NETWORKS AND TELECOMMUNICATIONS SERIES

Resource Optimization and Security for Cloud Services

Kaiqi Xiong

ISTE WILEY

Resource Optimization and Security for Cloud Services

Resource Optimization and Security for Cloud Services

Kaiqi Xiong

Series Editor
Harry Perros

ISTE WILEY

First published 2014 in Great Britain and the United States by ISTE Ltd and John Wiley & Sons, Inc.

ISTE Ltd
27-37 St George's Road
London SW19 4EU
UK

www.iste.co.uk

John Wiley & Sons, Inc.
111 River Street
Hoboken, NJ 07030
USA

www.wiley.com

Library of Congress Control Number: 2013953945

British Library Cataloguing-in-Publication Data
A CIP record for this book is available from the British Library
ISBN: 978-1-84821-599-3

Printed and bound in Great Britain by CPI Group (UK) Ltd., Croydon, Surrey CR0 4YY

Table of Contents

Preface

With the number of e-business applications dramatically increasing, service-level agreements (SLAs) will play an important part in distributed and cloud service computing. An SLA is a combination of several qualities of service (QoS) metrics, such as security, performance and availability, agreed between a customer and a service provider. Due to the complexity of these metrics, most existing research typically addresses only one of these QoS metrics. In the case of the response time as a performance metric, the average time to process and complete a job is typically used in the literature. However, this may not be of real interest to a customer. A statistically bounded metric, that is, a percentile response time, is more realistic than the average response time. Moreover, in cloud computing, customer requests are typically distinguished by different request characteristics and service requirements.

This book is a research monograph. It covers the state of the art in the resource optimization and security of cloud services. The book includes a study of trustworthiness, percentile response time, service availability and authentication among cloud service stations or sites that may be owned by different service providers. Cloud services are primarily supported through data centers and this book

mainly deals with the end-to-end performance and security of cloud services. Thus, this book focuses on the study of end-to-end performance and security between cloud users and data centers, instead of the discussion of cloud virtualization technologies. First, it contains an analysis of percentile response time, which is one of the most important SLA metrics. Effective and accurate numerical solutions for the calculation of percentile response time in single-class and multi-class queuing networks are obtained. Then, the numerical solution is incorporated into a resource allocation problem. Specifically, we present an approach for the resource optimization that minimizes the total cost of computer resources required while preserving a given percentile of the response time.

Second, we extend the approach to consider trustworthiness, service availability and the percentile of response time in Web services. We clearly define these QoS metrics and provide their quantitative analysis. Then, we take into account these QoS metrics in a trust-based resource allocation problem in which a set of computer resources is used by a service provider to host a typical Web services application for single-class and multiple-class customer services, respectively. We formulate the trust-based resource allocation problem as an optimization problem under SLA constraints in which we calculate the number of servers in each service site that minimize a cost function that reflects operational costs for single-class and multiple-class customer services, respectively. We solve this problem using an efficient numerical procedure. Experimental results show the applicability of the procedure and validate its accuracy.

Finally, we first present a thorough performance evaluation of two notable public key cryptography-based authentication techniques, public-key cross-realm authentication in Kerberos (PKCROSS) and public key

utilizing tickets for application servers (PKTAPP, also known as KX.509/KCA), in terms of computational and communication times. We then demonstrate their performance difference using queuing networks. PKTAPP was proposed to address the scalability issue of PKCROSS. However, our in-depth analysis of these two techniques shows that PKTAPP does not perform better than PKCROSS in a large-scale system. Thus, we propose a new public-key cryptography-based group authentication technique. Our performance analysis demonstrates that the new technique can perform better than PKCROSS and PKTAPP.

As mentioned above, this book is a research monograph. It collects the author's recent studies in the field. The book may be used as a reference book by those researchers and engineers who work and those students who study in the fields of distributed computing, cloud computing, service computing, networks and telecommunication, and network security.

Chapter 1

Introduction

This book is concerned with resource optimization problems subject to various constraints including service availability, performance and security, and the problem of public-key cryptography-based group authentication. The motivation of our research is discussed in section 1.1, the formulation of the optimization problems subject to various constraints is given in section 1.2 and the performance of public-key cryptography-based group authentication is presented in the same section as well. Section 1.3 summarizes the contributions of our research, and section 1.4 gives the organization of this book.

1.1. Motivation

The management of the Quality-of-Service (QoS) is fundamental to a distributed enterprise computing system including cloud computing. The increasing pervasiveness of network connectivity and the proliferation of on-demand e-business applications and services in public domains, corporate networks as well as home environments, give rise

to the need for the design of appropriate service solutions. When evaluating how a cloud service provider can support the customer's requirements of an enterprise, several questions often arise:

– How can a service provider more effectively manage its service resources to support the customer's requirements?

– How can a service provider achieve higher return on investment through improved utilization of its existing service resources?

– How can a customer receive his/her service from a reliable service provider?

– How can a customer receive better QoS at a lower fee?

– How can multiple service providers authenticate each other such that they can work together to provide the QoS for a customer?

– How does the process of authentication among service providers affect the overall QoS?

Accurately predicting e-business application performance based on system statistics and a customer's perceived quality allows a service provider not only to assure the QoS, but also to avoid overprovisioning to meet a service-level agreement (SLA).

An SLA sets the expectations between a customer and a service provider, and helps define the relationship between these two parties. It is the cornerstone of how the service provider sets and maintains commitments to the customer. Usually, an SLA is a set of QoS metrics and a price agreed between a customer and a service provider. It plays an important role in an e-business application. The set of the QoS metrics generally consists of availability, security and performance that are defined in the following:

– *Availability* is the percentage of time that a service provider can offer services. Capabilities in the availability

component are determined by the resiliency of the environment of service resources. They are related to recovery mechanisms and data replications that improve service survivability.

– *Security* can be categorized as *identity security* and *behavior security*. Identity security includes the authentication and authorization between a customer and a service provider, data confidentiality and data integrity. Behavior security includes the trustworthiness among multiple resource sites or stations (both terms are used indistinguishably in this book), and the trustworthiness of these resource sites by customers, including the trustworthiness of computing results provided by these sites. Every IT organization faces the issue of managing the security of many different resources within its enterprise. Behavior security may also include service survivability and vulnerability.

– *Performance* includes QoS metrics related to the underlying network that interconnects the resource sites, such as throughput, link utilization, packet loss rate and end-to-end transfer delay. It also includes similar QoS metrics for the resource sites, such as response time and throughput. The response time is the time it takes for a service request to be satisfied, and the throughput is the service rate that a service provider can offer.

Customers will receive higher levels of QoS if they are willing to pay more. Therefore, having different SLAs with different associated costs is a common practice approach.

Cloud services are primarily supported through data centers and this book is mainly concerned with the end-to-end performance and security of cloud services. Thus, this book focuses on the study of end-to-end performance and security between cloud users and data centers, instead of the discussion of cloud virtualization technologies. Thus, the

approaches presented in this book can be applied to enterprise service computing including cloud computing.

Moreover, this book will focus on resource optimization whereby a service provider uses the least resources at each service station but it is still able to achieve the predefined SLA. These resources may include servers, storage devices, routers, network links, processors and so on (e.g. see [SHI 06] and [XIA 99]). They are often virtual resources in cloud computing. This book also presents a study of performance of public-key cryptography-based group authentication.

1.2. The problems

In this book, we consider a collection of computer resources used by a service provider[1] to host enterprise applications for business customers. An enterprise application running in such a computing environment is associated with an SLA (see [LEE 02], [MAT 05] and [MIC]). That is, the service provider is required to execute service requests from a customer within negotiated QoS requirements for a given price. Figure 1.1 depicts a scenario for such an environment. A customer represents a business that generates service requests at a given rate to be processed by the service provider's resource stations according to QoS requirements and for a given fee. As shown in Figure 1.1, a service request is transmitted to the service provider over a network provider.

After it is processed at the various resource stations of the service provider, the final result is sent back to the customer. For presentation purposes, we assume that each resource station has only one type of server associated with cost c_j. If they have multiple types of servers, we can divide each

1 The computer resources may or may not be owned by the service provider. The service provider may only act as a service broker.

resource station into several individual stations so that each one only contains one type of server with the same cost.

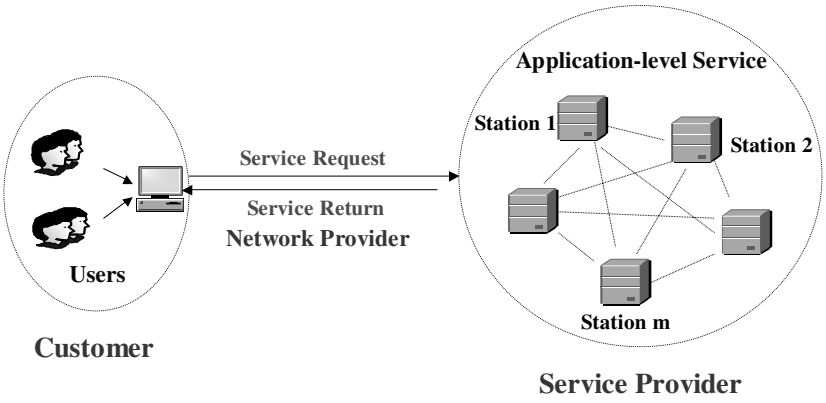


Figure 1.1. *The execution of service requests*

Let N_j be the number of servers at station j ($j = 1, 2, \cdots, m$). Thus, the resource allocation is quantified by solving for n_j ($n_j = 1, 2, \cdots, N_j$) in the following optimization problem:

$$I = \min_{n_1,\cdots,n_m} (n_1 c_1 + \cdots + n_m c_m) \qquad [1.1]$$

subject to SLA constraints. Performance and price are the two most important components for a variety of SLAs for business applications. Hence, in this book, we first consider the minimization of the overall cost of the service provider's computing resources allocated to a multiclass business customer so that $\gamma\%$ of the time, the response time, i.e. the time to execute a service request, is less than a predefined value.

In enterprise computing, customer requests often need to be distinguished, with different request characteristics and customer's different service requirements. In this book, we

further consider a set of computer resources used by a service provider to host enterprise applications for differentiated customer services subject to an SLA.

Imposing a priority structure with preemption-resume is one way to implement a service for satisfying multiple-class customer requests. A priority discipline does not depend on the state of a queue at the arrival of a customer, but is determined by a classification of arriving customers according to some criterion that is independent of the state of the queue. Suppose arriving customers to a single queue are classified into R different classes, where type r_1 customers ($r_1 = 1, 2, \cdots, R-1$) always have priority for service over those of type r_2 ($r_2 > r_1$; $r_1, r_2 = 1, 2, \cdots, R$), while customers of the same type are served in order of arrival (or First-In-First-Out (FIFO)). Then, it is said that the single queue operates according to a priority discipline with R classes.

In this book, we consider a *preemptive-resume* priority discipline, that is the service of a class r_2 customer can be interrupted if a higher priority customer of class r_1 ($r_2 > r_1$) arrives during his/her service. The interrupted customer resumes his/her service from where it stopped after the higher priority customer, and any other customer with priority higher than r_2 that may arrive during his/her service, complete their services. There are many other situations of practical interest (in the fields of modern computer systems, communication networks, computer server maintenance and computer security checking, for example) in which the order of servicing is determined by preemptive-resume.

Second, we present an approach for the resource optimization that minimizes the total cost of computer resources required while preserving a given percentile of the response time for priority-class customers. We calculate the

number of servers in each resource station that minimize a cost function that reflects operational costs.

Web services technology was introduced as a major component of .NET technology by Microsoft in June 2000, and it is widely considered as an emerging paradigm for the next generation of Internet computing [ZHA 05]. Web services technology has become the most popular computing paradigm for e-business applications and distributed environments. Many companies such as Google and Amazon are boosting their traffic using Web services Application Programming Interface (API) [MEN 04a]. By adopting service-oriented architectures (SOAs), services components from several universal service providers can be flexibly integrated into a composite service regardless of their location, platform and execution speed [BAR 03], [SIN 03]. In this type of enterprise's service applications, service resources may be located in different places. Thus, security becomes a big concern. Hence, our study further addresses how to choose reliable resource stations from a customer's perspective.

Usually, at the beginning of a service, there is no pre-existing relationship between a customer and a service provider. In this research, "trust" is used to establish the relationship between a customer and a service provider. It is a firm belief in the competence of a resource station that acts as expected. The trustworthiness of resource stations is an indicator of the QoS provided by these stations based on previous and current job completion experience. It often predicates the future behavior of the QoS at these stations. Moreover, we only consider the trustworthiness of resource sites from a customer's perspective. Hence, in Chapter 5, we simply assume that these resource stations trust each other.

In Chapter 5, we further consider a trust-based resource allocation problem that typically arises in the aforementioned Web services applications. We formulate the trust-based

resource allocation problem as an optimization problem under the SLA constraints: trustworthiness, performance and availability.

As discussed above, a trust model is suggested to establish the relationship between a customer and a service provider. However, receiving a reliable service response heavily depends on the security of the chosen service stations. When these chosen service stations are not secure, they cannot provide reliable services to the customer. Authentication is the process of reliably verifying the identity of someone or something. It is an essential part in enterprise service computing. Authentication must be done before or with data communication.

Kerberos [KOH 93] consists of a client, an application server and a key distribution center (KDC). It is a mature, reliable, secure network authentication protocol that allows a client to prove its identity to a server without sending confidential data across the network. Public-key cryptography has been extended to support Kerberos, since it simplifies the distribution of keys in Kerberos. It eliminates a single point of failure. Integrating public-key cryptography into Kerberos represents the enhancements of the current Kerberos standard. Several Kerberos-based authentication techniques using public-key cryptography have been proposed in the last decade. Among them include Public-Key Cross Realm Authentication in Kerberos (PKCROSS) [HUR 01] and Public-key Cryptography for Initial Authentication in Kerberos (PKINIT) [ZHU 06]. Moreover, the scalability of network security infrastructures is becoming a serious concern as the explosive growth of collaborative applications such as Web services continues unabated. Public-key-based Kerberos for Distribution Authentication (PKDA) [SIR 97] and Public Key Utilizing Tickets for Application Servers (PKTAPP, also known as

(a.k.a.) KX.509/KCA) [KX 07] and [MED 97] have been proposed to enhance the security and scalability of Kerberos. But, the actual costs (e.g. computational and communication times) associated with these techniques have been poorly understood so far. It remains unknown which technique performs better in a large network where there are multiple KDC remote realms. Both PKCROSS and PKTAPP use variations of PKINIT message types and data structures for integrating public-key cryptography with Kerberos in different authentication stages. PKTAPP was originally introduced as PKDA. It implemented PKDA using the message formats and exchanges of PKINIT. Hence, this book only considers these two notable techniques: PKCROSS and PKTAPP.

In this book, we first present a thorough performance evaluation of PKCROSS and PKTAPP in terms of computational and communication times. Then, we demonstrate their performance difference using open queuing networks. An in-depth analysis of these two techniques shows us that PKTAPP does not perform better than PKCROSS. Thus, we propose a new public-key cryptography-based group authentication technique. Our performance analysis demonstrates that the new technique can achieve better scalability as compared to PKCROSS and PKTAPP.

1.3. Summary of contributions

The contributions of this research are summarized as follows:

– *End-to-end response time*: the calculation of the response time often becomes critical in solving the resource optimization problem. Existing work addressing resource allocation uses the average response time (or average execution time). Though the average response time is relatively easy to calculate, it does not address the concerns of

a customer. Typically, a customer is more inclined to request a statistical bound on its response time than an average response time. To obtain the statistical bound on its response time, we need to calculate the probability density function of the end-to-end response time, which is not an easy task in a complex computing environment involving many computing nodes. We develop effective and accurate numerical solutions of this probability density function in both a tandem queuing network and a queuing network with feedback, which we then incorporate in this resource allocation problem.

– *Multiclass queuing networks*: in enterprise computing, customer requests often need to be distinguished, with different request characteristics and service requirements. Imposing a priority structure with preemption-resume is one way to implement a service for satisfying multiple-class customer requests. However, it is difficult to calculate the probability density function of an end-to-end response time even for a single node in the case of multiple priority customers. In our study, we first develop an efficient and accurate numerical solution for inverting the Laplace–Stieltjes transforms (LSTs) of a multiclass priority customer's response time numerically at a single node, and then extend the result to multiclass queuing networks under our study. A contributing factor is that typically we are interested in values of the cumulative distribution of the response time that correspond to very high percentiles for which our approximate results seem to have a very good accuracy.

– *Trust-based resource optimization in Web services*: most existing Web services products do not support an SLA that guarantees a level of service delivered to a customer for a given price. It is not easy to solve resource allocation problem when we consider all the constraints of trustworthiness, an end-to-end response time and service availability. We provide an efficient approach to solve the trust-based resource optimization problem in Web services whereby we are

required to minimize the total cost of service providers under the constraints of trustworthiness, an end-to-end response time and service availability.

– *Performance analysis of public key cryptography-based group authentication*: several authentication techniques have been proposed in the last decade. But, the actual costs (e.g. computational and communication times) associated with these techniques have been poorly understood so far. It remains unknown which technique performs better in a large network where there are multiple KDC remote realms. Based on complexity analysis and queuing theory, the book proposes a performance methodology that is an effective way to analyze the performance of security protocols and suggests a new group authentication that improves the scalability of PKCROSS and PKTAPP. It should be pointed out that the proposed performance method can be extended to analyze other security techniques as well.

1.4. The organization of this book

This book is organized as follows. Related work is reviewed in Chapter 2. We give the solution of the resource optimization problem with numerical validations for single-class customers in Chapter 3 and for multiple-class customers in Chapter 4. Chapter 5 addresses a trust-based resource allocation problem, and provides an approach to solve this problem for single- and multiple-class customers. Numerical examples are given, which demonstrates the validity of this approach for both cases. In Chapter 6, we give an in-depth performance evaluation of authentication techniques: PKCROSS and PKTAPP (also known as KX.509/KCA), by using complexity analysis and queuing networks, and propose a new group authentication technique. We conclude our results and discuss future work in Chapter 7.

Chapter 2

Current Approaches for Resource Optimization and Security

In this chapter, we discuss existing approaches for addressing service-level agreement (SLA) metrics including service availability, trustworthiness, performance and public-key cryptography-based group authentication. Then, we provide a review of how a resource optimization problem subject to an SLA is solved in the literature.

The chapter is organized as follows. Section 2.1 gives the definition of service availability and describes its measure methods. The characteristics of trustworthiness and performance metrics are presented in sections 2.2 and 2.3. Section 2.4 gives a literature review of the resource optimization problem subject to various Quality-of-Service (QoS) metrics. The performance of public-key cryptography-based group authentication is discussed in section 2.5.

2.1. Service availability

Availability is a critical metric in today's computer design [HEN 99]. It is the percentage of time that a service provider can offer services. A computer system can be unavailable due to a variety of causes, such as network failure, hardware failure, software failure or security attacks. Detecting and preventing these failures and attacks is beyond the scope of our study in this book. Cisco has asserted that the operational failure causes 80% of non-availability [CIS 13c]. Hence, increasing network availability is becoming a key priority for enterprise and service provider organization, as discussed in [CIS 13a]. Martin and Nilsson [MAR 02] give an example of how network service availability is defined in Sprint's SLA and WorldCom's SLA.

Availability has been extensively studied for a variety of computer systems in the literature. It has been studied to improve dependability for computer and telephone networks in [GRA 01]. The dependability can be defined as the property of a system such that reliance can justifiably be placed on the service it delivers, as defined in [BAR 95]. Systems with high availability tend to have large quorum[1] sizes and high load. Improving the availability of a system has been discussed in the literature (e.g. see [AIY 05], [AMR 85], [NAO 98] and [SKE 84]). Aiyer *et al*. [AIY 05] studied the availability of non-strict quorum systems and proposed K-quorums that can provide higher availability than the strict quorum systems. Naor and Wool [NAO 98] analyzed the load and the availability of traditional quorum systems.

Brown and Patterson [BRO 00] defined a new availability metric to capture the variations of the system QoS over time.

1 A quorum system is a collection of sets called quorums such that any two quorums have a non-empty interaction.

It is defined by the number of requests satisfied per second (or the latency of a request service) and the number of server failures that can be tolerated by a system.

In this research, our interest is potential hardware (i.e. servers within each station) failures and their effect on unavailability. To determine this, the service provider that owns these stations needs to understand the mean time to failure ($MTTF$) of all station components and the mean time to recover ($MTTR$) for hardware problems for all devices at each station, where $MTTF$ is the average time of a server failure, and $MTTR$ is the average time for recovering a server at each resource station. $MTTF$ information can be obtained from a hardware provider. For example, it is mentioned in [CIS 13c] that MTTF information is available for all Cisco components and is available upon request to a local account manager. $MTTR$ is determined by evaluating how quickly a station owner can repair broken servers. It is a major factor of server availability. To improve service availability, it is necessary to reduce the frequency time of failure, as indicated in Brewer [BRE 01].

Network availability data may be found on the Internet. For example, the University of Houston maintains current and historical network availability data on a Website [UH 12]. Cisco [CIS 13b] presented a formula for the calculation of network availability.

In this book, we consider the percentage of time that a resource is “up” or “down” as a metric, which is the traditional way to define service availability. Then, a two-state Markov chain with the states “up” and “down” is proposed to study the service availability at each service station. The detailed analysis and calculation of the service availability is given in section 5.3.3.

2.2. Trustworthiness

"Trust" is used to deal with the notion of trustworthiness. In this book, trust is defined as a firm belief in the competence of a resource station to act as expected. Many researchers have studied the trustworthiness of service entities, and suggested several different trust metrics. Zhang *et al*. [ZHA 04], and Ziegler and Lausen [ZIE 02] classified various trust metrics. Vu *et al*. [VU 05] proposed a new QoS-based semantic Web services selection and ranking solution using a trust and reputation management and assuming known QoS qualities.

The trustworthiness of resource stations is an indicator of the QoS provided by these stations based on previous and current job completion experience. It often predicates the future behavior of the QoS at these stations. Most trust models (e.g. [KAM 03], [LEE 03] and [RIC 03]) assume that the domain of trustworthiness ranges from 0 to 1, which is considered in this research. In this book, we use a rank- and a threshold-based approach. That is, according to the trust information of each station maintained by the trust manager, our study addresses how the trust manager selects service stations using a rank- and threshold-based approach. The rank-based approach is to rank the trustworthiness of all sites that provide the same type of services, such as EigenRep [KAM 03]. A threshold-based approach is to choose any of those service sites that can meet trust requirements predefined by customers. For example, if the trustworthiness of a service site is more than 0.9 and the predefined trust requirement of a customer is 0.9, then the trust manager can select the site to serve the customer's service request. The relationship between the rank- and the threshold-based approaches is discussed in [ZHA 04].

Our study introduces a trust manager who represents customers. The trust manager uses the collected trustworthy

information of the resource stations to evaluate their security behavior. We consider security behavior by modeling the behavior trusts of all sites, and quantify the trustworthiness of these stations. This approach is based on previous job completion experience assessed by the trust manager and customers. Multiple trust managers often exist in today's complex computer systems. The assessment also adopts the opinion of those trust managers besides its own customer's feedback. The domain of feedback is also assumed to be [0, 1].

The feedback is a statement issued by the trust manager's customers about the QoS provided by those chosen service stations for each service request or for a set of service requests during a certain period of time. These customers only provide feedback about their received services rather than those chosen service sites. (Note that the trust manager's customer is usually not aware of which service sites process his/her service request.) The opinion is defined as the information of trustworthiness provided by those trust managers who are neighbors of the trust manager. These neighbors are a small subset of the trust manager's acquaintances, adaptively selected based on their usefulness. The opinion aggregates the overall impression of those neighborhood trust managers for the service stations. A similar definition is given in [GOL 04], [KAM 03], [LEE 03], [MUI 02] and [RIC 03].

In this research, we consider all these factors by combining the feedback of the trust manager's customers and the opinions of neighborhood trust managers with the past trust information of the trust manager. The detailed discussion of updating the trust information for each station will be discussed in section 5.2.

2.3. Performance

The SLA performance metric defined in section 1.1 includes *throughput* and *response time*. As an end user, a customer is in general concerned about response time rather than throughput. So, in this book, we only consider the percentile of the response time as the performance metric. This is the time it takes for a service request to be executed on the service provider's multiple resource stations.

To obtain the statistical bound on its response time, i.e. the percentile of the response time, we are required to calculate the probability density function of the response time, whose detailed discussion will be given in section 3.1.

The calculation of the response time often becomes critical in solving the resource optimization problem. The computation of the response time has been extensively studied for a variety of computing systems. However, only the average response time is calculated rather than a percentile of the response time. Menasce and Bennani [MEN 03] designed self-managing systems to control QoS. Levy *et al*. [LEV 03] presented a performance management system for cluster-based Web services. In both studies, the average response time is used as a metric. Chandra [CHA 03] employed an online measurement method, and considered a resource allocation problem based on measured response times.

Martin and Nilsson [MAR 02] measured the average response time of a service request over IP networks. In [OSO 03], Osogami and Wierman analyzed an $M/GI/k$ system with two priority classes and a general phase-type distribution, and evaluated the optimal number of servers based on overall *mean* response time. A framework for service management in grid computing was defined in [MEN 04b],

but they did not provide a method for calculating the probability distribution of the response time.

To compute a percentile of the response time, we have to first find the probability distribution function (pdf) of the response time. This is not an easy task in a complex computing environment involving many computing nodes. The calculation of the pdf of the response time is relatively simple for a tandem network in [REI 57] and [REI 63] and overtake-free paths in Jackson and Gordon–Newell networks [WAL 80], [DAD 84]. Reich [REI 57] proved that the response times of a customer in each of two $M/M/1$ queues of a tandem network are independent, and he later extended the result to an arbitrary number of such queues in a tandem network [REI 63]. However, Reich [REI 57] proved that the result cannot be extended to a queuing network with $E_2/E_2/1$ queues where E_2 is a two-stage Erlangian distributions. Burke [BUR 72] further showed that in a tandem network with exponential server queues and customers arriving in a Poisson stream, whose first and last nodes are multiserver queues, while all other nodes are single-server queues, the response times of nodes are independent. Moreover, Walrand and Varaiya [WAL 80] generalized this result and proved that in any single-server open Jackson network, the response times of a customer at the various nodes of an overtake-free path are all mutually independent where the service discipline is first come first served (FCFS). They also proved that when a three-node tandem network has two parallel paths that are not overtaken-free as shown in Figure 2.1, the response time of nodes 1 and 2 and of nodes 2 and 3 is independent, but the response time of nodes 1 and 3 is not independent. Daduna [DAD 84] further showed that the same result is valid for overtake-free paths in Gordon–Newell networks with multiple-class customers. In addition, the pdf of the response time was derived for a closed queuing

network in [MUP 94], and the passing time distribution was calculated for large Markov chains in [HAR 02].

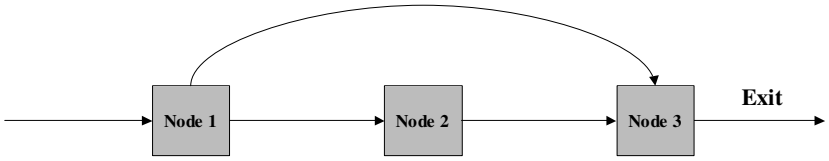


Figure 2.1. *A three-node tandem network with overtaking*

We develop effective and accurate numerical solutions of the probability density function of the response time for a variety of service models under our study, which we then incorporate in this resource allocation problem in Chapters 3 – 5.

2.4. The resource optimization problem subject to an SLA

Resource optimization problems subject to performance metrics such as response time, throughput, packet loss rate and link utilization have been extensively studied [AIB 04], [BOU 02], [BOL 93], [FLO 00], [MAR 02], [MEN 04b], [NOW 02], [XIO 09]. Link utilization is defined as the percentage of the time that the network node is utilized. Packet loss rate is the probability that the packet is lost during transmission through a network since its buffer is full.

The resource optimization problems subject to some of these performance metrics (or QoS metrics) were studied for multiple packet networks in [BOU 02], different classes of flows at network nodes in [CHA 02], IP networks in [MAR 02], wireless networks in [AIB 04], grid computing in [MEN 04b] and optical networks in [NOW 02]. Xiong and Perros [XIO 09] studied network optimization subject to the

percentile of response time, network availability, network utilization and packet loss rate.

These QoS metrics can be estimated by using measurement techniques (see [AIK 03], [ALL 03], [GUM 02] and [SOM 05]). In [CHA 02], Chassot *et al.* dealt with a communication architecture with guaranteed end-to-end QoS in an IPv6 environment. The end-to-end QoS includes an end-to-end delay (i.e. a response time). They only discussed and measured the maximal, minimal and average values of the response time. In [NOW 02], Nowak *et al*. developed a C++ event driven simulator to measure these three values in optical networks. But, measurement techniques are difficult to incorporate into solving a resource optimization problem with multiple constraints.

Calabrese [CAL 92] studied optimal workload allocation at each station in open networks of multiserver queues and formulated it as two nonlinear programming problems. One of their approaches is to maximize average throughput, subject to average number of jobs in the queues. The second approach is to minimize the average number of jobs in the queues, given a fixed average throughput. In [SHA 88], Shanthikumar and Yao discussed a server allocation problem in a single-class, closed queuing network. The problem was formulated as an optimization problem in which the average throughput is maximized.

Web services are often contracted through SLAs, which typically specify a certain QoS in return for a certain price. Although QoS was not defined in the initial Universal Description, Discovery and Integration (UDDI) standard for Web services, many studies have been carried out to extend the initial UDDI, such as Web Service Level Agreement (WSLA) [KEL 03], Web Service Offerings Language (WSOL) [TOS 02] and Quality of service Modeling Language (QML) [DOB 02]. The issue of a reputation-based SLA has been

studied by Jurca and Faltings [JUR 05] in which the cost is determined by the QoS that was actually delivered.

Resource allocation problems in distributed systems have been widely studied based on one of the metrics: trustworthiness and response time (e.g. see [NAB 04]). In Chapter 5, we will consider a resource allocation problem in Web services subject to all three QoS metrics, expressed by trustworthiness, percentile response time and service availability.

When we consider all QoSs consisting of trustworthiness, percentile response time and service availability, it becomes extremely difficult to solve a trust-based resource allocation problem whereby a set of computer resources is used by a service provider to host a typical distributed computing application for single-class and multiple-class customer services, respectively. To the best of our knowledge, our research is the first attempt toward solving the complex problem. We will formulate the trust-based resource allocation problem as an optimization problem under all these SLA constraints in which we calculated the number of servers in each service station that minimize a cost function that reflects operational costs for single-class and multiple-class customer services, respectively, in Chapter 5.

2.5. Public-key cryptography-based authentication

Kerberos has been revived over the past 15 years and rapidly since 1999. There have been numerous proposals to integrate public-key cryptography into Kerberos [DAD 84, HUR 01, MED 97, NEU 96, ZHU 06, SIR 97]. These proposals address various concerns of Kerberos in distributed networks, for instance security, scalability and portability. Neuman *et al*. [NEU 96] proposed PKINIT to enable the use of public-key cryptography for an initial authentication

between the client and its local key distribution center (KDC). PKINIT is an extension of the Kerberos protocol (RFC1510 [KOH 93]); its mechanism has not been standardized yet and the specification is still under development directed by the IETF Kerberos WG (see [ZHU 06]). Due to the cost of using public-key cryptography, Davis [DAV 96] suggested that public-key cryptography is best suited to securing communications between servers, between sites and between organizations.

PKCROSS [HUR 01] has been proposed to simplify the administrative burden of maintaining cross-realm keys so that it improves the scalability of Kerberos in large multirealm networks. Public-key cryptography takes place only in KDC-to-KDC authentication in PKCROSS. PKINIT and PKCROSS are centralized KDC protocols. Sirbu and Chuang [SIR 97] extended these two protocols to create PKDA for improving scalability and addressing the single point of failure on the KDC. PKTAPP is only a slight variation on the PKDA specification. Both PKDA and PKTAPP use lengthy public-key message exchanges between the client and the application servers, so they may not be any more efficient than public-key enabled authentication with a KDC and faster secret-key cryptography for subsequent encryption with application servers [HAR 01]. PKTAPP is also known as KX.509/KCA [DOS 01] and Windows has its own protocol that is the equivalent to KX.509/KCA (see [ALT 07a]). In spite of the protocol, the structure of PKTAPP has not been dramatically changed. We believe that PKCROSS and PKTAPP will be revived soon. PKCROSS was listed as Project 10 in the proposal given by the consortium. [CON 07].

Performance evaluation is a fundamental consideration in the design of security protocols. However, performance associated with most of these protocols has been poorly

understood. To analyze the performance of a protocol, we can first implement the protocol and then analyze measurement data taken from the implementation. But, the implementation of a protocol is very time-consuming and constricted to development resources and funding. While the implementation of KX.509 has been released [KX 07], PKCROSS has still not been implemented due to a matter of lack of development resources and funding [ALT 07b]. Hence, performance modeling has become an attractive approach since it can quickly provide a performance guidance used for a protocol design. Existing studies in [HAR 01] employed the performance modeling approach for examining the impact of PKCROSS and PKTAPP on network throughput based on their skeleton implementations and construction of *closed* queuing networks for a relatively simple case: there is only a single remote realm. In the present research, we also employ a performance modeling approach for the study of PKCROSS and PKTAPP but extend [HAR 01] in several essential and important aspects. First, complexity analysis has often been used to evaluate algorithm performance (e.g. [BLA 96]). While the performance of PKCROSS and PKTAPP was studied in [HAR 01], it remains unknown which technique performs better in multiple remote realms that are typical in an increasingly large network these days. It is much more difficult to analyze the case of multiple remote realms due to the complexity of authentication message exchanges. The difficulty is in the complexity analysis of these protocols and the building and analysis of queuing network models, which reflects the workload of authentication requests for these protocols. Second, we explicitly derive the formulas for calculating the computational and communication times of these protocols so as to easily determine which technique is better. Third, using a closed queuing network in [HAR 01] assumes that there exist *constant* authentication requests in the queuing network. Although the assumption can simplify our performance analysis, it is not usually true in the real

world. Rather, we adopted an *open* queuing network where the client requests authentication at a given rate, i.e. the number of authentication requests in a computing system under study is *not constant*. Fourth, due to a performance trade-off between these two protocols according to our scalability analysis, we propose a new hybrid technique. Our analysis shows that the new technique has better scalability than these two protocols in most cases. We discuss the open queuing network, give the complexity analysis of PKCROSS and PKTAPP and present the new authentication protocol in Chapter 6.

Chapter 3

Single Class Customers

This chapter gives the definition of the percentage of response time with an illustrative example and discusses a resource optimization problem subject to the percentile of response time and a fee for service models with single-class customers. We calculate the number of servers in each resource station that minimize a cost function that reflects operational costs. First, we analyze an overtake-free open tandem queuing network and then extend our work to an open tandem queuing network with feedback. This chapter is mainly based on the results presented in [XIO 06d] and [XIO 06a].

The remain of the chapter is organized as follows. In section 3.1, we define the service-level agreement (SLA) performance metric considered in this chapter. Section 3.2 formulates the resource optimization problem. In section 3.3, we present two typical real-life models and propose an approach for solving the optimization problem. In section 3.4, numerical simulations demonstrate the applicability and validity of the proposed approach. Finally, the conclusions are given in section 3.7.

3.1. The percentile of response time

An SLA is a contract between a customer and a service provider that defines all aspects of the service that is to be provided. An SLA generally uses response time as one performance metric.

As discussed in section 2.3, in this chapter we are interested in the percentile of the response time. This is the time that a job takes to be executed in a computing environment consisting of multiple computing nodes.

Assume that $f_T(t)$ is the probability distribution function of a response time T. T^D is a desired target response time that a customer requests and agrees with his/her service provider based on a fee paid by the customer. The SLA performance metric that a $\gamma\%$ SLA service is guaranteed is as follows:

$$\int_0^{T^D} f_T(t)\, dt \geq \gamma\% \qquad [3.1]$$

That is, $\gamma\%$ of the time a customer will receive its service in less than T^D.

As an example, let us consider an $M/M/1$ queue with an arrival rate λ and a service rate μ. The service discipline is first-in first-out (FIFO). The steady-state probability of the system is

$$p_0 = 1 - \rho,$$

and

$$p_k = (1 - \rho)\rho^k, \qquad k > 0$$

where $\rho = \frac{\lambda}{\mu}$.

The response time T is exponentially distributed with the parameter $\mu(1-\rho)$, i.e. its probability distribution function is given by (see [BOL 98] and [PER 94])

$$f_T(t) = \mu(1-\rho)e^{-\mu(1-\rho)t}$$

Using the definition given in [3.1], we have that

$$\int_0^{T^D} f_T(t)\,dt = 1 - e^{-\mu(1-\rho)T^D} \geq \gamma\% \qquad [3.2]$$

or

$$\mu \geq \frac{-\ln(1-\gamma\%)}{T^D} + \lambda \qquad [3.3]$$

This means that in order to guarantee higher SLA service levels, μ increases when T^D decreases.

Similarly, for any given arrival rate λ and service rate μ, we can use [3.2] to find the percentile of γ.

For example, when $\lambda = 100$ and $T^D = 0.05$, Figure 3.1 gives the cumulative distribution of the response time. As shown in Figure 3.1, the percentile of response time will increase as the service rate increases. Table 3.1 gives the numerical values for the cumulative distribution of the response time. From the table, we see that this service rate has to be bigger than 150 in order that 90% of the response time is less than 0.05.

3.2. A resource optimization problem for service models with single-class customers

In this section, we will discuss the minimization of the overall cost of the service provider's computing resources allocated to the single class customer so that $\gamma\%$ of the time the response time, i.e. the time to execute a service request, is

less than a predefined value. Based on section 3.1, the resource optimization problem can be formulated as follows:

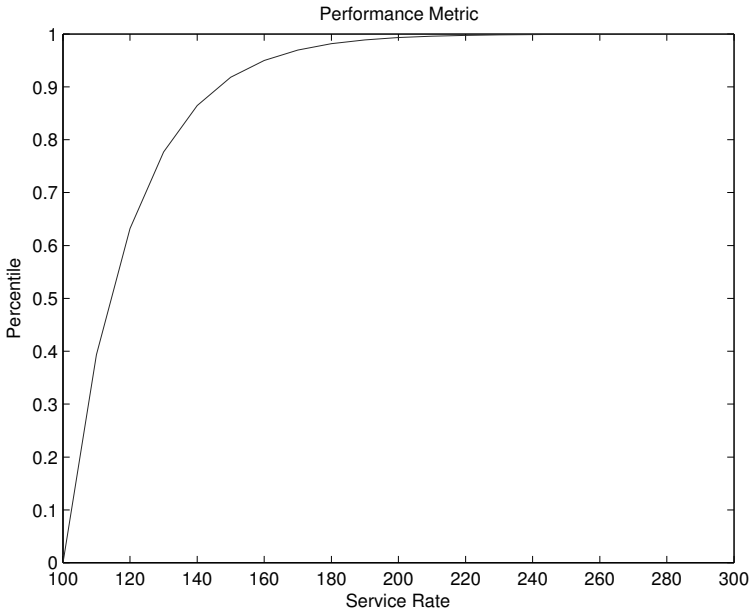


Figure 3.1. *Percentile response time versus service rate*

Service rate	100	120	140	150
CDF	0.0000	0.6321	0.8647	0.9179
Service rate	160	170	180	200
CDF	0.9502	0.9698	0.9817	0.9933
Service rate	220	240	280	300
CDF	0.9975	0.9991	0.9999	1.0000

Table 3.1. *The cumulative distribution function (CDF) of the response time versus service rate*

– *Resource optimization problem:* Find integers n_j ($1 \leq n_j \leq N_j$; $j = 1, 2, \cdots, m$) in the m-dimensional integer optimization problem [1.1] under the constraint of a percentile response time as expressed by [3.1], and the constraint: $I \leq C^D$, where C^D is a fee negotiated and agreed between a customer and a service provider.

