{1,n}^{(MAP)}$ in Table 6.1.

Table 6.2: The second largest eigenvalue of $(-\mathbf{D}_{0,n}^{(\text{ME})})^{-1}\mathbf{D}_{1,n}^{(\text{ME})}$.

30% Util		80% Util	
n	2nd Eigenvalue	n	2nd Eigenvalue
3	0.787447	3	0.657528
4	0.790878	4	0.648375
5	0.791682	5	0.754235
10	0.826577	10	0.891940
25	0.931856	25	0.957455
50	0.950909	50	0.980542
100	0.956804	100	0.990440
200	0.958508	200	0.997386
400	0.958970	400	0.998944
600	0.959059	600	0.999291

6.4 Experimental Results

We use the BMAP(3)/H₂/1 system as the first queue that is the same as the one in Section 6.3. At the second queue, we present experiments with two different service processes for comparison purposes. In the first experiment, the service process has an Erlang-2(E2) distribution with mean rate equal to 1.6667 and SCV equal to 0.50 as in Section 6.2.1. In the second experiment, the service process is an MMPP with the same mean rate as 1.6667, SCV equal to 20, and ACF starting at 0.4 at lag 1 and decaying to 0.001 beyond lag 36. Same as before, the rates of the service processes of the two nodes are scaled in order to achieve light system load (30% utilization) and high system load (80% utilization) across both nodes, i.e., we use all four combinations of utilization levels 30%-30%, 30%-80%, 80%-30%, and

80%-80%.

Figure 6.7 presents performance results at the second queue with MAP approximations of different size (i.e., block level n) as input. We focus on how well the output approximations of the first server can capture the mean queue length of the second server as the mean queue length is a performance measure that is truly affected by autocorrelated flows. Average throughputs and utilizations remain the same because the marginal distributions of the approximations are identical. The figure presents the average queue length error as a function of n . The figure shows that small block levels n consistently underestimate performance. The degree of error changes significantly from case to case (see the various ranges in the y-axes).

Under 30% utilization in the first queue, n as small as 25 is enough to provide nearly exact results, see Figure 6.7(I)-(II). For the 80% utilization cases in the first queue (Figure 6.7(III)-(IV)), the figures show that deciding the ideal n of the output of the first queue depends on the utilization level of the second queue. For small utilization of the second queue, small n is sufficient. The more bursty service process in the second experiment with the MMPP requires a higher block level n to reach a similar error level as in the first experiment with the more stable Erlang-2 service process.

The harder case is the 80%-80% one. Figure 6.7(IV) shows that $n = 400$ is required to achieve comparable error. This is in agreement with the asymptotic results presented in Section 6.3. Values of n as high as 100 result in around -5.7% error, making the cost of the method prohibitive for network decomposition in networks that operate under high utilization levels for all queues. In the next section we propose a remedy to this problem that strikes a balance between the size of the approximation and performance accuracy.

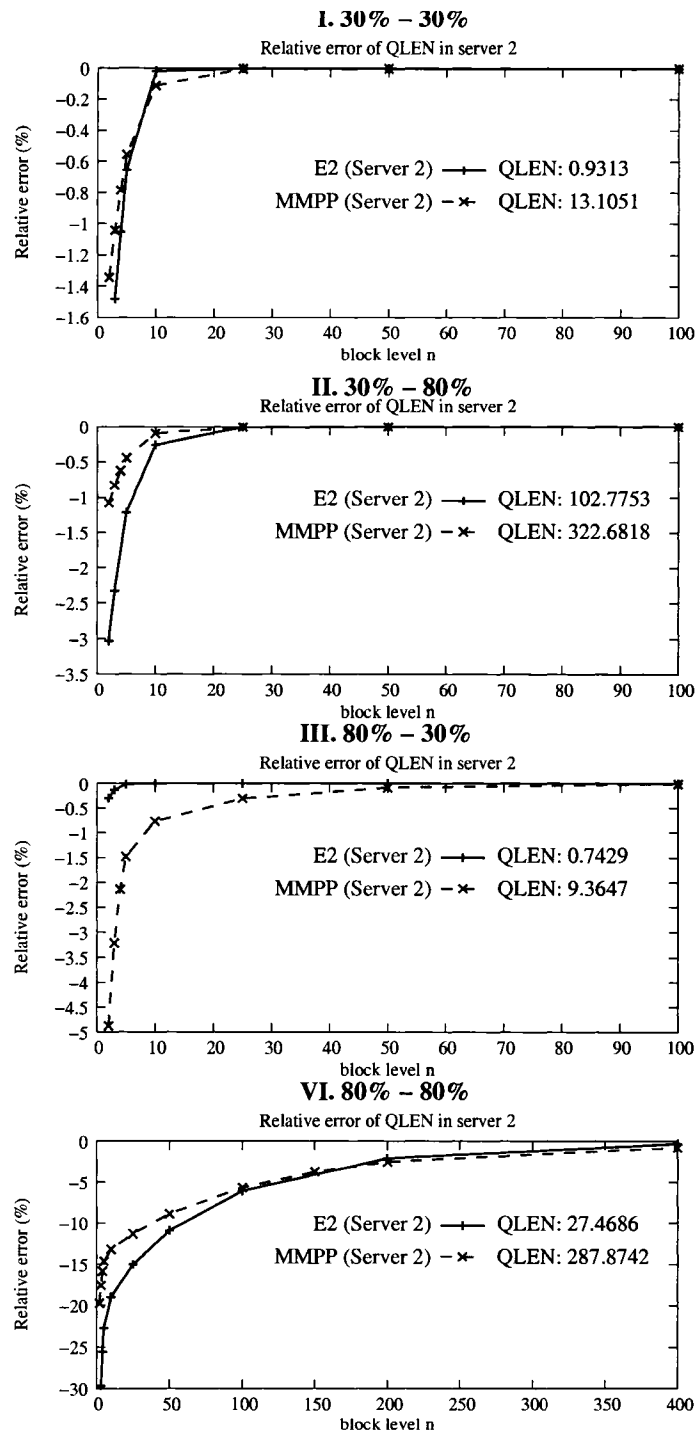


Figure 6.7: Experimental results of MAP approximation: mean queue length relative errors at server 2 of I. 30%-30% system, II. 30%-80% system, III. 80%-30% system and IV. 80%-80% system.

6.5 Improving Accuracy with Multiple Lumping

In Section 6.1 we showed that the lumped level eliminates the existing dependence among the various levels beyond n of the original infinite process, therefore higher n results in more accurate models. Nonetheless, the size of n needs to be kept at a minimum for any practical consideration. Here, we propose to break the single lumped level into several lumped levels, such that the dependence of the original process is recovered to some extent. Following this idea, we lump multiple levels up into to a certain predefined level as in Figure 6.8. Let w

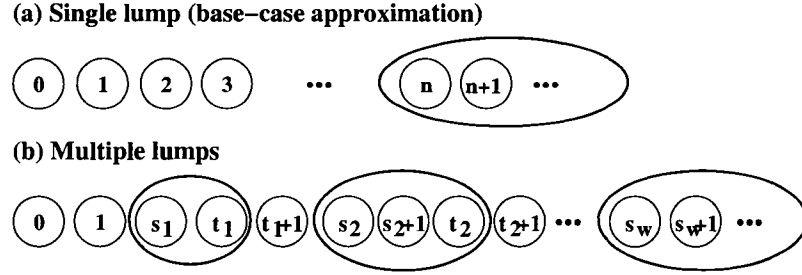


Figure 6.8: High level idea of lumping multiple levels.

denote the number of lumps of the departure process, and $[s_i, t_i]$ be the set of levels to be lumped in the i th lump, where $1 \leq i \leq w$, $s_i \leq t_i$, and t_w is equal to ∞ . Note that s_1 must be larger than or equal to 1 to avoid losing correctness of the marginal distributions of the approximation.

Lemma 6.2 *If the boundary states are not lumped (i.e., $s_1 \geq 1$), then the approximated departure process using multiple lumps preserves exactly the marginal distributions of the interdeparture times of a BMAP/MAP/1 queue, and it preserves the exact autocorrelation up to lag $(s_1 - 1)$.*

Proof: Since the stationary probabilities of the lumped levels are exactly derived by the probabilities of the Markov chain in the original BMAP/MAP/1 system, one can easily extend the proof of Theorem 6.1. ■

Consider the 80%-80% utilized system in Section 6.4 again. Here, we use multiple lumps by setting the number of levels that are lumped together to C , as illustrated in Figure 6.9.

All lumps are contiguous, i.e., $s_1 = 1$, $t_i = s_i + C - 1$, $s_{i+1} = t_i + 1$ for all $1 \leq i \leq w - 1$. Note

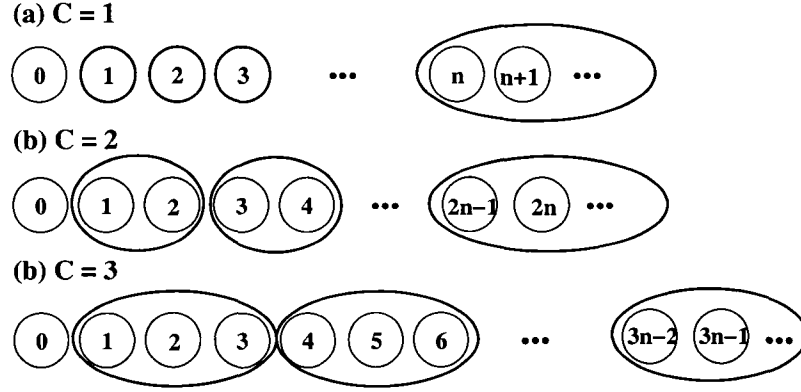


Figure 6.9: High level idea of multiple lumping: consecutive C levels are lumped in single lumps.

that the limit case of $C = 1$ is the original approximation, i.e., all levels beyond n are lumped into a single level. Figure 6.10 provides the autocorrelation of the approximated departure process with different values of C . The number of block levels of these approximations is

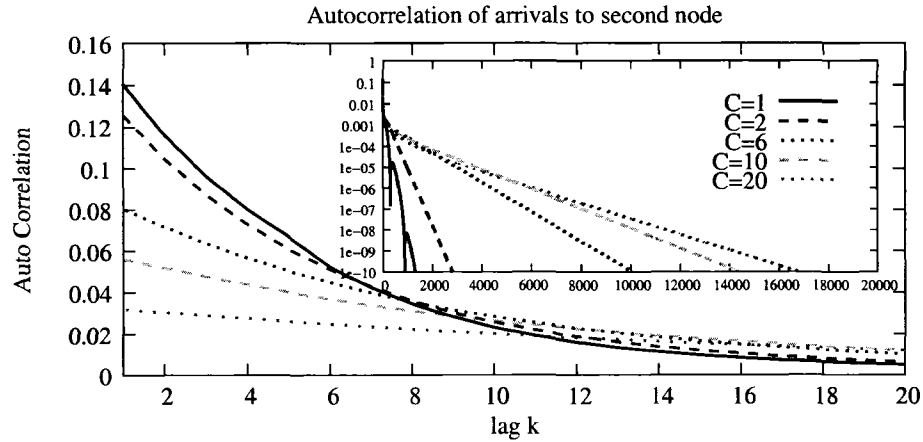


Figure 6.10: Total number of block levels is 51: autocorrelation of the departures from server 1/arrivals to server 2.

the same and equal to 51, i.e., their MAP matrices have the same size. $C = 1$ preserves exactly the ACF up to lag 49. When $C > 1$, the approximated process loses its short range dependence starting from lag 1. The error increases as C increases, because dependence in

the initial boundary levels is ignored. Focusing our attention to higher lags (see the inset graph in Figure 6.10), large C 's reflect better the asymptotic behavior of the second largest eigenvalue of the original model. Judiciously selecting C may balance the trade-off between the effect of small and large lags.

In the following, we propose a greedy algorithm to decide C for a pre-determined block level. Assuming that the block level n is pre-defined, our purpose is to find a good multiple-

```

I.  $C_1 \leftarrow 1$            /* match exact acf for the first chunk */
II. for  $i = 2 \rightarrow m$        /* decide  $[C_2, \dots, C_m]$  */
    1. /* initialization */
         $c \leftarrow 0$            /* lumping level is initialized as 0 */
         $QLEN_2 \leftarrow 0$      /* the avg. qlen in server 2 is set as 0 */
    2. /* increase lumping level gradually till the optimal
        performance in the second server is found */
        do
            a.  $QLEN_{prev} \leftarrow QLEN_2$ 
            b.  $C_j \leftarrow c + 1$  for  $i \leq j \leq m$ 
            c. approximate the departure process using
                 $[C_1, C_2, \dots, C_m]$ 
            d. use the departure process as the arrival process in
                server 2 and get its average queue length  $QLEN_2$ 
        while ( $QLEN_2 > QLEN_{prev}$ )
    3.  $C_i \leftarrow c$ 

```

Figure 6.11: Setting $[C_1, C_2, \dots, C_m]$ using a greedy algorithm.

lumping that achieves good performance given the restriction of the size n . Assume that n is to be divided in m chunks. Let the i th chunk consist of b_i blocks so that $\sum_{i=1}^m b_i = n$. For each chunk i , the number of levels that is collapsed into a single lump is constant and is denoted by C_i . As both the stationary probabilities and absolute ACF values decrease geometrically, the matches of the first lags are important. $C_1 = 1$ always, so that the exact autocorrelation values up to lag $(b_1 - 1)$ are exactly preserved. $[C_2, C_3, \dots, C_m]$ are determined by the algorithm in Figure 6.11.

