Availability modeling and evaluation of web-based services

Magnos Martinello

▶ **To cite this version:**

Magnos Martinello. Availability modeling and evaluation of web-based services. Computer Science [cs]. Institut National Polytechnique (Toulouse), 2005. English. NNT : 2005INPT043H . tel-04595476

HAL Id: tel-04595476
https://hal.science/tel-04595476v1

Submitted on 31 May 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

N° d'ordre : 2275 Année 2005

THÈSE

préparée au
Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS

en vue de l'obtention du
Doctorat de l'Institut National Polytechnique de Toulouse
Ecole doctorale : Systèmes
Spécialité : Systèmes Informatiques

par

Magnos MARTINELLO

Modélisation et évaluation de la disponibilité de services mis en œuvre sur le web - Une approche pragmatique

Availability modeling and evaluation of web-based services - A pragmatic approach

---

Soutenue le 4 novembre 2005 devant le jury :

Président	M.	**Daniel**	**NOYES**
Directeur de thèse	M.	**Mohamed**	**KAÂNICHE**
Rapporteurs	M.	**Yves**	**DUTUIT**
	M.	**András**	**PATARICZA**
Examinateurs	M.	**Nicolae**	**FOTA**
	Mme.	**Karama**	**KANOUN**

Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS
7, avenue du Colonel Roche, 31077 Toulouse Cedex 4

Rapport LAAS N° 0000

In Memory of my grandmother Leocadia.

Avant-propos

Les travaux présentés dans ce mémoire ont été effectués au Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS. Je remercie Messieurs Jean-Claude Laprie et Malik Ghallab, qui ont assuré la direction du LAAS-CNRS depuis mon entrée, de m'avoir accueilli au sein de ce laboratoire.

Je remercie également Messieurs David Powell et Jean Arlat, Directeurs de Recherche CNRS, responsables successifs du groupe de recherche Tolérance aux fautes et Sûreté de Fonctionnement informatique (TSF), de m'avoir permis de réaliser mes travaux dans ce groupe.

J'exprime ma profonde reconnaissance à Mohamed Kaâniche, Chargé de Recherche CNRS, pour avoir dirigé mes travaux de thèse, pour ses conseils, sa détermination impressionnante et notamment une méthodologie remarquable. Je suis également très reconnaissant à Karama Kanoun, Directeur de Recherche CNRS, pour ses orientations et ses critiques constructives. Je la remercie pour ses remarques pertinentes et pour avoir su canaliser mon énergie. J'ai tiré de nombreux enseignements de leur compétence et de leur expérience.

Je remercie Monsieur Daniel Noyes, Professeur à l'École Nationale des Ingénieurs de Tarbes, pour l'honneur qu'il me fait en présidant mon jury de thèse, ainsi que:

- Monsieur Yves Dutuit, Professeur à l'Université de Bordeaux I
- Monsieur Nicolae Fota, Responsable du Pôle Safety à SOFREAVIA
- Monsieur Mohamed Kaâniche, Chargé de Recherche CNRS
- Madame Karama Kanoun, Directeur de Recherche CNRS
- Monsieur András Pataricza, Professeur à Budapest University of Technology and Economics

pour l'honneur qu'ils me font en participant à mon jury. Je remercie tout particulièrement Messieurs Dutuit et Pataricza d'avoir accepté la charge d'être rapporteurs.

Les travaux développés dans cette thèse ont été partiellement effectués dans le cadre du projet européen DSOS - Dependability Systems of Systems et financés en partie par la CAPES. J'exprime ma vive reconnaissance à la société brésilienne d'avoir soutenu mes études à travers la bourse de doctorat fournie par la CAPES.

Je remercie vivement tous les membres du groupe TSF, permanents, doctorants et stagiaires. C'était un grand plaisir et je dirais un privilège d'échanger des idées, de discuter, de rigoler, enfin de vivre cette expérience parmi vous. J'associe à ces remerciements Joëlle, Gina et Jemina pour leur collaboration dans l'organisation de déplacements et, surtout, les derniers préparatifs de la soutenance.

Les différents services techniques et administratifs du LAAS-CNRS, par leur efficacité et leur disponibilité, m'ont permis de travailler dans d'excellentes conditions. Je les remercie sincèrement.

Il m'est impossible de ne pas mentionner le front philosophique de libération du bureau 20. J'ai trouvé de véritables compagnons de route qui ont su maintenir l'ambiance de travail plutôt décontractée et animée.

Un grand merci à tous mes amis qui ont été toujours là, même virtuellement et avec qui j'ai partagé des moments formidables. Vous étiez une source de motivation et grâce à vous je me suis rechargé tout en gardant mon enthousiasme.