3.3. Approaches for the resource optimization

In this section, we study two queuing network models that depict the path that service requests have to follow through the service provider's resource stations. These two models are shown in Figures 3.2 and 3.3. We refer to these two queuing models as service models since they depict the resources used to provide a service to a customer.

The first service model consists of a single infinite server and m stations numbered sequentially from 1 to m as shown in Figure 3.2. Each station j is modeled as a single FIFO queue served by n_j identical servers, each providing a service at the rate μ_j. Let Λ be the external arrival rate to the infinite server, and let λ and λ_j be the effective arrival rates to the infinite server and station j ($j = 1, 2, \cdots, m$). We assume that all service times are exponentially distributed and the external arrival to the infinite server occurs in a Poisson manner.

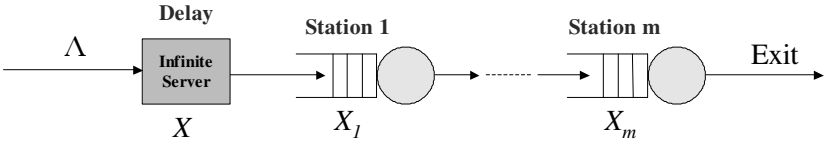


Figure 3.2. *A tandem-station service model for single-class customer services*

The infinite server represents the total propagation delay from the user to the service provider and back, and also from station 1 to m. Each station carries out a particular function. For instance, it could be a database server, a file server, a Web server, a group of CPUs and local disks, etc. In this chapter, we consider only a single-class customer.

In the following discussion, each station is modeled as a single $M/M/1$ queue with arrival rate λ_j and service rate $\psi(n_j)\mu_j$, where $\psi(n_j)$ is a known function of n_j and depends on the configuration of servers at each station. It is non-decreasing and can be inverted, i.e. ψ^{-1} exists. For instance, suppose that a station represents a group of CPUs. Then, $\psi(n)$ can be seen as a CPU scaling factor for the number of CPUs from 1 to n. According to [CHA 05], $\psi(n) = \xi^{\log_2 n}$, where ξ is a basic scaling factor from 1 CPU to 2. So, $\psi^{-1}(n) = \xi^{-\log_2 n}$.

Since the queuing network is overtake-free, the waiting time of a customer at a station is independent of its waiting times at other stations (see [DAD 84] and [WAL 80]). Let X be the service time at the infinite server and X_j be the time elapsed from the moment a customer arriving at station j to the moment it departs from the station. Then, the total response time is

$$T = X + X_1 + X_2 + \cdots + X_m,$$

and hence the Laplace-Stieltjes transform (LST) of the response time T is

$$L_T(s) = L_X(s)L_{X_1}(s)\cdots L_{X_m}(s) \qquad [3.4]$$

where $L_X(s)$ is the LST of the service time X given by

$$L_X(s) = \frac{\lambda}{s+\lambda} \qquad [3.5]$$

and $L_{X_j}(s)$ is the LST of the response time X_j at the jth station given by

$$L_{X_j}(s) = \frac{\psi(n_j)\mu_j(1-\rho_j)}{s+\psi(n_j)\mu_j(1-\rho_j)}, \qquad [3.6]$$

where $\rho_j = \frac{\lambda_j}{\psi(n_j)\mu_j}$ $(j = 1, 2, \cdots, m)$.

From [3.4]–[3.6], we have that

$$L_T(s) = \frac{\lambda}{s+\lambda}\,\Pi_{j=1}^{m}\frac{\psi(n_j)\mu_j(1-\rho_j)}{s+\psi(n_j)\mu_j(1-\rho_j)}$$

We observe that $f_T(t)$ and $F_T(t)$ are usually nonlinear functions of t and n_j. Hence, the resource optimization problem is an m-dimensional linear optimization problem subject to nonlinear constraints. In general, it is not easy to solve this problem. However, the complexity of the problem can be significantly reduced by requiring that the service rates of the queues in the service model in Figure 3.2 are all equal. That is, we find the optimum value of $n_1, \cdots, n_m$ such that

$$\psi(n_1)\mu_1 = \cdots = \psi(n_m)\mu_m$$

called *balanced utilization* or *balanced condition*. (We note that in production lines, it is commonly assumed that the service stations are balanced).

From the traffic equations:

$$\lambda = \lambda_j = \Lambda$$

for $j = 1, 2, \cdots, m$, we have that the utilization of each station

$$\rho_j = \frac{\lambda_j}{\psi(n_j)\mu_j} = \frac{\Lambda}{\psi(n_j)\mu_j}$$

Thus, we have

$$\hat{a}_i = \psi(n_i)\mu_i(1-\rho_i) = \psi(n_j)\mu_j(1-\rho_j) = \hat{a}_j \stackrel{\triangle}{=} \hat{a},$$

which implies $n_j = \psi^{-1}(\frac{\hat{a}+\lambda_j}{\mu_j})$ $(i, j = 1, 2, \cdots, m)$. Hence, from [3.4], we have

$$f_T(t) = L^{-1}\{\frac{\lambda}{s+\lambda} \cdot \frac{\hat{a}^m}{(s+\hat{a})^m}\},$$

and subsequently we obtain

$$F_T(t) = L^{-1}\{\frac{\lambda}{s(s+\lambda)} \cdot \frac{\hat{a}^m}{(s+\hat{a})^m}\} \qquad [3.7]$$

Consequently, $\sum_{j=1}^{m} n_j c_j$ reduces to a function of variable $\hat{a}$ due to $n_j = \lceil \psi^{-1}(\frac{\hat{a}+\lambda_j}{\mu_j}) \rceil$. Thus, we have the following algorithm for the resource optimization problem.

ALGORITHM 3.1.–

1) Find $\hat{a}$ in the one-dimensional optimization problem:

$$\hat{a}^{min} \leftarrow \arg\min_{\hat{a}} F_T(t)|_{t=T^D}$$

subject to the constraint $F_T(t)|_{t=T^D} \geq \gamma\%$ at $\hat{a} = \hat{a}^{min}$, where $F_T(t)$ is given by [3.7].

2) Compute integers n_j by using

$$n_j = \lceil \psi^{-1}(\frac{\hat{a}^{min}}{\mu_j(1-\rho_j)}) \rceil,$$

and check if $1 \leq n_j \leq N_j$ $(j = 1, 2, \cdots, m)$ and $I \leq C^D$ are satisfied. If yes, the obtained n_j is the number of servers required at each station. Otherwise, print "the problem cannot be solved".

In order to allow for a more complex execution path of a service request, we extended the above model to a service

model with feedback, as shown in Figure 3.3. Infinite server 1 represents the total propagation delay within a network provider and infinite server 2 represents the propagation delay within the service provider, i.e. among stations 1 to m. In this figure, a customer upon completion of its service at the mth station exits the system with probability α, or returns to the beginning of the system with probability $1 - \alpha$.

We note that the model shown in Figure 3.3 can be easily extended to a network of queues arbitrarily connected. We reuse the notation in the first model shown in Figure 3.2: Λ as the external arrival rate, λ_d, λ and λ_j as the effective arrival rates to the second infinite server and station j, and μ_j as the service rate at station j, where $j = 1, 2, \cdots, m$.

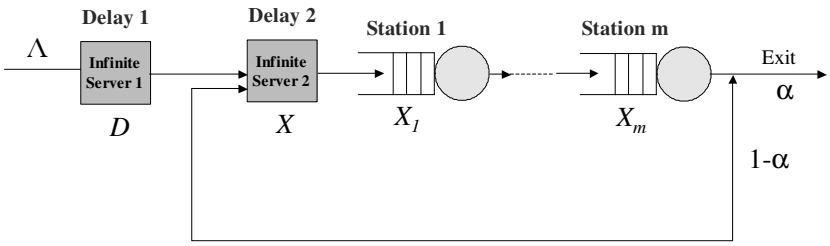


Figure 3.3. *A service model with feedback for single-class customer services*

The main difficulty of this resource optimization problem is to find $f_T(t)$, the probability distribution function of T. We obtain this probability distribution function assuming that the waiting time of a customer at a station is independent of its waiting time at other stations and each visit at the same station j is independent of the others. (We note that this assumption of independence does not hold in queuing networks with feedback. However, as will be discussed in

section 2.4, the solution obtained has a good accuracy). We first have the traffic equations:

$$\lambda_d = \Lambda, \quad \lambda = \Lambda + (1-\alpha)\lambda_m, \quad \text{and} \quad \lambda_j = \lambda,$$

which implies $\lambda_j = \lambda = \frac{\Lambda}{\alpha}$, and the utilization of each station is

$$\rho_j = \frac{\lambda_j}{\psi(n_j)\mu_j} = \frac{\Lambda}{\alpha\mu_j\psi(n_j)} \quad (j = 1, 2, \cdots, m)$$

Furthermore, the response time of the kth pass at the infinite station and the jth station is considered as the sum of $m+2$ random variables

$$T(k) = D + X + X_1 + \cdots + X_m,$$

where we assume that the waiting time of a customer at a station is independent of its waiting times in other visits to the same station. D and X are the service times at the first and second infinite servers, respectively, and X_j is the time elapsed from the moment a customer arrives at station j to the moment it departs from it. Then, the total response time is

$$T = \sum_{k=0}^{\infty} p(k)T(k),$$

where $p(k)$ is the steady-state probability that a request will circulate k times at the infinite station and the jth station through the computing system. $p(k)$ is determined by

$$p(k) = \alpha(1-\alpha)^{k-1}$$

Thus, the LST of the response time T is

$$L_T(s) = L_D(s) \sum_{k=0}^{\infty} p(k) L_X^k(s) L_{X_1}^k(s) \cdots L_{X_m}^k(s),$$

which can be rewritten as follows:

$$L_T(s) = \frac{\alpha L_D(s) L_X(s)\, \Pi_{j=1} L_{X_j}(s)}{1 - (1-\alpha)\, L_X(s) \Pi_{j=1}^{m} L_{X_j}(s)} \qquad [3.8]$$

where $L_D(s)$ is the LST of the service time D given by

$$L_D(s) = \frac{\Lambda}{s + \Lambda},$$

and replacing $L_X(s)$ and $L_{X_j}(s)$ $(j = 1, 2, \cdots, m)$ with [3.5] and [3.6] in [3.8], we have

$$L_T(s) = \frac{\Lambda^2 \Pi_{j=1}^{m} \hat{a}_j}{(s+\Lambda)[(s+\lambda)\Pi_{j=1}^{m}(s+\hat{a}_j) - (1-\alpha)\lambda \Pi_{j=1}^{m} \hat{a}_j]} \qquad [3.9]$$

To find the response time distribution $f_T(t)$, we need to invert the above LST using partial fraction decomposition of a rational function. However, the partial fraction decomposition of the rational function requires searching for roots of a high-order polynomial. It is usually not an easy task when the order of the polynomial is more than 5. Instead, in this chapter, the LST is inverted numerically.

Similarly, we want to find $n_1, \cdots, n_m$ such that the best utilization of these stations is achieved, which implies that each station has the same maximal service capacity. That is

$$\hat{a}_i = \hat{a}_j = \hat{a} \quad (i, j = 1, \cdots, m)$$

Then, from equation [3.9] and $F_T(t) = L^{-1}\{L_T(s)/s\}$, we have

$$F_T(t) = L^{-1}\{\frac{\Lambda^2 \hat{a}^m}{s(s+\Lambda)[(s+\lambda)(s+\hat{a})^m - (1-\alpha)\lambda \hat{a}^m]}\} \qquad [3.10]$$

Thus, we have the following algorithm for the resource optimization problem in the model shown in Figure 3.3.

ALGORITHM 3.2.–

Steps 1 and 2 are the same as steps 1 and 2 in algorithm 3.1 except $F_T(t)$ given by [3.10].

Note that if we cannot get a solution for the resource optimization problem using algorithm 3.1 (or 3.2), then the service provider cannot execute the service request for the service model 1 (or 3.2) due to at least one of the following reasons:

1) The service provider has an insufficient resource (i.e. N_j is too small).

2) A prespecific fee is too low (i.e. $I > C^D$).

3) A network connection is either too slow or has a problem so that [3.1] cannot be satisfied.

Using this information, we may detect and debug either a network problem or a service provider's capacity problem, or the SLA needs to be renegotiated.

In algorithms 3.1 and 3.2, the run-time for step 2 is $O(m)$. Thus, the run-time of either algorithm 3.1 or 3.2 is a sum of $O(m)$, the run-time for inverting the LST of the response time and the run-time for finding the maxima of the resulting function (or $F(t)$). While an efficient approach for finding the maxima of $F(t)$ can be found in [PRE 97], inverting the LST of the response time can be easily done by using the methods presented in [GRA 01].

3.4. Numerical validations

In this section, we demonstrate the accuracy and applicability of our proposed approximation method.

Two types of errors are introduced in our proposed approximation method. The first, hereafter referred to as

class I error, comes from numerically inverting the Laplace transform. The other, hereafter referred to as class II error, is due to the assumptions that the waiting time of a customer at each station is independent of the waiting times at the other stations, and it is also independent of its waiting times in other visits to the same station.

The relative error % is used to measure the accuracy of the approximate results compared to model simulation results and it is defined as follows:

$$\text{Relative error } \% = \frac{\text{Approximate ersult} - \text{simulation result}}{\text{Simulation result}} \times 100 \quad [3.11]$$

We study the accuracy of our proposed approximation method using two examples below.

First, we verify the accuracy of our approach for the first service model shown in Figure 3.2. Let $m = 8$, $\lambda = 100$, $N_j = 100$, $c_j = 2$, $\psi(n_j) = 1.5^{\log_2 n_j}$ and $C^D = 400$ ($j = 1, \cdots, 8$). The service rates of these eight stations are listed in Table 3.2.

Service rates	μ_1	μ_2	μ_3	μ_4	μ_5	μ_6	μ_7	μ_8
Values	52	18	80	35	41	15	25	35

Table 3.2. *The service rates of the eight stations in model 1*

We simulated the queuing network using Arena and the analytical method was implemented in Mathematica. The simulation results are considered as "exact" since the simulation model is an exact representation of the queuing network under study.

Table 3.3 shows the simulated and approximate cumulative distribution of the response time. In the table, the

column labeled "Simul" gives the simulation result, the column labeled "Approx" gives the approximate result and the column labeled "R-Err %" gives their relative errors. The same abbreviations are also used in Table 3.7. It seems that the results obtained by algorithm 3.1 are very accurate. The optimal number of servers required for 97.5% of the response time to be less than $T^D = 0.16$ is shown in Table 3.4. The exact optimal number of servers, obtained by exhaustive search using the simulation model, and assuming that each station has the same utilization, or balanced utilization, is consistent with the ones shown in Table 3.4. Thus, $I = 382 < C^D$. We point out that the relative errors shown in Table 3.3 are only due to the class I error since the class II error is not present in this service model.

Response time	Simul	Approx	R-Err %
0.04	0.0213	0.0214	0.4393
0.06	0.1517	0.1528	0.7004
0.08	0.4070	0.4075	0.1112
0.10	0.6681	0.6672	-0.1377
0.12	0.8468	0.8450	-0.2158
0.14	0.9398	0.9379	-0.1974
0.16	0.9785	0.9780	-0.0498
0.18	0.9931	0.9929	-0.0157
0.20	0.9979	0.9979	0.0000
0.22	0.9995	0.9994	-0.0077
0.24	0.9999	0.9998	-0.0051
0.26	1.0000	1.0000	0.0000

Table 3.3. *The cumulative distribution of the response time in model 1*

Let us now consider an example of the service model 2 shown in Figure 3.3. We choose $m = 8$, $\Lambda = 100$, $\alpha = 0.67$, $N_j = 250$, $c_j = 1$, $\psi(n_j) = 1.5^{\log_2 n_j}$, and $C^D = 580$ ($j = 1, \cdots, 8$). The service rates of these eight stations are listed in

Table 3.5. Thus, it follows from equation $\lambda = \frac{\Lambda}{\alpha}$ that $\lambda = 149.25$.

Station	1	2	3	4	5	6	7	8
#Servers	11	62	5	20	16	84	35	20

Table 3.4. *The optimal number of servers in model 1*

Service rates	μ_1	μ_2	μ_3	μ_4	μ_5	μ_6	μ_7	μ_8
Values	10	45	100	20	32	18	8	85

Table 3.5. *The service rates of the eight stations in model 2*

We obtained the cumulative distribution of the response time by solving [3.10] using the package of the inverse Laplace transforms given in [GRA 01]. Table 3.6 shows the number of servers in the eight stations necessary to ensure the 95% SLA guarantee for $T^D \leq 0.6$.

Station	1	2	3	4	5	6	7	8		
#Servers	168	13	4	52	23	62	246	5		

Table 3.6. *The optimal number of servers in model 2*

We also simulated the tandem queuing network and validated using the brute-force approach that these numbers of servers obtained by our approximate method are in fact optimal, provided that each station has balanced utilization. The optimal number of servers is given in Table 3.6. It derives that $I = 560 < C^D$, i.e. step 2) in algorithm 3.2 is met. Table 3.7 gives the cumulative distribution of the response time obtained using the approximate method and the simulation method, and the relative error $\%$. The relative error comes from both classes I and II errors. We note that our approximate method has a very good accuracy.

Response time	Simul	Approx	R-Err %
0.20	0.4865	0.4781	-1.7284
0.30	0.7418	0.7267	-2.0336
0.40	0.8551	0.8541	-0.1201
0.50	0.9189	0.9226	0.4067
0.60	0.9538	0.9589	0.5327
0.70	0.9733	0.9782	0.5000
0.80	0.9845	0.9884	0.3967
0.90	0.9908	0.9938	0.3070
1.00	0.9946	0.9967	0.2136
1.10	0.9967	0.9983	0.1563
1.20	0.9980	0.9991	0.1079
1.30	0.9988	0.9995	0.0714
1.40	0.9997	0.9997	0.0000
1.60	0.9999	0.9999	0.0026
1.80	0.9999	1.0000	0.0079
2.00	1.0000	1.0000	0.0000

Table 3.7. *The cumulative distribution of the response time in model 2*

In both examples as above, the run-time for the approximate method is less than 1 s when algorithm 3.1 or 3.2 was implemented in Mathematica, and the run-time for the simulation result is less than 1 min when Arena was used.

Extensive numerical results point to the fact that the independence assumption has little impact on the accuracy of the results when the number of nodes is large. A contributing factor is that typically we are interested in values of the cumulative distribution of the response time that correspond to very high percentiles for which the approximate results seem to have a very good accuracy through a comparison with simulation results that are considered as “exact.”

Additionally, to the best of our knowledge, this is the first work that provides an analytical solution of the resource optimization problem subject to the constraints of a percentile response time and a price. Hence, we do not give a comparison of our proposed method with other methods in this chapter.

3.5. The balanced condition

The aforementioned method to solve the resource optimization problem requires a balanced condition on all server stations. From a practical point of view, this condition is a fairly reasonable assumption in the resource optimization problem. This is because each server station may be owned by different entities, such as different organizations and different service providers, who have their individual objectives. These entities would price their services in such a way that they might maximize their profits. When a customer pays a certain fee for the service, these entities have to collaborate in allocating enough resources and determining the prices that they charge the customer. Otherwise, as studied in [HE 05], a self-centered competition for more revenues through either the insufficient allocation of resources or the high pricing may inflate the price charged to a customer and reduce the potential demand for the service. This means that these entities should collaborate with each other in allocating sufficient resources. Therefore, imposing the balanced condition on all server stations is one of the good ways in which these entities are evenly utilized.

In this section, we specifically study the balanced condition from a purely mathematical point of view. We want to investigate the change of the obtained solution, i.e. the number of servers in each station, when the condition is not imposed. This section restricts our discussion to the case of two stations in tandem without an infinite server in order to

reduce the amount of unnecessary formalization that would lead to tedious computations giving no new insights into the study. The tandem is shown in Figure 3.4. A similar extension to the case of m stations in tandem can be obtained for $m > 2$.

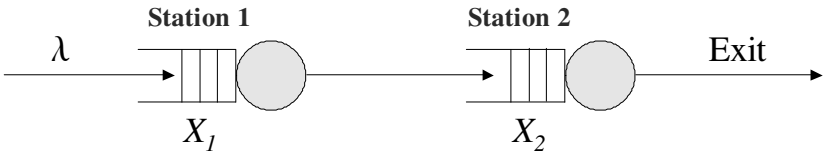


Figure 3.4. *A two-station tandem model*

The LST of the response time is computed by

$$L_T(s) = \Pi_{j=1}^{2} \frac{\hat{\mu}_j(1-\rho_j)}{s+\hat{\mu}_j(1-\rho_j)}$$

where $\hat{\mu}_j = \psi(n_j)\mu_j$ and $\rho_j = \frac{\lambda_j}{\hat{\mu}_j}$ for $j = 1, 2$. Thus, it derives the probability distribution function (PDF) of the response time given by

$$f_T(t) = \frac{\alpha_1\,\alpha_2}{\alpha_2 - \alpha_1}\left(e^{-\alpha_1 t} - e^{-\alpha_2 t}\right)$$

where $\alpha_j = \hat{\mu}_j(1-\rho_j) = \hat{\mu}_j - \lambda_j$ for $j = 1, 2$.

Furthermore, the percentile response time given in [3.1] can be rewritten as

$$G(T^D) = \int_{T^D}^{+\infty} f_T(t)\,dt = \frac{1}{\alpha_2 - \alpha_1}\left(\alpha_2\, e^{-\alpha_1 T^D} - \alpha_1\, e^{-\alpha_2 T^D}\right) \leq 1 - \gamma\%.$$

Again, in order to reduce the amount of unnecessary formalization, which would lead to tedious computations

giving no new insights into the study, we reformulate the resource optimization problem given in section 3.2 as

$$\min_{\hat{\mu}_1, \hat{\mu}_2 \geq 0} (\hat{c}_1 \hat{\mu}_1 + \hat{c}_2 \hat{\mu}_2) \qquad [3.12]$$

where $\hat{c}_j$ is the cost at a per unit of service rate in station j for $j = 1, 2$, under the constraint:

$$G(T^D) = \frac{1}{\alpha_2 - \alpha_1} \left(\alpha_2 \, e^{-\alpha_1 T^D} - \alpha_1 \, e^{-\alpha_2 T^D} \right) \leq 1 - \gamma\%$$

which is equivalent to the constraint:

$$G(T^D) = \frac{1}{\hat{\mu}_2 - \hat{\mu}_1} \left[(\hat{\mu}_2 - \lambda_2) \, e^{-(\hat{\mu}_1 - \lambda_1) T^D} - (\hat{\mu}_1 - \lambda_1) \, e^{-(\hat{\mu}_2 - \lambda_2) T^D} \right] \leq 1 - \gamma\% \qquad [3.13]$$

due to $\alpha_j = \hat{\mu}_j - \lambda_j$, where $\alpha_j \geq 0$. Note that by using the traffic equation in queuing theory we have $\lambda_1 = \lambda_2$.

Next, by imposing the balanced condition, i.e. $\alpha_1 = \alpha_2 \stackrel{\triangle}{=} \alpha$, we have

$$L_T(s) = \frac{\alpha^2}{(s + \alpha)^2}$$

Thus, the PDF of response time is given by

$$p(t) = \alpha^2 \, t e^{-\alpha t}$$

Since $\alpha_j = \hat{\mu}_j - \lambda_j$ and $\lambda_1 = \lambda_2 \stackrel{\triangle}{=} \lambda$, we obtain $\hat{\mu}_1 = \hat{\mu}_2 \stackrel{\triangle}{=} \hat{\mu}$. Subsequently, the objective function in the resource optimization problem is written as

$$c_1 \hat{\mu}_1 + c_2 \hat{\mu}_2 = (c_1 + c_2) \hat{\mu} \qquad [3.14]$$

and the percentile response time constraint can be computed by

$$H(T^D) = \int_{T^D}^{+\infty} p(t)\,dt = [(\hat{\mu} - \lambda) + 1]e^{-(\hat{\mu}-\lambda)T^D} \leq 1 - \gamma \qquad [3.15]$$

Note that $G(T^D)$ is a function of variables $\hat{\mu}_1$ and $\hat{\mu}_2$, and $H(T^D)$ is a function of variable $\hat{\mu}$. To simplify the notation, we reuse notations G and H, and write them as $G(\hat{\mu}_1, \hat{\mu}_2)$ and $H(\hat{\mu})$, i.e. $G(\hat{\mu}_1, \hat{\mu}_2) = G(T^D)$ and $H(\hat{\mu}) = H(T^D)$. Moreover, it is easy to see that

$$\frac{d\,(H(\hat{\mu}))}{d\,\hat{\mu}} = -(T^D)^2(\hat{\mu} - \lambda)\,e^{-(\hat{\mu}-\lambda)T^D} \leq 0 \qquad [3.16]$$

due to $\hat{\mu} \geq \lambda$, which implies that $H(\hat{\mu})$ is a decreasing function of variable $\hat{\mu}$. By considering [3.12]–[3.16], we can rewrite the resource optimization problem as follows.

When the balanced condition is not imposed, the resource optimization problem is to find $\hat{\mu}_1$ and $\hat{\mu}_2$ in the minimization problem:

$$\min_{\hat{\mu}_1, \hat{\mu}_2} (\hat{c}_1\hat{\mu}_1 + \hat{c}_2\hat{\mu}_2)$$

(Problem I) subject to

$$G(\hat{\mu}_1, \hat{\mu}_2) \leq 1 - \gamma\%, \text{ and } \hat{\mu}_j \geq \lambda_j, \quad j = 1, 2$$

where $G(\hat{\mu}_1, \hat{\mu}_2) = G(T^D)$ is given in [3.13].

When the balanced condition is imposed, the resource optimization problem is to find $\hat{\mu} = \hat{\mu}_1 = \hat{\mu}_2$ in the maximization problem:

$$\arg\max_{\hat{\mu}} H(\hat{\mu})$$

(Problem II) subject to

$$H(\hat{\mu}) \leq 1 - \gamma\%, \text{ and } \hat{\mu} \geq \lambda$$

where $H(\hat{\mu}) = H(T^D)$ is given in [3.15].

We are interested in investigating the difference between the solutions of problems I and II. Let us first study the property of these solutions. Assume that $\hat{\mu}_1 = \hat{\mu}_1^*$ and $\hat{\mu}_2 = \hat{\mu}_2^*$ be the solution of *problem I*, denoted by

$$(\hat{\mu}_1^*, \hat{\mu}_2^*) \leftarrow (\textbf{Problem I})$$

Clearly, the constraint function $G(\hat{\mu}_1, \hat{\mu}_2) - (1 - \gamma\%)$ is a symmetric function with respect to the line $\hat{\mu}_1 = \hat{\mu}_2$. Furthermore, if $\hat{c}_1 = \hat{c}_2$, then the objective function $\hat{c}_1 * \hat{\mu}_1 + \hat{c}_2 * \hat{\mu}_2$ is a symmetric function with respect to the line $\hat{\mu}_1 = \hat{\mu}_2$ as well, which implies that $\hat{\mu}_1 = \hat{\mu}_2^*$ and $\hat{\mu}_2 = \hat{\mu}_1^*$ is the solution of problem I as well. Hence, when there exists a unique solution of problem I, $\hat{\mu}_1^*$ should be equal to $\hat{\mu}_2^*$. This derives that problem I has the same solution as problem II. We summarize the conclusion below.

PROPOSITION 3.1.– Suppose that $\hat{c}_1 = \hat{c}_2$ and $(\hat{\mu}_1^*, \hat{\mu}_2^*)$ is a unique solution of problem I. Then, $\hat{\mu}_1^* = \hat{\mu}_2^*$, that is, problem I has the same solution as problem II.

This means that the balanced condition is satisfied when the two entities representing two resource stations charge the same rate at a per unit of service rate. Hence, the balanced condition is a fairly reasonable assumption also from the mathematical point of view.

Note that $H(\hat{\mu})$ is a continuous and non-increasing function. Hence, problem II is equivalent to the problem in finding a solution of $H(\hat{\mu}) = 1 - \gamma\%$. Thus, in this case, the solution of problem II can be easily obtained. In general, when the assumption of proposition 3.1 does not hold, problem II can be solved numerically.

Next, we present numerical implementations of these problems. Let us first assume that $\hat{c}_1 = \hat{c}_2$. We choose $\lambda = 100$

and $T^D = 3.5$ with varied γ. Table 3.8 gives the solutions of problems I and II.

Clearly, both optimization problems have the same solution, i.e. $\hat{\mu}_1 = \hat{\mu}_2 = \hat{\mu}$, which confirms our analysis as shown in proposition 3.1.

γ	80	85	90	95	97
$\hat{\mu}_1$ by problem I	100.856	100.964	101.111	101.355	101.530
$\hat{\mu}_2$ by problem I	100.856	100.964	101.111	101.355	101.530
$\hat{\mu}$ by problem II	100.856	100.964	101.111	101.355	101.530

Table 3.8. *The solutions of problems I and II with varied* γ

We further consider the case of $\hat{c}_1 \neq \hat{c}_2$. Let us choose $\lambda = 100$, $T^D = 3.5$ and $\gamma = 95$ with a fixed $\hat{c}_1 = 1$ and varied $\hat{c}_2$. We also choose $\mu_1 = 10$, $\mu_2 = 20$ and $\psi_j(n_j) = 1.5^{\log_2 n_j}$, where $j = 1, 2$. Table 3.9 shows the solutions of problems I and II.

$\hat{c}_2$	1	2	4	8	16
$\hat{\mu}_1$ by problem I	101.355	101.572	101.868	102.27	102.825
$\hat{\mu}_2$ by problem I	101.355	101.202	101.097	101.026	100.977
$\hat{\mu}_1 - \hat{\mu}_2$	0.000	0.370	0.771	1.244	1.848
$\hat{\mu}$ by problem II	101.355	101.355	101.355	101.355	101.355
$\hat{\mu} - \hat{\mu}_1$	0.000	-0.217	-0.513	-0.915	-1.470
$\hat{\mu} - \hat{\mu}_2$	0.000	0.153	0.258	0.329	0.378

Table 3.9. *The solutions of problems I and II with varied* $\hat{c}_2$

Note that the solution of problem II does not change with varied $\hat{c}_2$ because problem II is not related to $\hat{c}_2$. Table 3.9 demonstrates that the differences among $\hat{\mu}$, $\hat{\mu}_1$ and $\hat{\mu}_2$ are small. Furthermore, by using $\mu_1 = 10$, $\mu_2 = 20$ and $\psi_j(n_j) = 1.5^{\log_2 n_j}$ where $j = 1, 2$, we found the number of servers required to ensure that 95% of the service requests

from a customer can be received in less than $T^D = 3.5$, as shown in Table 3.10.

$\hat{c}_2$	1	2	4	8	16
n_1 by problem I	53	53	53	54	54
n_1 by problem II	53	53	53	53	53
n_2 by problem I	17	16	16	16	16
n_2 by problem II	17	17	17	17	17

Table 3.10. *A comparison of the number of servers by using problems I and II*

As is seen in Table 3.10, there is only, at most, one server difference in the number of servers obtained by problems I and II. Again, according to proposition 3.1 and the above numerical implementations, we see that the balanced condition does not introduce a significant inaccuracy. We have also done a lot of simulations with varied λ that are not reported here. These simulation results have shown that solving problem II can provide an accurate solution for problem I.

In addition, the balanced condition seems to be a fairly reasonable assumption from the practical point of view.

REMARK.– We can similarly show that proposition 3.1 holds in the case of m stations in tandem with or without feedback when the cost of each station at a per unit of service rate is the same.

3.6. Services Performance Modeling and Analysis in a Simple Scenario of Cloud Computing

As described before, the proposed approach can be extended to a general queuing network in which its nodes are

arbitrarily linked as long as they can be quantified. In the following section, we study a two-station queuing network that describes a simple scenario of cloud services. We demonstrate how to apply our proposed approach to cloud service performance analysis. This section is based on the results of Xiong and Perros [XIO 09b].

3.6.1. *Overview*

Cloud computing is an Internet-based computing paradigm that enabling ubiquitous, convenient, on-demand network access to a shared pool of computing resources. It has become an IT buzzword over the past few years. Cloud computing has been often used with synonymous terms such as "software as a service" (SaaS), "grid computing", "cluster computing", "autonomic computing" and "utility computing" [KIM 09]. SaaS is only a special form of services that cloud computing provides. Grid computing and cluster computing are two types of underlying computer technologies for the development of cloud computing. Autonomic computing means computing system services capable of *self-management*, and utility computing is the *packaging* of computing resources such as computational and storage devices [WIK 09] and [VAQ 09].

Loosely speaking, cloud computing is a style of computing paradigm in which typically real-time scalable *resources* such as files, data, programs, hardware and third-party services can be accessible from a Web browser via the Internet to users (or customers, alternatively). These customers pay only for the used computer resources and services by means of customized SLA, as well as having no knowledge of how a service provider uses an underlying computer technological infrastructure to support them. The SLA is a contract negotiated and agreed between a customer and a service provider. That is, the service provider is required to execute

service requests from a customer within negotiated Quality-of-Service (QoS) requirements for a given price. Thus, accurately predicting customer service performance based on system statistics and a customer's perceived quality allows a service provider to not only assure QoS but also avoid overprovisioning to meet an SLA. Due to a *variable* load derived from customer requests, dynamically provisioning computing resources to meet an SLA and allow for an optimum resource utilization will not be an easy task.

As stated in [WIK 09] and [TEC 09], the majority of current cloud computing infrastructures as of 2009 consist of services that are offered up and delivered through a service center, such as a data center, that can be accessed from a Web browser anywhere in the world. In this section, we study a computer service performance model for the cloud infrastructure as shown in Figure 3.5. This model consists of customers and a cloud architecture (or simply called a cloud) that has *service centers* such as data centers. (For presentation simplicity, we only consider one service center in this section). The cloud then, in this model, is *a single point of access* for the computing needs of the customers being serviced [TEC 09] through a Web browser supported by *a Web server*. The service center is a collection of service resources used by a service provider to host service applications for customers, as shown in Figure 3.5. A service request sent by a user is transmitted to the Web server and the service center that are owned by the service provider over network [MAR 02]. As discussed previously, a service application running in such a computing environment is associated with an SLA since a customer pays only for used resources and services.

The service load in cloud computing is dynamically changed upon end users' service requests. That is, the customer in the preceding discussion may represent multiple

users and generates service requests at a given rate to be processed at the service center hosted by the service provider through the cloud, according to QoS requirements and for a given fee. Clearly, a customer is, in general, concerned about response time rather than throughput in QoS requirements. So, we do not include throughput as a metric in this study. Other metrics may be defined in an SLA as well, but they are beyond the scope of our study in this section.

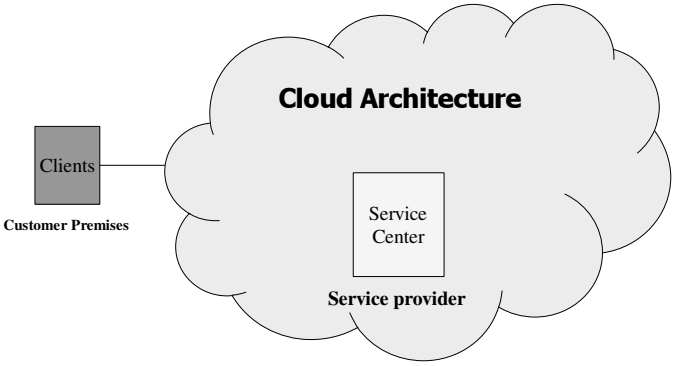


Figure 3.5. *A computer service scenario in cloud computing*

Existing works addressing QoS requirements in computer service performance usually use the average response time (or average execution time). Although the average response time is relatively easy to calculate, it does not address the concerns of a customer. Typically, a customer is more inclined to request a statistical bound on its response time than an average response time. For instance, a customer can request that 95% of the time, his/her response time should be less than a given value. Therefore, in this section, we consider a percentile of the response time that characterizes the statistical response time. That is, the time to execute a service request is less than a predefined value with a certain percentage of time. The metric has been used by researchers [MAT 03] at IBM; also it has been called a percentile delay

by scientists at Cisco [DAV 04] and MIT Communications Future Program [CON 07]. It has been defined as the p-percentile in the standards iIETF RFC5166 and RFC 2679 as well as MEF 10.1 [MEF 06]. Siripongwutikorn [SIR 06] has shown the difference of an average delay and percentile delay in per-flow network traffic analysis. By considering this percentile of response time metric, we study the relationship among the maximal number of customers, the minimal service resources and the highest level of services as discussed in the introduction. We will specifically restate the following *three important but challenging questions* for customer service performance in cloud computing:

1) For a given arrival rate of service requests and given service rates at the Web server and the service center, what level of QoS services can be guaranteed?

2) What are minimal service rates required at the Web server and the service center, respectively, so that a given percentile of the response time can be guaranteed for a given service arrival rate from customers?

3) What number of customers can be supported so that a given percentile of the response time can be still guaranteed when service rates are given at the Web server and the service center, respectively?

The problem of computer service performance modeling subject to QoS metrics, such as response time, throughput, network utilization, has been extensively studied in the literature (for example, see [KAR 04], [LU 05], [MEI 06], [MEI 06] and [SLO 95]). Readers are referred to Chapter 2 for the details of related work.

In order to compute a percentile of the response time, we first need to find the probability distribution of the response time. This is not an easy task in a complex computing environment involving many computing nodes. Walrand and

Varaiya [WAL 80] showed that in any open Jackson network, the response times of a customer at various nodes of overtake-free path are all mutually independent. Daduna [DAD 84] further proved that the same result is valid for overtake-free paths in Gordon–Newell networks. In this section, we derive an approximation method for the calculation of the probability and cumulative distributions of the response time, and show the accuracy of the proposed approximation method. Based on the obtained percentile response time (or the cumulative distribution of response time), we derive propositions and corollaries to answer the aforementioned service performance questions in cloud computing.

3.6.2. *A computer service performance model*

As discussed previously, the calculation of the percentile of response time plays a key role in answering the aforementioned performance questions. In this section, we derive the calculation.

3.6.2.1. *The response time distribution*

Modeling the customer service requests as a queuing network model is one of the best ways to make it possible to not only compute percentile response time but also characterize a *variable* load in cloud computing. The cloud computing service model shown in Figure 3.5 is modeled as a queuing network model as depicted in Figure 3.6, which consists of a Web server and a service center. Both the Web server and the service center may have several components. But, each can be viewed as an integral element that is modeled as a single queue. External arrivals to the computing station are distributed with a rate λ. Let μ_i ($i = 1, 2$) be the service rates at the first and second queues, respectively. Upon completion of a service at the service

center, the customer exits the system with probability $1 - \beta$ or continues to be served at the Web server with probability β. Furthermore, after being processed at the Web server, the customer returns to the beginning of the cloud computing system with probability $1 - \alpha$, or it exits the system with probability α.