Figure 6.12 shows the relative error of the average queue length in the second queue using the multiple lumping algorithm of Figure 6.11. The results are for the 80%-80%

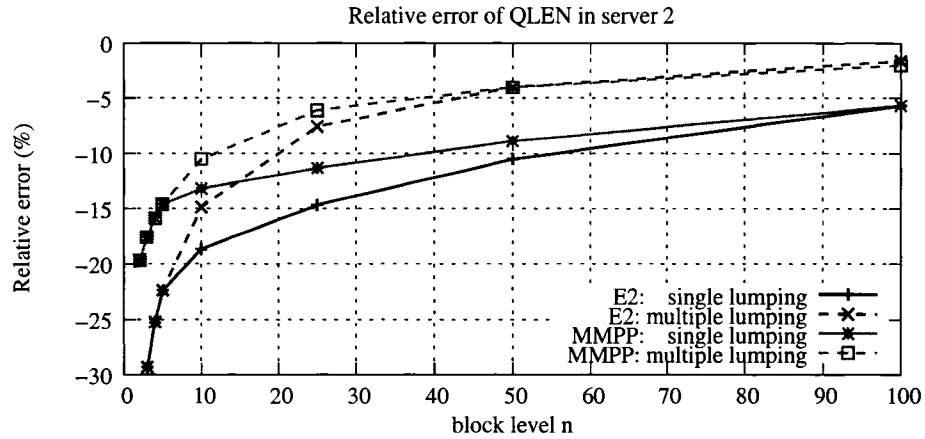


Figure 6.12: Relative error of average queue length in server 2.

systems in Section 6.4, and are compared with the approximation performance using the base-case approximation, i.e., single lumping. For very small n , i.e., 2, 3, 4 and 5, to avoid significant errors in the first lags, we only use single lumping even under the multiple lumping approximation, i.e., there is only one chunk and $C_1 = 1$. For $n = 10$ we equally partition the blocks into 2 chunks, and for $n \leq 25$, we equally partition the blocks into 4 chunks. The effectiveness of the approximation now improves: for $n = 25$, the relative error of the second queue length decreases to -7.6% and -6.1% for the experiments with Erlang-2(E2) and

MMPP service processes in server 2 respectively, while the corresponding values are -14.7% and -11.3% with the base-case approximation; for $n = 100$, the relative errors are only -1.6% and -2.0% for these two experiments.

6.6 Summary

In this chapter, we derive a family of finite MAP approximation models of the departure process of a BMAP/MAP/1 queue based on lumpability arguments, and compare it with the ME approximation models presented in Chapter 5. For the MAP output model, stationary probability vectors need to be computed additionally, which makes the construction of MAP representation more expensive than the construction of ME representation. However the MAP approximation can match the first $(n - 1)$ correlation coefficients of the inter-departure time while the ME representation can only match the first $(n - 2)$ values. The benefit of matching one more correlation coefficient on the performance accuracy at the downstream node is also demonstrated via experimentation.

This chapter also gives asymptotic analysis to best determine the size of the approximation n . Based on eigenvalue analysis of two key matrices of the MAP model as well as on analysis of the decay of probabilities of the lumped level for increasing n levels in the approximation, the block level n can be determined. We have shown that the size of n depends on both the utilization level of the BMAP/MAP/1 process and the utilization level of the queue that it feeds to. We have further presented a second category of approximations that aims at significantly reducing the size of the output model by lumping several intermediate block levels without performance loss. Extensive experimentation illustrates the approximation quality.

Chapter 7

Policy Development for Handling Systems with Autocorrelated Flows

Both workload characterization and analytic modeling aim toward the development of practical policies that can effectively improve the performance of systems with autocorrelation. Although correlated flows are observed in the our work as well as in many other previous papers, they are seldom considered in actual system design. In this chapter we illustrate how autocorrelation can be used to improve performance for load balancing in a clustered system.

Effective Load Balancing Under Autocorrelated Flows

In the past few years there has been a renewed interest in the development of load balancing policies for clustered systems with a single system image, i.e., systems where a set of homogeneous hosts behaves as a single host. Jobs (or requests) arrive at a dispatcher which then forwards them to the appropriate server.¹ While there exists no central waiting queue at the dispatcher, each server has a separate queue for waiting jobs and a separate

¹In this chapter we use the terms “jobs” and “requests” interchangeably.

processor, see Figure 7.1. The dispatching policy is critical for system performance and strongly depends on the stochastic characteristics of the jobs that request service as well as on the performance measures that the system strives to optimize.

Prior research has shown that the job service time distribution is critical for the performance of load balancing policies in such a setting [33, 32]. If job service times are highly variable, including job service times that are best characterized using heavy-tailed distributions, then policies that balance the load in the system by using *only* the size of each incoming job to determine the server that will be dispatched to, have been shown optimal if the performance goal is to minimize the expected job completion time, job waiting time, and job slowdown [23, 101].

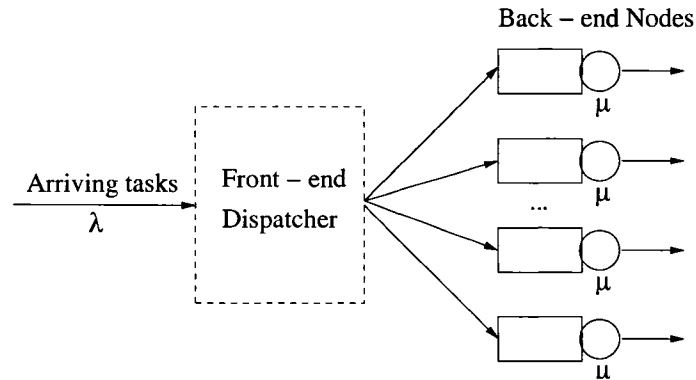


Figure 7.1: Model of a clustered server.

Several types of clustered systems can take advantage of size-based policies. Locally-distributed Web server cluster architectures that provide replicated services where a switch acts as the initial interface between the cluster nodes and the Internet, are one example of such systems [5, 101, 65]. For static requests in such systems, e.g., transfers of image files, the job service time is analogous to the size of the transferred file, thus it can be immediately used by the dispatcher to forward the request to the appropriate server. Content-distribution networks and media-server clusters that provide streaming of high quality audio and video

from a central server configuration to a large number of clients are a second example of a centralized system where size-based policies provide good balancing solutions [98, 16]. Finally, large storage systems which deploy mirroring for enhanced performance and data availability are another case of a clustered system where load balancing based on the job size is beneficial.

A significant body of research in task scheduling and load balancing has been developed over the years, but only recently there has been a consensus that traditional load balancing policies, i.e., join-the-shortest queue or join-the-least-loaded server, fail to balance the load if job service times are highly variable and/or heavy-tailed [33]. For workloads with highly variable service times *size-based* policies that advocate dedicating servers to jobs of similar sizes have been shown to achieve high performance. Assuming that there are N servers, the job sizes are partitioned into N intervals, $[s_0 \equiv 0, s_1)$, $[s_1, s_2)$, \dots , $[s_{N-1}, s_N \equiv \infty)$, so that server i is responsible for satisfying requests of size between s_{i-1} and s_i . By dedicating servers to requests of similar size, these policies aim at reducing the average job slowdown through separation of long and short jobs. Despite the fact that size-based policies are oblivious of the instantaneous load in each server, they successfully load each server with approximately the same amount of work so that they are equally utilized [32, 23]. Note that size-based policies are based solely on *a priori* knowledge of the distribution of the incoming job sizes. If this distribution is known, then size-based policies can minimize the expected job waiting time and job completion time.

Even if the job service time distributions are *not* known *a priori*, on-line versions of size-based policies have shown to maintain high performance for workloads that are highly variable *across time*, i.e., workloads that show transient behavior [101]. ADAPTLLOAD has been developed as an on-line version of a size-based policy that monitors the incoming work-

load and self-adjusts the above interval boundaries according to changes in the operational environment such as rapid fluctuations in the arrival intensities or service demand.

Nonetheless, size-based solutions are not adequate if the arrival streams in the dispatcher are *autocorrelated*. Indeed, conventional wisdom has it that the arrival process in Internet servers is not independent and it is an effect of the self-similar nature of the network traffic [86]. Furthermore, autocorrelated flows in the arrival process has been observed in systems including multi-tiered systems [54], large storage systems [28], an effect that has been shown to be detrimental for performance [22]. To alleviate the negative effects of autocorrelation, traffic shaping has been used by dropping, reordering, or delaying selected requests [12, 97, 17, 1].

In this chapter, we show that size-based load balancing policies cease to be effective if the workload arrival process is autocorrelated. We show that as autocorrelation in the arrival process increases, the performance benefits of size-based policies diminish. Based on our observations, we propose a size-based load balancing policy that aims at reducing the performance degradation due to autocorrelation in each server, while maintaining the property of similar job sizes been served by the same server. This new policy, called **D_EQAL**, strives to equally distribute work guided by autocorrelation and load, and effectively unbalances the load in the system: not all servers are equally utilized any more, but overall system performance increases dramatically. **D_EQAL** does not assume any *a priori* knowledge of the job service time distribution nor any knowledge of the intensity of the dependence structure in the arrival streams. By observing past arrival and service characteristics as well as past performance, it self-adjusts its configuration parameters. To the best of our knowledge this is the first time that dependence in the arrival process becomes a critical aspect of load balancing.

This chapter is organized as follows. In Section 7.1 we compare the performance of a size-based policy with several classic policies in the presence of autocorrelated arrival flows in the system. The proposed on-line size-based policy is presented in Section 7.2 and its performance is evaluated via simulation. Section 7.3 gives a summary of this chapter.

7.1 Autocorrelation Effects on Load Balancing Policies

In this section, we use trace driven simulation to examine the performance impacts of autocorrelated arrivals in load balancing policies in the simple cluster depicted in Figure 7.1. We assume that the number of nodes is equal to four.²

The service process is obtained from traces of the 1998 World Soccer Cup Web site,³ that have been used in several studies to evaluate the performance in load balancing policies in clustered web servers [101, 81, 84]. Trace data were collected during 92 days, from 26 April 1998 to 26 July 1998, see [7] for more details. Here, we use part of the June 24th trace (10 million requests), that corresponds to nearly ten hours of operation and we extract the file size of each transferred request. Because the Web site contained only static pages, the size of the requested file is a good approximation of the request service time. In the trace used for the experiments, the average size of a requested file is 5059 bytes and its coefficient of variation (CV) is 7.56. Figure 7.2(b) plots the average request size for batches of 10,000 requests for the duration of the trace, and shows that the average transferred size varies across time.

Unfortunately, we cannot use the arrival process of the World Cup trace data because it

²Experiments with larger number of nodes have been also done but results are qualitatively the same and are not reported here.

³Available from the Internet Traffic Archive at <http://ita.ee.lbl.gov>.

is not detailed enough: arrival timestamps of requests are provided in seconds, as a result there are *multiple* requests that arrive within one second periods. To examine the effect of autocorrelation in the arrival process, we use a 2-stage MMPP, which with appropriate parameterization allows for changing *only* the ACF while maintaining the same mean and *CV*, that are equal to 1 and 4.5, respectively. The ACF of the three arrival processes that we use here is illustrated in Figure 7.2(a).

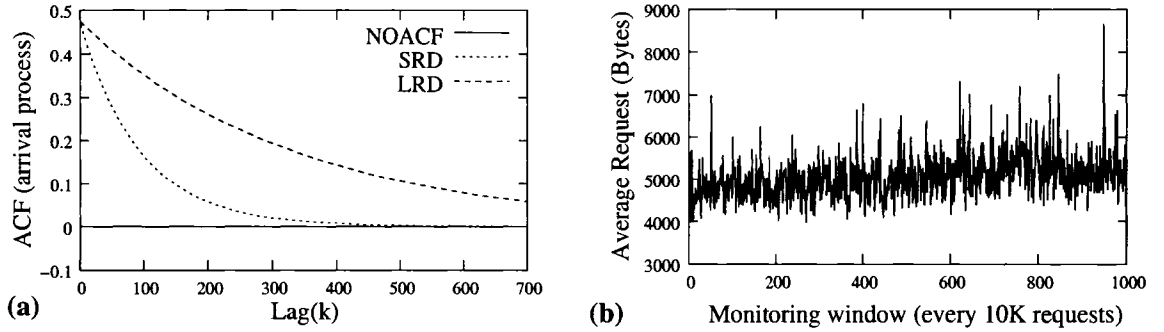


Figure 7.2: (a) ACF for the three arrival processes used in the simulation and (b) Average request size for every 10000 requests in the ten million sample space.

7.1.1 Load Balancing Policies

We compare the performance of the following policies: **ADAPTL**OAD, a size-based policy that does not require *a priori* knowledge of the service time distribution and has been shown to be effective under changing workload conditions [101], the *Join Shortest Weighted Queue* (JSWQ) policy [101], *Join Shortest Queue* (JSQ) [39], and *Round Robbin* (RR). The policies are summarized as follows:

- **AdaptLoad:** In a cluster with N server nodes, **ADAPTL**OAD partitions the possible request sizes into N intervals, $\{[s_0 \equiv 0, s_1), [s_1, s_2), \dots, [s_{N-1}, s_N \equiv \infty)\}$, so that if the size of a requested file falls in the i th interval, i.e., $[s_{i-1}, s_i)$, this request is routed to server i , for $1 \leq i \leq N$. These boundaries s_i for $1 \leq i \leq N$ are determined

by constructing the histogram of request sizes and partitioning it in equal areas, i.e., representing equal work for each server, as shown by the following equation:

$$\int_{s_{i-1}}^{s_i} x \cdot dF(x) \approx \frac{\bar{S}}{N}, \quad 1 \leq i \leq N, \quad (7.1)$$

where $F(x)$ is the CDF of the request sizes and the amount of total work is \bar{S} . By sending requests of similar sizes to each server, the policy improves average job response time and average job slowdown by avoiding having short jobs been stuck after long jobs in the queue. For a transient workload, the value of the $N - 1$ size boundaries s_1, s_2, \dots, s_{N-1} is critical. ADAPTLLOAD self-adjusts these boundaries by predicting the incoming workload based on the histogram of the last K requests. In the simulations, we set the value of K equal to 10000.