Eu quero deixar um agradecimento especial à todos os brasileiros, latinos e agregados com quem eu pude passar vários momentos maravilhosos na França e fora da França. Eu posso dizer sinceramente que esta lista é enorme, e prefiro resumir dizendo que tive uma sorte imensa de encontrar pessoas como vocês, porque simplesmente vocês são fantásticos.

Je remercie ma famille pour leur encouragement permanent, malgré l'océan qui nous sépare. Apesar de fisicamente estarmos distantes, vocês estiveram presentes em pensamento e graças à vocês eu cheguei onde cheguei.

Enfin, mon amour, celle qui a tout partagé avec moi pendant ces années d'études. Robertinha, você é a minha fonte de inspiração e a pessoa mais especial deste mundo. Eu só posso te dizer do fundo do meu coração: obrigado por me amar.

Acknowledgments

First of all, I would like to express my sincere gratitude to all the people who made this work together with me, participating intensively along these four years of adventure. Four years of intensive learning with a mix of excellent experiences and difficulties, but certainly a wonderful period of my live.

My gratitude to the directors of the LAAS-CNRS, Jean-Claude Laprie and Malik Ghallab and also to David Powell and Jean Arlat the successive heads on dependable computing and fault tolerance research group (TSF), for their support allowing me to work in the best conditions.

A very special thanks to the main responsibles of this adventure. Mohamed Kaâniche for being an advisor always ready to give all his energy with an inestimable determination and a remarkable methodology. Karama Kanoun for leading our discussions with her strategic vision, constant help and guidance in the research. I need to say that my multiple intrusions in their office were answered with availability and much attention. Moreover, the remarks and the way both conducted this work were not only fundamental to this thesis, but also they had a special value for me in the process of formation.

My sincere thanks to the people of group TSF with whom I had the pleasure of interacting and collaborating not only on technical aspects but specially personally. I think the most important lesson learned is how to do scientific research.

Special thanks go to my friends. For me, you are like brothers and sisters and an incredible source of motivation.

Finally, I thank my family for their love and support during these years of study. My parents Darci and Silvia two of the most wonderful people in the world. Tiago and Diego who are fantastic brothers. My wife Robertinha, who is my inspiration and the most special person in the world. I just thank you from the bottom of my heart for loving me.

Contents

List of Figures

List of Tables

Introduction

Deal with the faults of others as gently as with your own.

Chinese proverb

OVER the past years, the Internet has become an huge infrastructure used daily by millions of people in the world. The world wide web (www or web) is a publishing medium used to disseminate information quickly through this infrastructure. The web has had a rapid growth in size and usage, with an extensive development of web sites delivering a large variety of personal, commercial and educational material. Virtual stores on the web allow to buy books, cds, computers, and many other products and services. New web applications such as e-commerce, digital libraries, video on-demand and distance learning make the issue of dependability evaluation increasingly important, in particular with respect to the service perceived by web users.

Businesses and individuals are increasingly depending on web-based services for any sort of operations. Web-based services [1] connect departments within organizations, multiple companies and the population in general. In addition, the web is often used for critical applications such as online banking, stock trading, booking systems requiring high availability and performance of the service. In those applications, a temporary service unavailability may have unacceptable consequences in terms of financial losses.

A period of service unavailability may cost millions to the site depending on the duration and on the importance of the period [2]. It may be very difficult to access a major newspaper or TV site after some important news due to site overload. Those periods are usually most important to the web service provider because unplanned

[1] In this thesis, web services refer to the services delivered by a web application. Such services do not refer to the standards and protocols proposed by W3C and OASIS (both responsible for the architecture and standardization of web services).

[2] According to [Patterson et al. 2002] well-managed servers today achieve an availability of 99.9% to 99%, or equivalently between 8 to 80 hours of downtime per year. Each hour can be costly e.g. $200,000 per hour for Amazon.

unavailability is more expensive than planned unavailability [Brewer 2001]. For instance, ebay web site was unavailable for a period of 22 hours on June 1999, leading to a lost revenue of approximatively 5 billion dollars.

Dependability analysis and evaluation methods are useful to understand, analyze, design and operate web-based applications and infrastructures. Quantitative measures need to be evaluated in order to estimate the quality of service and the reliance that can be placed on the provided service. This evaluation may help web site designers to identify the weak parts of the architecture which can be used for improving the provided service. Dependability evaluation consists in estimating quantitative measures allowing to analyze how hardware, software or human related failures affect the service dependability. It can be carried out by dependability modeling with the goal of analyzing the various design alternatives in order to choose the final solution that better satisfies the requirements.