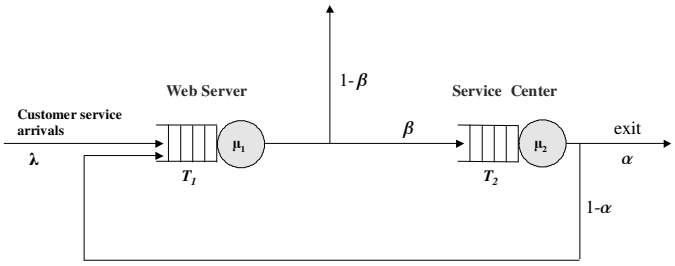


Figure 3.6. *A queuing performance model for computer services in cloud computing*

We are going to derive the Laplace–Stieltjes transform (LST–simply called the Laplace transform alternatively) of response time below. Let (i, j) be the number of visits in the Web server and the service center where i and j are the number of visits in the Web server and the number of visits in the service center, respectively. Let $p(i, j)$ be the probability of i visits to the Web server and j visits to the service center. There may be one time visit difference between the Web server and the service center. This means that either $j = i$, or $j = i - 1$. Let $T_1(\lambda, \mu_1, \mu_2)$ (or $T_2(\lambda, \mu_1, \mu_2)$) be the waiting time, that is, the time from the moment a customer arrives at the Web server (or the service center) to the moment it leaves the Web server (or the service center). For notational simplicity, $T_1(\lambda, \mu_1, \mu_2)$ and $T_2(\lambda, \mu_1, \mu_2)$ are written as T_1 and T_2. We further assume that T_1 and T_2 are the same for each visit. Then, we have

# Visits	Response time	Probability
$(1,0)$	T_1	$p(1,0) = 1-\beta$
$(1,1)$	$T_1 + T_2$	$p(1,1) = \beta\alpha$
$(2,1)$	$(T_1 + T_2) + T_1$	$p(2,1) = \beta(1-\alpha) \times(1-\beta)$
$(2,2)$	$2(T_1 + T_2)$	$p(2,2) = \beta^2(1-\alpha) \times\alpha$
...	...	...

Therefore, the average response time $E(T)$ is the weighted sum of individual response times

$$E[T] = \sum_{j=1}^{\infty} \beta^{j-1}(1-\alpha)^{j-1}(1-\beta) \times [(j-1)(T_1 + T_2) + T_1] + \sum_{j=1}^{\infty} \beta^{j}(1-\alpha)^{j-1}\alpha[j(T_1 + T_2)] \quad [3.17]$$

Now, let T_1^i and T_2^i be the response time at the Web server and the service center for the ith visit, respectively. Then, we have the following expression for $E(T)$

$$E[T] = \sum_{j=1}^{\infty} \beta^{j-1}(1-\alpha)^{j-1}(1-\beta)[\sum_{i=1}^{j-1}(T_1^i + T_2^i) + T_1^j] + \sum_{j=1}^{\infty} \beta^{j}(1-\alpha)^{j-1}\alpha[\sum_{i=1}^{j}(T_1^i + T_2^i)] \quad [3.18]$$

Let $R(j, j-1)$ be the response time for j visits to T_1 and $j-1$ visits to T_2, and $R(j, j)$ be the response time for j visits

to T_1 and T_2. Then

$$R(j,\, j-1) = \sum_{i=1}^{j-1}(T_1^i + T_2^i) + T_1^j$$

$$R(j,\, j) = \sum_{i=1}^{j}(T_1^i + T_2^i) \qquad [3.19]$$

Assume that T_1^i and T_1^k, T_2^i and T_2^k, and T_1^i and T_2^i are mutually independent ($i \neq k$). Under these assumptions, the conditional LSTs of the response times $R(j,\, j-1)$ and $R(j,\, j)$ can be expressed as follows (refer to Bolch *et al*. [BOL 98]).

$$L_{R(j,\,j-1)}(s) = (L_{T_1}(s))^j \times (L_{T_2}(s))^{j-1}$$

$$L_{R(j,\,j)}(s) = (L_{T_1}(s))^j \times (L_{T_2}(s))^j \qquad [3.20]$$

By combining equations [3.18] and [3.20], we obtain the LST of the total response time:

$$L_T(s) = \sum_{j=1}^{\infty} \beta^{j-1}(1-\alpha)^{j-1}(1-\beta)L_{T_1}^j(s)\, L_{T_2}^{j-1}(s)$$

$$+ \sum_{j=1}^{\infty} \beta^j(1-\alpha)^{j-1}\alpha L_{T_1}^j(s)\, L_{T_2}^j(s)$$

That is

$$L_T(s) = \frac{(1-\beta)L_{T_1}(s) + \alpha\beta L_{T_1}(s)L_{T_2}(s)}{1-\beta(1-\alpha)L_{T_1}(s)L_{T_2}(s)} \qquad [3.21]$$

We can further obtain the cumulative distribution of response time by inverting the Laplace transform of [3.21]:

$$F_T(t, \lambda, \mu_1, \mu_2) = L^{-1}\{L_T(s)/s\} \quad [3.22]$$

where L^{-1} is an inverse Laplace transform. Generally speaking, there is no closed-form solution for the inversion of the above Laplace transform. Hence, the inversion is usually done numerically. Thus, the answers to the first two questions in section 3.6.1 can be expressed by the following corresponding two propositions.

PROPOSITION 3.1.– For a given arrival rate λ, service rates μ_1 in the Web server and μ_2 in the service center, and a set of parameters α, β, and T^D, the level of QoS-guaranteed services (γ) will be no more than $100F_T(T^D, \lambda, \mu_1, \mu_2)$.

PROOF.– This is because of the percentile of response time is equal to $1 - F_T(T^D, \lambda, \mu_1, \mu_2)$. By a use of this, it is easy to derive this proposition.

PROPOSITION 3.2.– For a given arrival rate λ, a level of QoS services γ, and a set of parameters α, β, and T^D, service rate μ_1 in the Web server and service rate μ_2 in the service center are determined by solving for μ_1 and μ_2 in the optimization problem below:

$$\text{argmin}_{\mu_1 \in \mathcal{R}_1, \mu_2 \in \mathcal{R}_2} \left(F_T(T^D, \lambda, \mu_1, \mu_2) - 1\right)$$

subject to $F_T(T^D, \lambda, \mu_1, \mu_2) \geq \gamma\%$, where $\mathcal{R}_1$ and $\mathcal{R}_2$ are sets of all permission values for service rates μ_1 and μ_2, respectively. For example, $\mathcal{R}_1 = \mathcal{R}_2 = R^+$, which is a set of positive real numbers, i.e. a positive real line.

Furthermore, assume that there are n customers, each with an equivalent arrival rate of λ_0. Let $\lambda = n\lambda_0$. Then, an answer to the third question raised in section 3.6.1 can be expressed by the following proposition.

PROPOSITION 3.3.– For service rates μ_1 and μ_2, a level of QoS services γ, and a set of parameters α, β, and T^D, the maximal number of customers that can be supported without a violation of a predefined level of QoS services $\gamma\%$ is determined by solving for n in the integer optimization problem below:

$$\text{argmax}_{n\in\mathcal{I}} \left(F_T(T^D, n\lambda_0, \mu_1, \mu_2) - 1\right)$$

subject to $F_T(T^D, n\lambda_0, \mu_1, \mu_2) \geq \gamma\%$, where $\mathcal{I}^+$ is a set of all permission values for the number of customers n. For example, $\mathcal{I}^+$ is a set of all positive integers.

Similar to the proof of proposition 3.1, we can easily prove propositions 3.2 and 3.3.

We see that $F_T(T^D, \lambda, \mu_1, \mu_2)$ will play a key role in propositions 3.1–3.3 for answering the three questions of section 3.6.1. As stated previously, its expression can be numerically found.

Next, an interesting case is considered below in which the closed-form expression of $F_T(T^D, \lambda, \mu_1, \mu_2)$ is derived.

Assume that the Web server and the service center are each modeled as an $M/M/1$ queue. The service discipline is FIFO. External arrivals to the computing station are Poisson distributed and the service times at the Web server and the service center are exponentially distributed. Let λ_i $(i = 1, 2)$ be the arrival rates at the first and second $M/M/1$ queues, respectively. Then, traffic equations are given by $\lambda_1 = \lambda + (1 - \alpha)\lambda_2$ and $\lambda_2 = \beta\lambda_1$. Thus, the following expressions of λ_1 and λ_2 can be derived from these local balance equations: $\lambda_1 = \frac{\lambda}{1-(1-\alpha)\beta}$ and $\lambda_2 = \frac{\lambda\beta}{1-(1-\alpha)\beta}$.

Moreover, the LSTs of the response time at each queue are $L_{T_1}(s) = \frac{a_1}{s+a_1}$,where $a_1 = \mu_1(1 - \rho_1),\ \rho_1 = \frac{\lambda_1}{\mu_1}$, and $L_{T_2}(s) = \frac{a_2}{s+a_2}$,where $a_2 = \mu_2(1 - \rho_2),\ \rho_2 = \frac{\lambda_2}{\mu_2}$.

It follows from [3.21] that

$$L_T(s) = \frac{a_1(1-\beta)(s+a_2) + a_1 a_2 \alpha\beta}{(s+a_1)(s+a_2) - a_1 a_2 \beta(1-\alpha)} \quad [3.23]$$

whose de-numerator is a quadratic polynomial with respect to variable s that has the roots

$$s_{1,2} = \frac{-(a_1+a_2) \pm \sqrt{(a_1+a_2)^2 - 4a_1 a_2(1-\beta+\alpha\beta)}}{2} \quad [3.24]$$

Note that both α and β range from 0 to 1. Hence, $1-\beta+\alpha\beta$ is non-negative. This means that s_1 and s_2 must be non-positive, and either s_1 or s_2 is zero if and only if $1-\beta+\alpha\beta = 1-\beta(1-\alpha) = 0$, i.e., $\alpha = \beta = 1$, which is a less interesting case.

Moreover, from [3.23], we can have

$$\begin{aligned} L_T(s) &= \frac{a_1(1-\beta)s + a_1 a_2(1-\beta+\alpha\beta)}{(s-s_1)(s-s_2)} \\ &\triangleq \frac{B_1}{s-s_1} + \frac{B_2}{s-s_2} \end{aligned} \quad [3.25]$$

where constants B_1 and B_2 are given by

$$\begin{cases} B_1 = \frac{a_1 s_1(1-\beta)+a_1 a_2(1-\beta+\alpha\beta)}{s_1-s_2} \\ B_2 = -\frac{a_1 s_2(1-\beta)+a_1 a_2(1-\beta+\alpha\beta)}{s_1-s_2} \end{cases} \quad [3.26]$$

Therefore, it follows from [3.25] that the probability distribution function of response time T is

$$f_T(t) = B_1\, e^{s_1 t} + B_2\, e^{s_2 t} \quad [3.27]$$

The closed-form expression of $F_T(t, \lambda, \mu_1, \mu_2))$ is thus given by

$$F_T(t, \lambda, \mu_1, \mu_2)) = 1 + \left(\frac{B_1}{s_1}\, e^{s_1 T^D} + \frac{B_2}{s_2}\, e^{s_2 T^D} \right)$$

To ensure that $\gamma\%$ of the response time for customer service requests are not more than a desired target response time T^D, we require that $G(T^D) = 1 - F_T(t, \lambda, \mu_1, \mu_2)) \leq 1 - \gamma\%$. That is

$$\left(\frac{B_1}{s_1}\, e^{s_1 T^D} + \frac{B_2}{s_2}\, e^{s_2 T^D}\right) \geq \gamma\% - 1 \qquad [3.28]$$

From equation [3.28] and proposition 3.1, we can first determine the service level ($= \gamma\%$) for a given arrival rate and given service rates. That is, we have the following corollary for answering question 1) presented in section 3.6.1.

COROLLARY 3.1.– The level of QoS guaranteed services (γ) will be no more than

$$100\left[1 + \left(\frac{B_1}{s_1}\, e^{s_1 T^D} + \frac{B_2}{s_2}\, e^{s_2 T^D}\right)\right]$$

Second, from equation [3.28] and proposition 3.2, we can find service rates necessary to ensure a certain service level as in [3.2] and [3.3]. It can be shown that $G(T^D)$ is a decreasing function with respect to service rates μ_1 and μ_2. Hence, more specifically, proposition 3.2 can be rewritten as follows:

COROLLARY 3.2.– Service rate μ_1 in the Web server and service rate μ_2 in the service center are the solution of an optimization problem below:

$$\text{argmin}_{\mu_1 \in \mathcal{R}_1,\, \mu_2 \in \mathcal{R}_2} \left(\frac{B_1}{s_1}\, e^{s_1 T^D} + \frac{B_2}{s_2}\, e^{s_2 T^D}\right)$$

subject to $\gamma\% \leq 1 + \left(\frac{B_1}{s_1}\, e^{s_1 T^D} + \frac{B_2}{s_2}\, e^{s_2 T^D}\right)$, where s_1, s_2, B_1, and B_2 are given in [3.24] and [3.26].

Finally, proposition 3.3 is reduced as follows.

COROLLARY 3.3.– The maximal number of customers that can be supported without a violation of a predefined level of QoS

services $\gamma\%$ is the solution of an integer optimization problem below:

$$\text{argmax}_{n\in\mathcal{I}}\left(\frac{B_1}{s_1}\,e^{s_1T^D}+\frac{B_2}{s_2}\,e^{s_2T^D}\right)$$

subject to $\gamma\% \leq 1+\left(\frac{B_1}{s_1}\,e^{s_1T^D}+\frac{B_2}{s_2}\,e^{s_2T^D}\right)$.

The constrained optimization problems are two- and one-dimensional, which can be easily solved by using existing numerical tools (e.g. Matlab).

3.6.3. *A numerical validation*

In this section, we validate the correctness of propositions 3.1–3.3 and corollaries 3.1–3.3 and demonstrate how to find answers to the three questions given in section 3.6.1.

Since the expression of percentile response time is usually not a closed form and it is required to find the solutions of optimization problems in propositions 3.2–3.3 and corollaries 3.2–3.3, the correctness of these propositions and corollaries can be only validated numerically by a comparison of model simulation results. That is, the way to find the answers of the three questions in section 3.6.1 is a numerical approximate method. As stated before, the relative error percentage is used to measure the accuracy of the approximate results compared to model simulation results, and it is defined by:

$$\text{Relative error }\% = \frac{\text{Approximate Result} - \text{simulation result}}{\text{Simulation result}} \times 100.$$

We study the accuracy of our proposed approximation method using an example below.

We will verify the accuracy of the approximate method for the computing system analyzed in section 3.6.2 where the Web server and the service center are modeled as an $M/M/1$ queue. We let $\lambda = 100$, $\mu_1 = 380$, $\mu_2 = 200$, and α and β were varied.

We simulated the queuing network using Arena (see [ALT 01]), and the analytical method was implemented in Matlab and also in Mathematica. The simulation model in Arena exactly represents the computer service performance model under study; so the simulation results in Arena are considered "exact".

Table 3.11 shows the simulated and approximate cumulative distribution function of the response time for different values of α and β. In the table, the column labeled "Simul" gives the simulation result, the column labeled "Approx" gives the approximate result and the column labeled "R-Err %" gives their relative errors. These abbreviations are also used in other tables in this section. It appears that the results obtained by our approximate method are fairly good. For example, when $\alpha = 0.5$, $\beta = 0.2$, both results reflect that 98% of time the customer requests will be responded to in less than $T^D = 0.025$ as shown in Table 3.11. This means that when $\lambda = 380$, $\mu_1 = \mu_2 = 200$, $\alpha = 0.5$, $\beta = 0.2$, and $T^D = 0.025$, the level of QoS services $\gamma\%$ is no less than 98%.

Second, both propositions 3.2 and 3.3 (or corollaries 3.2 and 3.3) require us to solve an optimization problem. In order to obtain service rates μ_1 and μ_2 for a given arrival rate λ, we are required to find a solution of the two-dimensional optimization problem given in proposition 3.2 (or corollary 3.2). Meanwhile, in order to obtain the maximal number of customers n for given service rates μ_1 and μ_2, we are required to find a solution of the *integer* optimization problem given in proposition 3.3 (or corollary 3.3). Generally speaking, an integer optimization problem is more difficult to solve than a two-dimensional optimization problem whose parameters can

be chosen as real positive numbers. Hence, in the following simulation, we only consider proposition 3.3 (or corollary 3.3) to demonstrate how to find a solution of the integer optimization problem. Thus, the numerical example for proposition 3.2 (or corollary 3.2) is omitted.

Response time	α=0.65, β=0.4			α=0.65, β=0.2			α=0.5, β=0.2		
	Simul	Approx	R-Err %	Simul	Approx	R-Err %	Simul	Approx	R-Err %
0.005	0.5179	0.5166	−0.26	0.6395	0.6407	0.19	0.6291	0.6303	0.20
0.010	0.7478	0.7457	−0.28	0.8566	0.8556	−0.11	0.8436	0.8441	0.06
0.015	0.8624	0.8610	−0.17	0.9379	0.9375	−0.04	0.9278	0.9287	0.09
0.020	0.9232	0.9227	−0.05	0.9714	0.9718	0.04	0.9652	0.9659	0.08
0.025	0.9569	0.9568	−0.01	0.9863	0.9870	0.07	0.9824	0.9834	0.10
0.030	0.9752	0.9758	0.06	0.9936	0.9939	0.03	0.9910	0.9918	0.09
0.035	0.9854	0.9864	0.10	0.9968	0.9972	0.04	0.9953	0.9960	0.07
0.040	0.9914	0.9924	0.10	0.9983	0.9987	0.04	0.9975	0.9980	0.05
0.045	0.9950	0.9957	0.07	0.9991	0.9994	0.03	0.9985	0.9990	0.05
0.050	0.9969	0.9976	0.07	0.9994	0.9997	0.03	0.9992	0.9995	0.03
0.055	0.9981	0.9986	0.05	0.9996	0.9999	0.03	0.9994	0.9998	0.04
0.060	0.9989	0.9992	0.03	0.9998	0.9999	0.01	0.9996	0.9999	0.03
0.065	1.0000	0.9996	−0.04	0.9999	1.0000	0.01	0.9997	0.9999	0.02
0.070	1.0000	0.9998	−0.02	0.9999	1.0000	0.01	0.9998	1.0000	0.02
0.075	1.0000	0.9999	−0.01	1.0000	1.0000	0.00	0.9999	1.0000	0.01
0.080	1.0000	0.9999	−0.01	1.0000	1.0000	0.00	0.9999	1.0000	0.01
0.085	1.0000	1.0000	0.00	1.0000	1.0000	0.00	1.0000	1.0000	0.00

Table 3.11. *The cumulative distribution functions of the response time*

Let $\lambda_0 = 10$, $\mu_1 = 400$, $\mu_2 = 250$, $\alpha = 0.6$, $\beta = 0.4$, $T^D = 0.02$ and $\gamma\% = 94.5$. Then, λ_1 and λ_2 are calculated by $\lambda_1 = \frac{\lambda}{1-0.4\times0.4} = \frac{n}{0.084} \approx 11.905$ and $\lambda_2 = \lambda_1\beta = \frac{n}{0.21} \approx 4.762n$.

Furthermore, it follows from [3.24] that

$$s_{1,2} = \left\{-(650 - 16.666n) \pm \left[(650 - 16.666n)^2 - 3.36(400 - 11.905n)(250 - 4.762n)\right]\right\}/2$$

and from [3.26] we can obtain expressions of B_1 and B_2.

Then, by using either existing numerical tools (e.g. Mathematica, Matlab and Maple) or developing our own

numerical tool, we can solve for n in the integer optimization problem presented in proposition 3.3 (or corollary 3.3). In the numerical implementation, we obtain $n = 9$. This means that at most, nine customers can be supported so that 94.5% of the times all nine customers' requests can be completed in 0.02. We also simulated the computer service model and validated using the brute-force approach that $n = 9$ by using the closed-form solution is actually optimal. Table 3.12 gives the cumulative distribution of the response time obtained using proposition 3.3 (or corollary 3.3) and the simulation method. We noticed that proposition 3.3 (or corollary 3.3) gives us a very accurate solution.

In this approximation method, we assume that the waiting time of a customer at the Web server (or the service center) is independent of the waiting times at the service center (or the cloud), and it is also independent of his/her waiting times in other visits to the same Web server (or the service center). Hence, the relative error as shown in Table 3.11 is due to the above assumptions.

3.6.4. *Discussions*

We have studied three important but challenging questions for computer service performance in cloud computing: (1) for given service resources, what level of QoS services can be guaranteed? (2) For a given number of customers, how many service resources are required to ensure that customer services can be guaranteed in terms of the percentile of response time? (3) For given service resources, how many customers can be supported to ensure that customer services can be guaranteed in terms of the percentile of response time?

The main difficulty for answering these three questions in order to understand the characteristics of computer service

performance lies in the computation of the probability distribution function of response time. In this section, first we have proposed a queuing network model for studying the performance of computer services in cloud computing and then developed an approximation method for computing the Laplace transform of a response time distribution in the cloud computing system. Therefore, we have derived three propositions and three corollaries for answering the above three questions. The answers to the above three questions can be obtained by using a numerical approximate method in these propositions and corollaries.

We have further conducted numerical experiments to validate our approximate method. Numerical results showed that the proposed approximate method provided a good accuracy for the calculation of cumulative distributions of the response time, and the maximal number of customers for given computer service resources in cloud computing in which customer services can be guaranteed in terms of the percentile of response time. Hence, the proposed method provides an efficient and accurate solution for the calculation of probability and cumulative distributions of a customer's response time. It will be useful in the services' performance prediction of cloud computing. We plan to apply our proposed method to test cloud computing infrastructure, once it is available to us, in a real-world test. Additionally, the Web server and the service center have been modeled as a infinite queue for single-class customers in this section. The methods given in [XIO 06b] and [XIO 06d] can be applied to discuss a finite queue for single- and multiple-class customers.

3.7. Concluding remarks

We proposed an approach for resource optimization in a service provider's computing environment, whereby we minimize the total cost of computer resources allocated to a

customer so that it satisfies a given percentile of the response time. We have formulated the resource optimization problem as an optimization subject to SLA constraints for a service model with or without feedback. In the model without feedback, the obtained LST of a customer's response time is exact, and in the case of the model with feedback, it is approximate. We also developed an efficient and accurate numerical solution for inverting the LST of a customer's response time numerically. Validation tests showed that our approach has a very good accuracy. As an example, we specifically gave service performance modeling and analysis in a simple scenario of cloud computing. An extensive discussion of resource provisioning for cloud service job scheduling can be found in [XIO 10a].

Response time	Simul	Approx	R-Err%
0.005	0.5595	0.5594	−0.0258
0.010	0.7924	0.7954	0.3786
0.015	0.8996	0.8997	0.0092
0.020	0.9510	0.9495	−0.1598
0.025	0.9760	0.9741	−0.1973
0.030	0.9883	0.9861	−0.2179
0.035	0.9942	0.9933	−0.0947
0.040	0.9972	0.9963	−0.0880
0.045	0.9986	0.9980	−0.0617
0.050	0.9993	0.9989	−0.0422
0.055	0.9997	0.9994	−0.0267
0.060	0.9998	0.9996	−0.0237
0.065	0.9999	0.9998	−0.0120
0.070	1.0000	0.9998	−0.0161
0.075	1.0000	0.9999	−0.0081
0.080	1.0000	0.9999	−0.0091

Table 3.12. *The cumulative distribution functions of the response time*

In this chapter, we assumed that service requests are served in a queue in a FIFO manner. Priority service disciplines will be discussed in Chapter 4.

Chapter 4

Multiple-Class Customers

This chapter considers a set of computer resources used by a service provider to host enterprise applications subject to a service-level agreement (SLA) for differentiated customer services under a *preemptive-resume* priority. We present an approach for the resource optimization that minimizes the total cost of computer resources required while preserving a given percentile of the response time for priority-class customers. We first analyze an open tandem queuing network, and then we extend our work to an open tandem queuing network with feedback. In this chapter, we only consider two priority customers. High-priority-class customers are indexed 1 and low-priority-class customers are indexed 2. The obtained results, in this chapter, can be easily extended to the case of multiple priority customers by using a class-based decomposition. (This extension is not considered in this book). This chapter is mainly based on the results obtained in [XIO 06c].

This chapter is organized as follows. In section 4.1, we define the SLA performance metric considered in this book. Section 4.2 formulates the resource optimization problem for

differential customer services. In section 4.3, we first give the probability distribution of the response time distribution for a single priority queue with preemptive-resume. Then, we present two typical real-life models and propose an approach for solving the optimization problem for two priority-class customers. In section 4.4, numerical simulations demonstrate the applicability and validity of the proposed approach. Finally, the conclusions are given in section 4.5.

4.1. The SLA performance metric in the case of multiple class customers

Let us recall that an SLA is a contract between a customer and a service provider that defines all aspects of the service that is to be provided. In this chapter, the SLA consists of service performance and a fee under multiple customer services. We consider the percentile of the response time as the performance metric. This is the time it takes for a service request to be executed on the service provider's multiple resource stations.

Assume that $f_T(t)$ is the probability distribution function of a response time T of a certain priority class. For example, in the case of two priority classes, $T = T^{(1)}$ for the high-priority class and $T = T^{(2)}$for the low-priority class. $T_D^{(r)}$ is a desired target response time for priority class r ($r = 1, 2$) that a customer requests and agrees upon with his/her service provider based on a fee paid by the customer. The SLA performance metric used in this chapter can be expressed as follow:

$$\int_0^{T_D^{(r)}} f_{T^{(r)}}(t)\, dt \geq \gamma^{(r)}\%, \quad (r = 1, 2) \qquad [4.1]$$

That is, $\gamma^{(r)}\%$ of the time a customer will receive his/her service in less than $T_D^{(r)}$ ($r = 1, 2$).

As an example, let us consider an $M/M/1$ queue with an arrival rate $\lambda^{(r)}$ and a service rate $\mu^{(r)}$ ($r = 1, 2$). The service discipline is preemptive-resume. We want to describe the SLA performance metric [4.1] for the high-priority class. As discussed in section 4.1, in this case, the steady-state probability of the system as far the high-priority class is concerned is $p_0 = 1 - \rho^{(1)}$ and $p_k = (1 - \rho^{(1)})(\rho^{(1)})^k$, $k > 0$, where $\rho^{(1)} = \frac{\lambda^{(1)}}{\mu^{(1)}}$ (see [PER 94]). The response time $T^{(1)}$ is exponentially distributed with the parameter $\mu^{(1)}(1 - \rho^{(1)})$, i.e. the probability distribution of the high-priority response time is given by

$$f_{T^{(1)}}(t) = \mu^{(1)}(1 - \rho^{(1)})e^{-\mu^{(1)}(1-\rho^{(1)})t} \qquad [4.2]$$

Using the definition given in [4.1], we have

$$\int_0^{T_D^{(1)}} f_{T^{(1)}}(t)\, dt = 1 - e^{-\mu^{(1)}(1-\rho^{(1)})T_D} \geq \gamma^{(1)}\% \qquad [4.3]$$

or $\mu^{(1)} \geq \frac{-\ln(1-\gamma^{(1)}\%)}{T_D^{(1)}} + \lambda^{(1)}$. This means that in order to guarantee a higher SLA service level, $\mu^{(1)}$ increases when $T_D^{(1)}$ decreases. Similarly, for any given arrival rate $\lambda^{(1)}$ and service rate $\mu^{(1)}$, we can use [4.3] to find the percentile of $\gamma^{(1)}$.

4.2. The resource optimization problem for multiple customer services

The computer resource optimization problem for multiple customer services can be formulated as the following optimization problem.

4.2.1. *Resource optimization problem for multiple class customers*

Find integers n_j ($1 \leq n_j \leq N_j$; $j = 1, 2, \cdots, m$) in the n-dimensional integer optimization problem [1.1] under the constraints of percentile response times for differential customer services as expressed by [4.1], and the constraint: $I \leq C_D$, where C_D is a fee negotiated and agreed upon between a customer and the service provider.

4.3. Approaches for resource optimization

In this section, we propose an approach for solving the resource optimization problem for two typical service models that depict the path that service requests have to follow through the service provider's resource stations. Before presenting the approach, we need to derive the Laplace–Stieltjes transforms (LSTs) of the response time distributions for priority-class customers.

4.3.1. *The LSTs of response time distributions for two priority customers*

Let us recall that high-priority-class customers are indexed 1 and low-priority-class customers are indexed 2. In this section, we assume that the arrival processes of the two classes are independent of each other. Let $B^{(r)}(t)$ be the service-time cumulative distribution of class r with mean $1/v^{(r)}$ and second moment $1/v_2^{(r)}$ ($r = 1, 2$). The total service-time distribution $B(t)$ is given by $B(t) = \frac{\lambda^{(1)}}{\lambda}{}^{(1)} + \lambda^{(2)} B^{(1)}(t) + \frac{\lambda^{(2)}}{\lambda^{(1)}+\lambda^{(2)}} B^{(2)}(t)$, and the arrival rate into the queue is $\lambda = \lambda^{(1)} + \lambda^{(2)}$. It follows that the total utilization $\rho = \rho^{(1)} + \rho^{(2)}$, where $\rho^{(r)} = \frac{\lambda^{(r)}}{v^{(r)}}$, is equal to the occupation time given by $\lambda \int_0^\infty t \, d\, B(t) = \frac{\lambda^{(1)}}{v^{(1)}} + \frac{\lambda^{(2)}}{v^{(2)}} = \rho^{(1)} + \rho^{(2)}$. Assume

that the stability condition of the queuing system holds, i.e. $\rho = \rho^{(1)} + \rho^{(2)} < 1$.

The LST of the service-time distribution of class r is

$$g^{(r)}(s) = \int_0^\infty e^{-st}\, d\, B^{(r)}(t), \quad r = 1, 2 \qquad [4.4]$$

and the LST of the residual service-time distribution for class r (r = 1, 2) is

$$g_e^{(r)}(s) = \int_0^\infty e^{-st}\,(1-B^{(r)}(t))\, v^{(r)} dt = \frac{(1 - g^{(r)}(s))\, v^{(r)}}{s} \qquad [4.5]$$

The busy time period is the time between an interruption moment at which the server becomes busy due to an arriving customer and the first moment at which the server becomes available again. The LST of the busy time distribution of the high-priority class, denoted by $\delta^{(1)}(s)$, is the smallest root of the *Kendall functional equation* (Cohen [COH 82])

$$\delta^{(1)}(s) = g^{(1)}(s + \lambda^{(1)}(1 - \delta^{(1)}(s))) \qquad [4.6]$$

where $g^{(1)}$ is defined by [4.4].

Let c be the completion time of a customer, that is the time elapsed from the moment when the service of the customer begins to the moment where the customer is completely served. As derived in [4.51] of [COH 82], the LST, $L_c(s)$, of the complete time c for both preemptive-resume and non-preemptive-resume is

$$L_{c(s)} = g^{(2)}(z(s)) \qquad [4.7]$$

where

$$z(s) = s + \lambda^{(1)}(1 - \delta^{(1)}(s)) \qquad [4.8]$$

For the preemptive-resume discipline, class 2 does not exist as far as class 1 is concerned. Thus, in this case, the waiting-time distribution of the high-priority class is the same as in the First-In-First-Out (FIFO) queue without priorities. The probability distribution of the high-priority class was given in [4.2]. Hence, in the following discussion of this section, we only need to find the LSTs of the waiting-time distribution of the low-priority class.

Denote by $W^{(2)}$ the cumulative distribution of the steady-state waiting time of the low-priority customers. According to [4.63] and [4.67] in [COH 82], the LST of the low-priority waiting time $W^{(2)}(t)$ is given by

$$L_{W^{(2)}}(s) = \frac{1-\rho}{1-\rho f(s)}, \quad s > 0 \qquad [4.9]$$

where

$$f(s) = \frac{\rho^{(1)}}{\rho} h_0^{(1)}(s) + \frac{\rho^{(2)}}{\rho} g_e^{(2)}(z(s)) \qquad [4.10]$$

$$h_0^{(1)}(s) = \frac{1-\delta^{(1)}(s)}{\frac{1}{v^{(1)}} s + \rho^{(1)}(1-\delta^{(1)}(s))} \qquad [4.11]$$

Note that $g_e^{(2)}(z(s))$ and $\delta^{(1)}(s)$ are determined by [4.4] and [4.6], respectively.

Therefore, the low-priority response time denoted by $T^{(2)}$ is equal to the sum of the low-priority waiting time and the low-priority completion time whose LST is given by [4.7]. The low-priority response time distribution is a convolution of the distributions of the low-priority waiting time and the low-priority completion time. Thus, the LST of the

low-priority response time distribution denoted by $L_{T^{(2)}}(s)$ is: $L_{T^{(2)}}(s) = L_{W^{(2)}}(s) \times L_{c(s)}$. That is

$$L_{T^{(2)}}(s) = \frac{(1-\rho) \times g^{(2)}(z(s))}{1 - \rho f(s)}, \qquad s > 0 \qquad [4.12]$$

The distributions of the low-priority response time can be obtained by numerically inverting the LST of [4.12], respectively. This procedure can usually be performed via the software package mentioned in [GRA 01].

For presentation purposes, we only consider an $M/M/1$ queue in this book since the distributions of the low-priority response time given by [4.12] can be specified and simplified, as discussed in the following section.

4.3.1.1. *The LST of the low-priority response time distribution in an $M/M/1$ queue*

In an $M/M/1$ priority queue, assume that the service rates are $\mu^{(r)}$ ($r = 1, 2$). This means that $v^{(r)} = \mu^{(r)}$. In this case, from [4.4] and [4.5], we have, for $r = 1, 2$,

$$g^{(r)}(s) = \frac{\mu^{(r)}}{s + \mu^{(r)}} \qquad [4.13]$$

$$g_e^{(r)}(s) = \frac{\mu^{(r)}}{s + \mu^{(r)}} \qquad [4.14]$$

Therefore, from [4.6] and [4.13], we derive the relation

$$\delta^{(1)} = \frac{1 - \delta^{(1)}}{\frac{1}{u^{(1)}}\, s + \rho^{(1)}(1 - \delta^{(1)})} \qquad [4.15]$$

and by solving for $\delta^{(1)}$, we obtain

$$\delta^{(1)} = \frac{\eta - \sqrt{\eta^2 - 4\,\lambda^{(1)}\,\mu^{(1)}}}{2\,\lambda^{(1)}} \qquad [4.16]$$

where η is determined by

$$\eta = s + \lambda^{(1)} + \mu^{(1)} \qquad [4.17]$$

Furthermore, it follows from [4.10], [4.11] and [4.15] that $h_0^{(1)}(s) = \delta^{(1)}$, and

$$f(s) = \frac{\rho^{(1)}}{\rho} \frac{\eta - \sqrt{\eta^2 - 4\,\lambda^{(1)}\,\mu^{(1)}}}{2\lambda^{(1)}} + \frac{\rho^{(2)}}{\rho} \frac{\mu^{(2)}}{z(s) + \mu^{(2)}} \qquad [4.18]$$

where $z(s)$ is given by [4.8].

Moreover, from [4.6], [4.8] and [4.13], we have

$$\delta^{(1)} = \frac{\mu^{(1)}}{z(s) + \mu^{(1)}}, \quad \text{or} \quad z(s) = \mu^{(1)}[(\delta^{(1)})^{-1} - 1] \qquad [4.19]$$

Thus, due to [4.19], the LST of the low-priority waiting-time distribution given in [4.9] reduces to

$$\begin{aligned} L_{W^{(2)}}(s) &= \frac{1-\rho}{1-\rho^{(1)}\,\delta^{(1)} - \rho^{(2)}\,\frac{\mu^{(2)}}{z(s)+\mu^{(2)}}} \\ &= \frac{(1-\rho)\{\mu^{(1)}[(\delta^{(1)})^{-1}-1]+\mu^{(2)}\}}{(\mu^{(1)}-\mu^{(2)})\rho^{(1)}\delta^{(1)}+\mu^{(1)}(\delta^{(1)})^{-1}-\xi} \end{aligned} \qquad [4.20]$$

where ξ is given by

$$\begin{aligned} \xi &= \mu^{(1)}(\rho^{(1)} + \rho^{(2)}) + (\mu^{(1)} - \mu^{(2)})\,(1 - \rho^{(2)}) \\ &= \mu^{(1)}\rho + (\mu^{(1)} - \mu^{(2)})\,(1 - \rho^{(2)}) \end{aligned} \qquad [4.21]$$

Hence, from [4.16] and [4.20], we have the LST of the low-priority waiting distribution for an $M/M/1$ queue

$$L_{W^{(2)}}(s) = \frac{(1-\rho)(\frac{\eta}{2} + \frac{1}{2}\sqrt{\eta^2 - 4\,\lambda^{(1)}\,\mu^{(1)}} - (\mu^{(1)} - \mu^{(2)}))}{(1 - \frac{\mu^{(2)}}{2\,\mu^{(1)}})\eta - \xi + \frac{\mu^{(2)}}{2\,\mu^{(1)}}\sqrt{\eta^2 - 4\,\lambda^{(1)}\,\mu^{(1)}}} \qquad [4.22]$$

As a result, replacing [4.12] gives the LST of the low-priority response time distribution.

$$L_{T^{(2)}}(s) = \mu^{(2)}(1-\rho)[\tfrac{\eta}{2} + \tfrac{1}{2}\sqrt{\eta^2 - 4\,\lambda^{(1)}\,\mu^{(1)}}$$
$$-(\mu^{(1)} - \mu^{(2)})]\{(z(s) + \mu^{(2)}) \times [(1 - \tfrac{\mu^{(2)}}{2\,\mu^{(1)}})\eta$$
$$-\xi + \tfrac{\mu^{(2)}}{2\,\mu^{(1)}}\sqrt{\eta^2 - 4\,\lambda^{(1)}\,\mu^{(1)}}]\}^{-1}, \quad s > 0 \qquad [4.23]$$

where $z(s)$, η and ξ are given by [4.8], [4.17] and [4.21], respectively.

In addition, when the service rates of the two priority classes are the same, i.e. $\mu^{(1)} = \mu^{(2)}$, [4.18] reduces to the following expression:

$$f(s) = \frac{\rho^{(1)}}{\rho}\,\delta^{(1)} + \frac{\rho^{(2)}}{\rho}\,\frac{\mu^{(2)}}{z(s) + \mu^{(2)}}$$
$$= \frac{\rho^{(1)}}{\rho}\,\delta^{(1)} + \frac{\rho^{(2)}}{\rho}\,\frac{\mu^{(1)}}{z(s) + \mu^{(1)}} = \delta^{(1)}$$

due to [4.19]. Hence, equation [4.9] becomes $L_{W^{(2)}}(s) = \frac{1-\rho}{1-\rho\,\delta^{(1)}}$. Thus, the LST of the low-priority response time distribution is

$$L_{T^{(2)}}(s) = \frac{(1-\rho)\,\delta^{(1)}}{1 - \rho\,\delta^{(1)}} \qquad [4.24]$$

on the basis of [4.12], [4.19] and $\mu^{(1)} = \mu^{(2)}$. It should be pointed out that when the utilization ρ is fixed, the LST of low-priority response time distribution does not depend on the arrival rate $\lambda^{(2)}$ of the low-priority class.