- **JSWQ:** The length of each queue in the system is weighed by the size of queued requests, therefore each incoming request is routed to least loaded server.
- **JSQ:** When a request arrives, it is assigned to a server with the smallest waiting queue. If multiple servers have the same queue length, then a server is selected randomly from this group of servers.
- **RR:** In the round-robin algorithm, requests are routed to servers in a rotated order.

7.1.2 Performance Analysis

Using trace-driven simulation we compare the performance of the four policies. In all our experiments, we consider a cluster of four homogeneous back-end servers that serve requests in a *first-come-first-serve* (FIFO) order.

We evaluate the effect of autocorrelated inter-arrival times on the performance of load balancing policies by analyzing the response time (i.e., wait time plus service time), the average queue length (i.e., the total number of jobs in the server, both waiting and in service), the average slowdown (i.e., the ratio of the actual response time of a request to its service time), and the mean utilization. Figure 7.3 plots performance results for the four load balancing policies in the three different experiments. Similar to the results in the previous

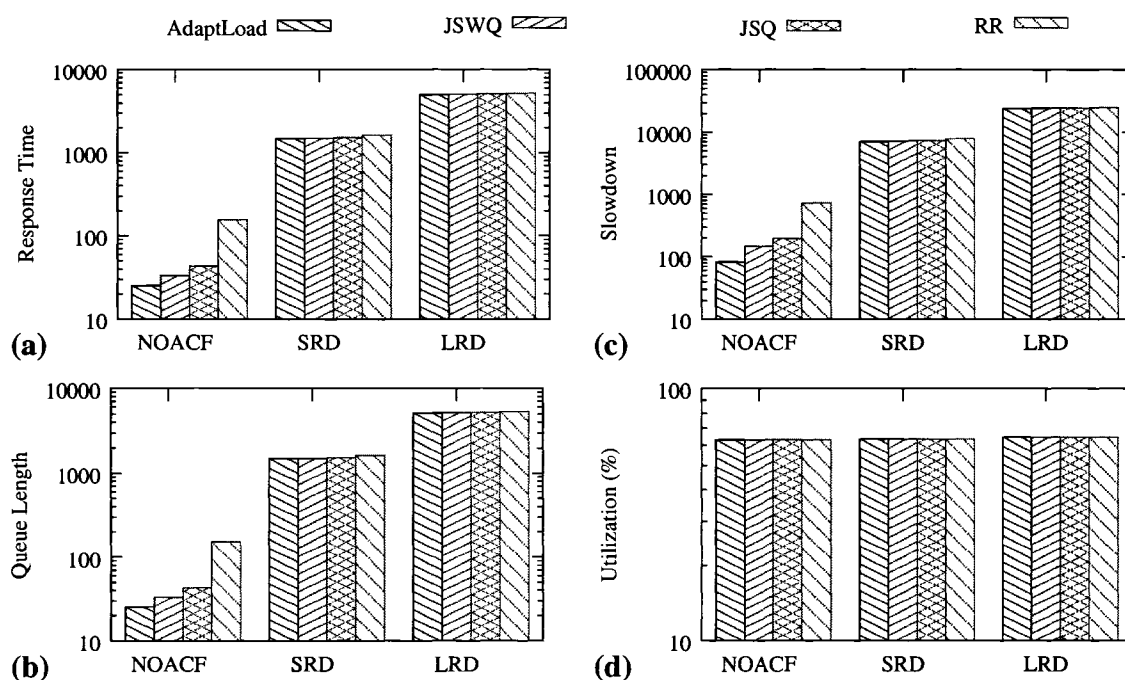


Figure 7.3: Performance metrics under four load balancing policies: (a) average response time, (b) average queue length, (c) average slowdown, and (d) average utilization.

chapters, Figure 7.3 shows that correlation in the arrival process degrades overall system performance for all four policies. For example the overall performance under independent arrivals (NOACF) is two orders of magnitude better than under SRD inter-arrivals, and three orders of magnitude better than under LRD inter-arrivals, despite the fact that average system utilizations are exactly the same for all experiments, i.e., the average utilizations are

about 62%, see Figure 7.3(d).⁴ Most importantly, the figure also shows that ADAPTLLOAD outperforms all policies under independent inter-arrivals *only*, see Figure 7.3(a)-(c). Under correlated arrival processes, ADAPTLLOAD's performance is comparable to the three other policies, essentially showing that separating requests according to their sizes is not sufficient.

To better understand this behavior, we turn to the autocorrelation of the arrival process in *each* server. Figure 7.4 shows the ACF of the arrival process at each back-end server, as well as the ACF of the arrival process at the front-end dispatcher (labeled as "original stream" in the figure). When there is no autocorrelation in the inter-arrivals at the front-end dispatcher, the ACF of inter-arrivals at each back-end server is almost zero for all policies except ADAPTLLOAD, see Figure 7.4(a). Because only a few requests are for large files, the size-based ADAPTLLOAD routes them to the fourth server only, and these requests are spaced in such a way that there is autocorrelation in their arrival process to server four. The middle column of graphs in Figure 7.4 shows the ACFs for the experiments with short-range dependence in the arrival process, and the right column of graphs in Figure 7.4 shows the ACFs for the experiments with long-range dependence in the arrivals. JSWQ and JSQ have the weakest dependence while RR has the strongest dependence. Because ADAPTLLOAD is a size-based policy and the workload is heavy-tailed, most requests are for small files and the first server receives most of requests. Therefore the ACF of its arrival process is very similar to the original ACF of the arrival process at the dispatcher. High ACF in the arrivals at the first server does not affect its utilization, which remains almost the same as the rest of the servers, but its response time and slowdown deteriorate significantly and negatively affect the overall performance.⁵ This suggests that perhaps in such systems unbalancing the load

⁴Per server utilizations for all experiments remain the same, and equal to about 62%.

⁵Weak ACFs in the arrival processes of all servers under JSWQ/JSQ help performance but because short and long jobs are now be served on the same server in both policies, their performance remains low.

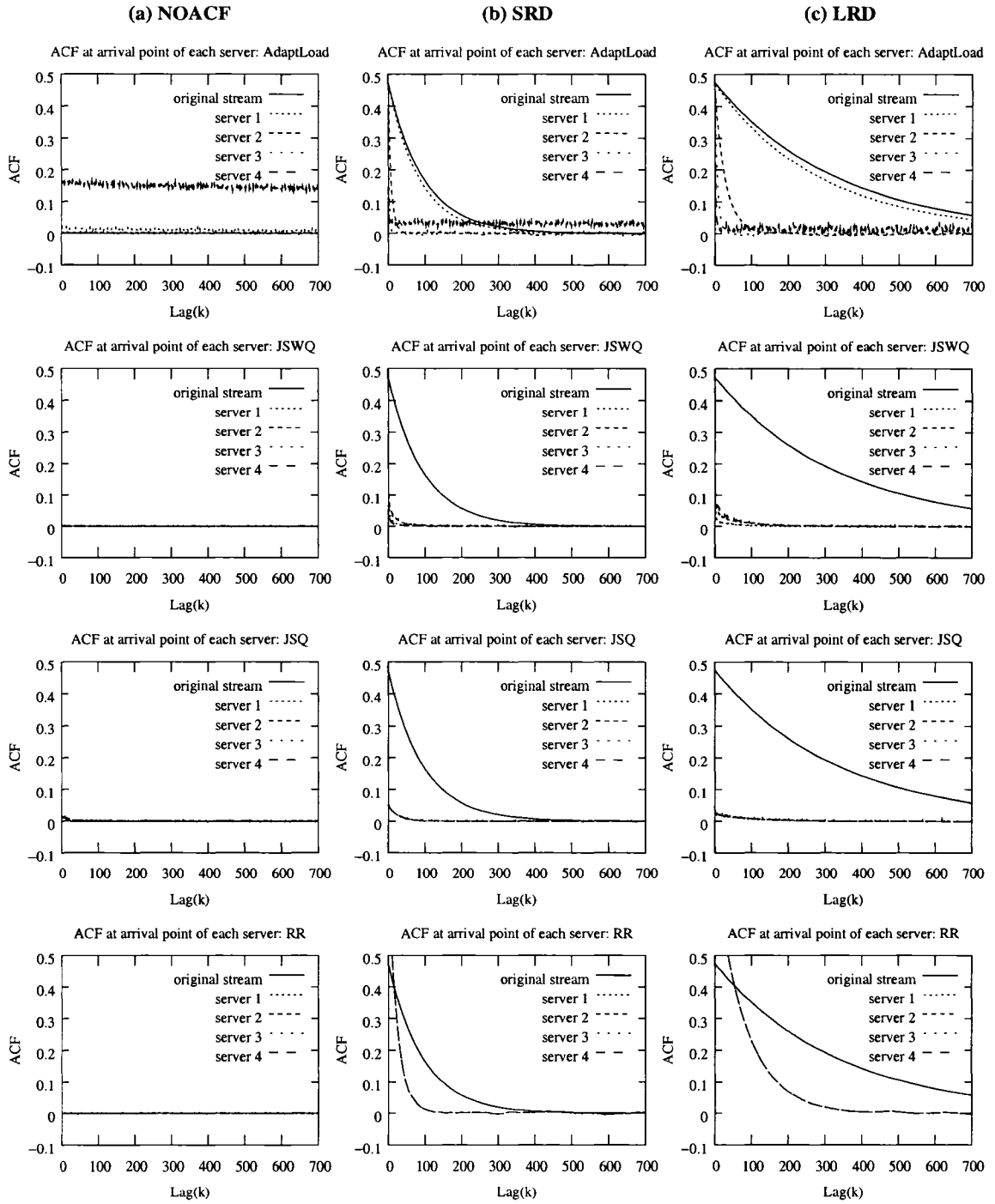


Figure 7.4: ACF in inter-arrivals at each server, where the arriving requests at the front-end dispatcher have (a) no dependence, (b) short range dependence (SRD), and (c) long range dependence (LRD).

(i.e., reducing the utilization level of the server with correlated arrivals) while maintaining the property of serving jobs of equal size in the same server may improve performance.

7.2 Unbalancing Load to Improve Performance

In this section, we propose an enhancement to the ADAPTLLOAD policy that accounts for dependence in the arrival process by relaxing ADAPTLLOAD's goal to balance the work among all nodes of the cluster. The proposed policy strives to judiciously *unbalance* the load among the nodes by moving jobs from the nodes with a strongly correlated arrival process to nodes with weaker correlation in their inter-arrival times. First we present a static version of the policy where the load of the servers with correlated interval times is reduced by a static percentage while the load of servers with no autocorrelation in their arrival process increases. Then, we present a dynamic version of the same policy where measured workload characteristics and policy performance measures guide load unbalancing in the system to improve overall system performance.

7.2.1 S_EQAL: Static Policy

Recall that ADAPTLLOAD is based on the idea that given that in an N -server cluster the amount of total work is \bar{S} , then the best performance is achieved if requests are assigned to the servers such that each server serves \bar{S}/N of the work, i.e., load is well balanced across all servers. Associating the request size with the work a server has to do, ADAPTLLOAD equally distributes the work among servers by determining boundaries of request sizes for each server. These boundaries s_i for $1 \leq i \leq N$ are determined by constructing the histogram of request sizes and partitioning it in equal areas, i.e., representing equal work for each server, as shown

by Eq. (7.1) .

S_EQUAL uses the same histogram information, but sets the new boundaries s'_i by weighting the work assigned to each server as a function of the degree of correlation in the arrival process based on the observation that in order to achieve similar performance levels under autocorrelated arrivals, the system utilization must be lower than under independent arrivals.

We introduce a shifting percentage vector $\mathbf{p} = (p_1, p_2, \dots, p_N)$, so that the work assigned at server i is now equal to $(1 + p_i) \frac{\bar{S}}{N}$ for $1 \leq i \leq N$. Note that p_i can take both negative and positive values. A negative p_i indicates that the amount of work assigned at server i should be less than the equal share of \bar{S}/N . A positive p_i indicates that the amount of work assigned at server i should be higher than the equal share of \bar{S}/N . Because the shifting percentage p_i simply shifts the amount of work from one server to another it should satisfy the equation $\sum_{i=1}^N p_i = 0$ for $1 \leq i \leq N$. The following equation formalizes this new load distribution:

$$\int_{s_{i-1}}^{s_i} x \cdot dF(x) \approx (1 + p_i) \frac{\bar{S}}{N}, \quad 1 \leq i \leq N. \quad (7.2)$$

Figure 7.5 gives an illustration of the high level idea of this new policy.