Evaluation can be carried out using two complementary approaches: i) *measurement* and ii) *modeling*. Measurements provide information for characterizing the current and past behavior of already existing systems. Recently, much research effort has been devoted to the analysis of service availability using measurements based on monitoring of operational web sites. On the other hand, it is fundamental to anticipate future behavior about the infrastructure supporting the service. Modeling is useful to guide the development of a web application during its design phase by providing quantitative measures characterizing its dependability. In the context of web-based services, modeling has been mainly used for performance evaluation purposes. However, less attention has been devoted to the dependability modeling and evaluation of web-based services and applications, specially from the user perceived perspective.

The user perceived availability of web-based services is affected by a variety of factors (e.g., user behaviors and workload characteristics, fault tolerance and recovery strategies, etc.). Due to the complexity of the web-based services and the difficulty to combine various types of information, a systematic and pragmatic modeling approach is needed to support the construction and processing of dependability models. The contributions presented in this thesis are aimed at fulfilling these objectives, introducing a pragmatic approach for analyzing the availability of such services from the user point of view.

The traditional notion of availability is extended in order to include some of the main causes of service unavailability relying on a performability modeling approach. Our modeling approach is based on a combination of Markov reward models and queuing theory, in which we investigate the potential existing closed-form equations. In fact, analysis from pure performance viewpoint tends to be optimistic because it ignores the failure/repair behavior of the system. On the other hand, pure dependability analysis that does not include performance levels of service delivery tends to be too conservative. Therefore, a performability based approach is well suited to capture the various degraded states, measuring not only whether the service is up or down but also operational degraded states. The causes of service unavailability considered explicitly in our modeling based approach fall into the following categories: i) hardware and

software failures affecting the servers; and ii) performance-related failures including: overload, loss of requests, and long response time.

Thesis outline

The core of this thesis deals with web service availability modeling and evaluation and is structured in four chapters. Figure 1 presents the interdependence of chapters.

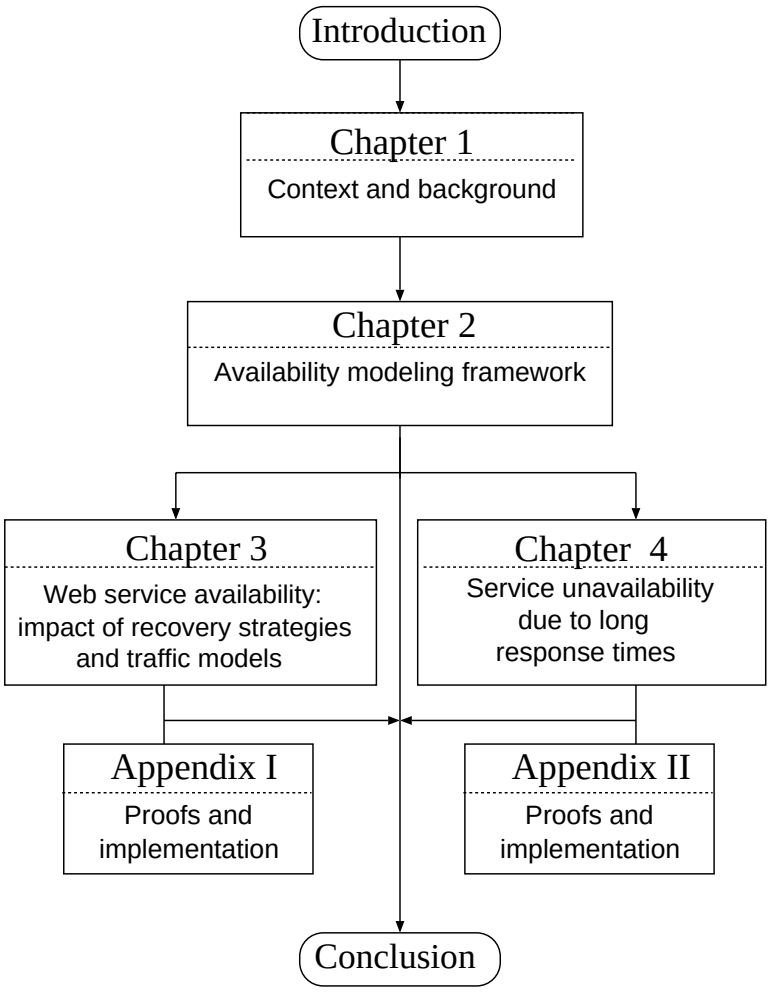


Figure 1: Interdependence of chapters

Chapter 1 states the motivation and the context of the work. It briefly presents the theory and techniques for dependability modeling providing a background for our investigation. We provide a discussion on dependability modeling starting by the main concepts of dependability. A brief overview of dependability evaluation is presented with some existing methods useful to build and to solve models. Some approaches in the probabilistic evaluation domain are reported. Also, the related works are reviewed presenting prior studies and contributions on web availability evaluation including measurements and modeling approaches.