4.3.2. *Algorithms for the resource optimization problem*

In this section, we study two queuing network models that depict the path that service requests have to follow through

the service provider's resource stations. These two models are shown in Figures 4.1 and 4.2. We refer to these two queuing models as service models since they depict the resources used to provide a service to a customer.

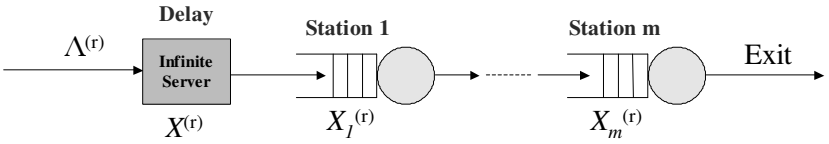


Figure 4.1. *A tandem-station service model for multiple-class customer services*

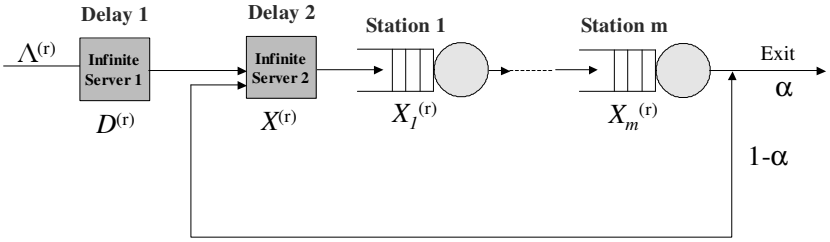


Figure 4.2. *A service model with feedback for multiple-class customer services*

The first service model consists of a single infinite server and m stations numbered sequentially from 1 to m as shown in Figure 4.1. Each station j is modeled as a priority queue served by n_j identical servers, each providing a service at the rate $\mu_j^{(r)}$, where $r = 1, 2$. Let $\Lambda^{(r)}$ be the external arrival rate to the infinite server, and let $\lambda^{(r)}$ and $\lambda_j^{(r)}$ be the effective arrival rates to the infinite server and station j, $j = 1, 2, \cdots, m$. We assume that all service times are exponentially distributed and the external arrival to the infinite server occurs in a Poisson manner.

The infinite server represents the total propagation delay from the user to the service provider and back to the user,

and also from station 1 to m. Each station carries out a particular function. For instance, it could be a database server, a file server, a Web server, a group of CPUs and local disks, etc. We consider two priority classes of customers in this book. In the following discussion, each station is modeled as a single $M/M/1$ priority queue with arrival rate $\lambda_j^{(r)}$ and service rate $\psi^{(r)}(n_j)\mu_j^{(r)}$, where $\psi^{(r)}(n_j)$ is a known function of n_j ($r = 1, 2$), and depends on the configuration of servers and the type of customers at each station. It is non-decreasing and can be inverted, i.e. $(\psi^{(r)})^{-1}$ exists ($r = 1, 2$). For instance, suppose that a station represents a group of CPUs. Then, $\psi^{(r)}(n)$ can be seen as a CPU scaling factor for the number of CPUs from 1 to n. According to [CHA 05], $\psi^{(r)}(n) = (\xi^{(r)})^{\log n}$, where $\xi^{(r)}$ is a basic scaling factor from CPU 1 to CPU 2, and it ranges from 1 to 2 ($r = 1, 2$). So, $(\psi^{(r)})^{-1}(n) = (\xi^{(r)})^{-\log n}$. In addition, each station that is modeled as a *single* $M/M/1$ priority queue with service rate $\psi^{(r)}(n_j)\mu_j^{(r)}$ can only serve either high- or low-priority customers at one time. Hence, $\psi^{(1)}(n_j)\mu_j^{(1)}$ is considered to be the same as $\psi^{(2)}(n_j)\mu_j^{(2)}$.

Since the queuing network is overtake-free, the waiting time of a customer at a station is independent of his/her waiting times at other stations (see [DAD 84] and [WAL 80]). Let $X^{(r)}$ be the service time at the infinite server and $X_j^{(r)}$ be the time elapsed from the moment a customer arrives at station j to the moment he/she departs from the station ($r = 1, 2$). Then, the total response time is $T^{(r)} = X^{(r)} + X_1^{(r)} + X_2^{(r)} + \cdots + X_m^{(r)}$, and hence the LST of the response time T is

$$L_{T^{(r)}}(s) = L_{X^{(r)}}(s)L_{X_1^{(r)}}(s)\cdots L_{X_m^{(r)}}(s) \qquad [4.25]$$

where $L_{X^{(r)}}(s)$ is the LST of the service time $X^{(r)}$ given by

$$L_{X^{(r)}}(s) = \frac{\lambda^{(r)}}{s + \lambda^{(r)}} \quad [4.26]$$

and $L_{X_j^{(r)}}(s)$ is the LST of the response time $X_j^{(r)}$ at the j-th station, where $r = 1, 2$.

Because of the preemptive-resume priority, the high-priority response time is the same as in the single-class FIFO $M/M/1$ whose probability distribution can be expressed as [4.2]. Thus, $L_{X_j^{(1)}}(s)$ is determined by

$$L_{X_j^{(1)}}(s) = \frac{\psi^{(1)}(n_j)\mu_j(1-\rho_j^{(1)})}{s + \psi^{(1)}(n_j)\mu_j(1-\rho_j^{(1)})}, \quad (j = 1, 2, \cdots, m) \quad [4.27]$$

and $L_{X_j^{(2)}}(s)$ is the LST of the low-priority response time given by

$$L_{X_j^{(2)}}(s) = \frac{(1-\rho_j)\delta_j^{(1)}}{1 - \rho_j\,\delta_j^{(1)}}, \quad (j = 1, 2, \cdots, m) \quad [4.28]$$

due to [4.24], where $\rho_j^{(r)} = \frac{\lambda_j^{(r)}}{\psi^{(r)}(n_j)\mu_j}$, $\rho_j = \frac{\lambda_j^{(1)} + \lambda_j^{(2)}}{\psi^{(r)}(n_j)\mu_j}$, $\delta_j^{(1)}$ is given by

$$\delta_j^{(1)} = \frac{\eta_j - \sqrt{\eta_j^2 - 4\,\psi^{(1)}(n_j)\lambda_j^{(1)}\,\mu_j^{(1)}}}{2\,\psi^{(1)}(n_j)\lambda_j^{(1)}}$$

and $\eta_j = s + \lambda_j^{(1)} + \psi^{(1)}(n_j)\mu_j^{(1)}$ for $j = 1, 2, \cdots, m$.

From [4.25]–[4.28], we have

$$L_{T^{(1)}}(s) = \frac{\lambda^{(1)}}{s + \lambda^{(1)}} \Pi_{j=1}^{m} \frac{\psi^{(1)}(n_j)\mu_j(1-\rho_j^{(1)})}{s + \psi^{(1)}(n_j)\mu_j(1-\rho_j^{(1)})}$$

and

$$L_{T^{(2)}}(s) = \frac{\lambda^{(2)}}{s+\lambda^{(2)}} \Pi_{j=1}^{m} \frac{(1-\rho_j)\delta_j^{(1)}}{1-\rho_j\,\delta_j^{(1)}}$$

We observe that $f_{T^{(r)}}(t)$ and $F_{T^{(r)}}(t)$ $(r = 1, 2)$ are usually nonlinear functions of t and n_j. Hence, the resource optimization problem is an n-dimensional linear optimization problem subject to nonlinear constraints. In general, it is not easy to solve this problem. However, the complexity of the problem can be significantly reduced by requiring that the service rates of the queues in the service model in Figure 4.1 are all equal. That is, we find the optimum value of $n_1, \cdots, n_m$ such that $\psi^{(r)}(n_1)\mu_1^{(r)} = \cdots = \psi^{(r)}(n_m)\mu_m^{(r)}$ $(r = 1, 2)$, called *balanced utilization* or *balanced condition,* as defined in Chapter 3.

From the traffic equations $\lambda^{(r)} = \lambda_j^{(r)} = \Lambda^{(r)}$ $(j = 1, 2, \cdots, m)$, we have that the utilization of each station $\rho_j^{(r)} = \frac{\lambda_j^{(r)}}{\psi^{(r)}(n_j)\mu_j^{(1)}} = \frac{\Lambda^{(r)}}{\psi^{(r)}(n_j)\mu_j^{(1)}}$. Thus, we have that for the high-priority queue, $\hat{a}_i = \psi^{(1)}(n_i)\mu_i^{(1)}(1-\rho_i^{(1)}) = \psi^{(1)}(n_j)\mu_j(1-\rho_j^{(1)}) = \hat{a}_j \stackrel{\triangle}{=} \hat{a}$ that implies $n_j = (\psi^{(1)})^{-1}(\frac{\hat{a}+\lambda_j^{(1)}}{\mu_j})$ $(i, j = 1, 2, \cdots, m)$. Hence, from [4.25], we have $f_{T^{(1)}}(t) = L^{-1}\{\frac{\lambda^{(1)}}{s+\lambda^{(1)}} \cdot \frac{\hat{a}^m}{(s+\hat{a})^m}\}$, and subsequently we obtain

$$F_{T^{(1)}}(t) = L^{-1}\{\frac{\lambda^{(1)}}{s(s+\lambda^{(1)})} \cdot \frac{\hat{a}^m}{(s+\hat{a})^m}\} \qquad [4.29]$$

Moreover, for the low-priority queue, we have $\rho_{j_1} = \rho_{j_2} \stackrel{\triangle}{=} \hat{\rho}$, and $\delta_{j_1}^{(1)} = \delta_{j_2}^{(1)} \stackrel{\triangle}{=} \hat{\delta}^{(1)}$, for $j_1,\ j_2 = 1, 2, \cdots, m$,

due to $\psi^{(2)}(n_1)\mu_1^{(2)} = \cdots = \psi^{(2)}(n_m)\mu_m^{(2)} \triangleq \hat{b}$. Thus, $L_{T^{(2)}}(s)$ reduces to

$$L_{T^{(2)}}(s) = \frac{\lambda^{(2)}}{s+\lambda^{(2)}} \frac{(1-\hat{\rho})^m(\hat{\delta}^{(1)})^m}{(1-\hat{\rho}\,\hat{\delta}^{(1)})^m} \quad [4.30]$$

which is a function of only one variable $\hat{b}$, since $\hat{\rho}$ and $\hat{\delta}^{(1)}$ are considered as functions of only one variable $\hat{b}$. As a result, $\sum_{j=1}^{m} n_j c_j$ reduces to a function of variable $\hat{a}$ due to

$$n_j^{(1)} = \lceil (\psi^{(1)})^{-1}(\frac{\hat{a}+\lambda_j^{(1)}}{\mu_j^{(1)}}) \rceil$$

for a high-priority customer and $n_j^{(2)} = \lceil (\psi^{(2)})^{-1}(\frac{\hat{b}}{\mu_j^{(2)}}) \rceil$ for a low-priority customer. Thus, the resource optimization problem can be decomposed into the two one-dimensional resource optimization problems for both high-priority and low-priority customers, respectively. We have the following algorithm for the resource optimization problem.

ALGORITHM 4.1.–

1) The minimization problem for high-priority customers: find $\hat{a}$ in the one-dimensional optimization problem:

$$\hat{a}^{min} \leftarrow \arg\min_{\hat{a}} F_{T^{(1)}}(t)|_{t=T_D^{(1)}}$$

subject to the constraint $F_{T^{(1)}}(t)|_{t=T_D^{(1)}} \geq \gamma^{(1)}\%$ at $\hat{a} = \hat{a}^{min}$, where $F_{T^{(1)}}(t)$ is given by [4.29].

2) The minimization problem for low-priority customers: find $\hat{b}$ in the one-dimensional optimization problem:

$$\hat{b}^{min} \leftarrow \arg\min_{\hat{b}} F_{T^{(2)}}(t)|_{t=T_D^{(2)}}$$

subject to the constraint $F_{T^{(2)}}(t)|_{t=T_D^{(2)}} \geq \gamma^{(2)}\%$ at $\hat{b} = \hat{b}^{min}$, where $F_{T^{(2)}}(t)$ is given by [4.30].

3) Compute integers $n_j^{(1)}$ and $n_j^{(2)}$ by using the expressions:

$$n_j^{(1)} = \lceil (\psi^{(1)})^{-1}(\frac{\hat{a}^{min}}{\mu_j^{(1)}(1-\rho_j^{(1)})}) \rceil \text{ and } n_j^{(2)} = \lceil (\psi^{(2)})^{-1}(\frac{\hat{b}^{min}}{\mu_j^{(2)}}) \rceil$$

Then, calculate $n_j = \max\{n_j^{(1)}, n_j^{(2)}\}$ ($j = 1, 2, \cdots, m$).

4) Check if $1 \leq n_j \leq N_j$ ($j = 1, 2, \cdots, m$) and $I \leq C^D$ are satisfied. If yes, the obtained n_j is the number of servers required at each station. Otherwise, print "the problem cannot be solved".

Note that when the balanced utilization discussed above is satisfied, the suboptimal solution obtained by algorithm 4.1 is optimal for the resource optimization problem. This statement is also true for algorithm 4.2 that will be discussed later.

To allow for a more complex execution path of a service request, we extended the above model to a service model with feedback, as shown in Figure 4.2. Infinite server 1 represents the total propagation delay within a network provider and infinite server 2 represents the propagation delay within the service provider, i.e. among stations 1 to m. In this figure, a customer upon completion of his/her service at the mth station exits the system with probability α or returns to the beginning of the system with probability $1 - \alpha$. We note that the model shown in Figure 4.2 can be easily extended to a network of queues arbitrarily connected. We reuse the notation in the first model shown in Figure 4.1: $\Lambda^{(r)}$ as the external arrival rate; $\lambda_d^{(r)}$, $\lambda^{(r)}$ and $\lambda_j^{(r)}$ as the effective arrival rates to the second infinite server and station j; and $\mu_j^{(r)}$ as the service rate at station j, where $j = 1, 2, \cdots, m$ and $r = 1, 2$.

The main difficulty of this resource optimization problem is to find $f_{T^{(r)}}(t)$, the probability distribution function of $T^{(r)}$ ($r = 1, 2$). We obtain this probability distribution function assuming that the waiting time of a customer at a station is

independent of his/her waiting time at other stations and each visit at the same station j is independent of the others. (We note that this assumption of independence does not hold in queuing networks with feedback. However, as will be discussed in the validation section, the solution obtained has a good accuracy). We first have the traffic equations: $\lambda_d^{(r)} = \Lambda^{(r)}$, $\lambda^{(r)} = \Lambda^{(r)} + (1-\alpha)\lambda_m^{(r)}$ and $\lambda_j^{(r)} = \lambda^{(r)}$ that implies $\lambda_j^{(r)} = \lambda^{(r)} = \frac{\Lambda^{(r)}}{\alpha}$, and the utilization of each station is $\rho_j^{(1)} = \frac{\lambda_j^{(r)}}{\psi^{(r)}(n_j)\mu_j} = \frac{\Lambda^{(r)}}{\alpha\mu_j\psi^{(r)}(n_j)}$ $(j = 1, 2, \cdots, m)$.

Then, the high-priority and low-priority response times of the kth pass at the infinite station and the jth station are considered as the sum of $m+2$ random variables $T^{(r)}(k) = D^{(r)} + X^{(r)} + X_1^{(r)} + \cdots + X_m^{(r)}$, where we assume that the high-priority (or low-priority) waiting time of a customer at a station is independent of his/her high-priority (or low-priority) waiting times in other visits to the same station. $D^{(r)}$ and $X^{(r)}$ are the service times of class r at the first and second infinite servers, respectively, and $X_j^{(r)}$ is the time elapsed from the moment a class r customer arrives at station j to the moment he/she departs from it. Then, the total response time is $T^{(r)} = \sum_{k=0}^{\infty} p(k)T^{(r)}(k)$, where $p(k)$ is the steady-state probability that a request will circulate k times at the infinite station and the jth station through the computing system. $p(k)$ is determined by $p(k) = \alpha(1-\alpha)^{k-1}$. Thus, the LST of the response time $T^{(r)}$ is $L_{T^{(r)}}(s) = L_D(s)\sum_{k=0}^{\infty} p(k)L_X^k(s)L_{X_1^{(r)}}^k(s)\cdots L_{X_m^{(r)}}^k(s)$, which can be rewritten as follows

$$L_{T^{(r)}}(s) = \frac{\alpha L_{D^{(r)}}(s)L_X(s)\,\Pi_{j=1}L_{X_j^{(r)}}(s)}{1-(1-\alpha)\,L_X(s)\Pi_{j=1}^m L_{X_j^{(r)}}(s)} \qquad [4.31]$$

where $L_{D^{(r)}}(s)$ is the LST of the service time $D^{(r)}$ given by $L_{D^{(r)}}(s) = \frac{\Lambda^{(r)}}{s+\Lambda^{(r)}}$, and replacing $L_X(s)$ and $L_{X_j^{(r)}}(s)$

($j = 1, 2, \cdots, m$) by [4.26]–[4.28] in [4.31], we have

$$L_{T^{(1)}}(s) = \frac{(\Lambda^{(1)})^2 \Pi_{j=1}^{m} \hat{a}_j}{(s + \Lambda^{(1)})[(s + \lambda^{(1)}) \Pi_{j=1}^{m} (s + \hat{a}_j) - (1 - \alpha)\lambda^{(1)} \Pi_{j=1}^{m} \hat{a}_j]} \quad [4.32]$$

where $\hat{a}_j = \psi^{(1)}(n_j)\mu_j^{(1)}(1 - \rho_j^{(1)})$, and

$$\begin{aligned} L_{T^{(2)}}(s) = {} & (\Lambda^{(2)})^2 \Pi_{j=1}^{m} [(1 - \rho_j)\delta_j^{(1)}](s + \Lambda^{(2)})^{-1} \\ & \times \{(s + \lambda^{(2)}) \Pi_{j=1}^{m} (1 - \rho_j \, \delta_j^{(1)}) \\ & - (1 - \alpha)\lambda^{(2)} \Pi_{j=1}^{m} [(1 - \rho_j)\delta_j^{(1)}]\}^{-1} \end{aligned} \quad [4.33]$$

To find the response time distribution $f_{T^{(r)}}(t)$, we are required to invert the LST given by [4.32] using partial fraction decomposition of a rational function. However, the partial fraction decomposition of the rational function requires searching for roots of a high-order polynomial. It is usually not an easy task when the order of the polynomial is more than 5. Instead, in this chapter, the LST given by [4.32] is inverted numerically as [4.33].

Similarly, we want to find $n_1, \cdots, n_m$ such that the best utilization of these stations is achieved, which implies that each station has the same maximal service capacity. That is, $\hat{a}_i = \hat{a}_j = \hat{a}$, $\hat{\rho}_i = \hat{\rho}_j = \hat{\rho}$ and $\hat{\delta}_i^{(1)} = \hat{\delta}_j^{(1)} = \hat{\delta}^{(1)}$ ($i, j = 1, \cdots, m$). Then, from equations [4.32] and [4.33], and $F_{T^{(r)}}(t) = L^{-1}\{L_{T^{(r)}}(s)/s\}$ ($r = 1, 2$), we have

$$F_{T^{(1)}}(t) = L^{-1}\{\frac{(\Lambda^{(1)})^2 \hat{a}^m}{s(s + \Lambda^{(1)})[(s + \lambda^{(1)})(s + \hat{a})^m - (1 - \alpha)\lambda^{(1)}\hat{a}^m]}\} \quad [4.34]$$

and

$$F_{T^{(2)}}(t) = L^{-1}\{(\Lambda^{(2)})^2[(1-\hat{\rho})\hat{\delta}^{(1)}]^m s^{-1}(s+\Lambda^{(2)})^{-1} \times\{(s+\lambda^{(2)})(1-\hat{\rho}\,\hat{\delta}^{(1)})^m -(1-\alpha)\lambda^{(2)}[(1-\hat{\rho})\hat{\delta}^{(1)}]^m\}^{-1}\} \quad [4.35]$$

Thus, we have the following algorithm for the resource optimization problem in the model shown in Figure 4.2.

ALGORITHM 4.2.–

Steps 1–4 are the same as steps 1–4 in algorithm 4.1 except $F_{T^{(2)}}(t)$ and $F_{T^{(2)}}(t)$ given by [4.34] and [4.35], respectively.

Note that if we cannot obtain a solution for the resource optimization problem using algorithm 4.1 (or 4.2), then the service provider cannot execute the service request for the service model 1 (or 2) due to at least one of the following reasons: (1) the service provider has an insufficient resource (i.e. N_j is too small), (2) a prespecific fee is too low (i.e. $I > C_D$) and (3) a network connection is either too slow or has a problem so that [4.1] cannot be satisfied. Using this information, we may detect and debug either a network problem or a service provider's capacity problem, or the SLA needs to be renegotiated.

4.4. Numerical validations

In this section, we demonstrate the accuracy and applicability of our proposed approximation method.

Two types of errors are introduced in our proposed approximation method. The first error, hereafter referred to as class I error, comes from numerically inverting the Laplace

transform. The second error, hereafter referred to as class II error, is due to the assumptions that the waiting time of a customer at each station is independent of the waiting times at the other stations, and it is also independent of his/her waiting times in other visits to the same station.

Let us recall that the relative error % is used to measure the accuracy of the approximate results compared to model simulation results and it is defined as follows

$$\text{Relative error } \% = \frac{\text{Approximate result} - \text{Simulation result}}{\text{Simulation result}} \times 100$$

as given in [3.11].

As can be seen, our proposed approximation method mainly depends on the computation of the inverse Laplace transform

$$f(t) = \frac{1}{2\pi i} \int_{c-i\infty}^{c+i\infty} e^{t\,s} \hat{f}(s)\, ds$$

where $\hat{f}(s)$ is the image function of the inverse Laplace function $f(t)$, and it is defined by

$$\hat{f}(t) = \int_0^{\infty} e^{-s\,t} f(t)\, dt$$

where $f(t)$ is called the original function of $\hat{f}(s)$ and it is a real- or complex-valued function defined on the positive part $\mathcal{R}_+$.

The numerical inversion of the Laplace transform has been widely studied and several efficient methods have been proposed in the past few decades (see [GRA 01]). However, as is described in [GRA 01], the numerical computation of an inverse Laplace transform is an ill-posed problem. The inverse Laplace transform is determined by the singularities of the image function $\hat{f}(s)$. This means that the behavior of

the image function near the singularities determines the inverse Laplace transform. Hence, we need to consider the singularities of the image function in our numerical validations. In addition, although some numerical methods work for certain image functions well, they may provide poor results for other image functions. No single method works for every image function. Thus, in our numerical validations, we used several different numerical methods for a given image function. If two or more methods can reach approximately the same results, then we are confident that the derived numerical inverse Laplace transform is correct. These numerical methods include the inversion methods using Laguerre functions and Fourier functions (see [GRA 01]), Gaussian quadrature formulas (see [PIE 71]) and the method by Gaver [GAV 66] and Stehfest [STE 70].

We study the accuracy of our proposed approximation method using several examples in the following.

Contrary to the distribution of the high-priority response time whereby it is the single-class FIFO $M/M/1$ case, as given in section 4.1, the distribution of the low-priority response time in an $M/M/1$ queue does not have a closed-form solution. The distribution whose Laplace transform is given by [4.24] has to be evaluated numerically by using one of the numerical methods for inverting a Laplace transform (e.g. [GRA 01]). Hence, we first verify the accuracy of the approximate method for the single queue given by [4.24]. Let $\mu^{(1)} = \mu^{(2)} = \frac{1000}{9} = 111.11$ and $\lambda^{(r)}$ is varied ($r = 1, 2$).

We simulated the queuing network using Arena (see [ALT 01]) and the analytical method was implemented in Mathematica using the package of the inverse Laplace transform in [GRA 01]. The simulation results are considered as “exact” since the simulation model is an exact representation of the queuing network under study.

Tables 4.1 and 4.2 show the simulated and approximate cumulative distributions of the low-priority response times for the two different cases: (1) $\lambda^{(1)} = \lambda^{(2)} = 50$; (2) $\lambda^{(1)} = 75$ and $\lambda^{(2)} = 25$. In these two tables, the column labeled "Simul" gives the simulation result, the column labeled "Approx" gives the approximate result and the column labeled "R-Err %" gives their relative errors. The same abbreviations are also used in other tables in the rest of this section. It appears that using the package of the inverse Laplace transform in [GRA 01], we can obtain a good accuracy for the numerical inversion of the low-priority response time distribution given by [4.24].

Response time	Simul	Approx	R-Err %
0.1	0.5183	0.4821	−6.9844
0.3	0.8691	0.8318	−4.2918
0.5	0.9691	0.9447	−2.5178
0.7	0.9911	0.9818	−0.9384
0.9	0.9997	0.9940	−0.5702
1.1	1.0000	0.9980	−0.2000
1.3	1.0000	0.9994	−0.0600
1.5	1.0000	0.9998	−0.0200
1.7	1.0000	0.9999	−0.0100
1.9	1.0000	1.0000	0.0000

Table 4.1. *The cumulative distribution of the low-priority response time for* $\lambda^{(1)} = \lambda^{(2)} = 50$

Then, we verify the accuracy of our approach for the first service model shown in Figure 4.1. Let $m = 8$, $\lambda^{(1)} = 100$, $\lambda^{(2)} = 50$, $\mu_1^{(1)} = 48$, $\mu_2^{(1)} = 18$, $\mu_3^{(1)} = 85$, $\mu_4^{(1)} = 32$, $\mu_5^{(1)} = 49$, $\mu_6^{(1)} = 24$, $\mu_7^{(1)} = 28$, $\mu_8^{(1)} = 38$, $\mu_1^{(2)} = 42$, $\mu_2^{(2)} = 15$, $\mu_3^{(2)} = 60$, $\mu_4^{(2)} = 25$, $\mu_5^{(2)} = 41$, $\mu_6^{(2)} = 18$, $\mu_7^{(2)} = 26$ and $\mu_8^{(2)} = 35$. We also choose $N_j = 100$, $c_j = 1$, $\psi^{(1)}(n_j) = 1.5^{\log n_j}$, $\psi^{(2)}(n_j) = 1.55^{\log n_j}$ and $C_D = 300$ ($j = 1, \cdots, 8$).

Response time	Simul	Approx	R-Err %
0.1	0.4175	0.3837	−8.0958
0.3	0.7115	0.6783	−4.6662
0.5	0.8617	0.8225	−4.5491
0.7	0.9338	0.9003	−3.5875
0.9	0.9669	0.9435	−2.4201
1.1	0.9835	0.9679	−1.5862
1.3	0.9922	0.9817	−1.0583
1.5	0.9970	0.9895	−0.7523
1.7	0.9982	0.9940	−0.4208
1.9	0.9996	0.9966	−0.3001
2.1	1.0000	0.9980	−0.2000
2.3	1.0000	0.9999	−0.0100
2.5	1.0000	1.0000	0.0000

Table 4.2. *The cumulative distribution of the low-priority response time for* $\lambda^{(1)} = 75$ *and* $\lambda^{(2)} = 25$

Tables 4.3 and 4.4 show the simulated and approximate cumulative distributions of the high-priority and low-priority response times, respectively. It appears that the results obtained by algorithm 4.1 are very accurate. It is shown in Table 4.5 that the optimal number of servers is required for 97.5% of the high-priority response time to be less than $T_D^{(1)} = 0.16$ and for 97.5% of the low-priority response time to be less than $T_D^{(2)} = 0.8$. Moreover, the optimal number of servers is required for satisfying both the high-priority and the low-priority response times presented in Table 4.5. The exact optimal number of servers, obtained by exhaustive search using the simulation model and assuming that each station has balanced utilization, is consistent with the ones presented in Table 4.5. So, $I = 214 < C_D$. We point out that the relative errors presented in Tables 4.3 and 4.4 are only due to the class I error since the class II error is not present for this model.

Response time	Simul	Approx	R-Err %
0.02	0.0002	0.0002	0.0000
0.04	0.0214	0.0214	0.0000
0.06	0.1542	0.1528	−0.9079
0.08	0.4085	0.4075	−0.2448
0.10	0.6691	0.6672	−0.2840
0.12	0.8459	0.8450	−0.1064
0.14	0.9385	0.9379	−0.0639
0.16	0.9781	0.9780	−0.0102
0.18	0.9931	0.9929	−0.0201
0.20	0.9980	0.9979	−0.0100
0.22	0.9995	0.9994	−0.0100
0.24	0.9998	0.9998	0.0000
0.26	0.9999	1.0000	0.0100
0.28	1.0000	1.0000	0.0000

Table 4.3. *The cumulative distribution of the high-priority response time in model 1*

Response time	Simul	Approx	R-Err %
0.2	0.1767	0.1473	−16.6384
0.3	0.4538	0.4396	−3.1291
0.4	0.6982	0.7082	1.4323
0.5	0.8541	0.8719	2.0841
0.6	0.9389	0.9503	1.2142
0.7	0.9774	0.9824	0.5116
0.8	0.9925	0.9942	0.1713
0.9	0.9978	0.9982	0.0401
1	0.9993	0.9995	0.0200
1.1	0.9998	0.9999	0.0100
1.2	1.0000	1.0000	0.0000

Table 4.4. *The cumulative distribution of the low-priority response time in model 1*

Let us now consider an example of the service model 2 shown in Figure 4.2. We choose $m = 8$, $\Lambda^{(1)} = 55$, $\Lambda^{(2)} = 42$, $\mu_1^{(1)} = 12$, $\mu_2^{(1)} = 46$, $\mu_3^{(1)} = 95$, $\mu_4^{(1)} = 25$, $\mu_5^{(1)} = 35$, $\mu_6^{(1)} = 20$, $\mu_7^{(1)} = 10$ and $\mu_8^{(1)} = 98$, and $\mu_1^{(2)} = 15$, $\mu_2^{(2)} = 42$, $\mu_3^{(2)} = 90$, $\mu_4^{(2)} = 18$, $\mu_5^{(2)} = 28$, $\mu_6^{(2)} = 15$, $\mu_7^{(2)} = 5$ and $\mu_8^{(2)} = 82$, and $\alpha = 0.67$. Let us also select $N_j = 250$, $c_j = 1$, $\psi^{(1)}(n_j) = 1.5^{\log n_j}$, $\psi^{(2)}(n_j) = 1.55^{\log n_j}$ and $C_D = 800$ ($j = 1, \cdots, 8$). Thus, it follows from equation: $\lambda^{(r)} = \frac{\Lambda^{(r)}}{\alpha}$ ($r = 1, 2$) that $\lambda^{(1)} = 82.09$ and $\lambda^{(2)} = 62.69$.

Station	1	2	3	4	5	6	7	8
High-priority customer	12	62	5	23	12	38	29	18
Low-priority customer	12	61	7	27	13	46	26	16
All the customers	12	62	7	27	13	46	29	18

Table 4.5. *The optimal number of servers in model 1*

We obtained the cumulative distribution of the response time by solving [4.34] and [4.35] using the package of the inverse Laplace transforms in [GRA 01]. Table 4.6 presents the number of servers in the eight stations necessary to ensure the 95% SLA guarantee for $T_D^{(1)} = 0.3$ and $T_D^{(2)} = 1.0$, respectively, and the number of servers in the eight stations necessary to ensure all the customers.

We also simulated the tandem queuing network and validated using the brute-force approach that these numbers of servers obtained by our approximate method are in fact optimal, provided that each station has balanced utilization. The optimal number of servers is presented in Table 4.6. It derives that $I = 617 < C_D$, i.e. step 4 in algorithm 4.2 is met. Tables 4.7 and 4.8 give the cumulative distributions of the high-priority and low-priority response times obtained using the approximate method and the simulation method, and their relative error %. The relative error comes from both

classes I and II error. We note that our approximate method has a very good accuracy when percentiles are high. Extensive numerical results (not reported here due to lack of space) point to the fact that the independence assumption has little impact on the accuracy of the results when the number of nodes is large. A contributing factor is that typically we are interested in values of the cumulative distribution of the response time that correspond to very high percentiles for which the approximate results seem to have a very good accuracy.

Station	1	2	3	4	5	6	7	8
High-priority customer	123	13	4	35	20	52	168	4
Low-priority customer	61	12	4	46	23	61	342	5
All the customers	123	13	4	46	23	61	342	5

Table 4.6. *The optimal number of servers in model 2*

Response time	Simul	Approx	R-Err %
0.1	0.3879	0.3850	−0.7476
0.2	0.8343	0.8332	−0.1318
0.3	0.9528	0.9547	0.1994
0.4	0.9867	0.9877	0.1013
0.5	0.9961	0.9966	0.0502
0.6	0.9989	0.9991	0.0200
0.7	0.9997	0.9998	0.0100
0.8	0.9999	0.9999	0.0000
0.9	1.0000	1.0000	0.0000

Table 4.7. *The cumulative distribution of the high-priority response time in model 2*

4.5. Concluding remarks

We proposed an approach for resource optimization in a service provider's computing environment, whereby we

minimize the total cost of computer resources allocated to a priority-class customer so that it satisfies a given percentile of the response time for each class customer. We have formulated the resource optimization problem as an optimization subject to SLA constraints for a service model with or without feedback. We have derived the LSTs of a customer's high-priority and low-priority response times. We further developed an efficient and accurate numerical solution for inverting the LSTs of a high-priority and low-priority customer's response time numerically. In the resource optimization problem, we are typically interested in values of the cumulative distribution of the response time that correspond to very high percentiles. Validation tests showed that our approach has a very good accuracy in this case. An extensive model can be found in [XIO 09] and [XIO 11a].

Response Time	Simul	Approx	R-Err %
0.2	0.1679	0.1467	−12.6266
0.4	0.6144	0.6116	−0.4557
0.6	0.8219	0.8207	−0.1460
0.8	0.9126	0.9177	0.5588
1.0	0.9567	0.9622	0.5749
1.2	0.9784	0.9826	0.4293
1.4	0.9889	0.9920	0.3135
1.6	0.9943	0.9963	0.2011
1.8	0.9971	0.9978	0.0702
2.0	0.9985	0.9992	0.0701
2.2	0.9993	0.9996	0.0300
2.4	0.9996	0.9998	0.0200
2.6	0.9998	0.9999	0.0100
2.8	0.9999	1.0000	0.0100
3.0	0.9999	1.0000	0.0100
3.2	1.0000	1.0000	0.0000

Table 4.8. *The cumulative distribution of the low-priority response time in model 2*

Chapter 5

A Trustworthy Service Model

This chapter considers a set of computer resources used by a service provider to host enterprise applications for customer services subject to a service-level agreement (SLA). The SLA defines three Quality-of-Service (QoS) metrics, namely trustworthiness, percentile response time and availability. We present an approach for resource optimization in such an environment that minimizes the total cost of computer resources used by a service provider for such an application while satisfying all three QoS metrics in a trust-based resource provisioning problem, which typically arises in Web services. We formulate the trust-based resource provisioning problem as an optimization problem under SLA constraints, and we solve it using an efficient numerical procedure. Part of the materials of this chapter has been published in [XIO 06e].

The rest of the chapter is organized as follows. In section 5.2, we present a framework for solving the trust-based resource provisioning problem, and give the calculation of SLA metrics in section 5.3. In section 5.4, we propose an approach for solving the trust-based resource provisioning problem for single-class and multiple-class customer services,

respectively. A numerical example is given in section 5.5 that demonstrates the validity of this approach. We conclude our results in section 5.6.

5.1. The trust-based resource optimization problem

The main standard in Web services is Extensible Markup Language (XML). XML provides a foundation for many core standards including Web Services Description Language (WSDL), Universal Description, Discovery and Integration specification (UDDI), and Simple Object Access Protocol (SOAP) [CUR 02]. WSDL allows developers to describe what services are offered, and it helps Web services of e-business to be accessed in public. UDDI defines XML-based registries in which businesses can upload information about themselves and the services they offer. SOAP gives us a way to move XML messages between locations, and it enables programs on separate computers to interact across any network. Thus, Web services allow us to exchange data with other applications on different computers by using Internet protocols.

However, most existing Web services products do not support SLAs that guarantee a level of service delivered to a customer for a given price. An SLA is a formal contract between a customer and a service provider that defines all aspects of the service being provided. In general, it consists of security, performance and availability. *Security* can be categorized as *identity security* and *behavior security*. Identity security includes the authentication and authorization between a customer and a service provider, data confidentiality and data integrity. Behavior security describes the trustworthiness among multiple resource sites, and the trustworthiness of these resource sites by customers, including the trustworthiness of computing results provided

by these sites. Performance includes the two following aspects [MEN 02].

1) *Response time* is the time for a service request to be satisfied. Namely, this is the time it takes for a service request to be executed on the service provider's multiple resource sites.

2) *Throughput* is the service rate that a service provider can offer. It is defined by the maximum throughput or by the undergoing change of throughput with service intensity.

Finally, *availability* is the percentage of time that a service provider can offer services.

In this chapter, we consider a resource management problem for Web services under SLA guarantees. Specifically, we define and solve a trust-based resource provisioning problem that occurs in Web services applications, subject to the constraints of trustworthiness, percentile response time and availability. Figure 5.1 depicts a typical scenario for these applications. A customer represents a business that submits a service request consisting of a stream of service jobs at a given rate with a certain price for a given QoS. After the trust manager that represents the customer negotiates an SLA with a service broker that represents resource sites (alternatively called service sites) S_1, $S_2, \cdots, S_M$ where $M > 0$, it checks the trustworthy information of the resource sites and selects m ($0 < m \leq M$) of those sites that meet predefined trustworthy requirements for serving the service request, simply say sites $1, 2, \cdots, m$. All service managers that manage the selected sites need to work together to achieve the SLA's requirement, and they overlook all resources within their sites. Upon the completion of a service job, the completed result is sent back to the trust manager. The trust manager forwards the completed job to the customer after checking and updating trustworthiness information in its database. Meanwhile, the customer

periodically sends feedback to the trust manager who uses it to update its trustworthiness information as well.

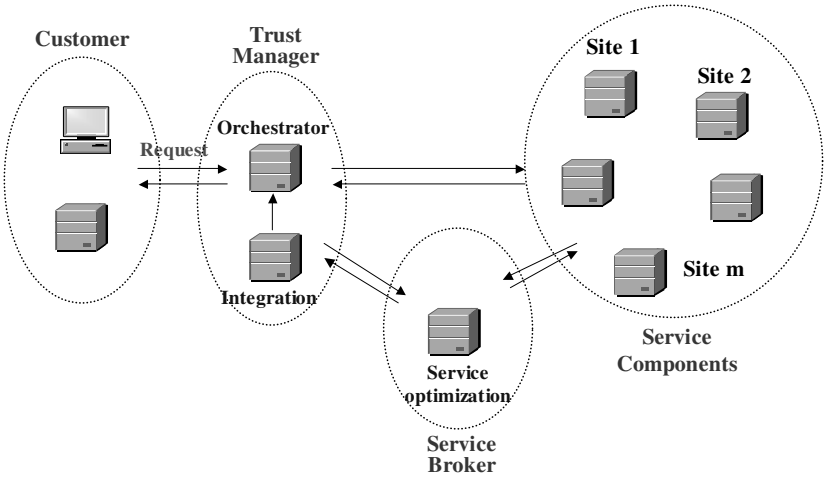


Figure 5.1. *An SLA-based Web services model*

The trust-based resource provisioning problem is minimizing the overall cost of the trusted computing resources required while satisfying SLA requirements. For presentation purposes, we assume that each resource site has only one type of server, each with cost c_j. Otherwise, if they have multiple types of servers, we can divide each resource site into several sites so that each one only contains one type of server with the same cost. Let N_j be the number of servers at site j ($j = 1, 2, \cdots, M$). Thus, the trust-based resource provisioning is quantified by solving for n_j ($1 \leq n_j \leq N_j$) in the following optimization problem:

$$\min_{n_1,\cdots,n_m} (n_1 c_1 + \cdots + n_m c_m) \qquad [5.1]$$

subject to constraints by an SLA. We discuss these constraints in section 5.3 and present a framework for the

detailed workload of the above Web services scenario in section 5.2.