First, we statically define the values of p_i for $1 \leq i \leq N$, by letting p_1 be equal to a pre-determined corrective constant R , $0\% \leq R < 100\%$, and then by calculating the rest of the shifting percentages p_i for $2 \leq i \leq N$ using a semi-geometric increasing method, as described by the algorithm in Figure 7.6. Because the first server is usually the one that serves the small requests and has strong autocorrelated inter-arrival times, the shifting percentage p_1 is negative, i.e., $p_1 = -R$. For example, if we define $R = 10\%$ then the shifting percentages for a 4-server cluster are $p_1 = -10\%$, $p_2 = -1.67\%$, $p_3 = 3.33\%$ and $p_4 = 8.34\%$. For $R = 20\%$ the shifting percentages are twice as high as in the case of $R = 10\%$. i.e., $p_1 = -20\%$,

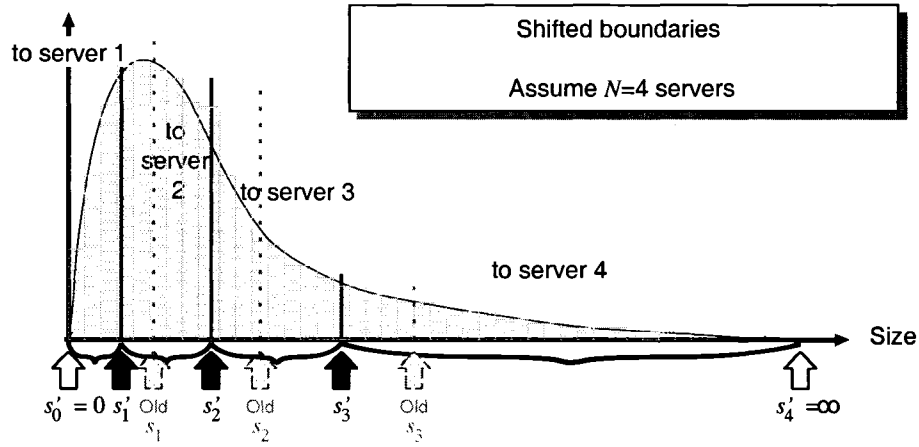


Figure 7.5: S_EQAL's high level idea to recalculate boundaries under autocorrelated inter-arrival times.

$p_2 = -3.34\%$, $p_3 = 6.67\%$, and $p_4 = 16.67\%$.

1. initialize variables	
a. initialize a variable <i>adjust</i>	$adjust = -R$
b. initialize the shifting percentages	$p_i = 0$ for all $1 \leq i \leq N$
2. for $i = 1$ to $N - 1$ do	
a. add <i>adjust</i> to p_i	$p_i \leftarrow p_i + adjust$
b. for $j = i + 1$ to N do	
equally distribute <i>adjust</i> to the remaining servers	$p_j \leftarrow p_j - \frac{adjust}{N-i}$
c. reduce <i>adjust</i> to half	$adjust \leftarrow adjust/2$

Figure 7.6: Setting the shifting percentages p_i for S_EQAL.

7.2.1.1 Arrival process with short-range dependence

We evaluate the performance of S_EQAL using the short range dependent arrival process used in Section 7.1.

First, we quantify the effect of the corrective constant R that we use to generate the values of the shifting percentages p_i for $1 \leq i \leq N$ by computing the average slowdown and average response time of requests under S_EQAL for different values of R . We present our findings in Figure 7.7. $R = 0\%$ corresponds to the the original ADAPTLOAD, i.e.,

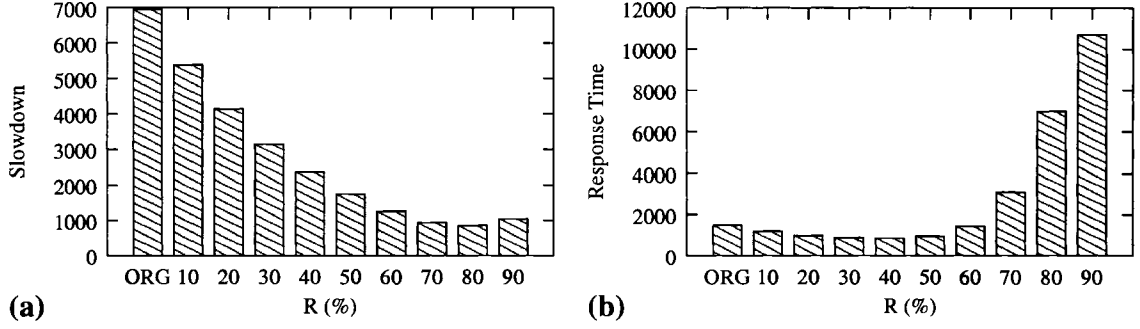


Figure 7.7: Average slowdown and average response time as a function of the corrective constant R under SRD inter-arrival times.

no shifting of boundaries. Figure 7.7(a) shows that the average slowdown of all requests improves as R increases (i.e., the boundaries are shifted to the left compared to the original ADAPTLOAD). We observe that the best performance is achieved for $R = 80\%$ (i.e., $p_1 = -80\%$). However, Figure 7.7(b) indicates that the best performance for response time is achieved when $R = 40\%$. Therefore, a good corrective constant is $R = 40\%$, where average slowdown improves by 75.1%. Average response time improves by 41.9% when compared to the original ADAPTLOAD.

We present the per server performance in Figure 7.8. Per server utilizations shown in Figure 7.8(d) verify that the shifting percentages p_i indeed imbalance work across the cluster. As R increases, the utilization of the first two servers decrease while the utilizations of the last two servers increase. The last server's utilization is now the highest in the cluster. Reducing utilization in the first server reduces its request slowdown, as shown in Figure 7.8(a), but the extra work that is now assigned to servers 3 and 4 do not increase their

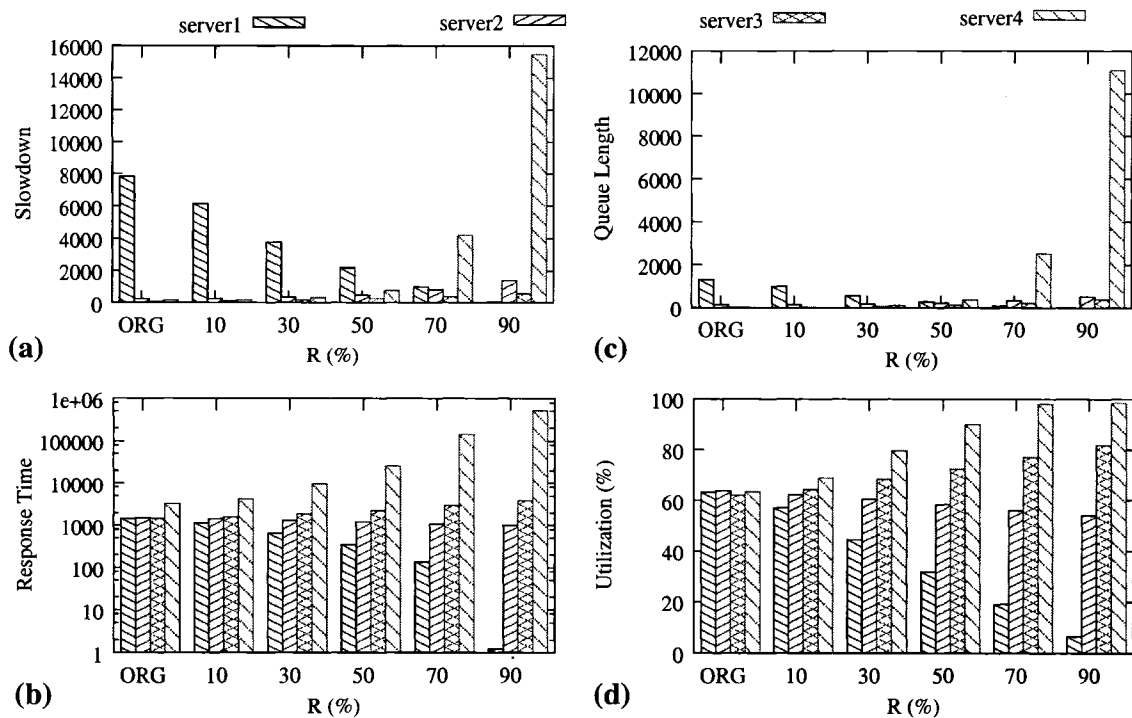


Figure 7.8: Per server performance measures: (a) average slowdown, (b) average response time, (c) average queue length and (d) average utilization as a function of the corrective constant R with SRD inter-arrival times. The order of bars for each policy reflect the server identity.

slowdown significantly for small values of R . For $R = 90\%$, slowdown at server 4 becomes very high, almost twice as high as for server 1 under the original ADAPTLLOAD. The average per-server queue length behaves similarly to the average slowdown (see Figure 7.8(c)). The average response time displayed in Figure 7.8(b) shows that small R values decrease average response time of the first server and increase the response time of the last server. If the portion of additional requests served by the last server is small, then the contribution of the last server performance values to the overall performance degradation is not significant. As R increases, more jobs are assigned to higher servers, which counterbalances the benefits of decreased utilization at the first servers.

7.2.1.2 Arrival process with long-range dependence

We evaluate the performance of S-EQUAL under long range dependent (LRD) inter-arrival times in the cluster. Figure 7.9 gives the average request slowdown and the average request response time as a function of the corrective constant R . In Figure 7.9 we observe the same performance trends as in Figure 7.7 but higher absolute values than under SRD. The smallest average request response time is achieved for $R = 40\%$, which represents an improvement of 49.2% compared with the original ADAPTL0AD (i.e., $R = 0\%$). The average request slowdown for the optimal R value is 67.2% better than with the original ADAPTL0AD.

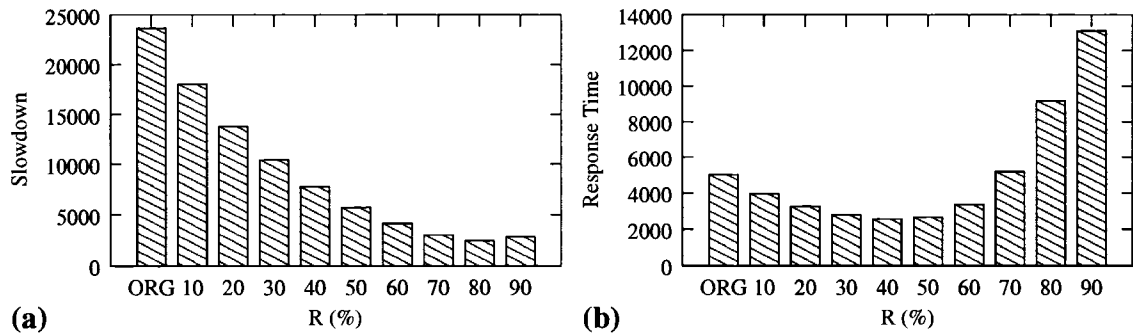


Figure 7.9: Average slowdown and average response time as a function of corrective constant R under LRD inter-arrivals times.

Figure 7.10 illustrates the per server performance under LRD traffic in the cluster. Although performance trends are similar as with the SRD case, they are more exaggerated here. Both average slowdown and average response time of the first server reduce as R increases (see Figures 7.10(a)-(b)), but a turning point exists where shifting more work to subsequent servers adversely affects slowdown. In the following section we present a dynamic algorithm that decides the amount of work to be shifted on-the-fly and strikes a good balance across both performance measures.

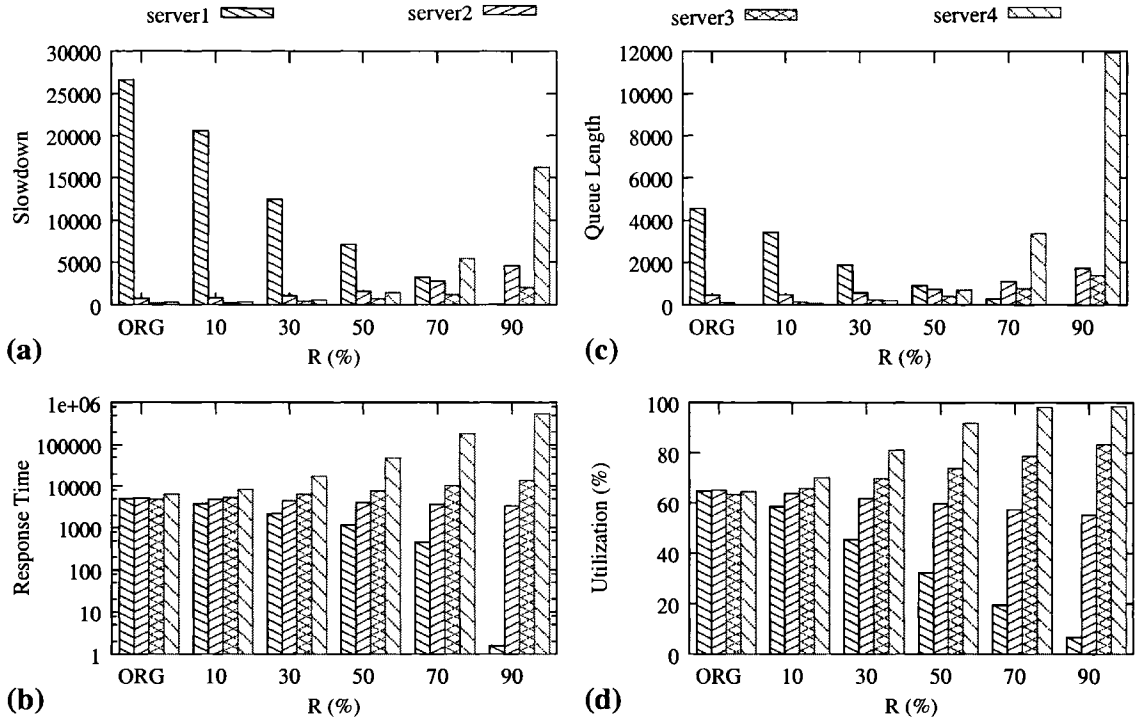


Figure 7.10: Per server performance metrics as a function of the corrective constant R under LRD traffic: (a) average slowdown, (b) average response time, (c) average queue length and (d) average utilization. The order of bars for each policy reflect the server identity.

7.2.2 D_EQAL: On-line Policy

In the previous section we gave a first proof of concept that load imbalancing can be beneficial for performance in clusters with autocorrelated inter-arrival times and heavy tailed service requests, but performance improvements depend on the degree of load imbalancing that is introduced by the corrective constant R . A good choice of R can result in significant performance improvements, but an unfortunate choice may also result in poor performance. Here we present an on-line version of the policy that monitors the workload as well as the effectiveness of load balancing, and its performance is now independent of the choice of R . Based on continuous monitoring, the policy readjusts the degree of load imbalancing on-the-fly while aiming at improving *both* average response time and average slowdown.

We use an updating window of C requests that have been served by the cluster. C must

be large enough to allow for statistically significant performance measurement but also small enough to allow for quick adaptation to transient workload conditions. In the experiments presented here C is set to 300K.⁶ The policy starts by setting R to zero, i.e., no load shifting is proposed beyond the computed ADAPTLLOAD intervals. For every batch of C requests, we compare the relative performance improvement/decline in comparison to the previous batch of C requests. The two performance measures that we examine are the average slowdown (Avg_{sl}) and the average normalized response time (Avg_{nres}), which is defined as follows:

$$Avg_{nres}(k) = \frac{\text{average response time of requests in the } k^{th} \text{ batch}}{\text{average file size of requests in the } k^{th} \text{ batch}}.$$

Then, according to the comparison of the values of average slowdown and normalized response times, we readjust R by a *small* pre-determined value adj_{left} or adj_{right} . The following four corrective actions can be taken:

- *Correct left:* $R \leftarrow R + adj_{left}$.
- *Correct right:* $R \leftarrow R - adj_{right}$.
- *Correct continuously:* If the previous adjustment is “correct left”, then correct left. If the previous adjustment is “correct right”, then correct right.
- *Correct reversely:* If the previous adjustment is “correct left”, then correct right. If the previous adjustment is “correct right”, then correct left.

In our experiments, adj_{left} and adj_{right} are set to 10% under medium or heavy load (i.e., 10% of the load is shifted left or right in the histogram of Figure 7.5 in order to recalculate

⁶We examined the robustness of D-EQUAL by different C values ranging from 100K to 800K, which achieve similar performance.

the interval boundaries). Under light load (i.e., system is under 30% utilized), adj_{left} is set to 20% to avoid slow adjustment of reaching the best performance. Note that we do not monitor the dependence structure of arrival streams in this online algorithm, but trust D_EQAL to find the optimal R .

The algorithm in Figure 7.11 describes how the corrective constant R is dynamically adjusted every C requests. Once a new value for R is set, the corrective factors p_i are computed according to the algorithm of Figure 7.6. Finally, the per server job size boundaries are computed according to Eq. (7.2) using the recalculated p_i .

7.2.2.1 Performance of D_EQAL

In this section, we evaluate the effectiveness of D_EQAL. As in the previous sections, each experiment is driven by the WorldCup 10 million request trace, the boundaries of ADAPTL0AD are computed every $K = 10K$ requests, while the adjustment of the corrective factors for D_EQAL happens every $C = 300K$ requests,

We compare the original ADAPTL0AD, S_EQAL with various values of its corrective constant R , and D_EQAL. Note that in the dynamic policy, we start with a value of $R = 0$, which indicates that we rely on the adaptive algorithm to find the best value of R . Results for various system utilizations of 30%, 62% and 80% are presented in Figure 7.12-7.14. In all the results, the on-line policy (labeled “D_EQAL”) is comparable to the best performing S_EQAL, where R is set to a set of static values. D_EQAL manages to adjust R such that both slowdown (Figures 7.12-7.14(a)) and response time (see 7.12-7.14(b)) are improved.

Figures 7.12-7.14(c) show how the value of the corrective constant R changes over time under NOACF, SRD and LRD arrivals under different utilizations, respectively. Observe that when system has no autocorrelation, R sits in value 0 most of time no matter what

1. initialize
 - a. set $R \leftarrow 0$
 - b. $k \leftarrow 0$
2. every C requests
 - a. compute the current system utilization and set adj_{left} and adj_{right}
 - b. compute the current performance metrics $Avg_{sld}(k)$ and $Avg_{nres}(k)$
 - c. if ($k = 0$)
 - then I. *Correct left*
 - II. go to 3.
 - d. if $\frac{Avg_{nres}(k) - Avg_{nres}(k-1)}{Avg_{nres}(0)} > \frac{Avg_{sld}(k) - Avg_{sld}(k-1)}{Avg_{sld}(0)}$
 - then I. *Correct right*
 - II. go to 3.
 - e. if ($Avg_{sld}(k) > Avg_{sld}(k-1)$ or $Avg_{nres}(k) > Avg_{nres}(k-1)$)
 - then *Correct reversely*
 - else *Correct continuously*
3. Compute p_i for $1 \leq i \leq N$ using the algorithm of Figure 7.6
4. $k \leftarrow k + 1$
5. goto 2.

Figure 7.11: D_EQAL: dynamically adjusting R .

the system utilization is. However under SRD or LRD arrival process, R , that starts from 0, converges toward the best performing. Under 20% utilized system, R oscillates around 60%, while under 62% and 80% systems, R is around 40% and 20% respectively. All these targeted values are the optimal R under S_EQAL in Section 7.2.

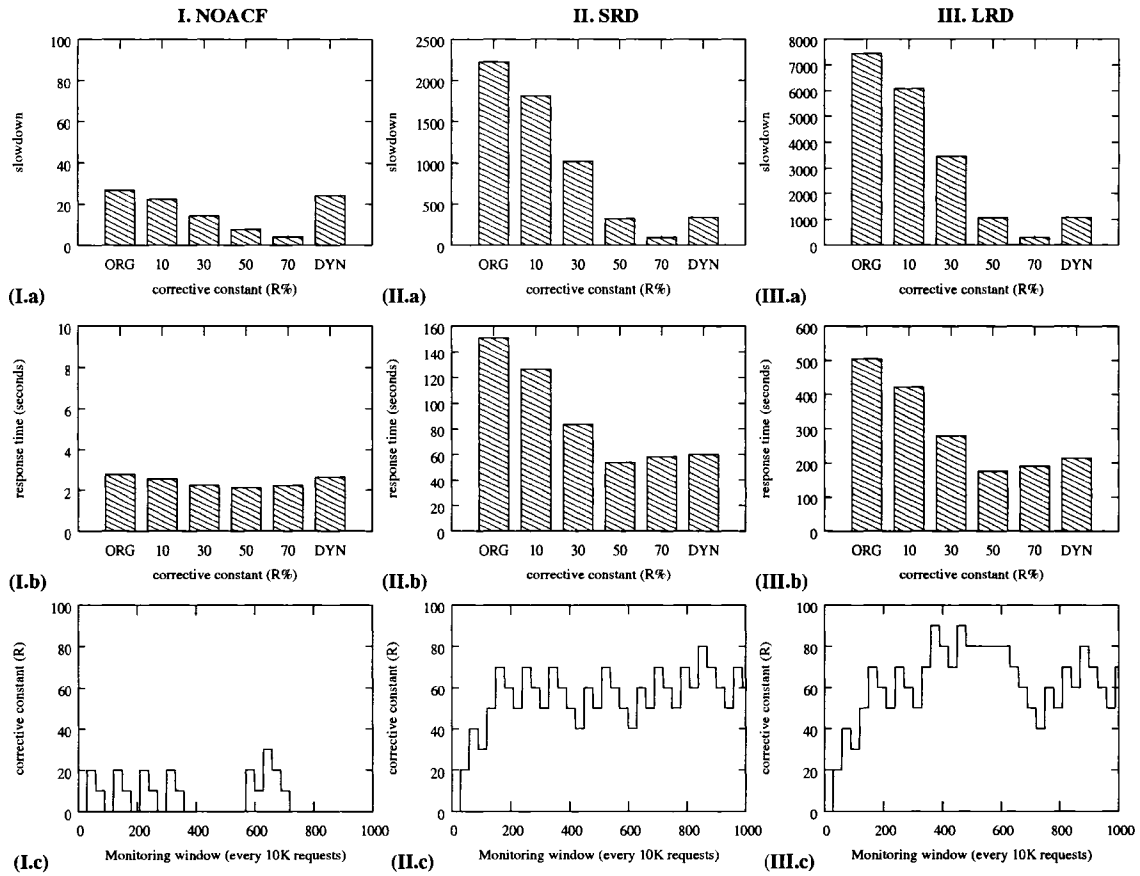


Figure 7.12: Performance of NOACF, SRD ad LRD traffic under average utilization 20%. The first two rows are average slowdown and average response time for the original ADAPTLOAD, S_EQAL with various values of R , and D_EQAL. The third row is the corrective constant R as a function of time (measured in processed requests) for $C = 300K$.

To further understand how D_EQAL adjusts R value, we present the performance of the on-line policy over time. Figure 7.15 gives results under medium load, i.e., 62% utilization (the x-axis of all the plots in Figure 7.15 represents the number of batches of K requests processed so far).⁷ Figures 7.15(b) and (c) show the average request slowdown and response time, respectively. Note the similar shape of both metrics across time, which indicates the effectiveness of load imbalance. As the average size of requests is increasing starting from the 600th monitoring window (see also Figure 7.2(b)), the utilization in this system becomes

⁷we observe the same over-time behavior of light load and heavy load as that of medium load so that we do not report them.

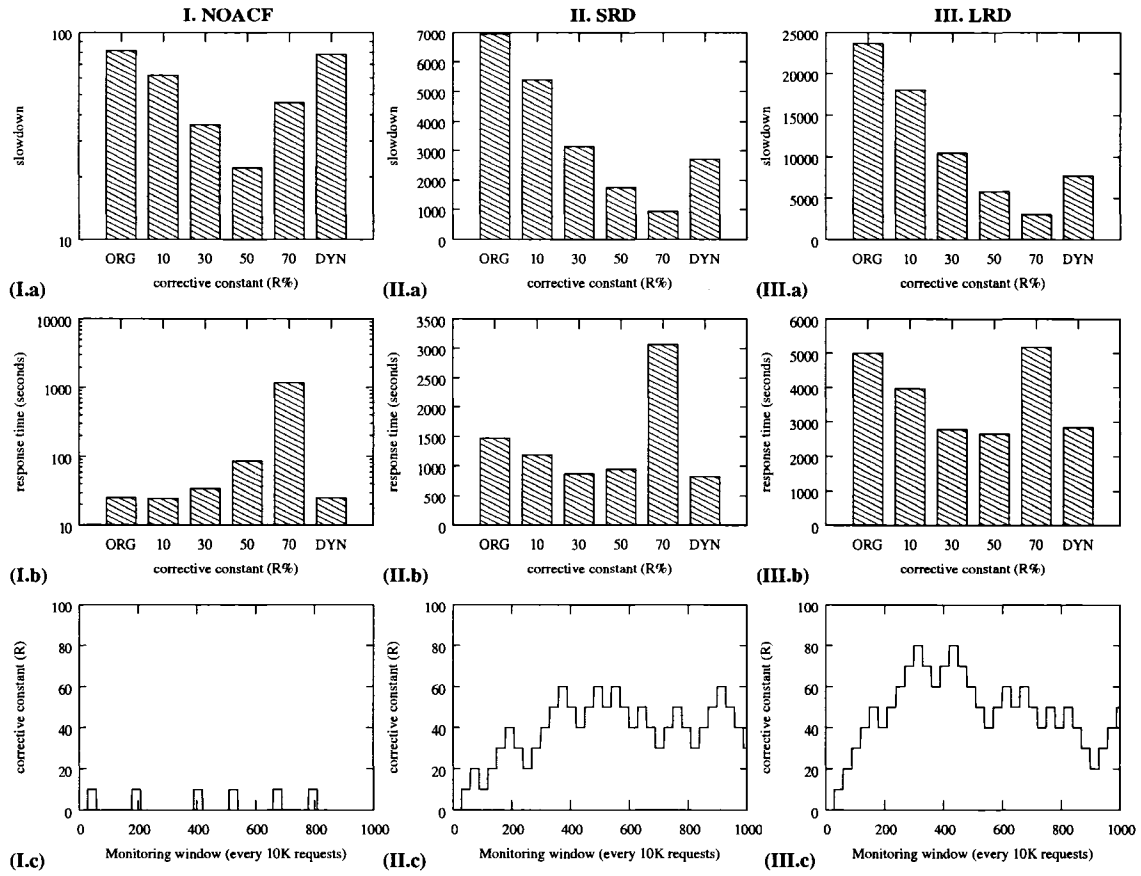


Figure 7.13: Performance of NOACF, SRD and LRD traffic under average utilization 62%. The first two rows are average slowdown and average response time for the original ADAPTLOAD, S_EQAL with various values of R , and D_EQAL. The third row is the corrective constant R as a function of time (measured in processed requests) for $C = 300K$.

larger thereafter. Correspondingly the R value decreases during this period, especially with LRD arrivals (see Figure 7.15(III.a)).