In this chapter, we present an approach for the resource optimization that minimizes the total cost of computer resources required while preserving three given QoS metrics for single-class and priority-class customers, respectively. We calculate the number of servers in each resource station that minimize a cost function that reflects operational costs in the trust-based resource provisioning problem. We analyze an open tandem queuing network. We note that the proposed approach can be also applied to queuing networks consisting of nodes arbitrarily linked.

5.2. A framework for solving the trust-based resource provisioning problem

Services components from several universal service providers can be flexibly integrated into a composite service with cross-language and cross-platform regardless of their location, platform, execution speed and process. Delivering quality services to meet customer's requirement under an SLA is very important and challenging due to the dynamical and unpredictable nature of Web services applications. In this chapter, we propose a Web services framework for solving the trust-based resource provisioning problem as shown in Figure 5.2. The framework consists of a customer, a trust manager, a service broker and a service processor. The functions of these service entities are explained as follows.

– The customer represents a business that negotiates a contract for particular Web services with a service broker and submits a service request to be processed by a number of resource sites. The customer consists of a number of business end users (simply called users), and the service request defines service jobs generated by the users at a given rate, and QoS requirements with a fee.

– The trust manager is an entity that plays the following roles:

- Service integration: according to the customer's request, it integrates services chosen from a number of resource sites.

- Trust monitoring and determination: it monitors and determines the trustworthiness of service sites.

- Service selection: it chooses service sites that meet trust level requirements predefined by the customer.

- Service orchestration: it defines the sequence and conditions in which one Web service invokes other Web services in order to realize some useful function. That is, it defines the pattern of interactions that service components must follow in order to process the customer's requests.

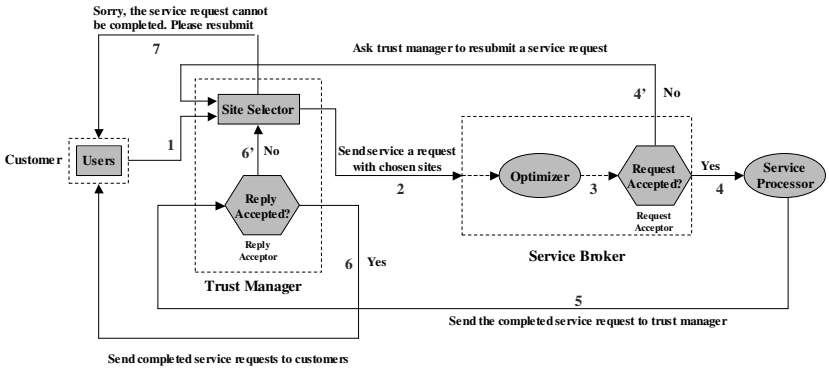


Figure 5.2. *A framework for solving the trust-based resource provisioning problem*

On the basis of the customer request, the trust manager generates a new service request that contains a set of service sites selected by the trust manager.

– The service broker is an entity that represents service sites. After it receives the trust manager's service request, it optimizes the number of service resources in each service

site to ensure the customer's QoS requirements. The service resource could be a piece of software, hardware or both, and it plays a key role in completing the customer's request. In this chapter, we consider it to be a hardware device such as a blade, a CPU, a storage device, a disk, a router and so on.

– The service processor is an entity that manages and instructs service sites to process the customer request based on the trust manager's service request. The service sites are entities that provide specific services. They may be owned by different service providers. Each site consists of a number of service resources managed by a service manager.

Furthermore, we outline the workflow for an approach to solving the trust-based resource provisioning problem in the framework as shown in Figure 5.2. The workflow can be presented in the following steps.

Step 1: a customer submits a service request to the trust manager that represents the customer. Let us recall that the customer represents a business consisting of a number of users within this business, and the service request consists of a number of service jobs generated by users.

Step 2: the site selector that is part of the trust manager selects service sites with the trust indices that meet the customer's trust requirement. Then, the trust manager submits the customer request to the service broker.

Step 3: an optimizer that is part of the service broker runs an optimization algorithm to find the number of servers required at each resource site to ensure the customer's SLA guarantee. Then, it will decide whether to accept the service request based on the profitability of the service broker.

Step 4: if the service broker can make an acceptable profit with a service provider that owns these chosen service sites,

then the service request is accepted. Subsequently, the service request is submitted to a service processor whereby these chosen service sites process the service request based on a rule defined in the trust manager's request. If the service request is not accepted, go to step 4'. The service processor is an entity that manages service sites.

Step 5: after these chosen service sites finish the service request, the service processor sends the completed service request back to the trust manager. Then, the reply acceptor that is part of the trust manager will decide whether to accept the service reply based on the trust indices of these chosen sites at the time when it receives the response from the service processor.

Step 6: if the trust indices of these chosen sites meet the customer's requirement, then the service reply is accepted. Subsequently, the completed service request is forwarded to the customer.

Step 6': if the trust indices of these chosen sites do not meet the customer's requirement, then the service reply cannot be accepted. Subsequently, the service acceptor forwards the service request to the site selector and asks it to resubmit the service request.

Step 4': if the service broker cannot make an acceptable profit with a service provider that owns these chosen service sites, then the service request cannot be accepted. Thus, the service broker notifies the trust manager to resubmit the service request by reselecting service sites.

Step 7: after the number of resubmissions is more than a threshold predefined by the trust manager or the service broker, then the trust manager will notify the customer that his/her request cannot be completed. Then, the customer

needs to either abort the service request or modify the QoS requirement with a fee, accordingly.

REMARK 5.1.–

– The trust manager and the service broker represent the customer and the chosen service sites, respectively. Hence, we assume that they only receive service commissions from the customer and the service processor, respectively, based on the number of service completions.

In general, the trust manager may be a business entity that not only receives a service commission from the customer, but also makes money since it may work with the service broker to find service sites that meet predefined trust requirements. Moreover, the chosen service sites have an overall lower fee.

– The trust manager selects service sites on behalf of its customer. Hence, it does not care about how much cost is needed to process the customer's request. But, if the number of service resubmissions surpasses a threshold predefined by the trust manager or the service broker, then the trust manager will notify the customer that her/his request cannot be completed, as mentioned in step 7.

– If the trust manager constantly resubmits a service request to the service broker, then the service request submissions constantly consume the service broker's resource. Thus, a denial of service attack may occur when an attacker impersonates the trust manager by continuing to submit bogus service requests to the service broker. We do not analyze such an attack here since it is beyond the scope of our study in this chapter.

– The above steps can be regarded as an SLA negotiation process between the trust manager and the service broker. The process is often finished before service delivery, i.e. the SLA is static. However, the SLA can be dynamically changed as well. In this case, the SLA will be periodically verified and/or negotiated. The length of the period of time for the verification

and/or negotiation depends on individual service types. It may be a week or a month.

5.3. The calculation of SLA metrics

Before presenting an algorithm to solve the above trust-based resource provisioning problem, we need to qualitatively analyze three SLA metrics: trustworthiness, percentile response time and service availability. In this section, we provide the calculation of three SLA metrics.

5.3.1. *The trustworthiness of resource sites*

In section 5.1, we classified security into *identity security* and *behavior security*, similar to identity trust and behavior trust defined in a grid computing system [ALT 01]. Identity security includes the authentication and authorization between a customer and a service provider, data confidentiality and data integrity. It has been widely studied in the literature (e.g. see [BIS 02], [KIM 00], [PER 01] and [ROS 04]). The identity security is beyond the scope of our study in this chapter. We will discuss group authentication in Chapter 6.

In this study, "trust" is used to deal with the notion of the trustworthiness in behavior security. Its definition is varied in the literature. In this chapter, trust is a firm belief in the competence of a resource site to act as expected. The trustworthiness of resource sites is an indicator of the QoS provided by these sites based on previous and current job completion experience. It often predicates the future behavior of the QoS at these sites. Moreover, we are only interested in the trustworthiness of resource sites from a customer's perspective. Hence, in this study, we simply assume that these resource sites trust each other.

Furthermore, assume that the trust manager is a trusted agent that represents customers. The trust manager uses the collected trustworthy information regarding the resource sites to evaluate their security behavior. We consider security behavior by modeling the behavior trusts of all sites, and quantify the trustworthiness of these sites using a rank and a threshold-based approach. This approach is based on previous job completion experience assessed by the trust manager and customers. The assessment may also include the opinion of other trust managers besides its own customer's feedback. The domain of feedback is assumed to be [0, 1].

The feedback is a statement issued by the trust manager's customers about the QoS provided by those chosen service sites in each service request or for a set of service requests during a certain period of time. These customers only provide feedback about their received services rather than those chosen service sites. (Note that the trust manager's customer is usually not aware of which service sites process its service request). Then, the trust manager scores the trust indices of those chosen service sites that serve these customers' requests. In the meantime, the trust manager may have its own feedback. In this case, the trust indices of those chosen service sites will aggregate both the trust manager's feedback and its customers' feedback.

The opinion is defined as the information of trustworthiness provided by those trust managers that are *neighbors* of the trust manager. These neighbors are a small subset of the trust manager's acquaintances, adaptively selected based on their usefulness. The opinion aggregates the overall impression of those neighborhood trust managers for the service sites.

We first discuss the case in which the trust indices of service sites are only determined by the trust manager's and

its customers' feedback. Let us consider the discrete times t_1, $t_2, \cdots, t_k, \cdots$ in an increasing order ($k = 1, 2, \cdots$). Let $I_j^{t_k}$ be the trust index of site j at time t_k. Then, *a trust function* is defined by

$$I_j^{t_{k+1}} = \xi \frac{R_j^s(k+1)}{R_j^a(k+1)} + (1-\xi) I_j^{t_k} \quad [5.2]$$

for each site j ($j = 1, 2, \cdots, M$), where $R_j^s(k+1)$ is the number of service jobs completed at site j that satisfied customers provided that the trust manager itself does not assess each completed job, or both the trust manager and its customers provided that both of them assess each completed job. $R_j^a(k+1)$ is the total number of service jobs submitted to site j during the time period $[t_k, t_{k+1}]$. The trust manager's satisfaction is assessed by the validation of a customer's SLA requirement after a service is completed, and a customer's satisfaction is based on the customer's feedback assessed by the customer through a comparison of the job completion experience with its expectation.

For presentation purposes, we simply assume that the trust manager has the same feedback as its customers. (Otherwise, the following discussion and notation need to be adjusted accordingly, which can be easily done). Denote $r_j(k+1)$ by

$$r_j(k+1) = \frac{R_j^s(k+1)}{R_j^a(k+1)} \quad [5.3]$$

Moreover, $r_j(k+1)$ is the satisfactory rate at site j from time t_k to t_{k+1} and it obviously ranges from 0 to 1. Clearly, when a set of the chosen sites is unchanged during this period of time, the number of completion jobs is the same for all chosen sites. Thereby, $r_j(k+1)$ is the same for these chosen sites as well. ξ is a parameter determined by the trust

manager. It is chosen depending on the type of resource sites. If a critical service job is processed at site j and site j's security is sensitive with the change of time, then ξ should be close to 1 (e.g. >0.7). Otherwise, it should be close to 0 (e.g. <0.3). Thus, $I_j^{t_k}$ ranges from 0 to 1, and it is called *a trust index*. It gives the percentage of a time that a resource site completes jobs to the satisfaction of the trust manager and its customers.

Then, we discuss the case in which the trust indices of service sites are determined by not only the trust manager's and its customers' feedback, but also the opinions of other trust managers. In today's large-scale complex computer system, a central trust manager would not be able to represent all customers, and would not know or would not be able to keep up with the trustworthiness information of all service sites in the system. This implies that there would exist multiple or many trust managers in the large-scale complex computer system. In this case, the trust index function in [5.2] should be modified by considering the opinions of the trust manager's neighbors besides its own customer's feedback. Thus, the satisfactory rate $r_j(k+1)$ consists of the following two components: the first component is based on the feedback of the trust manager and its own customers that is denoted by $r_j^b(k+1)$. Thus, $r_j^b(k+1)$ is given by [5.3], i.e.

$$r_j^b(k+1) = \frac{R_j^s(k+1)}{R_j^a(k+1)}$$

The second component is determined by aggregating all the opinions of the trust manager's neighbors and it is denoted by $r_j^o(k+1)$. Moreover, the satisfactory rate $r_j(k+1)$ is given by

$$r_j(k+1) = \xi^b \times r_j^b(k+1) + \xi^o \times r_j^o(k+1)$$

where $\xi^b + \xi^o = 1$. The two parameters are used to adjust the balance between the feedback and the opinion, and they are determined by the trust manager. Subsequently, the trust index function in [5.2] is rewritten as

$$\begin{aligned} I_j^{t_{k+1}} &= \xi\, r_j(k+1) + (1-\xi) I_j^{t_k} \\ &= \xi\, [\xi^b \times r_j^b(k+1) \\ &\quad + \xi^o \times r_j^o(k+1)] + (1-\xi) I_j^{t_k} \end{aligned}$$

The trust function describes the trustworthiness of resource sites monitored and updated by the trust manager. Therefore, the trust function reflects a probabilistic security behavior of the resource sites from a customer's perspective.

5.3.2. *The percentile response time*

The SLA performance metric defined in section 5.1 includes *throughput* and *response time*. As an end user, a customer is, in general, concerned about response time rather than throughput. So, in this study, we only consider *the response time*. In the trust-based resource provisioning problem for Web services, a response time is the time it takes for a job to be executed and completed in a distributed computing environment consisting of a trust manager and multiple resource sites.

In the literature, typically the average response time (or an average execution time) is used (e.g. see [MEN 02] and [MEN 04a]). The average response time is mainly influenced by "outliers", which occur in almost all measurements. Therefore, although the average response time is relatively easy to calculate, it may not address the concerns of a customer. Typically, a customer is more inclined to request a statistical bound on his/her response time than an average

response time. For instance, a customer can request that at least 95% of the times his/her response time should be less than a desired value. Hence, in this chapter, our aim is finding an optimal resource provisioning from trusted resource sites that meets a desired percentile response time.

Let us recall that $f_T(t)$ is the probability distribution function of a response time T of a customer as defined in section 3.1 for the case of single-class customers, and a certain priority-class customer as defined in section 4.1 for the case of multiple-class customers. For example, in the case of two priority classes, $T = T^{(1)}$ for the high-priority class and $T = T^{(2)}$for the low-priority class. $T_D^{(r)}$ is a desired target response time for priority class r ($r = 1, 2$) that a customer requests and agrees upon with his/her service provider based on a fee paid by the customer. Then, the SLA performance metric used in this chapter is

$$\int_0^{T_D} f_{T(t)}(t)\, dt \geq \gamma\% \qquad [5.4]$$

for the case of single-class customers. That is, $\gamma\%$ of the time a customer will receive his/her service in less than T_D, where T_D is a desired target response time that a customer requests and agrees upon with his/her service provider based on a fee paid by the customer. The SLA performance metric used in this chapter is

$$\int_0^{T_D^{(r)}} f_{T^{(r)}}(t)\, dt \geq \gamma^{(r)}\%, \quad (r = 1, 2) \qquad [5.5]$$

for the case of multiple-class customers. That is, $\gamma^{(r)}\%$ of the time a customer will receive his/her service in less than $T_D^{(r)}$ ($r = 1, 2$).

As an example given in section 4.1, when considering an $M/M/1$ queue with an arrival rate $\lambda^{(r)}$ and a service rate $\mu^{(r)}$

($r = 1, 2$). The service discipline is preemptive-resume. We have the probability distribution of the high-priority response time is given by

$$f_{T^{(1)}}(t) = \mu^{(1)}(1 - \rho^{(1)})e^{-\mu^{(1)}(1-\rho^{(1)})t} \quad [5.6]$$

5.3.3. *The service availability*

Availability is a critical metric in today's computer design [HEN 99]. Brown and Patterson [BRO 00] used an availability metric to describe the variations in system QoS over time. It is defined by the latency of a request service and the number of failures that can be tolerated by a system. The former was discussed in section 5.3.2. So, we only need to study the latter in this section. We consider the percentage of time that a resource is "up" or "down" as a metric, which is the traditional way to define service availability.

Let MTTF_j (mean time to failure) be the average time of a server failure and MTTR_j (mean time to recover) be the average time for recovering a server at the resource site j. Then, MTBF_j (mean time between failure) is a sum of MTTF_j and MTTR_j. MTBF information can be obtained from a hardware provider. For example, it is mentioned in [CIS 13c] that MTBF information is available for all Cisco components and is available upon request to a local account manager. MTTR is determined by evaluating how quickly a site owner can repair broken servers. It is a major factor of server availability. To improve service availability, it is necessary to reduce the frequency time of failure, as indicated in [BRE 01].

Network availability data may be also found on the Internet. For example, the University of Houston maintains current and historical network availability, see [UH 12].

Furthermore, each server fails at a rate of $\frac{1}{MTTF_j}$, denoted by a_j, and recovers (i.e. it is put back into operation) at a rate

of $\frac{1}{MTTR_j}$, denoted by b_j ($j = 1, \cdots, m$). Thus, a two-state Markov chain with the states "up" and "down" can be used to study the service availability at site j. The failure rate a_j is the state of transition from "up" to "down", and the recovery rate b_j represents the rate of transition from "down" to "up". Then, the probability p_j^i that i servers are down is given by (see [BOL 98])

$$p_j^i = \frac{N_j!}{i!(N_j - i)!}\, \eta_j^i\, p_j^0, \quad \text{for} \quad i = 1, \cdots, N_j \qquad [5.7]$$

where $\eta_j = \frac{a_j}{a_j + b_j}$ is the server unavailability rate, and p_j^0 is given by

$$p_j^0 = [N_j! \sum_{i=0}^{N_j} \frac{\eta_j^i}{i!(N_j - i)!}]^{-1} \qquad [5.8]$$

The probability that no more than $N_j - n_j$ servers at site j are down is

$$P_j(n_j, N_j) = p_j^0 \sum_{i=0}^{N_j - n_j} \frac{N_j!}{i!(N_j - i)!}\, \eta_j^i \qquad [5.9]$$

[5.9] can also be seen as the probability that at least n_j servers in site j are available.

5.4. An approach for solving the trust-based resource provisioning problem

As we saw in section 5.2, a Web services framework consists of a customer, a trust manager, a service broker and a service processor. In this framework, an optimizer within the service broker is an important key component. It calculates the number of service resources required to ensure that the response time of a service request meets the

requirement of a predefined percentile response time under a given fee. The calculation of the response time is a sum of the processing time of the trust manager and the execution time of chosen service sites for the customer request.

Without any confusion, we reuse m ($0 < m \leq M$) as the number of resource sites necessary for processing a customer's service job. Assume that the trust manager first selects m resource sites from the M resource sites. We consider the following two cases: single customer services and multiple priority customer services, respectively.

5.4.1. *Single-class customers*

We can model those m resource sites as a queuing network as shown in Figure 5.3. Without loss of generality, we assume that the first m sites are chosen. In Figure 5.3, the tandem queuing network consists of a trust server and m stations numbered sequentially from 1 to m. The trust server represents the trust manager and each station represents a resource site. Each station carries out a particular function, such as a database server, a computing server, a file server or a Web server. The execution procedure of a service request may be complicated in the real world, but the main idea of our proposed modelling approach can be extended to describe any collaborative relationship among resource sites as long as the relationship can be quantified.

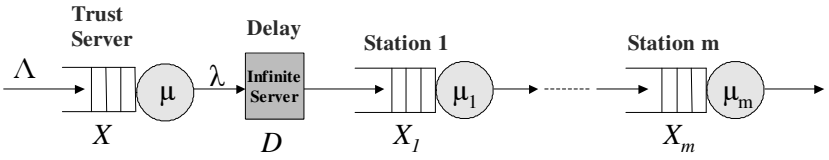


Figure 5.3. *A Web services computing system for single-class customer services*

In Figure 5.3, each station j is modeled as a single First-In-First-Out (FIFO) queue served by n_j identical servers, each providing a service at the rate μ_j. Let Λ be the external arrival rate to the infinite server, and let λ and λ_j be the effective arrival rates to the infinite server and station j ($j = 1, 2, \cdots, m$). The notion of server here is defined as a service resource at each site that processes users' jobs. For example, as mentioned previously, it could be a blade, a CPU, a disk, a storage device and so on. We assume that all service times are exponentially distributed and the external arrival to the trust server occurs in a Poisson manner. The trust server provides a service at the rate μ.

In Figure 5.3, the infinite server represents the total propagation delay from the user to the service provider and back to the user and also from station 1 to m. We only consider a single class of customer in this chapter.

In this chapter, we are interested in minimizing the overall cost of the above Web services computing system so that the desired SLA is guaranteed. Before presenting an algorithm for solving the minimization problem, we first need to calculate the response time of a customer's service request. Recall that the response time refers to the time elapsed from the moment a customer's service job joins the computing system to the time it departs. It is the most important QoS metric. The response time also reflects security behavior and the availability of services in some degree.

Let T be a random variable that represents the response time of a service job. Also, assume that $f_T(t)$ is the probability distribution of T. Denote the overall service cost [1.1] by

$$g(n_1, n_2, \cdots, n_m) = \sum_{j=1}^{m} n_j c_j \qquad [5.10]$$

where n_1, $n_2, \cdots$, n_m are the number of servers allocated to a specific stream of jobs with arrival rate λ.

Then, the trust-based resource provisioning problem can be formulated as the following three subproblems:

1) Select m resource sites within a predefined trust index at time $t = t_k$.

2) Solve for n_j in the m-dimensional integer optimization problem:

$$\min_{n_1,\cdots,n_m} g(n_1, n_2, \cdots, n_m) \quad [5.11]$$

Under the constraint of a percentile response time of [3.1], and the constraint of service availability:

$$P_j(n_j, N_j) \geq \delta_j\% \quad [5.12]$$

where T^D is a desired response time defined by a customer and δ_j is a desired percentage of service availability at site j. $P_j(n_j, N_j)$ is given by [5.9].

3) Update the trust indices of all M sites based on the activity during the time interval $[t_k, t_k + T^D]$. Then, the trust manager decides if a completed job is accepted. If each trust index at those selected sites that complete the service job meet a predefined index value, then the completed job is accepted. Otherwise, the completed job is discarded and the trust manager needs to resubmit the job.

Note that in the model, the verifying time of a trust index is ignored since we are interested in the processing time of a job at resource sites. While subproblems 1 and 3 are solved at the customer side to ensure a customer service received from reliable resource sites, subproblem 2 is solved by the service broker who represents these resource sites. Hence, in the current model, the trustworthiness of resource sites in

subproblems 1 and 3 are not considered as a constraint for the optimization problem in subproblem 2.

The above resource optimization problem as described in subproblem 2 of section 3.3 is to find a minimal cost presented in [5.11] such that $\gamma\%$ of the time a customer's response time is less than a predefined value T^D and $\delta_j\%$ of time the selected resource sites have n_j servers available for the job.

In the following discussion, each server station is modeled as a single $M/M/1$ queue with arrival rate λ_j and service rate $\psi_j\mu_j$, where $\psi_j \in [1, n_j]$ is a function of n_j to be determined by the configuration of servers at each station ($j = 1, 2, \cdots, m$). It is non-decreasing and can be inverted, i.e. ψ^{-1} exists. As discussed in section 3.3, (1) suppose that a domain represents a group of CPUs in a model as discussed in [SHI 06]. Then, $\psi(n)$ can be seen as a CPU scaling factor for the number of CPUs from 1 to n, and then $\psi(n)$ can be expressed as $\psi(n) = \xi^{\log_2 n}$, where ξ is a basic scaling factor from CPU 1 to CPU 2. So, $\psi^{-1}(n) = \xi^{-\log_2 n}$. (2) Suppose that a domain represents a group of routers in a network model. Then, $\psi(n) = 1$ when these n routers are serially placed, and $\psi(n) = n$ when they are in parallel.

Since the queuing network is overtake-free, the waiting time of a customer at a station is independent of its waiting times at other stations (see [DAD 84] and [WAL 80]). Let D be the service time at the infinite server, X be the time elapsed from the moment a customer arrives at the trust manager server to the moment he/she departs from the server and X_j be the time elapsed from the moment a customer arrives at station j to the moment he/she departs from the station. Then, as in Chapter 3, the total response time can be calculated by

$$T = X + D + X_1 + X_2 + \cdots + X_m$$

and hence the Laplace–Stieltjes transform (LST) of the response time T is

$$L_T(s) = L_X(s)L_D(s)L_{X_1}(s)\cdots L_{X_m}(s) \quad [5.13]$$

where $L_X(s)$ and $L_D(s)$ are the LST of the service time D and X, respectively, given by

$$L_X(s) = \frac{\mu(1-\rho)}{s+\mu(1-\rho)}, \qquad L_D(s) = \frac{\lambda}{s+\lambda} \quad [5.14]$$

and $L_{X_j}(s)$ is the LST of the response time X_j at the jth station given by

$$L_{X_j}(s) = \frac{\psi(n_j)\mu_j(1-\rho_j)}{s+\psi(n_j)\mu_j(1-\rho_j)} \quad [5.15]$$

where $\rho = \frac{\lambda}{\mu}$ and $\rho_j = \frac{\lambda_j}{\psi(n_j)\mu_j}$ $(j = 1, 2, \cdots, m)$.

From [5.13]–[5.15] we have

$$L_T(s) = \frac{\mu(1-\rho)}{s+\mu(1-\rho)} \cdot \frac{\lambda}{s+\lambda} \times \Pi_{j=1}^{m} \frac{\psi(n_j)\mu_j(1-\rho_j)}{s+\psi(n_j)\mu_j(1-\rho_j)}$$

Then, we can obtain the probability and cumulative distributions of the response time by solving the following:

$$f_T(t) = L^{-1}\{L_T(s)\} \quad \text{and} \quad F_T(t) = L^{-1}\{\frac{L_T(s)}{s}\}$$

We observe that $f_T(t)$ and $F_T(t)$ are usually nonlinear functions of t and n_j. Hence, the resource optimization problem is an m-dimensional linear optimization problem subject to nonlinear constraints. In general, it is not easy to solve this problem. However, the complexity of the problem can be significantly reduced by requiring that the service rates of the queues in the service model in Figure 5.3 are all

equal. That is, we find the optimum value of $n_1, \cdots, n_m$ such that

$$\psi(n_1)\mu_1 = \cdots = \psi(n_m)\mu_m$$

From the traffic equations

$$\lambda = \lambda_j = \Lambda$$

for $j = 1, 2, \cdots, m$, we have the utilization of each station

$$\rho_j = \frac{\lambda_j}{\psi(n_j)\mu_j} = \frac{\Lambda}{\psi(n_j)\mu_j}$$

Thus, we have

$$\hat{a}_i = \psi(n_i)\mu_i(1-\rho_i) = \psi(n_j)\mu_j(1-\rho_j) = \hat{a}_j \stackrel{\triangle}{=} \hat{a}$$

which implies $n_j = \psi^{-1}(\frac{\hat{a}+\lambda_j}{\mu_j})$ $(i, j = 1, 2, \cdots, m)$. Hence, from [5.13] we have

$$f_T(t) = L^{-1}\{\frac{\mu(1-\rho)}{s+\mu(1-\rho)} \cdot \frac{\lambda}{s+\lambda} \cdot \frac{\hat{a}^m}{(s+\hat{a})^m}\}$$

and subsequently we obtain

$$F_T(t) = L^{-1}\{\frac{\mu(1-\rho)}{s+\mu(1-\rho)} \cdot \frac{\lambda}{s(s+\lambda)} \cdot \frac{\hat{a}^m}{(s+\hat{a})^m}\} \qquad [5.16]$$

As a result, $\sum_{j=1}^{m} n_j c_j$ reduces to a function of variable $\hat{a}$ due to

$$n_j = \lceil \psi^{-1}(\frac{\hat{a}+\lambda_j}{\mu_j}) \rceil$$

Then, we have the following algorithm for the resource optimization problem.

Moreover, the constraint of service availability at each resource site in [5.12] is rewritten as

$$G_j(\hat{a}) \stackrel{def}{=} P_j(\lceil \psi^{-1}(\frac{\hat{a}+\lambda_j}{\mu_j})\rceil, N_j) \quad [5.17]$$

At this time, $g(n_1, \cdots, n_m)$ in [5.10] reduces to a function of one variable. Thus, the trust-based resource provisioning problem is solved using the following iterative algorithm.

ALGORITHM 5.1.–

1) At time t_k, select m resource sites within a predefined trust index $\hat{I}_j$, and the highest m trust indices. If such a selection is impossible, then the trust manager informs the customer "We cannot process the service request at this moment". After the customer request has waited for more than a given threshold time, the trust manager still cannot find those service sites that meet the predefined trust index $\hat{I}_j$. Then, the trust manager informs the customer "You need to either revise the trust requirement or abort the request and resubmit it later" Otherwise, continue to do step 2.

2) Find $\hat{a}$ in the following two one-dimensional optimization problems.

i) The minimization problem of a percentile response time

$$\hat{a}^{(1)} \leftarrow \arg\min_{\hat{a}} F_T(t)|_{t=T^D}$$

subject to the constraint $F_T(t)|_{t=T^D} \geq \gamma\%$ at $\hat{a} = \hat{a}^{(1)}$, where $F_T(t)$ is given by [5.16].

ii) The minimization problem of service availability

$$\hat{a}_j^{(2)} \leftarrow \arg\min_{\hat{a}} G_j(\hat{a})$$

subject to the constraint $G_j(\hat{a}^{(2)}) \geq \delta_j\%$, where $G(\hat{a})$ is given by [5.17].

3) Compute integers n_j by using $n_j = \lceil \frac{\hat{a}_j^M}{\mu_j(1-\rho_j)} \rceil$, where $\hat{a}_j^M = \max(\hat{a}^{(1)}, \hat{a}_j^{(2)})$ for $1 \leq n_j \leq N_j$ and $j = 1, 2, \cdots, m$.

4) Update $I_j^{t_k}$ to $I_j^{t_k+T^D}$ based on the trust function [5.2]. If $|I_j^{t_k+T^D}| \geq \hat{I}_j$, then the completed service job is accepted. Otherwise, repeat steps 1–3.

5) When the repeated number is more than a predefined threshold, the trust manager notifies the customer "We need to renegotiate an SLA with the service broker".

As mentioned previously, algorithm 5.1 is an iteration algorithm. The number of iterations will be determined by the trust manager and the service broker. When the number of iterations is more than a predefined threshold, the trust manager will notify the customer to modify the SLA. The customer may abort the service request or resubmit the service request with a new QoS requirement with a new fee. In addition, the trust manager selects those service sites with the highest trust indices that also meet predefined trust requirements in step 1 in algorithm 5.1. But, for its own benefit, the trust manager can choose those service sites that meet trust requirements predefined by a customer.

Moreover, algorithm 5.1 shows that the m-dimensional optimization problem in subproblem 2 significantly reduces to a one-dimensional optimization problem. Surprisingly, the optimal number of servers obtained by algorithm 5.1 is independent of server costs c_j because $\hat{a}_i = \hat{a}_j$ $(i, j = 1, 2, \cdots, m)$. In general, each of the above two one-dimensional optimization problems can be solved numerically.

Note that the service broker cannot provide the QoS at the moment in which an SLA negotiation is requested. In this case, the service broker may get a service penalty. For presentation purposes, it will not be discussed further here

since the above approach can be easily adjusted to deal with this case. In addition, we have only considered a single-class customer. The above results can be extended to the case of two-class customers in next section.

5.4.2. *Multiple priority customers*

We can model those m resource sites as a queuing network as shown in Figure 5.4. Without loss of generality, we assume that the first m sites are chosen. In Figure 5.4, the tandem queuing network consists of a trust server and m stations numbered sequentially from 1 to m. The trust server represents the trust manager and each station represents a resource site. Each station carries out a particular function, such as it could be a database server, a computing server, a file server and a Web server. The execution procedure of a service request may be complicated in the real world, but the main idea of our proposed modeling approach can still be extended to describe any collaborative relationship among resource sites as long as the relationship can be quantified.

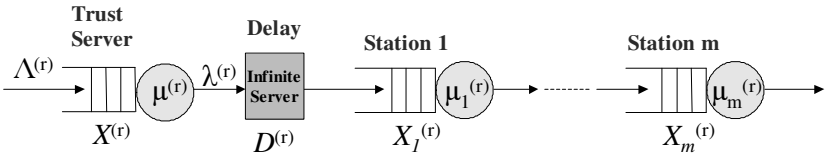


Figure 5.4. *A Web services computing system for multiple-class customer services*

In Figure 5.4, each station j is modeled as a single FIFO queue served by n_j identical servers, each providing a service at the rate $\mu_j^{(r)}$, where $r = 1, \cdots, R$. Let $\Lambda^{(r)}$ be the external arrival rate to the infinite server, and let $\lambda^{(r)}$ and $\lambda_j^{(r)}$ be the effective arrival rates to the infinite server and station j, $j = 1, 2, \cdots, m$. We assume that all service times are

exponentially distributed and the external arrival to the infinite server occurs in a Poisson fashion. The trust server provides a service at the rate $\mu^{(r)}$.

In Figure 5.4, the infinite server represents the total propagation delay from the user to the service provider and back to the user and also from station 1 to m. We only consider two classes of customers in this chapter. That is, $R = 2$.

In this chapter, we are interested in minimizing the overall cost of the above Web services computing system so that the desired SLA is guaranteed. Before presenting an algorithm for solving the minimization problem, we first need to calculate the response time of a customer's service request. Recall that the response time refers to the time elapsed from the moment a customer's service job joins the computing system to the time it departs. It is the most important QoS metric. The response time also reflects security behavior and the availability of services in some degree.

Let $T^{(r)}$ be a random variable that represents the response time of a service job with class r customers. Also, assume that $f_T^{(r)}(t)$ is the probability distribution of $T^{(r)}$. Denote the overall service cost [5.1] by

$$g(n_1^{(r)}, n_2^{(r)}, \cdots, n_m^{(r)}) = \sum_{j=1}^{m} n_j^{(r)} c_j \qquad [5.18]$$

where $n_1^{(r)}, n_2^{(r)}, \cdots, n_m^{(r)}$ are the number of servers allocated to a specific stream of jobs with arrival rate $\Lambda^{(r)}$ $(r = 1, 2)$.

Then, the trust-based resource provisioning problem can be formulated as the following three subproblems:

1) Select m resource sites within the predefined trust indices of all class customers at time $t = t_k$.

2) Solve for $n_j^{(r)}$ in the m-dimensional integer optimization problem:

$$\min_{n_1^{(r)},\cdots,n_m^{(r)}} g(n_1^{(r)}, n_2^{(r)}, \cdots, n_m^{(r)}) \qquad [5.19]$$

Under the constraint of a percentile response time of [5.5], and the constraint of service availability

$$P_j(n_j^{(r)}, N_j) \geq \zeta_j^{(r)}\% \qquad [5.20]$$

where $T_D^{(r)}$ is desired response time defined by class r customers and $\zeta_j^{(r)}\%$ is a desired percentage of service availability at site j. $P_j(n_j^{(r)}, N_j)$ is given by [5.9]. Then, calculate

$$n_j = \max\{n_j^{(1)}, n_j^{(2)}\} \quad (j = 1, 2, , \cdots, m)$$

3) Update the trust indices of all M sites based on the activity during the time interval $[t_k, t_k + T^D]$. Then, the trust manager decides if a completed job is accepted. If each trust index at those selected sites that complete the service job meet a predefined index value, then the completed job is accepted. Otherwise, the completed job is discarded and the trust manager needs to resubmit the job.

Note that in the model, the verifying time of a trust index is ignored since we are interested in the processing time of a job at resource sites. While subproblems 1 and 3 are solved at the customer side to ensure a customer service received from reliable resource sites, subproblem 2 is solved by the service broker who represents these resource sites. Hence, in the current model, the trustworthiness of resource sites in subproblems 1 and 3 is not considered as a constraint for the optimization problem in subproblem 2.

The above resource optimization problem as described in subproblem 2 of section 5.4 is to find a minimal cost

presented in [5.19] such that $\gamma^{(r)}\%$ of the time a customer's response time is less than a predefined value $T_D^{(r)}$, and $\zeta_j^{(r)}\%$ of time the selected resource sites have $n_j^{(r)}$ servers available for the job of class r ($r = 1, 2$).

In the following discussion, each station is modeled as a single $M/M/1$ priority queue with arrival rate $\lambda_j^{(r)}$ and service rate $\psi^{(r)}(n_j)\mu_j^{(r)}$, where $\psi^{(r)}(n_j)$ is a known function of n_j ($r = 1, 2$), and depends on the configuration of servers and the type of customers at each station. It is non-decreasing and can be inverted, i.e. $(\psi^{(r)})^{-1}$ exists ($r = 1, 2$). The detailed explanation of $(\psi^{(r)})^{-1}$ is given in section 4.3.2.

Since the queuing network is overtake-free, the waiting time of a customer at a station is independent of its waiting times at other stations (see [DAD 84] and [WAL 80]). Let $D^{(r)}$ be the service time at the infinite server, $X^{(r)}$ be the time elapsed from the moment a customer of class r arriving at the trust manager server to the moment it departs from the server and $X_j^{(r)}$ be the time elapsed from the moment a customer of class r arriving at station j to the moment it departs from the station. Then, the total response time is

$$T^{(r)} = X^{(r)} + D^{(r)} + X_1^{(r)} + X_2^{(r)} + \cdots + X_m^{(r)}$$

for the customer of class r and hence the LST of the response time $T^{(r)}$ is

$$L_{T^{(r)}}(s) = L_{X^{(r)}}(s)L_{D^{(r)}}(s)L_{X_1^{(r)}}(s)\cdots L_{X_m^{(r)}}(s) \qquad [5.21]$$

where $L_{D^{(r)}}(s)$ is the LST of the service time $D^{(r)}$ given by

$$L_{D^{(r)}}(s) = \frac{\lambda^{(r)}}{s + \lambda^{(r)}} \qquad [5.22]$$

Also, $L_{X^{(r)}}(s)$ is the LST of the response times $X^{(r)}$ at the trust server and $L_{X_j^{(r)}}(s)$ is the LST of the response time $X_j^{(r)}$ at the jth station, where $r = 1, 2$.