7.3 Summary

In this chapter, we evaluate the performance of sized-based load balancing policies for homogeneous clustered servers under correlated arrivals. We show that under correlated arrivals sized-based policies, which have been shown to successfully balance load and improve performance when service demands are highly variable, are now ineffective.

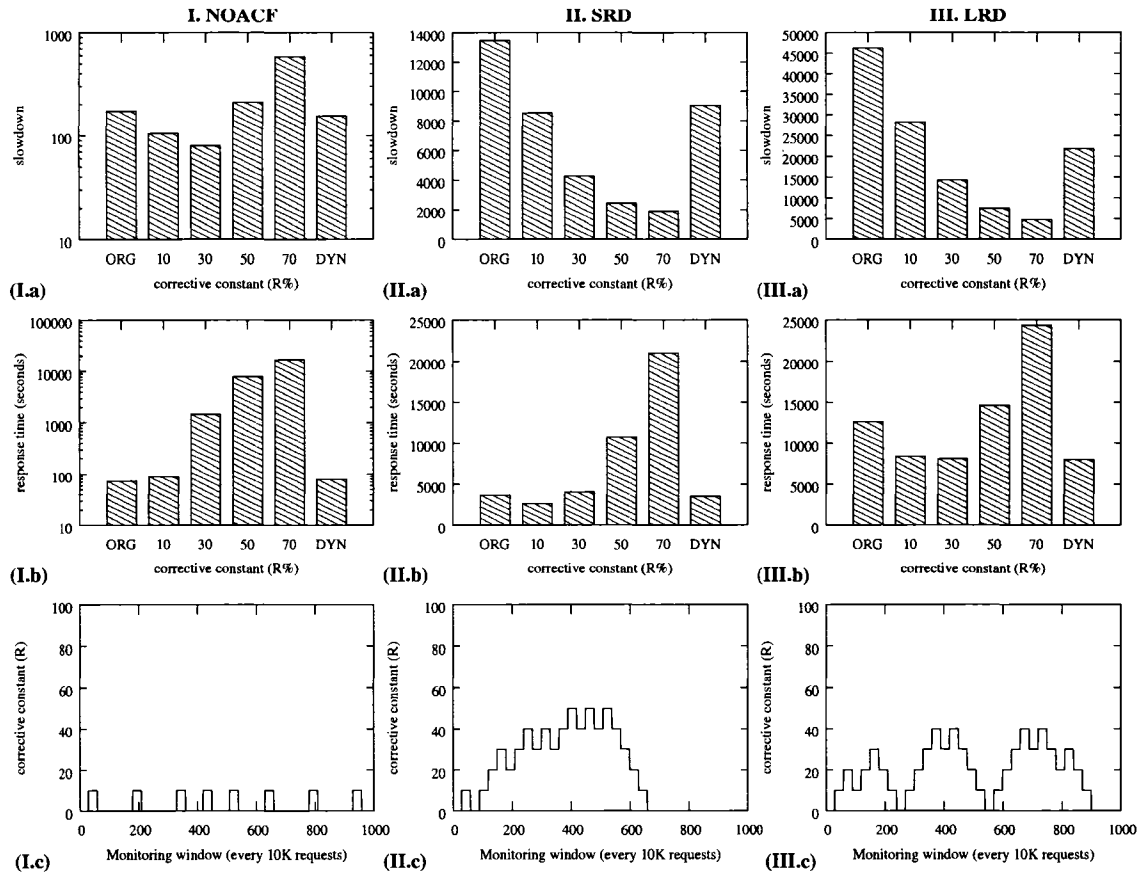


Figure 7.14: Performance of NOACF, SRD and LRD traffic under average utilization 80%. The first two rows are average slowdown and average response time for the original ADAPTLOAD, S_EQAL with various values of R , and D_EQAL. The third row is the corrective constant R as a function of time (measured in processed requests) for $C = 300K$.

Our experiments show that if the arrival process is correlated, then it is not enough for a size-based policy to equally distribute the work among the servers in the cluster – if the arrival streams to individual servers are correlated, then performance significantly degrades. We propose a new size-based load balancing policy, called D_EQAL, that strives to distribute the work such that the load to each server is proportional to the correlation structure of the arrival process to that server and still separates jobs to servers according to their sizes. As a result of this effort, not all servers are equally utilized (i.e., load in the system becomes unbalanced) but this imbalance results in significant performance benefits. D_EQAL does not require any prior knowledge of the correlation structure of the arrival stream or of the job

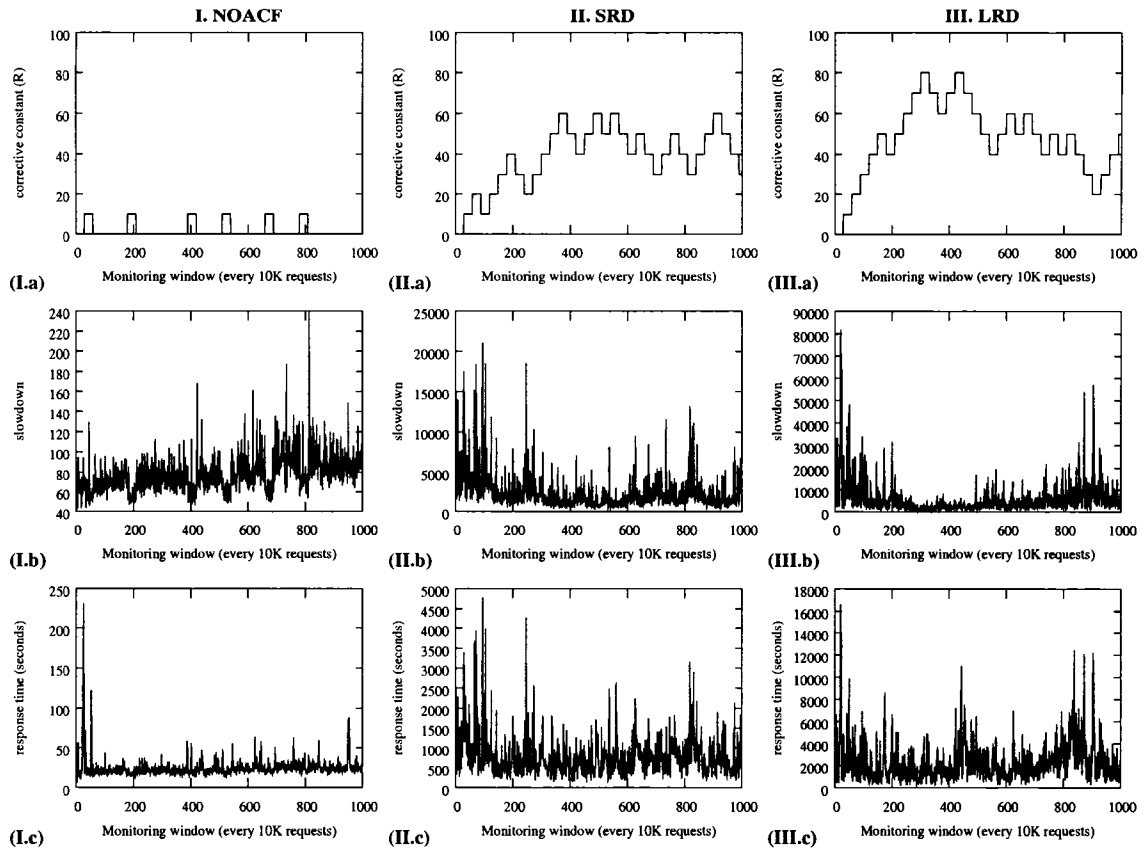


Figure 7.15: Corrective constant R , average slowdown, average normalized response time as a function of time (measured in processed requests) for $C = 300K$ under NOACF, SRD and LRD traffic. System utilization is 62%.

size distribution. Using trace-driven simulation, we show that D_EQAL is an effective on-line policy: by monitoring performance measures it self-adjusts its parameters to transient workload conditions.

Chapter 8

Conclusions and Future Work

This dissertation presents analysis of dependent flows and their impact on multi-tiered systems:

- by characterizing the propagation of autocorrelation and understanding its performance effects via experimentation of an on-line bookstore based on the TPC-W benchmark,
- by building analytic models that can be used to solve multi-tiered systems with dependent arrival/service processes, and
- by developing robust resource management policies under variable workload conditions.

More specifically, the existence of dependence in the service process is observed in the multi-tiered system driven by TPC-W. This is identified as the source of the dependent flows in *all* tiers of this system. Such dependent flows significantly deteriorate the end-to-end performance in spite of the fact that the bottleneck resource in the system is far from saturation and that the measured throughput and utilizations of other resources are also modest. When autocorrelation is not considered, this underutilization of resources falsely

indicates that the system can sustain higher capacities. Different from the tails of the overall response time distributions in the independent systems which reflect the time spent at the bottleneck tier, the tails are now shaped by the response time tail at the tier that is the source of autocorrelation. Based on this observation, an admission control algorithm is presented that takes autocorrelation into account and shows benefit in reducing the long tail of the response time distribution instead of doing admission control at the bottleneck tiers.

For systems with infinite buffer size, that performs as open systems, two families of analytic models that approximate the departure process of a BMAP/MAP/1 queue are presented, i.e., the ETAQA-based approximation and the lumpability-base approximation. These analytic models can be used to model each tier in isolation and to understand the following performance affected by dependent flows in multi-tiered systems. The properties of these two output approximation models are formally proved: the marginal distribution of the true departure process is exactly preserved; the lag correlations of the inter-departure times are matched up to lag $(n - 2)$ with the ETAQA-based approximation and up to lag $(n - 1)$ with the lumpability-based approximation for the output models with $n + 1$ block levels of the M/G/1-type process. The value of n can be automatically identified by the asymptotic properties of the approximations to maintain well the queueing behavior of the real departure process.

This dissertation also demonstrates how the knowledge of autocorrelation can be used to aid robust policy development by presenting D_EQAL, a size-based load balancing policy, for a cluster of homogeneous servers. D_EQAL aims at reducing the performance degradation due to autocorrelation and strives to distribute the work to each server in such a way that the assigned work is proportional to the correlation structure of the arrival process to that server. D_EQAL monitors performance measures as well as the dependence of the incoming traffic

to the cluster and self-adjusts its parameters to transient workload conditions. Performance evaluation using trace-driven simulation shows that D_EQAL achieves superior performance in comparison to other classic policies under correlated arrivals.

8.1 Future work

The research presented in this dissertation is introducing several open problems in two directions. The first direction is on the development of analytic models for closed and open systems with autocorrelated arrival and/or service processes. The second direction is on designing robust system policies where autocorrelation in arrivals/service flows is instrumental in policy decisions.

8.1.1 Analytic Direction

- **Analytic models of closed systems:** We will develop effective models similar to the traditional Mean Value Analysis (MVA) and convolution models that are traditionally used in closed queuing networks [43] to obtain approximate analytic solutions. We will use ideas for solutions of load-dependent queuing networks [13, 14] that have been recently proposed and apply them to closed queuing networks with autocorrelated service processes. In addition to the approximate analytic solutions, we expect to derive performance bounds of dependent systems in the spirit of the classic Asymptotic Bounds Analysis [43] and Balanced Job Bounds [43] that have been used for product-form networks.
- **Efficient models of open systems:** The analysis of multiple lumping method in Chapter 6 indicates the huge potential of obtaining better approximations of departure

processes by lumping states in different ways, intermixing lumps and exact states in the embedded Markov chain. We plan to further investigate this idea, understand how the ACF values of departure processes are controlled by more eigenvalues of the key matrix as well as the system utilization, and propose a more effective algorithm to optimize the approximation accuracy automatically.

- **Fitting correlated trace:** Different sample ordering of the same distribution results in processes with different dependent structures. Traditional fitting methods focus on matching the stochastic characteristics, i.e., the first moments of a real trace, while ignoring the development of dependence in the trace. We propose to devise MAP-fitting methods which will be aware of the temporal bias in a trace. This method will divide the trace into small portions within each no dependence is detected. Each portion can be modeled as an exponential or a phase-type process using simple fitting techniques. Then the transitions among these portions will be captured with Markov process and entire trace will be fitted into a MAP process.
- **Propagation characteristics:** We have observed that autocorrelation propagates in both open and closed systems, while lacking the knowledge that which process (arrival or service) dominates the departure process, especially under medium loads. The theory that we will develop will address the above question. Knowing which of the two processes dominates the departure flow has an immediate application in resource allocation, guiding the development of scheduling policies within each server that reduce correlation in the observed output, similar to input shaping that is widely used in networking research.

8.1.2 Application Direction

- **Admission control:** Our preliminary results on admission control guided by autocorrelation in flows achieve promising performance improvements (see Chapter 3). We will continue building on this idea and develop a sophisticated and practical policy that maintains comparable performance with the existing admission control policies by dropping a much smaller ratio of requests. We will specifically focus on web servers and storage systems.
- **Capacity planning:** The significant impact of dependence in system performance makes autocorrelation function a considerable factor in capacity planning and can not be overlooked. First, we will identify the system reasons that introduce dependence in the service process, e.g., the execution properties of application servers or the effect of caches. Once this knowledge is given, better system design, e.g., selection of the appropriate applications or hardware, might be able to reduce burstiness and improve system performance. For the systems where the dependence can not be eliminated, a robust capacity planning that takes correlation into account will be proposed to answer the typical question: How many number of clients can be supported by the existing system while satisfying pre-defined server level agreements.
- **Scheduling classification:** Research has shown that Shortest-Job-First (SJB) and its pre-emptive counterpart Shortest-Remaining-Processing-Time (SRPT) are superior to FCFS and yet do not starve large jobs if the workload is heavy-tailed [94, 93]. We will provide a new classification of these scheduling policies by showing what they do to autocorrelation of the service process. An immediate application of the observations for autocorrelation in scheduling can be used to improve disk scheduling. We anticipate

that based on on-line measurement of autocorrelation, we will be able to devise new scheduling policies that are oblivious of the size of requests, but use the statistical knowledge of the size of the next batch of jobs (as given by the autocorrelation values of the first lags) to determine an effective scheduling solution.

Appendix A

Equivalence of open/closed systems

Equivalence of open/closed systems

Open systems with finite queue capacity or admission control behave in essence like closed systems [63], therefore the performance effects of autocorrelation in a closed system directly apply to open systems with finite buffers. This equivalence of open/closed systems is illustrated here via a simple example. Figure A.1(a) shows an M/M/1 system, i.e., arrivals are Poisson and service times are exponential with parameters λ and μ , respectively. The system has a finite buffer size equal to M , i.e., incoming requests are rejected when the buffer is full. The state space in this system is denoted by an integer number i , number of jobs in the system, see the Markov chain of this system in Figure A.1(b).

A two-queue closed system with a limitation on the number of circulating jobs as expressed by the multiprogramming level $MPL = M$ is also shown in Figure A.1. The first queue “simulates” the arrival of jobs in the system from the outside world. The state space of this system is expressed by two integers (i, j) , $0 \leq i, j \leq M$, while $i + j = M$ for all states, see Figure A.1(d). There is a one-to-one correspondence of the two Markov chains, which implies that the probability distributions of the states in both chains are identical. Based on

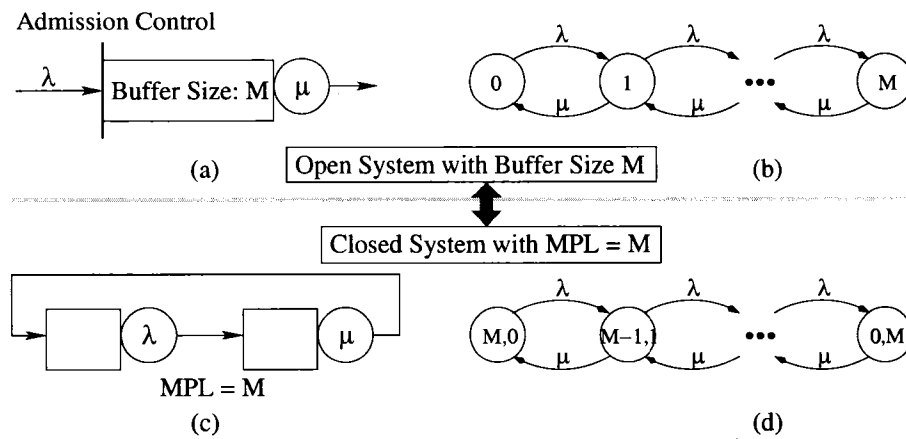


Figure A.1: Illustration of the equivalence of an open system with finite buffer of size M and a closed system with $MPL = M$.

this observation and Little's law, one can trivially show that the performance measures of the queue with rate μ are identical for both systems. This equivalence can be easily extended to systems with MAP distributions that describe the arrival/service processes.

Bibliography

- [1] D. ABENDROTH AND U. KILLAT. Intelligent shaping: Well shaped throughout the entire network? In *Proceedings of the IEEE INFOCOM 2002*, volume 2, pages 912–919, New York City, NY, June 2002.
- [2] A. M. ADAS AND A. MUKHERJEE. On resource management and QoS guarantees for long range dependent traffic. In *Proceedings of the IEEE INFOCOM 1995*, pages 779–787, Boston, MA, April 1995.
- [3] C. AMZA, E. CECCHET, A. CHANDA, A. COX, S. ELNIKETY, R. GIL, J. MARGUERITE, K. RAJAMANI, AND W. ZWAENEPOEL. Specification and implementation of dynamic content benchmarks. In *Proceedings of the 5th IEEE Workshop on Workload Characterization (WWC-5)*, Austin, TX, November 2002.
- [4] C. AMZA, A. COX, AND W. ZWAENEPOEL. A comparative evaluation of transparent scaling techniques for dynamic content servers. In *Proceedings of the 21st International Conference on Data Engineering (ICDE 2005)*, pages 230–241, Tokyo, Japan, April 2005.
- [5] M. ANDREOLINI, M. COLAJANNI, AND R. MORSELLI. Performance study of dispatching algorithms in multi-tier web architectures. *ACM SIGMETRICS Performance Evaluation Review*, 30(2):10–20, September 2002.
- [6] M. ARLITT, D. DRISHNAMURTHY, AND J. ROLIA. Characterizing the scalability of a large Web-based shopping system. *ACM Transactions on Internet Technology*, 1(1):44–69, August 2001.
- [7] M. ARLITT AND T. JIN. Workload characterization of the 1998 World Cup Web site. Technical Report HPL-1999-35R1, Hewlett-Packard Laboratories, September 1999.
- [8] G. BANGA AND P. DRUSCHEL. Measuring the capacity of a Web server. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems (USITS'97)*, pages 61–72, Monterey, CA, December 1997.
- [9] J. BERAN. *Statistics for Long-Memory Processes*. Chapman & Hall, New York, 1994.
- [10] P. BUCHHOLZ. An EM-algorithm for MAP fitting from real traffic data. In *Proceedings of the 13th International Conference on Modeling Techniques and Tools for Computer Performance Evaluation (Performance TOOLS 2003)*, P. Kemper and W.H. Sanders, editors, volume 2794 of *LNCS*, pages 218–236. Springer-Verlag, 2003.

- [11] P. BUCHHOLZ, G. HORVÁTH, AND M. TELEK. A MAP fitting approach with independent approximation of the inter-arrival time distribution and the lag correlation. In *Proceedings of the 2nd International Conference on Quantitative Evaluation of Systems (QEST'05)*, pages 124–133, Torino, Italy, September 2005.
- [12] D. BUSHMITCH, S. S. PANWAR, AND A. PAL. Thinning, striping and shuffling: Traffic shaping and transport techniques for variable bit rate video. In *Proceedings of the IEEE Global Communications Conference (GLOBECOM 2003)*, volume 2, pages 1485–1491, Taipei, November 2003.
- [13] G. CASALE. An efficient algorithm for the exact analysis of multiclass queueing networks with large population sizes. In *Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS/Performance 2006)*, pages 169–180, Saint Malo, France, June 2006.
- [14] G. CASALE. On single class load-dependent normalizing constant equations. In *Proceedings of the 3rd International Conference on Quantitative Evaluation of Systems (QEST'06)*, pages 333–342, Riverside, CA, September 2006.
- [15] L. CHERKASOVA AND P. PHAAL. Session based admission control: a mechanism for peak load management of commercial web sites. *IEEE Transactions on Computers*, 51(6):669–685, June 2002.
- [16] L. CHERKASOVA, W. TANG, AND S. SINGHAL. An SLA-oriented capacity planning tool for streaming media services. In *Proceedings of the International Conference on Dependable Systems and Networks, (DSN'2004)*, pages 743–752, Florence, Italy, June 2004.
- [17] K. J. CHRISTENSEN AND V. BALLINGAM. Reduction of self-similarity by application-level traffic shaping. In *Proceedings of the 22nd IEEE Conference on Local Computer Networks (LCN'97)*, pages 511–518, Minneapolis, MA, November 1997.
- [18] I. COHEN, J. S. CHASE, M. GOLDSZMIDT, T. KELLY, AND JULIE SYMONS. Correlating instrumentation data to system states: A building block for automated diagnosis and control. In *Proceedings of the 6th Symposium on Operating Systems Design and Implementation (OSDI'04)*, pages 231–244, San Francisco, CA, December 2004.
- [19] M. E. CROVELLA AND A. BESTAVROS. Self-similarity in world wide web traffic: evidence and possible causes. In *Proceedings of the Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'96)*, pages 160–169, May 1996.
- [20] R. L. CRUZ. Service burstiness and dynamic burstiness measures: A framework. *Journal of High Speed Networks*, 1(2):105–127, 1992.
- [21] S. ELNIKETY, E. NAHUM, J. TRACEY, AND W. ZWAENEPOEL. A method for transparent admission control and request scheduling in e-commerce Web sites. In *Proceedings of the 13th International Conference on World Wide Web (WWW2004)*, pages 276–286, New York, NY, 2004.
- [22] A. ERRAMILI, O. NARAYAN, AND W. WILLINGER. Experimental queueing analysis with long-range dependent packet traffic. *IEEE/ACM Transactions on Networking*, 4(2):209–223, 1996.

- [23] H. FENG, M. VISRA, AND D. RUBENSTEIN. Optimal state-free, size-aware dispatching for heterogeneous M/G/-type systems. *Performance Evaluation Journal*, 62(1-4):475–492, 2005.
- [24] H.-W. FERNG AND J.-F. CHANG. Departure processes of BMAP/G/1 queues. *Queueing Systems*, 39(2-3):109–135, 2001.
- [25] G. M. C. GAMA, W. MEIRA JR., AND M. L. B. CARVALHO. Resource placement in distributed e-commerce servers. In *The Evolving Global Communications Network (GLOBECOM 2001)*, volume 3, pages 1677–1682, San Antonio, TX, November 2001.
- [26] G. R. GANGER, B. L. WORTHINGTON, AND Y. N. PATT. The DiskSim simulation environment, Version 2.0, Reference manual. Technical report, Electrical and Computer Engineering Department, Carnegie Mellon University, 1999.
- [27] D.F. GARCIA AND J. GARCIA. TPC-W e-commerce benchmark evaluation. *IEEE Computer*, 36(2):42–48, February 2003.
- [28] M. E. GOMEZ AND V. SANTONJA. On the impact of workload burstiness on disk performance. *Workload Characterization of Emerging Computer Applications (The International Series in Engineering and Computer Science)*, pages 181–201, 2001.
- [29] D. GREEN. *Departure Processes from MAP/PH/1 Queues*. PhD thesis, Department of Applied Mathematics, University of Adelaide, South Australia, 1999.
- [30] D. GREEN. Lag correlations of approximating departure processes of MAP/PH/1 queues. In *Proceedings of the 3rd International Conference on Matrix-Analytic Methods in Stochastic Models (MAM3)*, pages 135–151, Leuven, Belgium, July 2000.
- [31] S. D. GRIBBLE, G. S. MANKU, D. ROSELLI, E. A. BREWER, T. J. GIBSON, AND E. L. MILLER. Self-similarity in file systems. In *Proceedings of the Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'98)*, pages 141–150, Madison, WI, June 1998.
- [32] M. HARCHOL-BALTER, M.E. CROVELLA, AND C.D. MURTA. On choosing a task assignment policy for a distributed server system. In *Proceedings of the 10th International Conference on Computer Performance Evaluation: Modeling Techniques and Tools (Performance TOOLS 1998)*, pages 231–242, Palma de Mallorca, Spain, September 1998.
- [33] M. HARCHOL-BALTER AND A. DOWNEY. Exploiting process lifetime distributions for dynamic load balancing. *ACM Transactions on Computer Systems*, 15(3):253–285, August 1997.
- [34] M. HARCHOL-BALTER, B. SCHROEDER, N. BANSAL, AND M. AGRAWAL. Size-based scheduling to improve web performance. *ACM Transactions on Computer Systems*, 21(1):207–233, May 2003.
- [35] A. HEINDL. *Traffic-Based Decomposition of General Queueing Networks with Correlated Input Processes*. PhD thesis, Department of Electrical Engineering and Computer Science, TU Berlin, Aachen, Germany, 2001.

- [36] A. HEINDL, Q. ZHANG, AND E. SMIRNI. ETAQA Truncation Models for the MAP/MAP/1 Departure Process. In *Proceedings of the 1st International Conference on Quantitative Evaluation of Systems (QEST'04)*, pages 90–99, Enschede, Netherlands, September 2004.
- [37] A. HORVÁTH AND M. TELEK. Markovian modeling of real data traffic: Heuristic phase-type and MAP fitting of heavy tailed and fractal-like samples. In *Tutorials of the IFIP WG7.3 Int. Symposium on Computer Performance Modeling, Measurement and Evaluation*, LNCS 2459, pages 405–434, Rome, Italy, September 2002.
- [38] A. KAMRA, V. MISRA, AND E. NAHUM. Yaksha: A Self-Tuning Controller for Managing the Performance of 3-Tiered Web sites. In *Proceedings of the Twelfth IEEE International Workshop on Quality of Service (IWQoS 2004)*, pages 47–58, Montreal, Canada, June 2004.
- [39] L. KLEINROCK. *Queueing Systems, Volume I: Theory*. Wiley, 1975.
- [40] A. KLEMM, C. LINDEMANN, AND M. LOHMANN. Modeling IP traffic using the Batch Markovian Arrival Process. *Performance Evaluation*, 54(2):149–173, 2003.
- [41] J. KUMARAN, K. MITCHELL, AND A. VAN DE LIEFVOORT. Characterization of the departure process from an ME/ME/1 queue. *RAIRO Recherche Operationelle / Operations Research*, 38(2):173–191, 2004.
- [42] G. LATOUCHE AND V. RAMASWAMI. *Introduction to Matrix-Analytic Methods in Stochastic Modeling*. Series on statistics and applied probability. ASA-SIAM, 1999.
- [43] E. D. LAZOWSKA, J. ZAHORJAN, G. S. GRAHAM, AND K. C. SEVCIK. *Computer System Analysis Using Queueing Network Models*. Prentice-Hall, Inc, New York, 1984.
- [44] L. H. LEEMIS AND S. K. PARK. *Discrete-event Simulation: A First Course*. Prentice Hall, 2005.
- [45] W. E. LELAND, M. S. TAQQU, W. WILLINGER, AND D. V. WILSON. On the self-similar nature of Ethernet traffic. *IEEE/ACM Transactions on Networking*, 2(1):1–15, 1994.
- [46] L. LIPSKY. *Queueing Theory: A linear algebraic approach*. MacMillan, New York, 1992.
- [47] Z. LIU, M. SQUILLANTE, AND J. WOLF. On maximizing service-level-agreement profits. In *Proceedings of the Third ACM Conference on Electronic Commerce (EC'01)*, pages 213–223, Tampa, Florida, October 2001.
- [48] D. M. LUCANTONI. New results on the single server queue with a batch Markovian arrival process. *Communications in Statistics, Stochastic Models*, 7(1):1–46, 1991.
- [49] D. MCWHERTER, B. SCHROEDER, N. AILAMAKI, AND M. HARCHOL-BALTER. Priority mechanisms for oltp and transactional web applications. In *Proceedings of the 20th International Conference on Data Engineering (ICDE 2004)*, pages 535–546, Boston, MA, April 2004.