Because the preemptive-resume priority, the high-priority response time is same as in the single-class FIFO $M/M/1$ whose probability distribution can be expressed as [5.6]. Thus, $L_{X^{(1)}}(s)$ is determined by

$$L_{X^{(1)}}(s) = \frac{\mu(1-\rho^{(1)})}{s+\mu(1-\rho^{(1)})}, \quad (j = 1, 2, \cdots, m) \qquad [5.23]$$

and $L_{X^{(2)}}(s)$ is the LST of the low-priority response time given by

$$L_{X^{(2)}}(s) = \frac{(1-\rho)\delta^{(1)}}{1-\rho\,\delta^{(1)}}, \quad (j = 1, 2, \cdots, m) \qquad [5.24]$$

due to (25) in [XIO 06c], where $\mu \stackrel{\triangle}{=} \mu^{(1)} = \mu^{(2)}$, $\rho^{(r)} = \frac{\lambda^{(r)}}{\mu}$, $\rho = \frac{\Lambda^{(1)}+\Lambda^{(2)}}{\mu}$ and $\delta^{(1)}$ is given by

$$\delta^{(1)} = \frac{\eta - \sqrt{\eta^2 - 4\,\Lambda^{(1)}\,\mu^{(1)}}}{2\,\Lambda^{(1)}} \quad \text{and} \quad \eta = s + \Lambda^{(1)} + \mu^{(1)}$$

Furthermore, $L_{X_j^{(1)}}(s)$ is given by

$$L_{X_j^{(1)}}(s) = \frac{\psi^{(1)}(n_j)\mu_j(1-\rho_j^{(1)})}{s+\psi^{(1)}(n_j)\mu_j(1-\rho_j^{(1)})}, \quad (j = 1, 2, \cdots, m) \qquad [5.25]$$

and $L_{X_j^{(2)}}(s)$ is the LST of the low-priority response time given by

$$L_{X_j^{(2)}}(s) = \frac{(1-\rho_j)\delta_j^{(1)}}{1-\rho_j\,\delta_j^{(1)}}, \quad (j = 1, 2, \cdots, m) \qquad [5.26]$$

due to (25) in [XIO 06c], where $\rho_j^{(r)} = \frac{\lambda_j^{(r)}}{\psi^{(r)}(n_j)\mu_j}$,s $\rho_j = \frac{\lambda_j^{(1)}+\lambda_j^{(2)}}{\psi^{(r)}(n_j)\mu_j}$ and $\delta_j^{(1)}$ is given by

$$\delta_j^{(1)} = \frac{\eta_j - \sqrt{\eta_j^2 - 4\,\psi^{(1)}(n_j)\lambda_j^{(1)}\,\mu_j^{(1)}}}{2\,\psi^{(1)}(n_j)\lambda_j^{(1)}} \quad \text{and}$$

$$\eta_j = s + \lambda_j^{(1)} + \psi^{(1)}(n_j)\mu_j^{(1)}$$

for $j = 1, 2, \cdots, m$.

From [5.21]–[5.26], we have

$$L_{T^{(1)}}(s) = \frac{\lambda^{(1)}}{s+\lambda^{(1)}} \frac{\mu(1-\rho^{(1)})}{s+\mu(1-\rho^{(1)})} \Pi_{j=1}^{m} \frac{\psi^{(1)}(n_j)\mu_j(1-\rho_j^{(1)})}{s+\psi^{(1)}(n_j)\mu_j(1-\rho_j^{(1)})}$$

and

$$L_{T^{(2)}}(s) = \frac{\lambda^{(2)}}{s+\lambda^{(2)}} \frac{(1-\rho)\delta^{(1)}}{1-\rho\,\delta^{(1)}} \Pi_{j=1}^{m} \frac{(1-\rho_j)\delta_j^{(1)}}{1-\rho_j\,\delta_j^{(1)}}$$

We observe that $f_{T^{(r)}}(t)$ and $F_{T^{(r)}}(t)$ $(r = 1, 2)$ are usually nonlinear functions of t and n_j. Hence, the resource optimization problem is an n-dimensional linear optimization problem subject to nonlinear constraints. In general, it is not easy to solve this problem. However, the complexity of the problem can be significantly reduced by requiring that the service rates of the queues in the Web services model in Figure 5.4 are all equal. That is, we find the optimum value of $n_1, \cdots, n_m$ such that $\psi^{(r)}(n_1)\mu_1^{(r)} = \cdots = \psi^{(r)}(n_m)\mu_m^{(r)}$ $(r = 1, 2)$, i.e. *balanced utilization*.

From the traffic equations $\lambda^{(r)} = \lambda_j^{(r)} = \Lambda^{(r)}$ $(j = 1, 2, \cdots, m)$, we have the utilization of each station

$\rho_j^{(r)} = \frac{\lambda_j^{(r)}}{\psi^{(r)}(n_j)\mu_j^{(1)}} = \frac{\Lambda^{(r)}}{\psi^{(r)}(n_j)\mu_j^{(1)}}$. Thus, we have for the high-priority queue, $\hat{a}_i = \psi^{(1)}(n_i)\mu_i^{(1)}(1 - \rho_i^{(1)}) = \psi^{(1)}(n_j)\mu_j(1 - \rho_j^{(1)}) = \hat{a}_j \triangleq \hat{a}$ that implies $n_j = (\psi^{(1)})^{-1}(\frac{\hat{a}+\lambda_j^{(1)}}{\mu_j})$ $(i, j = 1, 2, \cdots, m)$. Hence, from [5.21] we have

$$f_{T^{(1)}}(t) = L^{-1}\{\frac{\lambda^{(1)}}{s + \lambda^{(1)}} \cdot \frac{\mu(1 - \rho^{(1)})}{s + \mu(1 - \rho^{(1)})} \cdot \frac{\hat{a}^m}{(s + \hat{a})^m}\}$$

and subsequently we obtain

$$F_{T^{(1)}}(t) = L^{-1}\{\frac{\lambda^{(1)}}{s(s + \lambda^{(1)})} \cdot \frac{\mu(1 - \rho^{(1)})}{s + \mu(1 - \rho^{(1)})} \cdot \frac{\hat{a}^m}{(s + \hat{a})^m}\} \qquad [5.27]$$

Moreover, since $\hat{a}_i = \hat{a}_j = \hat{a}$ and $\psi^{(1)}(n_i)\mu_i^{(1)} = \psi^{(2)}(n_i)\mu_i^{(2)}$ $(i, j = 1, 2, \cdots, m)$, we have $\psi^{(2)}(n_1)\mu_1^{(2)} = \cdots = \psi^{(2)}(n_m)\mu_m^{(2)} \triangleq \hat{b}$. It is easy to see that $\hat{b} = \hat{a} + \lambda_j^{(1)}$. Thus, for the low-priority queue, we obtain $\rho_{j_1} = \rho_{j_2} \triangleq \hat{\rho}$ and $\delta_{j_1}^{(1)} = \delta_{j_2}^{(1)} \triangleq \hat{\delta}^{(1)}$, for $j_1,\ j_2 = 1, 2, \cdots, m$. Furthermore, $L_{T^{(2)}}(s)$ reduces to

$$L_{T^{(2)}}(s) = \frac{\lambda^{(2)}}{s + \lambda^{(2)}} \cdot \frac{(1 - \rho)\delta^{(1)}}{1 - \rho\,\delta^{(1)}} \cdot \frac{(1 - \hat{\rho})^m(\hat{\delta}^{(1)})^m}{(1 - \hat{\rho}\,\hat{\delta}^{(1)})^m} \qquad [5.28]$$

which is a function of only one variable $\hat{b}$, i.e. variable $\hat{a}$. This is because $\hat{\rho}$ and $\hat{\delta}^{(1)}$ are considered as functions of only one variable $\hat{b}$, i.e. variable $\hat{a}$. Hence, we obtain the cumulative distribution of the response time for a low-priority customer

$$F_{T^{(2)}}(s) = L^{-1}\{\frac{\lambda^{(2)}}{s(s + \lambda^{(2)})} \cdot \frac{(1 - \rho)\delta^{(1)}}{1 - \rho\,\delta^{(1)}} \cdot \frac{(1 - \hat{\rho})^m(\hat{\delta}^{(1)})^m}{(1 - \hat{\rho}\,\hat{\delta}^{(1)})^m}\} \qquad [5.29]$$

Since it is required that $n_j^{(1)} = \lceil(\psi^{(1)})^{-1}(\frac{\hat{a}+\lambda_j^{(1)}}{\mu_j^{(1)}})\rceil$ for a high-priority customer and $n_j^{(2)} = \lceil(\psi^{(2)})^{-1}(\frac{\hat{b}}{\mu_j^{(2)}})\rceil$ for a

low-priority customer, the constraint of service availability at each resource site in [5.20] for a high-priority customer is rewritten as

$$G_j^{(1)}(\hat{a}) \stackrel{def}{=} P_j^{(1)}(\lceil(\psi^{(1)})^{-1}(\frac{\hat{a}+\lambda_j^{(1)}}{\mu_j^{(1)}})\rceil, N_j) \quad [5.30]$$

and the constraint of service availability at each resource site in [5.20] for a low-priority customer is rewritten as

$$\begin{aligned} G_j^{(2)}(\hat{a}) &\stackrel{def}{=} P_j^{(2)}(\lceil(\psi^{(2)})^{-1}(\frac{\hat{b}}{\mu_j^{(2)}})\rceil, N_j) \\ &= P_j^{(2)}(\lceil(\psi^{(2)})^{-1}(\frac{\hat{a}+\lambda_j^{(1)}}{\mu_j^{(2)}})\rceil, N_j) \end{aligned} \quad [5.31]$$

As a result, $\sum_{j=1}^{m} n_j c_j$ reduces to a function of variable $\hat{a}$ due to the relations $n_j^{(1)} = \lceil(\psi^{(1)})^{-1}(\frac{\hat{a}+\lambda_j^{(1)}}{\mu_j^{(1)}})\rceil$ for a high-priority customer and to a function of variable $\hat{b}$ due to $n_j^{(2)} = \lceil(\psi^{(2)})^{-1}(\frac{\hat{b}}{\mu_j^{(2)}})\rceil$ for a low-priority customer. That is, $g(n_1, \cdots, n_m)$ in [5.18] reduces to a function of one variable. Thus, the resource optimization problem can be divided into one-dimensional resource optimization problems for both high-priority and low-priority customers, respectively. Thus, the trust-based resource provisioning problem is solved using the following iterative algorithm.

ALGORITHM 5.2.–

1) At time t_k, select m resource sites within predefined trust indices $\hat{I}_j^{(r)}$, and the highest m trust indices. If such a selection is impossible, then the trust manager informs the customer "We cannot process the service request at this

moment". After the customer request has waited for more than a given threshold time, the trust manager still cannot find those service sites that meet the predefined trust indices $\hat{I}_j^{(r)}$. Then, the trust manager informs the customer "You need to either revise the trust requirement or abort the request and resubmit it later". Otherwise, continue to do step 2.

2) Find $\hat{a}$ in the following two one-dimensional optimization problems.

i) The minimization problem of a percentile response time for class r customers:

$$\hat{a}^{(r)}[1] \leftarrow \arg\min_{\hat{a}} F_{T^{(r)}}(t)|_{t=T_D^{(r)}}$$

subject to the constraint $F_{T^{(r)}}(t)|_{t=T_D^{(r)}} \geq \gamma^{(r)}\%$ at $\hat{a} = \hat{a}^{(r)}[1]$, where $F_{T^{(r)}}(t)$ are given by [5.27] and [5.29], respectively. Then, computing the number of servers $n_j^{(r)}[1]$ required for availability guarantee is given by

$$n_j^{(r)}[1] = \lceil (\psi^{(r)})^{-1}(\frac{\hat{a}_j^{(r)}[1] + \lambda_j^{(1)}}{\mu_j^{(r)}}) \rceil$$

for $r = 1, 2$, and $j = 1, 2, \cdots, m$.

ii) The minimization problem of service availability

$$\hat{a}_j^{(r)}[2] \leftarrow \arg\min_{\hat{a}} G_j^{(r)}(\hat{a})$$

subject to the constraint $G_j^{(r)}(\hat{a}^{(r)}[2]) \geq \zeta_j^{(r)}\%$, where $G^{(r)}(\hat{a})$ are given by [5.30] and [5.31], respectively. Then, computing the number of servers $n_j^{(r)}[2]$ required for availability guarantee is given by

$$n_j^{(r)}[2] = \lceil (\psi^{(r)})^{-1}(\frac{\hat{a}_j^{(r)}[2] + \lambda_j^{(1)}}{\mu_j^{(r)}}) \rceil$$

for $r = 1, 2$, and $j = 1, 2, \cdots, m$.

3) Compute integers $n_j^{(r)}$ by using

$$n_j^{(r)} = \max\{n_j^{(r)}[1],\ n_j^{(r)}[2]\}$$

Thus, the number of servers required is $n_j = \max\{n_j^{(1)},\ n_j^{(2)}\}$ for station j $(j = 1, 2, \cdots, m)$.

4) Update $I_j^{t_k}$ to $I_j^{t_k+T_D^{(r)}}$ based on the trust function [5.2]. If $|I_j^{t_k+T_D^{(r)}}| \geq \hat{I}_j^{(r)}$, then the completed service job for class r customers is accepted. Otherwise, repeat steps 1–3.

5) When the repeated number is more than a predefined threshold, the trust manager notifies the customer "We need to renegotiate an SLA with the service broker".

As mentioned previously, algorithm 5.2 is an iteration algorithm. The number of iterations will be determined by the trust manager and the service broker. When the number of iterations is more than a predefined threshold, the trust manager will notify the customer to modify the SLA. The customer may abort the service request or resubmit the service request with a new QoS requirement with a new fee. In addition, the trust manager selects those service sites with the highest trust indices that also meet predefined trust requirements in step 1 in algorithm 5.2. But, for its own benefit, the trust manager can choose those service sites that only meet trust requirements predefined by a customer.

Moreover, algorithm 5.2 shows that the m-dimensional optimization problem in subproblem 2 significantly reduces to a one-dimensional optimization problem. Surprisingly, the optimal number of servers obtained by algorithm 5.1 is independent of server costs c_j because $\hat{a}_i = \hat{a}_j$ $(i, j = 1, 2, \cdots, m)$. In general, each of the above two one-dimensional optimization problems can be solved numerically.

Note that the service broker cannot provide the QoS at the moment in which an SLA negotiation is requested. In this case, the service broker may get a service penalty. For presentation purposes, it will not be discussed further here since the above approach can be easily adjusted to deal with this case. In addition, we have only considered two-class customers. The above results can be extended to the case of customers with more than two classes.

5.5. Numerical examples

In this section, we demonstrate how to apply our algorithm to solve the trust-based resource provisioning problem. We consider the cases of single customer services and multiple priority customer services, respectively, below.

5.5.1. *Single-class customers*

Let us first consider a 10 resource sites example modeled by the tandem queuing network presented in section 3.3. We choose $m = 7$, $\xi = 0.6$ and the trust index at time t_1 is listed in Table 5.1.

Station	1	2	3	4	5	6	7	8	9	10
I^{t_1}	0.8	0.5	0.2	0.9	0.6	0.4	0.8	0.7	0.3	0.6

Table 5.1. *The initial trust index of the ten stations for single-class customers*

Assume that $r_j(k)$ is uniformly distributed in [0.75, 1]. r_i and r_j are independent for any $i \neq j$ $(i, j = 1, \cdots, 10)$. Furthermore, we choose $\lambda = 100$, $T^D = 0.16$, $\gamma = 97.5$, $\mu = 200$ and the service rates of these 10 stations are given in Table 5.2. We also choose $c_j = 2$, $N_j = 200$, $\psi(n_j) = 1.5^{\log_2 n_j}$, $\delta_j = 99.999$, $\hat{I}_j = 0.9$ $(j = 1, \cdots, 10)$ and the server unavailable rates of these 10 stations are listed in Table 5.3.

Service rates	μ_1	μ_2	μ_3	μ_4	μ_5	μ_6	μ_7	μ_8	μ_9	μ_{10}
Values	28	18	80	35	39	41	15	25	82	35

Table 5.2. *The service rates of the ten stations for single-class customers*

Unavailable rates	η_1	η_2	η_3	η_4	η_5	η_6	η_7	η_8	η_9	η_{10}
Values	0.012	0.01	0.005	0.04	0.015	0.03	0.02	0.045	0.008	0.01

Table 5.3. *The server unavailability rates of the 10 stations for single-class customers*

A customer submits a service job at time t_5 with $t_{k+1} - t_k = 0.01$ ($k = 1, 2 \cdots$) and it requires two of these five resource sites satisfying the predefined $\hat{I}_j$ for processing the job.

First, we generated I^{t_k} (k = 2,$\cdots$,5,$\cdots, 21$) in Matlab as presented in Table 5.4. As we see, sites 1, 2, 3, 4, 6, 7, 8 and 10 meet the trust requirement at $t = t_5$. Thus, sites 2, 3, 4, 6, 7, 8 and 10 are selected because they have the highest seven trust indices.

	Station									
	1	2	3	4	5	6	7	8	9	10
I^{t_2}	0.8625	0.6412	0.5709	0.9156	0.7908	0.6651	0.8317	0.7892	0.3205	0.7428
I^{t_3}	0.8761	0.8190	0.7925	0.8501	0.8335	0.7698	0.8918	0.8593	0.4846	0.8865
I^{t_4}	0.8420	0.8532	0.8209	0.9014	0.7961	0.8725	0.9016	0.8946	0.4592	0.9132
I^{t_5}	0.9005	0.9129	0.9314	0.9248	0.8552	0.9153	0.9421	0.9189	0.5828	0.9588
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$I^{t_{20}}$	0.8728	0.9215	0.9338	0.9182	0.8736	0.9326	0.9419	0.9546	0.6223	0.9235
$I^{t_{21}}$	0.8812	0.9513	0.9602	0.9278	0.9046	0.9462	0.9392	0.9698	0.8588	0.9351

Table 5.4. *The trust indices of the 10 stations for single-class customers at times* $t = t_2, \cdots, t_5, t_{20}, t_{21}$

Then, we simulated the model in Arena 10.01. The simulation results are considered as "exact" since the simulation model is an exact representation of the queuing network under study. We further implemented [5.16] by using

the inverse Laplace transform method in [GRA 01], which is an approximate solution. The obtained cumulative distributions are presented in Table 5.5. It can be seen that our approximation result has a good accuracy, where the relative error in Table 5.5 was calculated by

$$\text{Relative error } \% = \frac{\text{Approximation result} - \text{Simulation result}}{\text{Simulation result}} \times 100$$

as given in [3.11]. It is used to measure the accuracy of the approximate results compared to model simulation results.

Response time	Simul	Approx	R-Err %
0.04	0.0213	0.0214	0.4393
0.06	0.1517	0.1528	0.7004
0.08	0.4070	0.4075	0.1112
0.10	0.6681	0.6672	−0.1377
0.12	0.8468	0.8450	−0.2158
0.14	0.9398	0.9379	−0.1974
0.16	0.9785	0.9780	−0.0498
0.18	0.9931	0.9929	−0.0157
0.20	0.9979	0.9979	0.0000
0.22	0.9995	0.9994	−0.0077
0.24	0.9999	0.9998	−0.0051
0.26	1.0000	1.0000	0.0000

Table 5.5. *The cumulative distribution of the response time for single-class customers*

Moreover, we obtained that $\hat{a}^{(1)} = 100$ and the optimal number of servers required for 97.5% of the response time to be less than $T^D = 0.16$ is presented in Table 5.6. The exact optimal number of servers, obtained by exhaustive search using the simulation model, and assuming that each station has the same utilization, or balanced utilization, is consistent with the ones presented in Table 5.6. We validated that they are consistent with the result obtained by a brute force

search using the simulation model in Arena, and assuming that each station has the same utilization or balanced utilization.

Station	2	3	4	6	7	8	10
#Servers necessary to ensure 97.5% response time	62	5	20	16	84	35	20
#Servers necessary to ensure 99.999% availability	10	7	22	18	15	23	10
#Servers necessary to ensure these two SLA metrics	62	7	22	18	84	35	20

Table 5.6. *The optimal number of servers for single-class customers*

Station	1	2	3	4	5	6	7	8	9	10
I^{t_1}	0.8	0.5	0.2	0.9	0.6	0.4	0.8	0.7	0.3	0.6

Table 5.7. *The initial trust index of the 10 stations for priority-class customers*

Furthermore, we obtain the number of servers required for 99.999% service availability in these seven stations using step 2 (b) of algorithm 5.1, as presented in Table 5.6. By doing the calculation in step 3 of algorithm 5.2, we obtained the numbers of servers required for the response time and service availability guarantees at these seven stations, respectively.

Finally, the trust manager needs to determine whether to accept the job completed by sites 2, 3, 4, 6, 7, 8 and 10 when it receives the completed job at $t = t_5 + T^D = t_5 + 0.16 = t_{21}$. As can be seen, these stations meet the trust requirement at $t = t_{21}$ and consequently the job is accepted.

The above ten-site example has demonstrated how to apply our approach to solve the trust-based resource provisioning problem in the case of single-class customers.

Next, we are going to demonstrate how to use our proposed algorithm to solve the trust-based resource provisioning problem in the case of multiple-class customers.

5.5.2. *Multiple priority customers*

In the case of multiple-class customers, we still consider a ten resource site example modeled by the tandem queuing network presented in section 5.4. We choose $m = 7$, $\xi = 0.6$ and the trust index at time t_1 is unchanged, i.e. it is listed in Table 5.1.

Again, assume that $r_j(k)$ is uniformly distributed in [0.75, 1]. r_i and r_j are independent for any $i \neq j$ $(i, j = 1, \cdots, 10)$. Moreover, we choose $\Lambda^{(1)} = 100$, $\Lambda^{(2)} = 50$, $T_D^{(1)} = 0.16$, $T_D^{(2)} = 0.8$, $\gamma^{(1)} = 97.5$, $\gamma^{(2)} = 99$, $\mu = 200$ and the service rates of these seven stations are given in Table 5.8. We also choose $c_j = 1$, $N_j = 200$, $\psi(n_j) = 1.55^{\log_2 n_j}$, $\xi_j^{(1)} = 99.999$, $\xi_j^{(2)} = 99.9$, $\hat{I}_j^{(1)} = 0.9$, $\hat{I}_j^{(2)} = 0.91$ $(j = 1, \cdots, 10)$ and the server unavailable rates of these 10 stations are listed in Table 5.9.

Service rates	μ_1	μ_2	μ_3	μ_4	μ_5	μ_6	μ_7	μ_8	μ_9	μ_{10}
$\mu_j^{(1)}$	88	18	85	32	31	49	24	28	21	38
$\mu_j^{(2)}$	96	15	60	25	55	41	18	26	31	35

Table 5.8. *The service rates of the 10 stations for priority-class customers*

Unavailable rates	η_1	η_2	η_3	η_4	η_5	η_6	η_7	η_8	η_9	η_{10}
Values	0.012	0.01	0.005	0.04	0.015	0.03	0.02	0.045	0.008	0.01

Table 5.9. *The server unavailability rates of the 10 stations for priority-class customers*

Again, a customer submits a service job at time t_5 with $t_{k+1} - t_k = 0.01$ $(k = 1, 2 \cdots)$ and it requires two of these five resource sites satisfying the predefined $\hat{I}_j$ for processing the job.

We first extended the generation of trust indices at the 10 stations. That is, we generated I^{t_k} ($k = 2, \cdots, 5, \cdots, 21, \cdots, 84, 85$) in Matlab presented in Table 5.10, as an extension of Table 5.4. As can be seen, while sites 1, 2, 3, 4, 6, 7, 8 and 10 meet the trust requirement for high-priority customers at $t = t_5$, only sites 2, 3, 4, 6, 7, 8 and 10 meet the trust requirement for low-priority customers at $t = t_5$. Thus, sites 2, 3, 4, 6, 7, 8 and 10 are selected because they meet both trust requirements.

	Station									
	1	2	3	4	5	6	7	8	9	10
I^{t_2}	0.8625	0.6412	0.5709	0.9156	0.7908	0.6651	0.8317	0.7892	0.3205	0.7428
I^{t_3}	0.8761	0.8190	0.7925	0.8501	0.8335	0.7698	0.8918	0.8593	0.4846	0.8865
I^{t_4}	0.8420	0.8532	0.8209	0.9014	0.7961	0.8725	0.9016	0.8946	0.4592	0.9132
I^{t_5}	0.9005	0.9129	0.9314	0.9248	0.8552	0.9153	0.9421	0.9189	0.5828	0.9588
...	...	...	...	...	...	...	...	...	...	...
$I^{t_{20}}$	0.8728	0.9215	0.9338	0.9182	0.8736	0.9326	0.9419	0.9546	0.6223	0.9235
$I^{t_{21}}$	0.8812	0.9513	0.9602	0.9278	0.9046	0.9462	0.9392	0.9698	0.8588	0.9351
...	...	...	...	...	...	...	...	...	...	...
$I^{t_{84}}$	0.9126	0.9318	0.9458	0.9288	0.8538	0.9225	0.9518	0.9449	0.7256	0.9338
$I^{t_{85}}$	0.8918	0.9815	0.9731	0.9388	0.9126	0.9565	0.9291	0.9588	0.8528	0.9588

Table 5.10. *The trust indices of the 10 stations for priority-class customers at times* $t = t_2, \cdots, t_5, \cdots, t_{20}, t_{21}, \cdots, t_{84}, t_{85}$

Then, we simulated the model in Arena 10.01. The simulation results are considered as "exact" since the simulation model is an exact representation of the queuing network under study. We further implemented [5.27] by using the inverse Laplace transform method in [GRA 01], which is an approximate solution. Tables 5.11 and 5.12 show the simulated and approximate cumulative distributions of the high-priority and low-priority response times, respectively. It appears that the results obtained by algorithm 5.2 are very accurate. Table 5.13 presents that the optimal number of servers is required for 97.5% of the high-priority response time to be less than $T_D^{(1)} = 0.16$ and for 99% of the low-priority response time to be less than $T_D^{(2)} = 0.8$. As can be seen in

Table 5.11, our approximation result has a good accuracy, where the relative error was calculated by [3.11]. Again, the relative error is used to measure the accuracy of the approximate results compared to model simulation results.

Response time	Simul	Approx	R-Err %
0.02	0.0002	0.0002	0.0000
0.04	0.0214	0.0214	0.0000
0.06	0.1542	0.1528	−0.9079
0.08	0.4085	0.4075	−0.2448
0.10	0.6691	0.6672	−0.2840
0.12	0.8459	0.8450	−0.1064
0.14	0.9385	0.9379	−0.0639
0.16	0.9781	0.9780	−0.0102
0.18	0.9931	0.9929	−0.0201
0.20	0.9980	0.9979	−0.0100
0.22	0.9995	0.9994	−0.0100
0.24	0.9998	0.9998	0.0000
0.26	0.9999	1.0000	0.0100
0.28	1.0000	1.0000	0.0000

Table 5.11. *The cumulative distribution of the high-priority response time*

Furthermore, the optimal number of servers required for satisfying both the high-priority and the low-priority response times is presented in Table 5.13. The exact optimal number of servers, obtained by exhaustive search using the simulation model, and assuming that each station has balanced utilization is consistent with the ones presented in Table 5.13. We validated that they are consistent with the result obtained by a brute force search using the simulation model in Arena, and assuming that each station has the same utilization or balanced utilization.

In addition, we obtain the number of servers required for 99.999% service availability guarantee for high-priority

customers and for 99.9% service availability guarantee for low-priority customers in these seven stations using step 2(ii) of algorithm 5.2, as presented in Table 5.13. By doing the calculation in step 3 of algorithm 5.2, we obtained the number of servers required for the response time and service availability guarantees at these seven stations, respectively.

Response time	Simul	Approx	R-Err %
0.2	0.1767	0.1473	-16.6384
0.3	0.4538	0.4396	-3.1291
0.4	0.6982	0.7082	1.4323
0.5	0.8541	0.8719	2.0841
0.6	0.9389	0.9503	1.2142
0.7	0.9774	0.9824	0.5116
0.8	0.9925	0.9942	0.1713
0.9	0.9978	0.9982	0.0401
1.0	0.9993	0.9995	0.0200
1.1	0.9998	0.9999	0.0100
1.2	1.0000	1.0000	0.0000

Table 5.12. *The cumulative distribution of the low-priority response time for priority-class customers*

The number of servers	Stations						
Metrics	2	3	4	6	7	8	10
97.5% Response time guarantee for high-priority customers	62	5	23	12	38	29	18
99% Response time guarantee for low-priority customers	61	7	27	13	46	26	16
99.999% Service availability guarantee for high-priority customers	10	7	22	18	15	23	10
99.9% Service availability guarantee for low-priority customers	7	5	14	15	11	19	7
#Servers necessary to ensure all these SLA metrics	62	7	27	18	46	29	18

Table 5.13. *The optimal number of servers for priority-class customers*

Finally, the trust manager needs to determine whether to accept the job completed by sites 2, 3, 4, 6, 7, 8 and 10 when it receives the completed job at $t = t_5 + T_D^{(1)} = t_5 + 0.16 = t_{21}$ for high-priority customers and at $t = t_5 + T_D^{(2)} = t_5 + 0.8 = t_{85}$ for low-priority customers, respectively. As can be seen, these stations meet the trust requirements at $t = t_{21}$ for high-priority customers and $t = t_{85}$ for low-priority customers, and consequently the completed jobs are accepted.

We have demonstrated how to apply our approach to solve the trust-based resource provisioning problem in the above ten-site example for the case of multiple-class customers.

As we know, most existing Web services products do not support an SLA that guarantees a level of service delivered to a customer for a given price. It is not easy to solve a resource provisioning problem when we consider all the following constraints: trustworthiness, an end-to-end response time and service availability. The last two sections demonstrated how to apply our efficient algorithm to solve the trust-based resource provisioning problem by using the above two illustrative examples in the cases of single- and multiple-class customers, respectively.

5.6. Concluding remarks

In this chapter, we have proposed a trust-based Web services model and provided a framework for studying the model. We have discussed a trust-based resource provisioning problem that arises in these typical Web services applications. The problem has been constructed by minimizing the total cost of service providers while satisfying SLA guarantees. We have further formulated it as an optimization problem under the constraints of percentile response time and service availability in the cases of

single-class customers and multiple-class customers, respectively.

In our approach, we considered the percentile response time that may better address a customer's concern as opposed to the average response time that is commonly used in the literature. We obtained an efficient and accurate numerical solution for calculating the percentile response time. Then, we proposed efficient approaches for solving the trust-based resource provisioning problem in the cases of single-class customers and multiple-class customers, respectively. We used two numerical examples to illustrate the use of our proposed algorithms in these two cases. Our numerical validations showed that our algorithms have provided a good accuracy.

Chapter 6

Performance Analysis of Public-Key Cryptography-Based Group Authentication

Several Kerberos-based authentication techniques using public-key cryptography have been proposed in the last decade. The purpose of using public-key cryptography is to eliminate the problem of a single point failure in a key distribution center (KDC), and achieve better scalability. Among them, the two notable techniques are public-key cryptography for cross-realm authentication in Kerberos (PKCROSS) and public key utilizing tickets for application servers (PKTAPP, also known as KX.509/KCA). The latter was proposed to improve the scalability of the former. However, the actual costs (e.g. computational and communication times) associated with these techniques have been poorly understood so far. It remains unknown which technique performs better in a large network where there are multiple KDC remote realms.

This chapter is organized as follows. An overview of public-key cryptography-based authentication is given in

section 6.1. In section 6.2, we first give an in-depth discussion of PKCROSS and PKTAPP and then present their performance evaluation using queuing theory. Section 6.3 presents a new public-key cryptography-based group authentication technique along with the details of message flows. We give a performance analysis of the proposed technique and compare it to PKCROSS and PKTAPP in section 6.4. Finally, we conclude our discussion in section 6.5.

6.1. Public-key cryptography-based authentication

In the last few years, we have witnessed an explosive growth in the usage of Internet collaborative applications, such as video and audio conferencing, replicated servers and databases, and in particular Web services whereby service components from several universal service providers can be flexibly integrated into a composite service regardless of their location, platform and execution speed [BAR 03]. To ensure quality-of-service, the service providers are required to work together. The rapid growth in collaborative applications has heightened the need for a reliable group communication system. In turn, it is impossible for the system to be reliable without group authentication.

Kerberos, [KOH 93] consisting of a client, an application server and a KDC, is a mature, reliable and secure network authentication protocol that allows a client to prove its identity to a server without sending confidential data across the network. Public-key cryptography has been extended to support Kerberos since it simplifies the distribution of keys in Kerberos. It eliminates a single point of failure. Integrating public-key cryptography into Kerberos represents the enhancements of the current Kerberos standard. Several Kerberos-based authentication techniques using public-key cryptography have been proposed in the last decade. Among them are PKCROSS [HUR 01] and public-key cryptography

for initial authentication in Kerberos (PKINIT) [ZHU 06]. Moreover, the scalability of network security infrastructures is becoming a serious concern as the explosive growth of collaborative applications such as Web services continues unabated. Public-key-based Kerberos for distribution authentication (PKDA) [SIR 97] and PKTAPP (also known as KX.509/KCA) [KX 07], [MED 97] have been proposed to enhance the security and scalability of Kerberos.

PKINIT is a core specification among these Internet drafts. Both PKCROSS and PKTAPP use variations of PKINIT message types and data structures for integrating public-key cryptography with Kerberos in different authentication stages. PKTAPP was originally introduced as PKDA. It implemented PKDA using the message formats and exchanges of PKINIT. Microsoft has adopted the Internet draft specification of PKINIT for the support of public-key cryptography in the Windows 2000 and 2003 implementations of Kerberos [GAR 03]. It has its own protocol that is the equivalent of KX509/KCA. There were preliminary discussions between the Kerberos WG and Microsoft about using a common protocol. The Massachusetts Institute of Technology (MIT) Kerberos consortium will drive these discussions [ALT 07b]. According to Altman [ALT 07a], the standardization of PKCROSS and PKTAPP will be standardized next for Kerberos and PKI integration. The Kerberos consortium at MIT was formed in September 2007 and listed PKCROSS as Project 10 in its proposal [CON 07]. We believe that PKCROSS and PKTAPP will be revived soon. Hence, this research only considers these two notable techniques: PKCROSS and PKTAPP.

It has been argued that PKCROSS would not scale well in large networks in [SIR 97]. PKTAPP was proposed to improve the scalability of PKCROSS. However, the actual costs associated with these techniques have been *poorly*

understood so far. PKTAPP has been shown in [HAR 01] to perform poorly when there are two or more application servers in one remote realm. In addition, it remains *unknown* as to which technique performs better in a large network where, as is typical in many applications, application servers are within *multiple* KDC remote realms.

In the next section, we first present a thorough performance evaluation of PKCROSS and PKTAPP in terms of computational and communication costs. Then, we demonstrate their performance difference using open queuing networks.

6.2. PKCROSS and PKTAPP

Kerberos [KOH 93] was developed at MIT in 1988. It is a network authentication protocol for providing a secure communication between a user workstation (also called a client) and application servers. The latest version of Kerberos is Version 5. It divides the world into realms, each with user workstations, a single primary KDC, backup KDCs and application servers in which the KDC is a trusted intermediary. In the Kerberos protocol, the client engages in a multiple-step authentication to obtain access to the application server whereby the client first obtains a relatively short-lived credential, a ticket-granting ticket (TGT) from the authentication service running on a KDC, and then obtains a session ticket for a particular application server by presenting the TGT to a centralized ticket-granting service (TGS) running on the KDC. The client presents the session ticket to the application server for authenticating herself/himself by showing knowledge of a secret session key. The secret session key was securely passed to the client by the KDC. Kerberos is stateless, and this is extremely valuable from the scalability point of view. Cross-realm

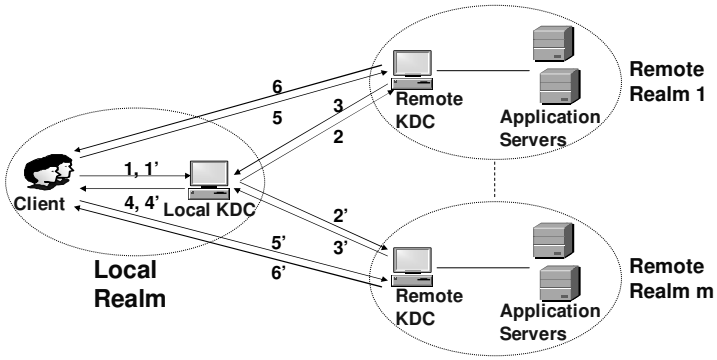
authentication is necessary when a client and an application server with different network domains fall into different Kerberos realms.

It is well known that a public-key security system is easier to administer, more secure, less trustful and more scalable than a symmetric-key security system. Public-key security does not use a trusted key management infrastructure since the burden of key management falls on public-key clients [DAV 96]. In a public-key infrastructure, public-key clients need to constantly and vigorously check the validity of the public keys that they use. Public-key cryptography shifts the burden of key management from the KDC/TGS in Kerberos to its certificate authority (CA) that may be considered as a trusted intermediary. CA issues a public-key certificate that is relatively long-lived credential. The burden is determined by the number of times clients want to authenticate to application servers in Kerberos. It might not be affordable in time-sensitive applications if one large-scale PKI deployment is needed for group authentication in a large network.

6.2.1. *Protocol analysis*

PKCROSS [HUR 01] is one notable protocol of integrating public-key cryptography with Kerberos to address the problem of network authentication among the client and the application servers in a large number of realms. Figure 6.1 illustrates the authentication steps of PKCROSS. The cross-realm KDC-to-KDC authentication is achieved by using public-key cryptography. First, just like Kerberos, the KDC in PKCROSS is burdened by the need to constantly renew short-lived TGTs, and TGS must be involved whenever a client wants to request a service from an application server. Thus, if a large number of users in a single realm request services, which may be a typical case in Web services, the KDC in the realm becomes a serious single point of failure

due to a KDC compromise and possibly a performance bottleneck. Recovering from such a compromise requires the reestablishment of secret keys for all users within this realm. When the number of users is large, such a recovery is very costly. PKINIT facilitates public-key-based authentication between users and their local KDC. Hence, one possible solution of eliminating the single point of failure is to combine PKCROSS with PKINIT, and thus public-key-based authentication becomes integrated throughout the *entire* Kerberos environment. However, it is easy to see that this solution is not feasible for time-sensitive applications, since the users might suffer from a long delay or even denial of services due to a high computational cost for the calculation of a public key used in the entire environment.

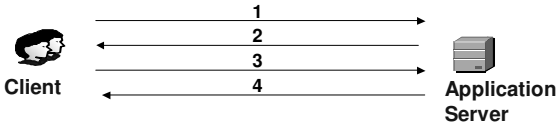


1 and 1'. Traditional cross-realm request for TGT to access a remote realm
2 and 2'. AS_REQ with PKCROSS flag set
3 and 3'. AS_REP with PKCROSS ticket encrypted using local KDC's public key
4 and 4'. Traditional reply with cross-realm TGT using SKC
5 and 5'. Ticket-Granting Service Request (TGS-REQ)
6 and 6'. Ticket-Granting Service Reply (TGS-REP)

Figure 6.1. *The PKCROSS message flow*

Second, in PKCROSS, the local KDC of the client issues all short-lived TGTs and all session tickets in its realm, and communicates with a remote KDC. Hence, it can easily

become a performance bottleneck since all these authentication transactions have to transit the KDC. Thus, PKTAPP [MED 97] has been proposed to address the issue. Figure 6.2 shows the message exchange of PKTAPP. The PKTAPP technique allows the client to communicate directly with the application servers so that the number of messages between them is reduced in an authentication process. But, as can be seen in [HAR 01], even though PKTAPP requires fewer message exchanges than PKCROSS during client authentication with remote application servers, PKCROSS outperforms PKTAPP when two or more application servers are in a single remote realm. This is because PKTAPP requires more public-key message exchanges than PKCROSS, which requires only one pair of public-key message exchanges. In the following, we study their performance in multiple remote realms.



1. A standard Kerberos version 5 message that includes PA-PK-AS-REQ pre-authentication field
2. A standard Kerberos version 5 message that includes PA-PK-AS-REP pre-authentication field
3. AS_REQ in which the client requests a service from the application server, using the traditional secret-key procedure defined in RFC1510
4. AS_REQ in which the client receives a reply from the application server, using the traditional secret-key procedure defined in RFC1510

Figure 6.2. *The PKTAPP message flow*

Table 6.1 summarizes the encryption and decryption operations performed in PKTAPP and PKCROSS, where m and n are the number of remote realms and the number of applications servers within these realms, respectively. Denote by c_1, c_2 and c_3 the computational times per secret-, private- or public-key operation, respectively. Then, $c_1 < c_2 < c_3$. Let $f_j(n, m)$, $j = 1, 2$, be the total computational times of

encryption and decryption operations in PKTAPP and PKCROSS. Then, from Table 6.1, we have:

$$f_1(n,m) = (5c_1 + 2c_2 + 7c_3)n + 2c_1 + c_2$$
$$f_2(n,m) = 4c_1 n + (4c_1 + 3c_2 + 7c_3)m + 11c_1 + c_2$$

Protocols	Entities	Number of secret keys	Number of private keys	Number of public keys
PKTAPP	Client	$2n$+1	n+1	$3n$
	Application server	$3n$+1	n	$4n$
PKCROSS	Total	$5n$+2	$2n$+1	$7n$
	Client	m+6	0	0
	Local KDC	5	m+1	$3m$
	Remote KDC	$3m$+n	m	$4m$
	Application server	$3n$	0	0
	Total	4(n+m)+11	$2m$+1	$7m$

Table 6.1. *The operations of encryption and decryption when n application servers are in m remote realms*

Note that $f_1(n,m)$ does not depend on m, which can be hence abbreviated as $f_1(n)$. Then, we have the following proposition.

PROPOSITION 6.1.– For each authentication, PKCROSS requires less computational time than PKTAPP if and only if the number of application servers n is more than $\lceil m + \frac{3(m+3)c_1}{c_1+2c_2+7c_3} \rceil$, or the number of remote realms m is less than the integer: $\lfloor n - \frac{3(n+3)c_1}{4c_1+2c_2+7c_3} \rfloor$.

PROOF.– According to $f_1(n,m)$ and $f_2(n,m)$, their difference is given by $f_1(n,m) - f_2(n,m) = -9c_1 + (c_1 + 2c_2 + 7c_3)n - (4c_1 + 2c_2 + 7c_3)m$. Hence, $f_1(n,m) - f_2(n,m) \geq 0$ if and only if $-9c_1 + (c_1 + 2c_2 + 7c_3)n - (4c_1 + 2c_2 + 7c_3)m \geq 0$, which implies proposition 6.1. The proof is complete.

It is anticipated that the client is connected to the local KDC by a local area network (LAN), the client and the local KDC are connected to a remote KDC by a wide area network (WAN), and a remote KDC and its application servers are connected by a WAN. Assume that all WANs have an identical communication time and a LAN has a negligible communication time compared to a WAN. From Figures 6.1 and 6.2, we note that the number of WAN communications are $4n$ for PKTAPP and $4m + 2n$ for PKCROSS. Let $g_j(n, m)$, $j = 1, 2$, be the transaction times of PKTAPP and PKCROSS. *The transaction time* is defined as the computational time of the total encryption and decryption operations plus the communication time per authentication in a technique. Also, denote by d the time spent in a WAN communication. Then, the following conclusion holds.

PROPOSITION 6.2.– For each authentication, PKCROSS uses less transaction time than PKTAPP if and only if the number of application servers n is more than the integer: $m + \lceil \frac{(3c_1+2d)m+9c_1}{c_1+2c_2+7c_3+2d} \rceil$.

PROOF.– For $j = 1, 2$, $g_j(n, m)$ can be computed by using $g_1(n, m) = f_1(n) + 4n$ and $g_2(n, m) = f_2(n, m) + 4m + 2n$ and thus their difference is given by $g_1(n, m) - g_2(n, m) = (c_1 + 2c_2 + 7c_3 + 2d)n - (4c_1 + 2c_2 + 7c_3 + 4d)m - 9c_1$, which easily derives proposition 6.2. The proof is complete.

Note that if $n = 1$ (so, $m = 1$), $m + \lceil \frac{(3c_1+2d)m+9c_1}{c_1+2c_2+7c_3+2d} \rceil > 1 = n$, and if $m = 1$, $m + \lceil \frac{(3c_1+2d)m+9c_1}{c_1+2c_2+7c_3+2d} \rceil = 1 + \frac{12c_1+2d}{c_1+2c_2+7c_3+2d} < 2$ since c_1 is significantly smaller than c_3. We have the following corollary:

COROLLARY 6.1.– When $m = 1$, we have that PKTAPP requires less transaction time than PKCROSS if $n = 1$ but more transaction time than PKCROSS if $n \geq 2$.

In proposition 6.2, we have shown that the number of application servers should be $\lceil \frac{(3c_1+2d)m+9c_1}{c_1+2c_2+7c_3+2d} \rceil$ more than the number of remote KDC realms so as to ensure that PKCROSS uses less transaction time than PKTAPP. But the transaction time does not take into account the time required to wait in a queue for a service (i.e. waiting time) when multiple authentication requests are present in any one of the authentication entities. Response time is used when such a case is considered. That is, *the response time* is the transaction time plus the waiting time per authentication request. A further discussion of the response time is given in the following section.

6.2.2. *The calculation of the response time via queuing networks*

A queuing network is an efficient tool for analyzing the scalability of a distributed system. To investigate the scalability of PKTAPP and PKCROSS, we first model the entities, the client, the local KDC, the remote KDCs, the application servers and communication networks, as a queuing network. The client may represent a single user or multiple users who request group authentication at a given rate and the authentication request is processed in these entities according to these two techniques. Figures 6.3 and 6.4 show two queuing networks that depict the message flows of PKCROSS and PKTAPP where each system resource is modeled as a queue associated with a queuing discipline. Since public-key cryptography requires a significantly higher computation cost than private-key cryptography, it is not reasonable to assume that all client requests are served at the same average service time. Instead, a class-switching technique as in [BRU 80] and [MUN 75] is employed to model the class transaction with switching from low- to high-priority class, as different types of encryption/decryption operations are required with different service times.

Preemption-resume is one good way to implement service for satisfying multiple-class client requests. In this chapter, we use a *preemptive-resume* priority discipline, i.e. the service of a class r_2 request can be interrupted if a higher priority request of class r_1 ($r_2 > r_1$) arrives during its service. The interrupted request resumes its service from where it stopped after the higher priority request, and any other request with priority higher than r_2 that may arrive during its service, complete their service.

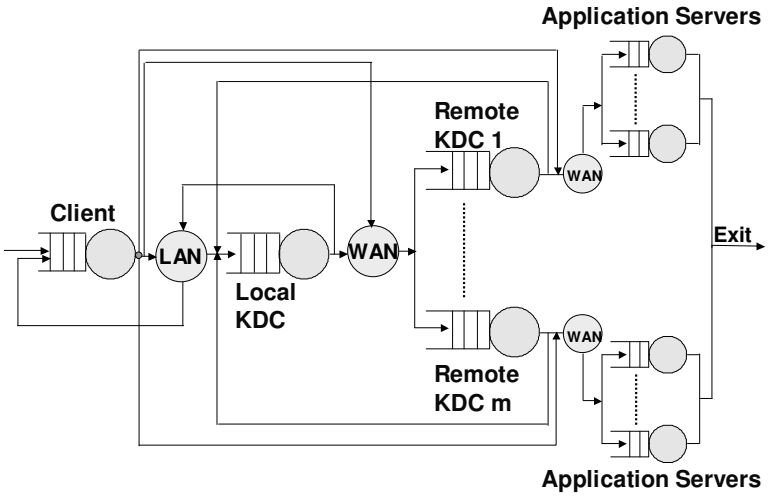


Figure 6.3. *A queuing network with m remote realms for PKCROSS*

Our goal is to use the queuing networks to calculate the response time of an authentication request. The calculation is given below. Assume that the client requests group authentication at a rate λ. On the basis of the forced law, the throughput of an entity (the client station, the local KDC, the remote KDCs or the application servers) is $X^{(j)} = \lambda v^{(j)}$ for class j job, where $v^{(j)}$ is the number of visits to the entity by class j jobs. Then, the total throughput X is a sum of $X^{(j)}$ over all job classes at the entity. Thus, according to the utilization law, the utilization of the entity by class j jobs is

$\rho^{(j)} = X^{(j)}\mu^{(j)} = \lambda v^{(j)}\mu^{(j)}$, where $\mu^{(j)}$ is the service time per visit to the entity by class j jobs. Hence, the total utilization ρ is a sum of $\rho^{(j)}$ over all classes at the entity. Denote by v the total number of visits at the entity. Thereby, the response time of the entity is $R = \frac{\mu}{1-\rho}$, where μ is an average service time of all classes at the entity, and the total response time per authentication request is a sum of vR over all entities.

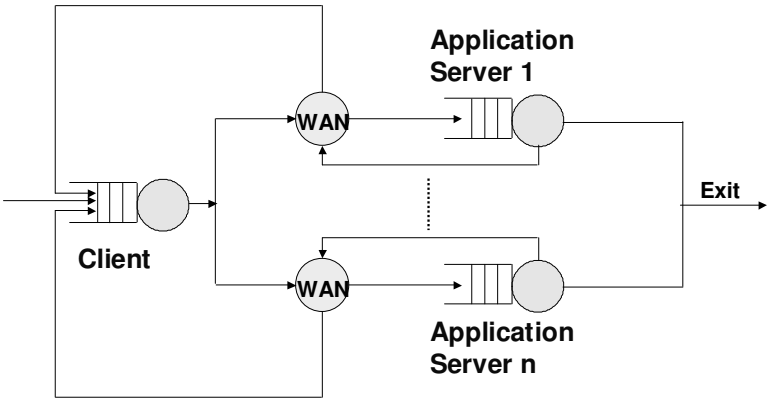


Figure 6.4. *A queuing network with n application servers for PKTAPP*

To validate the accuracy of the queuing networks, we first simulated the scenario of these entities as shown in Figure 6.5 by doing the reference implementations of these two techniques under Windows XP. Moreover, a public-key cipher is usually associated with the calculations of 1,024 bit precision numbers, so a public-key operation is computationally expensive and it costs as much as a factor of 1,000 more than an equivalent secret-key operation [DON 03]. In the reference implementations, we adopted the results from the Crypto++ ran on an Intel Core 2 1.83 GHz processor under Windows XP SP 2 in 32 bit mode [DAI 07]. Table 6.2 gives the time required to encrypt and decrypt a 64 byte block of data. Without the loss of generality, we chose

$c_1 = 0.000248$ ms, $c_2 = 0.07$ ms and $c_3 = 1.52$ ms. (The performance evaluation based on an ECC key will be discussed in my future research). Table 6.3 shows the accuracy of analytical response times obtained by the queuing methods compared to simulated results based on the scenario shown in Figure 6.5, where R-Err% is the relative error used to measure the accuracy of the analytic results compared to model simulation results, and it is defined by (analytic result simulated result)/simulated result $\times$ 100. As seen in the table, the analytic response times match the simulated results very well.

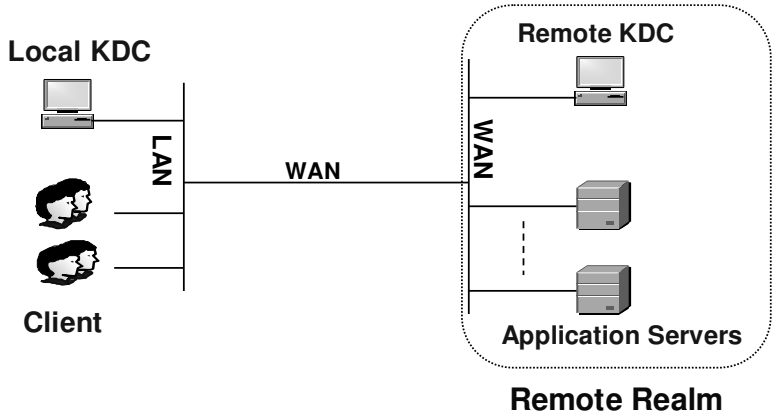


Figure 6.5. *A test scenario with a remote realm*

Protocols and operation	Key length	Computational times (ms)
AES/ECB	128	0.000248
AES/ECB	256	0.000312
RSA encryption	1,024	0.07
RSA decryption	1,024	1.52
RSA encryption	2,048	0.15
RSA decryption	2,048	5.95

Table 6.2. *The computational times of encryption and decryption operations*

m=1		The number of application servers				
Protocols	Methods	1	2	4	8	16
PKCROSS	Analytic	102.1	122.1	162.1	242.1	402.1
	Simulated	102.3	122.5	162.9	244.2	408.8
	R-Err%	−0.22	−0.30	−0.47	−0.85	−1.63
PKTAPP	Analytic	82.10	164.05	327.96	655.76	1311.38
	Simulated	82.23	164.56	329.97	663.87	1344.23
	R-Err%	−0.15	−0.30	−0.61	−1.22	−2.44

Table 6.3. *A comparison of analytic and simulated response times*

To investigate the performance with an increased number of application servers and remote realms, we further present response times as a function of authentication request rates, i.e. throughput, when there are two remote realms, as shown in Figure 6.6. As can be seen, PKCROSS has a slightly lower response time than PKTAPP when $n = 4$, called *the crossover number*. That is, PKCROSS is more efficient than PKTAPP if $n \geq 4$, but less efficient than PKTAPP if $n < 4$. Clearly, it follows from Table 6.3 that the crossover number is equal to 2 when $m = 1$. In general, Figure 6.7 shows crossover numbers with a varying number of remote realms. 99.8% of the crossover numbers fit perfectly along the straight line: $n = 1.8916m + 0.2879$. This means that PKCROSS is more efficient than PKTAPP when $n \geq \lceil 1.8916m + 0.2879 \rceil$. In other words, PKTAPP does not scale better than PKCROSS. That is different from what PKTAPP's designers expected. In the following section, we propose a hybrid approach that has better scalability than PKCROSS.

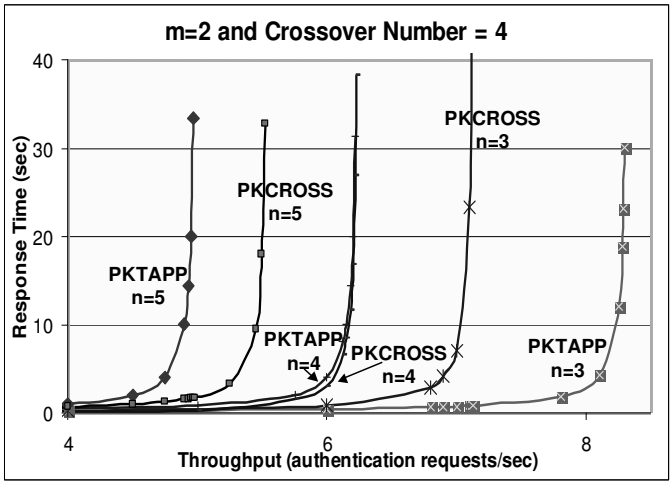


Figure 6.6. *Response time versus authentication request rate*

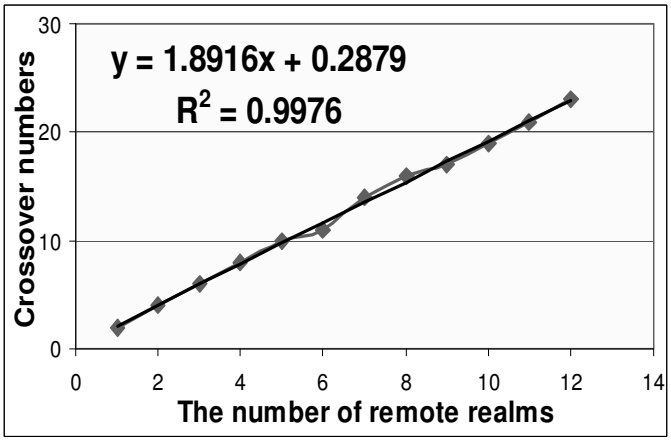


Figure 6.7. *Crossover numbers versus the number of remote realms*

6.3. A new group authentication technique using public-key cryptography

As seen above, in PKCROSS, the client is first required to communicate with the KDC before talking to the application server. PKTAPP uses a direct communication between the client and the application servers for the reduction of message exchanges and the relief of a single point of failure on the client's KDC. While PKTAPP is more efficient in the case of only a single application server and thus only a single remote realm too, PKCROSS performs significantly better if the number of application servers is more than a crossover number as shown in Table 6.3 and Figure 6.7. Our proposed technique below takes into consideration the advantages of both techniques. While it allows the client to deal directly with the application servers so that the number of messages is reduced in the authentication process like PKTAPP, the new technique still relies on the KDC for the authentication of the client and the application servers like PKCROSS.

6.3.1. *A single remote realm*

In this section, we define the group authentication technique between a client (denoted by C) and application servers (denoted by S_j) that are in a single remote realm (denoted by KDC_R), where $j = 1, \cdots, n$. The client C is a group key initiator, for example, representing either a group member in a video conference or a service provider in Web services applications who wants to initiate secure communication among groups or service providers. (How C is chosen is beyond the scope of our study in this book). The application servers S_j are the remainder of either group members or service providers. Table 6.4 presents the notation used in this section. Figure 6.8 shows the message flow of the new group authentication technique whose message exchanges are given in detail as follows.

C	Client
S_j	Application server j ($j = 1, 2, \cdots, n$)
KDC_L	Local KDC, i.e. client's KDC
KDC_R	Remote KDC, i.e. application servers' KDC
K_C	Secret key of C
K_{S_j}	Secret key of S_j
$K_{C,S}$	Group key shared by C and all S_j
$\{\text{Message}\}_{KDC_L}$	Message encrypted with KDC_L's public key
$\{\text{Message}\}_{KDC_R}$	Message encrypted with KDC_R's public key
$[\text{Message}]_{KDC_R}$	Message signed with KDC_R's private key
KB_{KDC_R}	Public key of KDC_R
$Cert_{KDC_L}$	Certificate of KDC_L
$Cert_{KDC_R}$	Certificate of KDC_R
N_C	Nonce generated by C
N_{KDC_R}	Nonce generated by KDC_R
TMS_C	Timestamp generated by C
$TMS1_R, TMS2_R$	Timestamps generated by KDC_R

Table 6.4. *The notation used in the case of a single remote realm*

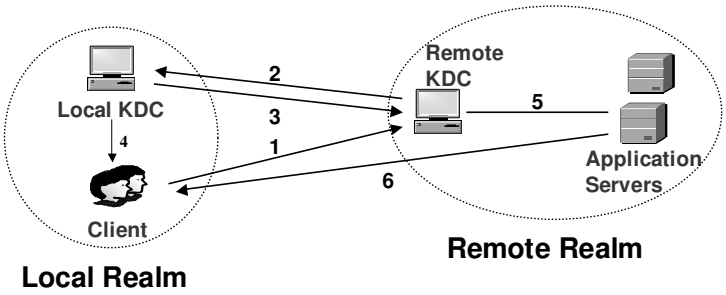


Figure 6.8. *The message flow of the new technique in a single remote realm*

Message 1, $C \rightarrow KDC_R$:
N_C, "C", "KDC_R: S_1, ..., S_n", $K_C\{N_C, K_{C,S}$, "KDC_R : $S_1, \cdots, S_n$", $TMS_C\}$

The client C initiates a group key request for a secure communication with the application servers S_j. To do this, it will encrypt the request message with a key, $K_{C,S}$, that the client C invents. To make sure that only S_j can get the key (of course, KDC_L and KDC_R should be able to get it as well), the key will be encrypted using the client C's key along with the destination, KDC_R: S_1, ..., S_n. In order for S_j to get $K_{C,S}$, the KDC_R will first need to be authenticated by the KDC of the client C, and then S_j will need to be authenticated by their KDC: KDC_R and have KDC_R give S_j the key, $K_{C,S}$. KDC_R will be responsible for making sure that S_j are valid recipients and the client C is a valid sender validated by the client C's KDC, KDC_L. A common nonce, N_C, will also be invented to track the transaction and permit the recipient to authenticate the transaction with the remote KDC server, KDC_R. A timestamp TMS_C is included to prevent replay attacks or prevent the key, $K_{C,S}$, from being given out more than once.

Message 2, $KDC_R \rightarrow KDC_L$:
N_C, "C", $K_C\{N_C, K_{C,S}$, "KDC_R : $S_1, ..., S_n$", $TMS_C\}$, $\{N_C, N_{KDC_R}, TMS1_R$, AuthInfo$\}_{KDC_L}$, where AuthInfo consists of "KDC_R", $Cert_{KDC_R}$, $[N_C, N_{KDC_R}, TMS1_R$, "KDC_L", $KB_{KDC_R}]_{KDC_R}$.

After receiving the message from the client C, KDC_R invents a nonce, N_{KDC_R}, and generates AuthInfo necessary to authenticate KDC_R. Then, it constructs a message to send to the client C's local KDC that contains N_{KDC_R}, N_C, $TMS1_R$ and AuthInfo, encrypted with KDC_L's public key. KDC_R will also forward the part of the original message encrypted with C's key. This is enough information for KDC_L to authenticate the client C and KDC_R to make sure only if KDC_L can decrypt this message. Note that the message uses "KDC_R" instead of "KDC_R: S_1, ..., S_n" for the sake of a privacy consideration. This is because the client C's local KDC is not

required for knowing who will be part of the secure communication except the client C. In addition, this message does not explicitly contain the identity KDC_R since it is included in AuthInfo. "KDC_R" and its public key KB_{KDC_R} are uniquely determined by $Cert_{KDC_R}$ in AuthInfo. Thus, $Cert_{KDC_R}$ is used to prevent man-in-the-middle attacks. $TMS1_R$ serves to avert replay attacks.

Message 3, $KDC_L \rightarrow KDC_R$:
$\{$"C", "KDC_R", $N_{KDC_R}, K_{C,S}, Cert_{KDC_L}, TMS_C, TMS1_R\}_{KDC_R}$

Using the key of the name given, KDC_L will decrypt the message encrypted with its public key and the original message encrypted by K_C, and verify KDC_R's signature. Then, it will check if the N_C encrypted by key K_C matches the N_C encrypted by its public key. If they match, then the client C is really the client C and its message is not altered. KDC_L will then read the destination, in this case, "KDC_R". (Again, it does not include "KDC_R: $S_1, \cdots, S_n$" in the message for the sake of a privacy consideration.) Furthermore, KDC_L will make sure if KDC_R is a valid member. If so, it continues the processing. Accordingly, KDC_L will make sure that the timestamp is within the allowed clock-skew and that the key for this request has not already been given out. KDC_L will then give out the key, $K_{C,S}$. KDC_L will also encrypt N_{KDC_R} with KDC_R's public key to authenticate KDC_L to KDC_R. KDC_R will verify that the N_{KDC_R} received from KDC_L matches the N_{KDC_R} sent to KDC_L, and it will distribute the returned $K_{C,S}$ to S_j.

Message 4, $KDC_L \rightarrow C$:
$K_C\{K_{C,S}$, "KDC_L", "$FromKDC_R$", $N_C, TMS_C, TMS_L\}$

The client decrypts the share key $K_{C,S}$, "$FromKDC_R$" and TMS_L. The client needs to make sure that the N_C matches the nonce of a pending request, the timestamp is within the

allowed clock-skew and $K_{C,S}$ matches the group key that was previously created. Using timestamps and nonce numbers makes the encrypted message random to some extent, and thus prevents a brute-force cryptographic attack.

Message 5, $KDC_R \rightarrow S_j$:
$K_{S_j}\{\text{“}C\text{”}, K_{C,S}\}$, $K_{C,S}\{N_C, TMS_C, TMS2_R\}$

The application servers S_j will decrypt the first message to get $K_{C,S}$, and then use the shared key $K_{C,S}$ to decrypt the second message to get TMS_C and $TMS2_R$. TMS_C is included so that S_j knows when the group key was inverted by C. To make sure that it is not a replay, S_j should keep all timestamps that were recently received, say in the last 5 min, which is a parameter set approximately for the maximum allowable time skew. Then, S_j should check that each received timestamp from a given request initiator is different from any of the stored values. All authentication requests older than 5 minutes (or whenever the value of the maximum allowable time skew is) would be rejected, so S_j would not remember values older than 5 minutes.

Messages 6, $S_j \rightarrow C$:
$K_{C,S}\{N_C, TMS_C\}$, where $j = 1, 2, \cdots, n$.

The application servers send replies to the client. Then, by using pregenerated $K_{C,S}$, the client decrypts the message to make sure that the group key $K_{C,S}$ is not altered. It also verifies nonce N_C and makes sure the received timestamp of the pending request is within the allowed clock-skew of the timestamp TMS_C.

In the new technique, note that $K_{C,S}$ can be replaced by K_{C,S_j} when C only wants to securely communicate with any one of application servers S_j. The aforementioned technique can be also modified so that a reply is issued to C by either

KDC_L or KDC_R whenever needed after C has been authenticated.

6.3.2. *Multiple remote realms*

In a large network, application servers are often within different network domains. For instance, in Web services, service providers may be divided into many different realms due to their location flexibility. To work together, a service provider may wish to gain access to other service providers' application servers in remote realms. To support "cross-realm" authentication, the service provider's KDC needs to establish either a direct or an indirect trust relationship with the other service providers' KDCs. Here, we briefly describe how the proposed technique is extended in multiple remote realms.

Assume that the n application servers S_j are distributed in m realms, each with KDC servers denoted by KDC_i ($i = 1, \cdots, m$). Let n_i be the number of application servers within KDC_i, where $0 \leq n_i \leq m$ and $\sum_{i=1}^{m} n_i = n$. Clearly, the case of a single remote realm is associated with $n_1 = n$ and $n_i = 0\,(i = 2, \cdots, m)$. We further let $S_{i,k}$ be those application servers that belong to the set $\{S_j\,|j = 1, \cdots, n\}$ within realm i, where $k = 1, \cdots, n_i$. Then, the set $\cup_{i=1}^{m}\{S_{i,k}\,|k = 1, \cdots, n_i\}$ is identical to the set $\{S_j\,|j = 1, 2, \cdots, n\}$. Table 6.5 presents additional notation employed in this section. Figure 6.9 shows how the client authenticates to application servers. Authentication messages are as follows:

Messages 1 and 1', $C \rightarrow KDC_i$ for each fixed i:
N_C, "C", "KDC_i: $S_{i,1}$, ..., S_{i,n_i}", $K_C\{N_C, K_{C,S}$, "KDC_i : $S_{i,1}, \cdots, S_{i,n_i}$", $TMS_C\}$

Messages 2 and 2', $KDC_i \rightarrow KDC_L$:
N_C, "C", $K_C\{N_C, K_{C,S}$, "KDC_i : $S_{i,1}, \cdots, S_{i,n_i}$", $TMS_C\}$,

$\{N_C, N_{KDC_i}, TMS1_i, \text{AuthInfo}_i\}_{KDC_L}$, where AuthInfo_i consists of "KDC_i", $Cert_{KDC_i}$ and $[N_C, N_{KDC_i}, TMS1_i,$ "KDC_L", $KB_{KDC_i}]_{KDC_i}$

$S_{i,k}$	Application server k in realm i
KDC_i	Remote KDC server in realm i
$K_{S_{i,k}}$	Secret key of $S_{i,k}$
KB_{KDC_i}	Public key of KDC_i
$\{\text{Message}\}_{KDC_i}$	Message encrypted with KDC_i's public key
$[\text{Message}]_{KDC_i}$	Message signed with KDC_i's private key
N_{KDC_i}	Nonce generated by KDC_i
$Cert_{KDC_i}$	Certificate of KDC_i
$TMS1_i$ and $TMS2_i$	Timestamps generated by KDC_i

Table 6.5. *The additional notation used in the case of multiple remote realms*

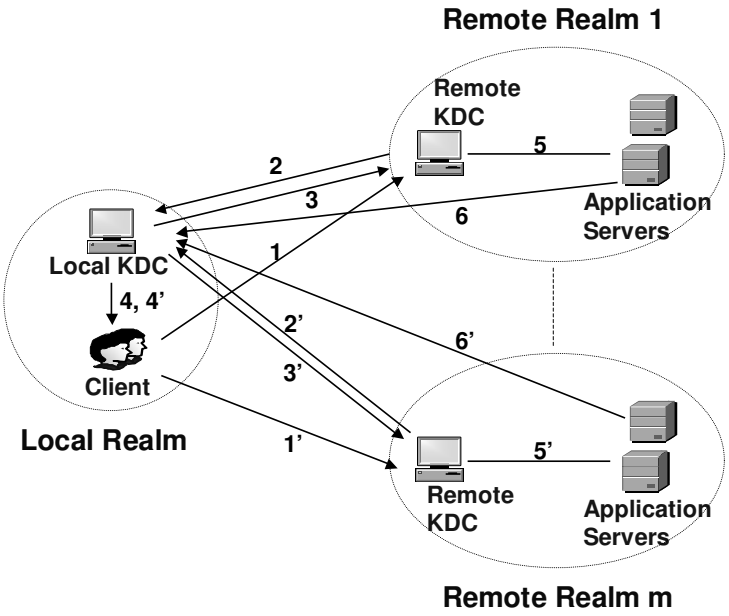


Figure 6.9. *The message flow of the new technique in multiple remote realms*

Messages 3 and 3', $KDC_L \rightarrow KDC_i$ where $i = 1, \cdots, n$:
$\{\text{“}C\text{”}, \text{“}KDC_i\text{”}, N_{KDC_i}, K_{C,S}, Cert_{KDC_L}, TMS_C\}_{KDC_i}$

Messages 4 and 4', $KDC_L \rightarrow C$:
$K_C\{K_{C,S}, \text{“}KDC_L\text{”}, \text{“}FromKDC_i\text{”}, N_C, TMS_C, TMS_L\}$

Messages 5 and 5', $KDC_i \rightarrow S_{i,k}$:
$K_{S_{i,k}}\{\text{“}C\text{”}, K_{C,S}\}$, $K_{C,S}\{N_C, TMS_C, TMS2_i\}$, where $i = 1, \cdots, m$ and $k = 1, \cdots, n_i$.

Messages 6 and 6', $S_{i,k} \rightarrow C$:
$K_{C,S}\{N_C, TMS_C\}$, where $i = 1, \cdots, m$ and $k = 1, \cdots, n_i$.

In the first three messages, the client C authenticates to his/her local KDC through remote KDC_i. Message 4 or 4' checks if the group key $K_{C,S}$ is valid, and Message 5 or 5' distributes the group key $K_{C,S}$ and authenticates designated application servers within their individual KDC realms. An explanation of these messages is similar to the one given in section 6.3.1.

6.4. Performance evaluation of the new proposed technique

In this section, we focus on studying the efficiency of the proposed group authentication technique. We first compute its computational and communication costs. Then, we give a thorough performance evaluation of the new technique using the queuing method proposed in section 6.2.

6.4.1. *The operations of encryption and decryption*

The baseline transactions are constructed with one or more application servers in a remote realm as shown in

Figure 6.8. Without any confusion, we directly consider the case of m remote realms, each with n_i application servers, where $0 < n_i \leq n$ and $i = 1, 2, \cdots, m$. Table 6.6 summarizes the number of encryption and decryption operations performed in the proposed technique. As is shown in Tables 6.1 and 6.6, in our technique, the computational burden on the client is mitigated to its local KDC compared to PKTAPP. Specifically, in terms of the calculation of public-key operations, the burden on the client's local KDC is $O(m)$ in both the proposed technique and PKCROSS but the burden on the client is $O(n)$ in PKTAPP. Let us recall that m is the number of remote KDC servers and n is a total number of application servers where $1 \leq m \leq n$. Hence, their computational burdens may be significantly different when $m \ll n$. Note that the proposed technique uses public-key cryptography to authenticate the client's KDC and the remote KDCs of application servers. So, it reduces the risk of a single failure on these KDCs.

Entities	Number of secret keys	Number of private keys	Number of public keys
Client	3	0	0
Local KDC	2	1	$4m$
Remote KDC	n+1	$2m$	$3m$
Application server	$3n$	0	0
Total	$4n$+6	$2m$+1	$7m$

Table 6.6. *The operations of encryption and decryption when n application servers are in m remote realms*

Next, let us consider the operation costs of the proposed technique. Denote by $f_3(n, m)$ the total computational time of its encryption and decryption operations. Then, it follows from Table 6.6 that $f_3(n, m)$ is computed by:

$$f_3(n, m) = 4c_1 n + (2c_2 + 7c_3)m + 6c_1 + c_2$$

Thus, we have the following proposition.

PROPOSITION 6.3.– (1) The proposed technique requires less computational time than PKCROSS. (2) For $n \geq 4$, the proposed technique requires less computational time than PKTAPP. But, when $1 \leq n < 4$, the proposed technique requires less computational time than PKTAPP if and only if the number of remote realms m is less than $\lfloor n + \frac{(n-4)c_1}{2c_2+7c_3} \rfloor$.

PROOF.– The differences of computational times among the three techniques are given by $f_1(n,m) - f_3(n,m) = -4c_1 + (c_1 + 2c_2 + 7c_3)n - (2c_2 + 7c_3)m$ and $f_2(n,m) - f_3(n,m) = 4c_1m + 5c_1$. Obviously, our technique requires less computational time than PKCROSS due to $f_2(n,m) - f_3(n,m) > 0$. Moreover, $f_1(n,m) - f_3(n,m) \geq 0$ if and only if $m \leq n + \frac{(n-4)c_1}{2c_2+7c_3}$, which implies (2) due to $n \geq m$. This proves proposition 6.3.

Similarly, we can have the following proposition:

PROPOSITION 6.4.– For $m \geq 4$, the proposed technique requires less computational time than PKTAPP. But, when $1 \leq m < 4$, the proposed technique requires less computational time than PKTAPP if and only if the number of application servers n is more than $\lceil m - \frac{(m-4)c_1}{c1+2c_2+7c_3} \rceil$.

Furthermore, let $g_3(n,m)$ be the transaction time of the proposed technique, i.e. the computational time of its encryption and decryption operations plus its communication time. Note that the number of WAN communications required in the technique is $3m$+$2n$, and recall that d is the time spent in a WAN communication. Then, the following statements hold.

PROPOSITION 6.5.– (1) The proposed technique has less transaction time than PKCROSS. (2) The proposed technique uses less transaction time than PKTAPP if and only if the

number of application servers n is more than $\lceil m + \frac{(d-c_1)m+4c_1}{c1+2c_2+7c_3+2d} \rceil$.

PROOF.– It is given by $g_1(n,m) - g_3(n,m) = [f_1(n,m) - f_3(n,m)] + (2n-3m)d = (c_1+2c_2+7c_3+2d)n - (2c_2+7c_3+3d)m - 4c_1$, and $g_2(n,m) - g_3(n,m) = [f_2(n,m) - f_3(n,m)]$. These easily derive (1) and (2) in this proposition. The proof is complete.

Note that when $n = 1$, we get that $m + \frac{(d-c_1)m+4c_1}{c1+2c_2+7c_3+2d} > 1 = n$ as usually $d > c_1$, which implies $g_1(n,m) - g_3(n,m) < 0$. Also, when $m = 1$ and $n \geq 2$, $m + \frac{(d-c_1)m+4c_1}{c1+2c_2+7c_3+2d} = 1 + \frac{d+3c_1}{c1+2c_2+7c_3+2d} < 2 \leq n$ as $c_1 < c_3$, which implies $g_1(n,m) - g_3(n,m) > 0$. Thus, we have that

COROLLARY 6.2.– PKTAPP is more efficient than the proposed technique when $n = 1$, but less efficient when $m = 1$ and $n \geq 2$, in terms of transaction time.

Using proposition 6.5, we calculated the minimal number of application servers so as to ensure that our technique requires a lower transaction time than PKTAPP with varied $d = 0.12$, 4.8 and 10 ms when $c_1 = 0.000248$ ms, $c_2 = 0.07$ ms and $c_3 = 1.52$ ms. The results are shown in Table 6.7. We observe that the minimal number of application servers is sensitive to d rather than c_j ($j = 1, 2, 3$). Table 6.8 further presents the difference of transaction times between our technique and PKCROSS. We also noted that our proposed technique requires significantly less transaction time than PKCROSS, and the difference of transaction times between the two techniques is independent of the number of application servers. That is, the proposed technique is more efficient than PKCORSS.

6.4.2. *The calculation of the response time via a queuing network*

Similar to section 6.2, we can use a queuing network to characterize the message flow of the proposed technique where each system resource is modeled as a queue associated with a queuing discipline shown in Figure 6.10.

	Number of remote realms									
Number of servers	1	2	3	4	5	6	7	8	9	10
d = 0.12	2	3	4	5	6	7	8	9	10	11
d = 4.8	2	3	4	5	7	8	9	10	12	13
d = 10	2	3	4	6	7	8	10	11	12	14

Table 6.7. *The minimal number of application servers*

Figures 6.11, 6.12 and 6.13 show response times as a function of authentication request rates, i.e. throughput, in the case of one, two or eight remote realms, respectively, with varying number of application servers. As can be seen, our technique performs better than PKCROSS in all cases. It has also been demonstrated that our technique is more efficient than PKTAPP when $n \geq 2$ for one remote realm in Figure 6.11, when $n \geq 4$ for two remote realms in Figure 6.12 and when $n \geq 12$ for eight remote realms in Figure 6.13 (roughly $n \geq 1.5m$ if $m > 1$). These crossover numbers from $m = 1$ to 12 are further depicted in Figure 6.14 where the crossover numbers are 99.6% perfectly fitted by the straight line: $n = 1.4406m + 0.6364$, which predicts the number of application servers required to ensure that our technique performs better than PKTAPP. Moreover, the line $n = 1.4406m + 0.6364$ is below the straight line $n = 1.8916m + 0.2879$ given in section 6.2 as $1.8916m + 0.2879 > 1.4406m + 0.6364$ when $m \geq 1$. This again confirms that our technique is more efficient than PKCROSS. We also noted that the numbers n in Table 6.7 are smaller than the

crossover numbers in Figure 6.14. This means that it is not sufficient *only if* transaction time is employed to analyze performance that researchers have often used.