- [50] D. MENASCE, V. ALMEIDA, R. RIEDI, F. PELEGRINELLI, R. FONSECA, AND W. MEIRA. In search of invariants for e-business workloads. In *Proceedings of the Second ACM Conference on Electronic Commerce (EC'00)*, pages 56–65, Minneapolis, MN, October 2000.
- [51] D. MENASCE, D. BARBARA, AND R. DODGE. Preserving qos of e-commerce sites through self-tuning: A performance model approach. In *Proceedings of the Third ACM Conference on Electronic Commerce (EC'01)*, pages 224–234, Tampa, Florida, October 2001.
- [52] D. A. MENASCE AND V. A. F. ALMEIDA. *Capacity planning for Web services: metrics models and methods*. Prentice Hall, Inc., NJ, 2001.
- [53] D. A. MENASCE, V. A. F. ALMEIDA, AND L.W. DOWDY. *Capacity Planning and Performance Modeling: From Mainframes to Client-Server Systems*. Prentice Hall, Inc., NJ, 1994.
- [54] N. MI, Q. ZHANG, A. RISKA, AND E. SMIRNI. Performance impacts of autocorrelation in tpc-w. Technical Report WM-CS-2005-35, Department of Computer Science, College of William and Mary, November 2005.
- [55] K. MITCHELL AND A. VAN DE LIEFVOORT. Approximation models of feed-forward G/G/1/N queueing networks with correlated arrivals. *Performance Evaluation*, 51(2-4):137–152, 2003.
- [56] J. C. MOGUL. Operating systems support for busy internet servers. In *Proceedings of the Fifth Workshop on Hot Topics in Operating Systems (HotOS-V)*, pages 1–15, Orcas Island, WA, May 1995.
- [57] A. W. MOORE. *Measurement-based management of network resources*. PhD thesis, Computer Laboratory, University of Cambridge, Cambridge, UK, 2002.
- [58] MySQL AB. *MySQL*. <http://www.mysql.com>.
- [59] V. A. NAOUMOV, U. KRIEGER, AND D. WAGNER. Analysis of a multi-server delay-loss system with a general Markovian arrival process. In *Proceedings of the Second International Conference on Matrix-Analytic Methods in Stochastic Models (MAM2)*, Chakravorthy and Alfa, editors, pages 43–66, New York, 1997. Marcel Dekker.
- [60] R. NELSON. *Probability, Stochastic Processes, and Queueing Theory*. Springer-Verlag, 1995.
- [61] M. F. NEUTS. *Structured Stochastic Matrices of M/G/1-type and their Applications*. Marcel Dekker, New York, NY, 1989.
- [62] M. F. NEUTS. *Algorithmic Probability: A Collection of Problems*. Chapman and Hall, 1995.
- [63] R. O. ONVURAL AND H. G. PERROS. Equivalences between open and closed queueing networks with finite buffers. *Performance Evaluation*, 9(4):263–269, 1989.

- [64] V. N. PADMANABHAN AND L. QIU. The content and access dynamics of a busy web site: Findings and implications. In *Proceedings of ACM SIGCOM'02*, pages 111–123, Stockholm, Sweden, August 2000.
- [65] V. S. PAI, M. ARON, G. BANGA, M. SVENDSEN, P. DRUSCHEL, W. ZWAENEPOEL, AND E. NAHUM. Locality-aware request distribution in cluster-based network servers. In *Proceedings of the Eighth International Conference on Architectural Support for Programming Languages and Operating Systems, (ASPLOS-VIII)*, pages 205–216, San Jose, CA, October 1998.
- [66] S. PANDEY, K. RAMAMRITHAM, AND S. CHAKRABARTI. Monitoring the dynamic web to respond to continuous queries. In *Proceedings of the Twelfth International World Wide Web Conference (WWW2003)*, pages 659–668, Budapest, Hungary, May 2003.
- [67] V. PAXSON AND S. FLOYD. Wide-area traffic: the failure of Poisson modeling. *IEEE/ACM Transactions on Networking*, 3(3):226–244, June 1995.
- [68] PHARM PROJECT. *Java TPC-W Implementation Distribution*. <http://www.ece.wisc.edu/pharm/>, Department of Electrical and Computer Engineering and Computer Sciences Department, University of Wisconsin-Madison.
- [69] V. RAMASWAMI. A stable recursion for the steady-state vector in Markov chains of m/g/1 type. *Communications in Statistics, Stochastic Models*, 4:183–263, 1988.
- [70] S. RANJAN, J. ROLIA, H. FU, AND E. KNIGHTLY. Qos-driven server migration for internet data centers. In *Proceedings of the Tenth IEEE International Workshop on Quality of Service (IWQoS 2002)*, pages 3–12, Miami Beach, FL, May 2002.
- [71] A. RISKÄ AND E. RIEDEL. Disk drive level workload characterization. In *Proceedings of the USENIX Annual Technical Conference*, pages 97–103, Boston, MA, June 2006.
- [72] A. RISKÄ AND E. SMIRNI. Exact aggregate solutions for M/G/1-type Markov processes. In *Proceedings of ACM SIGMETRICS Conference*, pages 86–96, Marina del Rey, CA, June 2002.
- [73] A. RISKÄ AND E. SMIRNI. MAMSolver: a matrix-analytic methods tools. In *Proceedings of the 12th International Conference on Modeling Tools and Techniques for Computer and Communication System Performance Evaluation (Performance TOOLS 2002)*, T. Field, P. Harrison, J. Bradley, and U. Harder, editors, pages 205–211, London, UK, April 2002. Springer-Verlag.
- [74] A. RISKÄ, M. SQUILLANTE, S. YU, Z. LIU, AND L. ZHANG. Matrix-analytic analysis of a MAP/PH/1 queue fitted to web server data. In *Matrix-Analytic Methods; Theory and Applications*, G. Latouche and P. Taylor, editors, pages 333–356. World Scientific, 2002.
- [75] J. A. ROLIA AND K. C. SEVCIK. The method of layers. *IEEE Transactions on Software Engineering*, 21(8):689–700, 1995.
- [76] Y. SAAD. *Numerical methods for large eigenvalue problems*. Manchester University Press, 1992.

- [77] R. SADRE AND B. HAVERKORT. Characterizing traffic streams in networks of MAP/MAP/1 queues. In *Proceedings of the 11th GI/ITG Conference on Measuring, Modeling and Evaluation of Computer and Communication Systems*, pages 195–208, Aachen, Germany, 2001.
- [78] F. SHEIKH AND C. M. WOODSIDE. Layered analytic performance modeling of distributed database systems. In *Proceedings of the International Conference on Distributed Computer Systems*, pages 482–490, Baltimore, MD, May 1997.
- [79] D. J. SORIN, J. L. LEMON, D. L. EAGER, AND M. K. VERNON. Analytic evaluation of shared-memory architectures. *IEEE Transactions on Parallel and Distributed Systems*, 14(2):166–180, February 2003.
- [80] C. STEWART AND K. SHEN. Performance modeling and system management for multi-component online services. In *Proceedings of USENIX NSDI*, Boston, MA, May 2005.
- [81] Y. M. TEO AND R. AYANI. Comparison of load balancing strategies on cluster-based web servers. *Simulation: Transactions of the Society for Modeling and Simulation International*, 77(5-6):185–195, November 2001.
- [82] THE APACHE SOFTWARE FOUNDATION. *Apache Web Server*. <http://www.apache.org>.
- [83] TRANSACTION PROCESSING AND PERFORMANCE COUNCIL. *TPC-W*. <http://www.tpc.org>.
- [84] V. UNGUREANU, B. MELAMED, P. G. BRADFORD, AND M. KATEHAKIS. Class-dependent assignment in cluster-based servers. In *Proceedings of the 2004 ACM symposium on Applied computing (SAC'04)*, pages 1420–1425, Nicosia, Cyprus, March 2004.
- [85] B. URGONKAR, G. PACIFICI, P.J. SHENOY, M. SPREITZER, AND A.N.S TANTAWI. An analytical model for multi-tier internet services and its applications. In *Proceedings of the Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'05)*, pages 291–302, Banff, Canada, June 2005.
- [86] U. VALLAMSETTY, K. KANT, AND P. MOHAPATRA. Characterization of e-commerce traffic. In *Advanced Issues of E-Commerce and Web-Based Information Systems (WECWIS 2002)*, pages 137–144, Newport Beach, California, June 2002.
- [87] D. VILLELA, P. PRADHAN, AND D. RUBENSTEIN. Provisioning Servers in the Application Tier for E-commerce Systems. In *Proceedings of the Twelfth IEEE International Workshop on Quality of Service (IWQoS 2004)*, pages 57–66, Montreal, Canada, June 2004.
- [88] VMWARE INC. *VMWare Workstation*. <http://www.vmware.com>.
- [89] Q. WANG, D. MAKAROFF, H. K. EDWARDS, AND R. THOMPSON. Workload characterization for an e-commerce web site. In *Proceedings of the 2003 conference of the Centre for Advanced Studies on Collaborative research (CASCON 2003)*, pages 313–327, Toronto, Ontario, Canada, October 2003.

- [90] H. W. CAIN, R. RAJWAR, M. MARDEN, AND M. H. LIPASTI. An architectural evaluation of java tpc-w. In *Proceedings of the Seventh International Symposium on High-Performance Computer Architecture (HPCA'01)*, pages 229–240, Nuevo Leone, Mexico, January 2001.
- [91] B. P. WELFORD. Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4:419–420, 1962.
- [92] M. WELSH AND D. CULLER. Adaptive overload control for busy Internet servers. In *Proceedings of the 4th USENIX Symposium on Internet Technologies and Systems, (USITS'03)*, Seattle, WA, 2003.
- [93] A. WIERMAN AND M. HARCHOL-BALTER. Classifying scheduling policies with respect to unfairness in an M/GI/1. In *Proceedings of the Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'03)*, pages 238–249, San Diego, CA, June 2003.
- [94] A. WIERMAN, M. HARCHOL-BALTER, AND T. OSOGAMI. Nearly insensitive bounds on smart scheduling. In *Proceedings of the Conference on Measurement and Modeling of Computer Systems (SIGMETRICS'05)*, pages 205–216, Banff, Canada, June 2005.
- [95] R. WOLSKI, N. T. SPRING, AND J. HAYES. Predicting the CPU availability of time-shared unix systems on the computational grid. *Cluster Computing: the Journal of Networks, Software Tools and Applications will provide a forum for presenting*, 3(4):293–301, 2000.
- [96] B. L. WORTHINGTON, G. R. GANGER, AND Y. N. PATT. Scheduling for modern disk drives and non-random workloads. Technical Report CSE-TR-194-94, Computer Science and Engineering Division, University of Michigan, 1994.
- [97] F. XUE AND S. J. B. YOO. Self-similar traffic shaping at the edge router in optical packet-switched networks. In *Proceedings of the IEEE International Conference on Communications (ICC'02)*, volume 4, pages 2449–2453, April 2002.
- [98] Q. ZHANG, L. CHERKASOVA, AND E. SMIRNI. Flexsplit: A workload-aware, adaptive load balancing strategy for media clusters. In *Proceedings of the 13th Annual Multimedia Computing and Networking (MMCN'06)*, volume 6071, pages 01(1–12), San Jose, CA, January 2006.
- [99] Q. ZHANG, A. HEINDL, AND E. SMIRNI. Characterizing the BMAP/MAP/1 departure process using matrix exponentials. Technical Report WM-CS-2005-10, Department of Computer Science, College of William and Mary, September 2005.
- [100] Q. ZHANG, A. HEINDL, AND E. SMIRNI. Characterizing the BMAP/MAP/1 departure process via the ETAQA truncation. *Stochastic Models*, 21(2-3):821–846, 2005.
- [101] Q. ZHANG, A. RISKI, W. SUN, E. SMIRNI, AND G. CIARDO. Workload-aware load balancing for clustered web servers. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 16(3):219–233, March 2005.

VITA

Qi Zhang

Qi Zhang received the BS and MS degrees in computer science from the Huazhong University of Science and Technology and the University of Science and Technology of China, in 1998 and 2001, respectively. She has been a PhD candidate of Computer Science at the College of William and Mary since 2003. Her research interests are system modeling and performance analysis, resource management and capacity planning in multi-tiered systems, quality of services provision in e-commerce Web servers, and load balancing in cluster systems.