(a) $m = 1, 2, 3, 4, 5$

	Number of remote realms				
Difference (ms)	1	2	3	4	5
d = 0.12	0.1222	0.2432	0.3642	0.4852	0.6062
d = 4.8	4.8022	9.6032	14.404	19.205	24.006
d = 10	10.002	20.003	30.004	40.005	50.006

(b) $m = 8, 12, 16, 20, 24$

	Number of remote realms				
Difference (ms)	8	12	16	20	24
d = 0.12	0.9691	1.45314	1.93711	2.42108	2.90504
d = 4.8	38.409	57.6131	76.8171	96.0210	115.225
d = 10	80.009	120.013	160.017	200.021	240.025

Table 6.8. *The difference of transaction times between our technique and PKCROSS (i.e. $g_2(n, m)$-$g_3(n, m)$)*

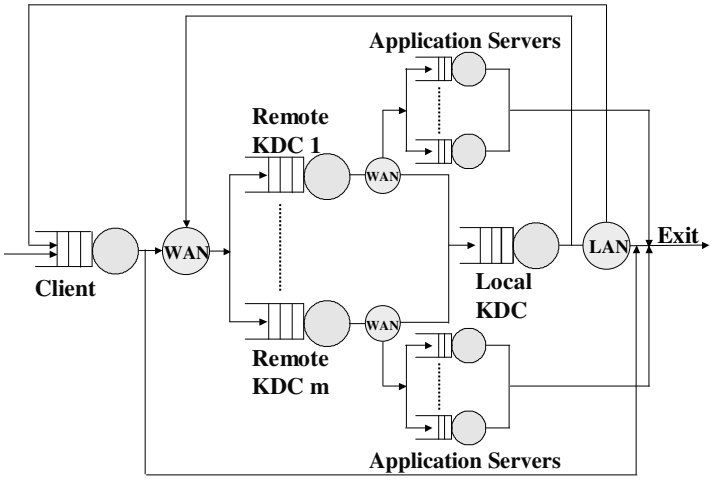


Figure 6.10. *A queuing network with n remote realms for the new technique*

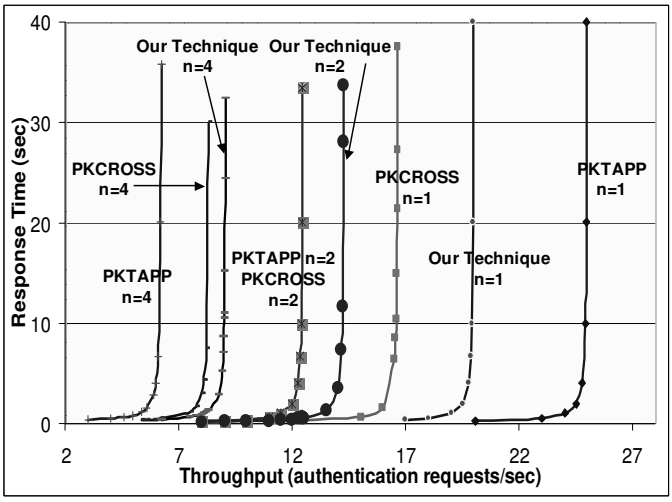


Figure 6.11. *Response times versus authentication request rates when* $m = 1$

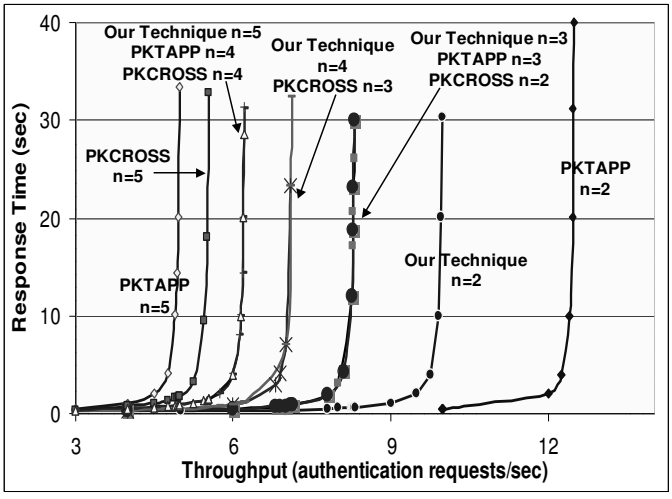


Figure 6.12. *Response times versus authentication request rates when* $m = 2$

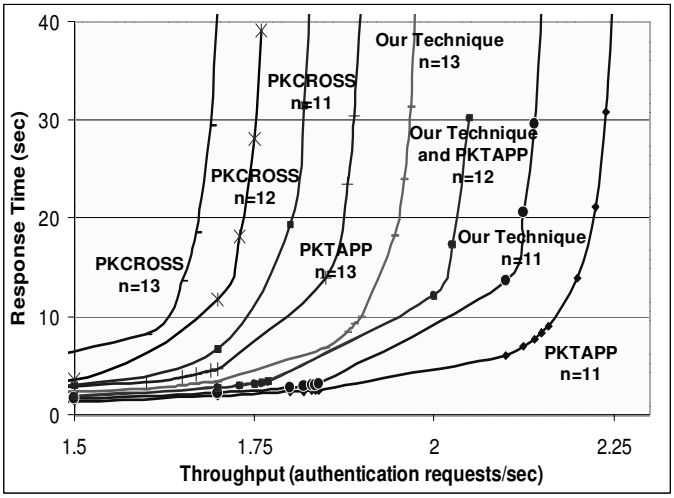


Figure 6.13. *Response times versus authentication request rates when* $m = 8$

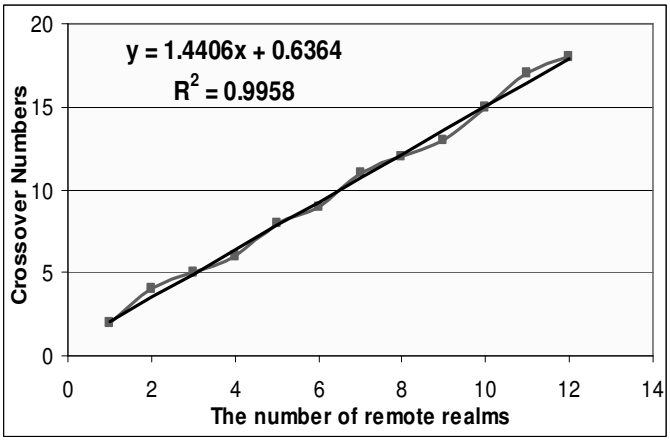


Figure 6.14. *Crossover numbers versus remote realms*

6.4.3. *Discussions*

The proposed technique is conceptually simple as it involves the client that generates a key and the KDCs that

authenticate and distribute the key to group members via a pairwise secure channel established with each group member. It works well in one-to-many multicast scenarios, for example secure communications among service providers in Web services.

While our technique has several strengths as compared to PKCROSS and PKTAPP, we have also realized that the client's request message passes through the remote KDC to the client's local KDC with no verification. It needs to securely migrate and maintain an authentication state at every remote KDC, and might result in serious denial of services (DOS) attacks. But our technique is motivated by secure communications in Web services applications and it is mainly geared toward e-business applications in which a trust-but-verify framework for Web services authorization has been proven an efficient method [SKE 84]. So, under the same framework we trust a client in a certain degree and allow the client to contact the KDC server of the application servers directly. Then, the authentication procedures proposed in this chapter will follow. However, if the remote KDC server is badly under DOS attacks due to unauthenticated messages from the client, then we leave the option to switch back to the way in which a client needs to be first authenticated by its local KDC, e.g., using PKCROSS.

In addition, in the proposed technique, the client is allowed to create the group key $K_{C,S}$. This is particularly useful when the client does not need to get a reply from its local KDC for a further verification. If the reply is required, then we can easily mitigate the creation of $K_{C,S}$ from the client to its local KDC by slightly modifying the proposed technique accordingly.

6.5. Concluding remarks

Public-key-enabled Kerberos-based techniques such as PKINIT, PKCROSS, PKDA and PKTAPP (also known as

KX.509/KCA) give a potentially effective way for cross-realm authentication. However, the authentication problem is simple to describe but hard to solve, especially for multiple realms. Their performance has been poorly understood. This chapter presented a detailed performance evaluation of PKCROSS and PKTAPP in terms of computational and communication times, and through the use of validated analytical queuing models. Our analysis revealed that PKTAPP does not perform better than PKCROSS in a large network where there are many application servers within either a single remote realm or multiple remote realms. Thus, we proposed a new public-key cryptography-based group authentication technique. Our performance analysis has shown that the proposed technique outperforms PKCROSS in terms of computational and communication times in propositions 6.3 and 6.5, and response time in Figures 6.11, 6.12 and 6.13. This chapter also gave the predicted minimal number of application servers so as to ensure that the proposed approach is more efficient than PKTAPP in multiple remote realms. Roughly speaking, the proposed technique performs better than PKTAPP when the number of application servers is approximately 50% more than the number of remote realms (i.e. $n \geq 1.5m$) as discussed in section 6.4.2. As shown, our performance methodology based on complexity analysis and queuing theory is an effective way to analyze the performance of security protocols.

Chapter 7

Summary and Future Work

This chapter presents the research summary of the book and provides research directions for future study.

7.1. Research summary of the book

In this book, we considered a collection of computer resources used by a cloud service provider to host enterprise applications for business customers including cloud users. An enterprise application running in such a computing environment is associated with a service-level agreement (SLA). That is, the service provider is required to execute service requests from a customer within negotiated Quality-of-Service (QoS) requirements for a given price. The QoS requirements may include service availability, performance and security, and the service requests may come from either single-class customers (e.g. one type of cloud users) or multiple-class customers (e.g. multiple types of cloud users). It is assumed that the service provider owns and controls a number of resource stations or sites. In Chapter 1, we gave an overview of service availability, performance and

security, and described a resource optimization problem subject to these QoS metrics, and a related group authentication problem in the distributed computing. A literature review was given in Chapter 2.

Chapter 3 presented an approach for service optimization in such an enterprise application environment that minimizes the total cost of computer resources required while satisfying an SLA for single-class customers. We considered a response time in the QoS. Typically, in the literature, the response time is taken into account through its mean. However, this may not be of real interest to a customer. In view of this, we studied percentiles of the response time, and calculated the number of servers in each resource station that minimize a cost function that reflects operational costs. Numerical results showed the applicability of the proposed approach and validated its accuracy in the case of single-class customers. We found that there is a relative error of less than 0.71% in the cumulative distribution function of response time calculated by our approximate algorithm and a simulation method for the service models under our study. We also found that the number of servers in each resource station obtained by our approximate algorithm is in fact optimal, provided that each station has balanced utilization.

However, in enterprise computing, customer requests often need to be distinguished, with different request characteristics and customer service requirements. In this book, we further considered a set of computer resources used by a cloud service provider to host enterprise applications for differentiated customer services subject to a percentile response time and a fee, as given in Chapter 4. We proposed algorithms for solving the resource optimization problem in the case of multiple-class customers. The accuracy of our algorithms was verified by numerical simulation. A contributing factor to our algorithms' accuracy is that

typically we are interested in values of the cumulative distribution of the response time that correspond to very high percentiles for which the approximate results seem to have a very good accuracy.

With the number of e-business applications increasing dramatically, the SLA will play an important part in Web services. It is often a combination of several QoS, such as security, performance and availability, agreed between a customer and a cloud service provider. Most existing research addresses only one of these QoS metrics.

In Chapter 5, we presented a study of three QoS metrics, namely trustworthiness, percentile response time and availability. Then, we considered all these three QoS metrics in a trust-based resource provisioning problem that typically arises in Web services, and formulated this problem as an optimization problem under SLA constraints in the cases of single-class and multiple-class customers, respectively. By the use of an effective and accurate numerical solution for the calculation of the percentile response time presented in Chapters 3 and 4, we solved the optimization problem using an efficient numerical procedure in the both cases.

Chapter 6 presented a thorough performance evaluation of PKCROSS and PKTAPP (a.k.a. KX.509/KCA) in terms of computational and communication times. Then, we demonstrated their performance difference using open queuing networks. An in-depth analysis of these two techniques showed that PKTAPP does not scale better than PKCROSS. Thus, we proposed a new public key cryptography-based group authentication technique. Our performance analysis demonstrated that the new technique can achieve better scalability as compared to PKCORSS and PKTAPP. This monograph is based on [XIO 07].

7.2. Future research directions

In this section, we present an extensive discussion of our approach for solving the resource optimization problem and the problem of public-key cryptography-based group authentication for our future study.

The selection of service sites: we introduce a trust manager who is a trusted agent representing customers. We assume that the trust manager is aware of the service sites that process its service request. Moreover, the trust manager is able to maintain the trustworthiness information of the service sites. A customer, however, does not have this information because the trust manager is his/her representative who deals with all service matters including an SLA negotiation, and service requests' submission and processing.

As shown in Figure 5.2, the service processor uses a composition of services selected from these service sites to complete the customer's request. These service sites are chosen by the trust manager on the basis of the trust information of each site. In our proposed algorithms, the trust manager keeps the ranking of all service sites. When the trust manager receives a service request from a customer, it will check the trustworthiness information of service sites in its database, and then select those service sites with the highest indices that meet trust requirements predefined by its customers as well. This selection maximizes the customer's benefit in terms of trustworthiness.

However, the more reliable service site usually requires more security measures so as to provide the better QoS. Thus, the higher the trust index of the service site, the higher the cost of the service. The trust manager on the other hand may only choose those service sites that meet predefined trust requirements with less costs. In this case, the trust manager

may require the service broker to find those service sites that meet predefined trust requirements but have the lowest total cost for the customer's request. The service broker can find those sites by modifying [1.1] into the following optimization:

$$\min_{\substack{\forall s_j \in S; I_{s_j} \leq \hat{I}_j \\ j=1,\cdots,m}} \min_{n_1,\cdots,n_m} (n_1 c_1 + \cdots + n_m c_m) \qquad [7.1]$$

and solving the problem subject to constraints by an SLA, where S is a set of service sites consisting of $S_1, S_2, \cdots, S_M$, and s_j ($j = 1, \cdots, m$) are any m of those service sites whose trust indices I_{s_j} are less than $\hat{I}_j$, as predefined by the customer. This approach maximizes the profitability of the trust manager. It may be a typical case in the real world, for example, when the trust manager and the service broker may be integrated into only one entity that represents both the customer and the service sites.

In this book, we only considered the case in which the trust manager selects those service sites with the highest indices, as used in section 3.3 for the best benefit of the customer. An extension to study the above alternative case will be of interest.

The relationship among trustworthiness, response time and availability: the trust indices of resource sites are an indicator of the QoS provided by these sites, and are based on previous and current job completion experience. Clearly, when the response time does not meet a predefined requirement, or resource sites are not able to serve a customer request, the trust manager will give these resource sites low trust indices.

Similarly, the longer the service at a resource site is unavailable, the longer the end-to-end response time

becomes. Particularly, when resource sites are under denial of service attacks, the end-to-end response time of a service request may become infinite. Hence, when a service request suffers from a much longer response time, it may mean that some of resource sites cannot meet the requirement of service availability. In other words, these three SLA metrics, trustworthiness, response time and service availability, interact each other, and their relationship is very complicated. A detailed study of these three metrics will be conducted in our future research.

The trust updating approach and the server repairing mechanism: in our approach for solving the trust-based resource provisioning problem, we used a rank- and a threshold-based approach to deal with the trust indices of resource sites, and assumed that a service discipline was First-In-First-Out (FIFO) and each resource site could intermediately repair a server. In our future study, we will extend our discussion by evaluating different trust approaches and using other service disciplines with a general repairing mechanism for secure resource management in Web service applications and other distributed computing environments.

The study of other security mechanisms: as is shown, our performance methodology, based on complexity analysis and queuing theory presented in Chapter 6, is an effective way to analyze the performance of security protocols. In the future, we plan to apply the methodology to other security protocols. In particular, it will be interesting to apply our method to those protocols that are used in either time-sensitive or resource-limited applications, e.g. those in wireless sensor networks.

The end-to-end security and performance assessment of cloud services: while many researchers have investigated the performance and security of cloud services in data centers,

there is a lack of studies on the end-to-end performance and security of cloud services. It will be interesting to study the end-to-end security and performance evaluation of cloud services theoretically and experimentally based on the proposed approaches in this book. Preliminary results have been obtained in [RED 13] and [SHE 12].

The study of power-aware resource optimization: power management becomes a critical issue in cloud computing. What is the relationship between cloud service performance and power consumption? How to ensure performance guarantee subject to predefined power consumption? How to minimize power consumption subject to predefined performance constraints? The partial answers to these questions can be found in [XIO 10b], [XIO 11b] and [XIO 13].

Bibliography

[AIB 04] AIB I., AGOULMINE N., PUJOLLE G., "The generalized service level agreement model and its application to the SLA driven management of wireless environments", *International Arab Conference on Information Technology*, ACIT, December 2004.

[AIK 03] AIKAT J., KAUR J., SMITH F., *et al.*, "Variability in TCP round-trip times", *Proceedings of the ACM SIGCOMM Internet Measurement Conference*, October 2003.

[AIY 05] AIYER A., ALVISI L., BAZZI R., "On the availability of non-strict quorum systems", *Proceedings of the 19th International Symposium on Distributed Computing (ISDC 2005)*, pp. 48–62, September 2005.

[ALL 03] ALLMAN M., EDDY W., OSTERMANN S., "Estimating loss rates with TCP", *ACM Performance Evaluation Review*, vol. 31, no. 3, pp. 12–24, December 2003.

[ALT 01] ALTIOK T., MELAMED B., *Simulation Modeling and Analysis with Arena*, Cyber Research, Inc. and Enterprise Technology Solutions, Inc., 2001.

[ALT 07a] ALTMAN J., "NIST PKI'06: integrating PKI and Kerberos", 2007. Available at www.secure-endpoints.com/talks/nist-pki-06-kerberos.pdf.

[ALT 07b] ALTMAN J., "Personal communication", 2007. Available at www.secure-endpoints.com/talks/nist-pki-06-kerberos.pdf.

[AMR 85] AMR EL ABBADI D.S., CRISTIAN F., "An efficient and fault-tolerant protocol for replicated data management", *Proceedings of PODS*, 1985.

[BAR 95] BARBACCI M., KLEIN M.H., LONGSTAFF T.H., *et al.*, Quality attributes, CMU techical report: CMU/SEI-95-TR-021, Software Engineering Institute, Carnegie Mellon University, 1995.

[BAR 03] BARRY D.K., *Web Services and Service-Oriented Architecture: Your Road Map to Emerging IT*, Morgan Kaufmann, 2003.

[BIS 02] BISHOP M., *Computer Security*, Addison-Wesley, Boston, MA, 2002.

[BLA 96] BLAZE M., *High-Bandwidth Encryption with Low-Bandwidth Smartcards*, Lecture Notes in Computer Science, no. 1039, 1996.

[BOL 93] BOLOT J., "End-to-end packet dealy and loss behavior in the Internet", *Proceedings of the ACM SIGCOMM*, 1993.

[BOL 98] BOLCH G., GREINER S., MEER H., *et al.*, *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications*, John Wiley & Sons, New York, 1998.

[BOU 02] BOUILLET E., MITRA D., RAMAKRISHNAN K., "The structure and management of service level agreements in networks", *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 4, pp. 691-699, 2002.

[BRE 01] BREWER E.A., "Lessons from giant-scale service", *IEEE Internet computing*, vol. 5, pp. 46–55, July–August 2001.

[BRO 00] BROWN A., PATTERSON D., "Towards availability benchmarks: a case study of software RAID systems", *Proceedings of 2000 USENIX Annual Technical Conference*, USENIX, June 2000.

[BRU 80] BRUELL S., BALBO G., *Computational Algorithms for Closed Queueing Networks*, Science Library, Elsevier North Holland, Inc., New York, 1980.

[BUR 72] BURKE P., "Output processes and tandem queues", *Proceedings of the Symposium on Computer Communication Networks and Teletraffic*, Brooklyn, April 1972.

[CAL 92] CALABRESE J., "Optimal workload allocation in open queueing networks in multiserver queues", *Management Science*, vol. 38, no. 12, pp. 1792–1802, December 1992.

[CHA 02] CHASSOT C., GARCIA F., AURIOL G., *et al.*, "Performance analysis for an IP differentiated services network", *Proceedings of IEEE International Conference on Communication (ICC'02)*, pp. 976–980, 2002.

[CHA 03] CHANDRA A., GONG W., SHENOY P., "Dynamic resource allocation for shared data centers using online measurements", *Proceedings of 11th International Conference on Quality of Service (IWQoSC 2003)*, June 2003.

[CHA 05] CHANG J., "Processor performance, update 1", 2005. Available at http://www.sql-server-performance.com.

[CIS 13a] CISCO, "Increasing network availability", 2013. Available at http://www.cisco.com/warp/public/779/largeent/learn/technologies/ina/IncreasingNetworkAvailability-Abstract.pdf.

[CIS 13b] CISCO, "Network availability: how much do you need? How do you get it?", 2013. Available at http://www.cisco.com/web/IT/unified_channels/area_partner/cisco_powered_network/net_availability.pdf.

[CIS 13c] CISCO, "Service level management: best practice", 2013. Available at http://www.cisco.com/en/US/tech/tk869/tk769/technologies_white_paper09186a008011e783.shtml.

[COH 82] COHEN J., *The Single Server Queue*, North-Holland, Amsterdam/New York/Oxford, 1982.

[CON 07] CONSORTIUM T.M.K., "Proposal for corporate sponsors", 2007. Available at http://www.kerberos.org/join/proposal.pdf.

[CUR 02] CURBERA F., DUFTLER M., KHALAF R., *et al.*, "Unraveling the Web services Web: an introduction to SOAP, WSDL, and UDDI", *IEEE Internet Computing*, vol. 6, no. 2, pp. 86–93, March–April 2002.

[DAD 84] DADUNA H., "Burke's theorem on passage times in Gordon-Newell networks", *Advances in Applied Probability.*, vol. 16, pp. 867–886, 1984.

[DAI 07] DAI W., "Crypto++ 3.1 benchmarks", 2007. Available at http://www.eskimo.com/weidai/benchmark.html.

[DAV 96] DAVIS D., "Compliance defects in public-key cryptography", *Proceedings of the 6th USENIX UNIX Security Symposium (USENIX Security' 96)*, San Jose, CA, July 1996.

[DAV 04] DAVIE B., "Interprovider QoS for MPLS networks", 2004. Available at http://www.isocore.com/mpls2004/Abstracts/Tutorial4.htm.

[DOB 02] DOBSON G., "Quality of service in service-oriented architectures", 2002. Available at http://digs.sourceforge.net/papers/qos.html.

[DON 03] DONGARA P., VIJAYKUMAR T.N., "Accelerating private-key cryptography via multithreading on symmetric multiprocessors", *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS 03)*, IEEE Press, pp. 58–69, 2003.

[DOS 01] DOSTER W., WATTS M., HYDE D., The KX.509 protocol, 2001. Available at http://www.citi.umich.edu/techreports/reports/citi-tr-01-2.pdf.

[FLO 00] FLOYD S., HANDLEY M., PADHYE J., *et al.*, "Equation based congestion control for unicast applications", *Proceedings of the ACM SIGCOMM*, 2000.

[GAR 03] GARMAN J., *Kerberos: The Definitive Guide*, O'Reilly, 2003.

[GAV 66] GAVER D., "Observing stochastic processes, and approximate transform inversion", *Operation Research*, vol. 14, no. 3, pp. 444–459, 1966.

[GOL 04] GOLBECK J., HENDLER J., "Accuracy of metrics for inferring trust and reputation in semantic Web-based social networks", *Proceedings of International Conference on Knowledge Engineering and Knowledge Management (EKAW)*, October 2004.

[GRA 01] GRAY J., "Dependability in the Internet era", *Presentation at the High Dependability Computing Consortium Meeting*, May 2001. Available at http://research.microsoft.com/gray/talks.

[GUM 02] GUMMADI K., SAROIU S., GRIBBLE S., "King: estimating latency between arbitrary Internet end hosts", *Proceedings of the ACM SIGCOMM Internet Measurement Workshop*, November 2002.

[HAR 01] HARBITTER A., MENASCE D., "Performance of public-key-enabled Kerberos authentication in large networks", *Proceedings of 2001 IEEE Symposium on Security and Privacy*, Oakland, CA, 2001.

[HAR 02] HARRISON P., KNOTTENBELT W., "Passage time distributions in large Markov chains", *Proceedings of the ACM SIGMETRICS*, 2002.

[HE 05] HE L., WALRAND J., "Pricing and revenue sharing strategies for Internet service providers", *Proceedings of the IEEE INFOCOM*, 2005.

[HEN 99] HENNESSY J., "The future of systems research", *IEEE Computer*, vol. 32, no. 8, pp. 27–33, August 1999.

[HUR 01] HUR M., TUNG B., RYUTOV T., *et al.*, "Public key cryptography for cross-realm authentication in Kerberos (PKCROSS)", May 2001. Available at http://tools.ietf.org/html/draft-ietf-cat-kerberos-pk-cross-07.

[JUR 05] JURCA R., FALTINGS B., "Reputation-based service level agreements for Web services", *3rd International Conference on Service Oriented Computing (ICSOC 2005)*, Amsterdam, The Netherlands, December 2005.

[KAM 03] KAMVAR S.D., SCHLOSSER M.T., GARCIA-MOLINA H., "EignRep: reputation management in P2P networks", *Proceedings of the World-Wide Web Conference*, 2003.

[KAR 04] KARLAPUDI H., MARTIN J., "Web application performance prediction", *Proceedings of the IASTED International Conference on Communication and Computer Networks*, pp. 281–286, 2004.

[KEL 03] KELLER A., LUDWIG H., "The WSLA framework: specifying and monitoring service level agreements for Web services", *Journal of Network and Systems Management, Special Issue on E-Business Management*, vol. 11, no. 1, pp. 27–33, March 2003.

[KIM 00] KIM Y., PERRIG A., TSUDIK G., "Simple and fault-tolerant key agreement for dynamic collaborative groups", *Proceedings of the 7th ACM Conference on Computer and Communications Security (ACM CCS)*, ACM, pp. 235–244, 2000.

[KIM 09] KIM W., "Cloud computing: today and tomorrow", *Journal of Object Technology*, vol. 8, pp, 65–72, 2009.

[KOH 93] KOHL J., NEUMAN C., RFC 1510: the Kerberos network authentication service (V5), 1993. Available at http://www.ietf.org/rfc/rfc1510.txt.

[KX 07] KX.509., KX.509 source, 2007. Available at http://tools.ietf.org/html/rfc6717.

[LEE 02] LEE J., BEN-NATAN R., *Integrating Service Level Agreements : Optimizing Your OSS for SLA Delivery*, Addison-Wesley Publishing Company, Reading, MA, 2002.

[LEE 03] LEE S., SHERWOOD R., BHATTACHARJEE B., "Cooperative peer groups in NICE", *Proceedings of the IEEE INFOCOM*, 2003.

[LEV 03] LEVY R., NAGARAJARAO J., PACIFICI G., *et al.*, "Performance management for cluster based Web services", *8th IFIP/IEEE International Symposium on Integrated Network Management (IM2003)*, IEEE, 2003.

[LU 05] LU J., WANG J., "Performance modeling and analysis of Web switch", *Proceedings of the 31st Annual International Conference on Computer Measurement (CMG)*, 2005.

[MAR 02] MARTIN J., NILSSON A., "On service level agreements for IP networks", *Proceedings of the IEEE INFOCOM*, June 2002.

[MAT 03] MATTHYS C., *On Demand Operating Environment: Managing the Infrastructure (Virtualization Engine Update)*, IBM Redbooks, 2003.

[MAT 05] MATTHYS C., BARI P., LIEURAIN E., *et al.*, *On Demand Operating Environment: Managing the Infrastructure (Virtualization Engine Update)*, IBM Redbooks, June 2005.

[MED 97] MEDVINSKY A., HUR M., NEUMAN C., Public key utilizing tickets for application servers (PKTAPP), January 1997. Available at http://tools.ietf.org/html/draft-ietf-cat-pktapp-00.

[MEF 06] MEF 10.1, Ethernet Service Attributes Phase 2, 2006.

[MEI 06] MEI R.D., MEEUWISSEN H.B., "Modelling end-to-end Quality-of-Service for transaction-based services in multi-domain environment", *Performance Challenges for Efficient Next Generation Networks*, in LIANG X.J., XIN Z.H., IVERSEN V.B., KUO G.S. (eds), *Proceedings of the 19th International Teletraffic Congress (ITC19)*, 2006. Available at http://dl.acm.org/citation.cfm?id=1173081

[MEI 06] MEI R., MEEUWISSEN H., PHILLIPSON F., "User perceived Quality-of-Service for voice-over-IP in a heterogeneous multi-domain network environment", *Proceedings of the IEEE International Conference on Web Services (ICWS)*, 2006.

[MEN 02] MENASCE D., "QoS issues in Web services", *IEEE Internet Computing*, vol. 6, no. 4, pp. 72–75, November–December 2002.

[MEN 03] MENASCE D., BENNANI M., "On the use of performance models to design self-managing computer systems", *Proceedings of the 2003 Computer Measurement Group Conference*, IEEE, 2003.

[MEN 04a] MENASCE D., "Response-time analysis of composite Web services", *IEEE Internet Computing*, vol. 8, no. 1, pp. 90–92, January–February 2004.

[MEN 04b] MENASCE D., CASALICCHIO E., "A framework for resource allocation in grid computing", *Proceedings of the 12th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'04)*, IEEE, pp. 259–267, October 2004.

[MIC] MICROSYSTEMS S., "Service level agreement in the data center", 2000. Available at http://www.sun.com/blueprints/0402/sla.pdf.

[MUI 02] MUI L., MOHTASHEMI M., HALBERSTADT A., "A computational model of trust and reputation", *Proceedings of the 35th Hawaii International Conference on System Science*, 2002.

[MUN 75] MUNTZ R., CHANDY K., BASKETT F., *et al.*, "Open, closed, and mixed networks of queues with different classes of customers", *Journal of the ACM*, pp. 248–260, April 1975.

[MUP 94] MUPPULA J., TRIVEDI K., MAINKAR V., *et al.*, "Numerical computation of response time distributions using stochastic reward nets", *Annals of Operations Research*, vol. 48, pp. 155–184, 1994.

[NAB 04] NABRZYSBI J., SCHOPF J., WEGLARZ J., *Grid Resource Management*, Kluwer Academic Publication, Boston, MA, 2004.

[NAO 98] NAOR M., WOOL A., "The load, capacity, and availability of quorum systems", *SIAM Journal on Computing*, vol. 27, no. 2, pp. 423–447, 1998.

[NEU 96] NEUMAN B., TUNG B., WAY J., *et al.*, Public key cryptography for initial authentication in Kerberos servers (PKINIT-02), October 1996. Available at http://ietf.org/internet-drafts/draft-ietf-cat-Kerberos-pk-init-02.txt.

[NOW 02] NOWAK D., PERRY P., MURPHY J., "Bandwidth allocation for service level agreement aware Ethernet passive optical networks", *IEEE GLOBECOM 2004*, pp. 1953–1957, 2002.

[OSO 03] OSOGAMI T., WIERMAN A., HARCHOL-BALTER M., *et al.*, How many servers are best in a dual-priority FCFS system?, CMU technical report: CMU-CS-03-201, Carnegie Mellon University, November 2003.

[PER 94] PERROS H., *Queueing Network with Blocking, Exact and Approximate Solutions*, Oxford University Press, 1994.

[PER 01] PERRIG A., CANETTI R., SONG D., *et al.*, "Efficient and secure source authentication for multicast", *Proceedings of Network and Distributed System Security Symposium*, February 2001.

[PIE 71] PIESSENS R., "Gaussian quadrature formulas for the numerical integration of Bromwich's integral and the inversion of the Laplace transform", *Journal of Engineering Mathematics*, vol. 5, no. 1, pp. 1–9, 1971.

[PRE 97] PRESS W., FLANNERY B., TEUKOLSKY S., *et al.*, *Numerical Recipes in Fortran*, Cambridge University Press, 1997.

[RED 13] REDDY S., SHETTY S., XIONG K., "'Security risk assessment of cloud carriers'", *Proceedings of the 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing (CCGrid)*, 2013.

[REI 57] REICH E., "Waiting times when queues are in tandem", *Annals of Mathematical Statistics*, vol. 28, no. 3, pp. 768–773, 1957.

[REI 63] REICH E., "Notes on queues are in tandem", *Annals of Mathematical Statistics*, vol. 34, pp. 338–341, 1963.

[RIC 03] RICHARDSON M., AGRAWAL R., DOMINGOS P., "Trust management for the semantic Web", *Proceedings of the 2nd International Semantic Web Conference*, 2003.

[ROS 04] ROSENBERG J., REMY D., *Securing Web Services with WS-Security: Demystifying WS-Security, WS-Policy, SAML, XML, Signature, and XML Encryption*, SAMS, Indianapolis, IN, 2004.

[SHA 88] SHANTHIKUMAR J.G., YAO D.D., "On server allocation in multiple center manufacturing systems", *Operations Research*, vol. 36, no. 2, pp. 333–342, November 1988.

[SHE 12] SHETTY S., LUNA N., XIONG K., "Assessing network path vulnerabilities for secure cloud computing", *Proceedings of the IEEE ICC Workshop on Clouds, Networks and Data Centers*, 2012.

[SHI 06] SHIN M., CHONG S., RHEE I., "Dual-resource TCP/AQM for processing-constrained networks", *Proceedings of the IEEE INFOCOM*, April 2006.

[SIN 03] SINGH M.P., HUHNS M.N., *Service-Oriented Computing*, John Wiley & Sons, Ltd., 2003.

[SIR 97] SIRBU M., CHUANG J., "Distributed authentication in Kerberos using public key cryptography", *IEEE Symposium on Network and Distributed System Security (NDSS' 97)*, 1997.

[SIR 06] SIRIPONGWUTIKORN P., BANERJEE S., "Per-flow delay performance in traffic aggregates", *Proceedings of the IEEE GLOBECOM*, 2006.

[SKE 84] SKEEN D., WRIGHT D.D., "Increasing availability in partitioned database systems", *Proceedings of PODS*, pp. 290–299, 1984.

[SLO 95] SLOTHOUBER L., "A model of Web server performance", 1995. Available at www.geocities.com/webserverperformance.

[SOM 05] SOMMERS J., BARFORD P., DUFFIELD N., *et al.*, "Improving accuracy in end-to-end packet loss measurement", *Proceedings of the ACM SIGCOMM*, August 2005.

[STE 70] STEHFEST H., "Algorithm 386, numerical inversion of Laplace transforms", *Communications of the ACM*, vol. 13, no. 1, pp. 47–49, January 1970.

[TEC 09] TECH, What is cloud computing, 2009. Available at http://jobsearchtech.about.com/od/historyoftechindustry/a/cloudcomputing.htm.

[TOS 02] TOSIC V., PATEL K., PAGUREK B., "WSOL – Web service offerings language", *Lecture Notes In Computer Science*, Revised Papers from the International Workshop on Web services, E-Business, and the Semantic Web, vol. 2512, pp. 57–67, 2002.

[UH 12] UH, Network availability, 2012. Available at http://www.telecomm.uh.edu/stats/netlogs/NetStatus/Averagedaily.txt.

[VAQ 09] VAQUERO L., RODERO-MERINO L., CACERES J., *et al.*, "A break in the clouds: towards a cloud definition", *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 50–55, 2009.

[VU 05] VU L., HAUSWIRTH M., ABERER K., QoS-based service selection and ranking with trust and reputation management, Infoscience's techical report, 2005. Available at http://infoscience.epfl.ch/search.py?recid=52732, 2005.

[WAL 80] WALRAND J., VARAIYA P., "Sojourn times and the overtaking condition in Jacksonian networks", *Advances in Applied Probability*, vol. 12, no. 4, pp. 1000–1018, 1980.

[WIK 09] WIKIPEDIA, "Cloud computing", 2009. Available at http://en.wikipedia.org/wiki/Cloudcomputing.

[XIA 99] XIAO X., NI L.M., "Internet QoS: a big picture", *IEEE Network*, vol. 13, pp. 8–18, March–April 1999.

[XIO 06a] XIONG K., PERROS H., "SLA-based service composition in enterprise computing", *Proceedings of the 16th IEEE International Workshop on Web Services (IWQoS)*, Enschede, The Netherlands, 2008.

[XIO 06b] XIONG K., PERROS H., "SLA-based resource allocation in cluster computing systems", *Proceedings of the 22nd IEEE International Parallel and Distributed Proceeding Symposium (IPDPS)*, Miami, Florida, 2008.

[XIO 06c] XIONG K., PERROS H., "Computer resource optimization for differentiated customer services", *Proceedings of Conference on Measurement and Simulation of Computer and Telecommunication Systems (MASCOTS)*, September 2006.

[XIO 06d] XIONG K., PERROS H., "Resource optimization subject to a percentile response time SLA for enterprise computing", *Proceedings of IEEE Globecom: Quality Reliability and Performance Modeling for Emerging Network Services (Globecom-QRPM)*, November–December 2006.

[XIO 06e] XIONG K., PERROS H., "Trust-based resource allocation in Web services", *Proceedings of IEEE International Conference on Web Services (ICWS)*, September 2006.

[XIO 07] XIONG K., Resource optimization and security in distributed computing, Ph.D. thesis, North Carolina State University, 2007.

[XIO 09] XIONG K., PERROS H., BLAKE S., "Bandwidth provisioning in ADSL access networks", *Journal of Network Management*, vol. 19, no. 5, pp. 427–444, 2009.

[XIO 09a] XIONG K., "'Multiple priority customer service guarantees in cluster computing", *Proceedings of the 23rd IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, May 2009.

[XIO 09b] XIONG K., PERROS H., "Service performance and analysis in cloud computing", *Proceedings of the 2009 Congress on Services - I*, SERVICES '09, IEEE Computer Society, Washington, DC, 2009.

[XIO 10a] XIONG K., SUH S., "Resource provisioning in SLA-based cluster computing", *The 15th International Workshop in Job Scheduling Strategies for Parallel Processing (JSSPP)*, Lecture Notes in Computer Science 6253, Springer, Atlanta, GA, April 2010.

[XIO 10b] XIONG K., "'Power-aware resource provisioning in cluster computing", *Proceedings of the 24th IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, May 2010.

[XIO 11a] XIONG K., KANG K., CHEN X., "A priority-type approach for resource allocation in cluster computing", *Proceedings of the 13th IEEE International Conference on High Performance Computing and Communications (HPCC)*, 2011.

[XIO 11b] XIONG K., "Power and performance management in priority-type cluster computing systems", *Proceedings of the 25th IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, May 2011.

[XIO 13] XIONG K., HE Y., "Power-efficient resource allocation in MapReduce clusters", *Proceedings of the 13th IEEE/IFIP Integrated Network Management Symposium (IM)*, 2013.

[ZHA 04] ZHANG Q., YU T., IRWIN K., "A classification scheme for trust functions in reputation-based trust management", *International Workshop on Trust, Security, and Reputation on the Semantic Web*, Hiroshima, November 2004.

[ZHA 05] ZHANG J., ZHANG L.J., "Editorial preface: a framework to ensure trustworthy Web services", *International Journal of Web Services Research*, vol. 2, no. 3, pp. 1–7, July–September 2005.

[ZHU 06] ZHU L., TUNG B., RFC 4556: Public key cryptography for initial authentication in Kerberos (PKINIT), June 2006. Available at http://www.ietf.org/rfc/rfc4556.txt.

[ZIE 02] ZIEGLER C., LAUSEN G., "Spreading activation models for trust propagation", *IEEE International Conference on e-Technology, e-Commerce, and e-Service (EEE '04)*, April 2002.

Index

Q, S

T, U, W