The general problem addressed in this thesis is introduced in Chapter 2. This chapter presents the proposed framework using a web-based travel agency as example, illustrating the main concepts and the feasibility of the framework. The framework is based on the decomposition of the web based application following a hierarchical description. The hierarchical description is structured into four levels of abstraction. The highest level describes the dependability of the web application as perceived by the users. Intermediate levels describe the dependability of functions and services provided to the users. The lowest level describes the dependability of the component systems on which functions and services are implemented. Sensitivity analyses are presented to show the impact of users operational profile, the fault coverage and the travel agency architecture on user perceived unavailability.

Chapter 3 provides a modeling based approach of web service availability supported by web cluster architectures. We are particularly interested in fault-tolerant web architectures. Our interest is justified by the fact that web clusters architectures are leading architectures for building popular web sites. Web designers require to find an adequate sizing of these architectures to ensure high availability and performance for the delivered services. Moreover, it is crucial to study the impact of recovery strategies supported by these architectures on web service availability.

Thus, we address especially recovery strategies issues and traffic burstiness effects on web service availability. Web cluster architectures are studied taking into account the number of nodes, recovery strategies after a node failure and the reliability of the nodes. Various causes of request loss are considered explicitly in the web service availability measure: losses due i) to buffer overflow or ii) to node failures, or iii) during recovery time. Closed-form equations for request loss probability are derived for both recovery strategies. Two simple traffic models (Markov Modulated Poisson Process (MMPP) and Poisson) are used to analyze the impact of traffic burstiness on web service availability.

From the user perspective, the service is perceived as degraded or even unavailable if the response time is too long compared to what the users are expecting. Certainly, the long response time has an impact on the overall service availability. To our knowledge, however, there has not been a quantitative evaluation of the long response time effects on service availability especially from the web user perspective. Chapter 4 introduces a flexible analytic modeling approach for computing service unavailability due to long response times. The proposed approach relies on Markov reward models and queuing theory. We introduce a mathematical abstraction that is general enough

to characterize the unavailability behavior due to long response times. The computation of the service unavailability measure is based on the evaluation of the response time distribution. Closed-form equations are derived for conditional response-time distribution and for the service unavailability due to long response time, considering single and multi-server queueing systems.

The developed models are implemented using tools such as gnu-octave and maple. Since we specifically focus on small models aiming to obtain closed-form equations as much as possible, these tools are enough for evaluating the obtained equations and for supporting the sensitivity analyses. Appendix I and II show the obtained equations proofs as well as the models implementation of the chapters 3 and 4.

Chapter 1

Context and background

> All models are wrong. Some models are useful.
>
> Albert Einstein

THIS chapter presents the motivation and the context of our work. We provide a discussion on dependability evaluation starting by the main concepts of dependability. A brief overview of dependability evaluation is presented through some existing methods useful to build and to solve models. Various approaches used in probabilistic evaluation domain are non-exhaustively reported, illustrating the considerable advances that have extended the capabilities of analytic models.

We introduce the modeling process indicating its phases. The phases are described presenting some of the main problems and methods used in each phase. After that, two major problems related to models construction and processing are discussed: largeness and stiffness. We report some approaches useful to build large models. Finally, we review the related studies that form the basis for our investigation on web availability evaluation including measurements and modeling approaches.

1.1 Context and motivation

The web is an evolving system incorporating new components and services at a very fast rate. A large number of new applications such as e-commerce, digital libraries, video on-demand, distance learning have been migrated to the web. Many web site projects are built in three or four months because they need to beat competitors and quickly establish a web presence. This requirement of becoming visible online often

comes without a careful design and testing, leading to some problems on dependability and performance.

Recently, many high-tech companies providing service on the web have experienced operational failures. Financial web services experienced intermittent outages as the volume of visitors has increased. A report presented in [Meehan 2000] showed that online brokerage companies were concerned with system outages and with the inability to accommodate growing numbers of online investors. During those outages, users and investors could not access real-time quotes. Such operational problems have resulted, in many cases, in degraded performance, unsatisfied users and heavy financial losses.

Quantitative methods are needed to understand, analyze, design and operate such large infrastructure. Quantitative measures need to be evaluated in order to estimate the quality of service and the reliance that can be placed on the provided service. This evaluation may help system designers to identify the weak parts of the system that should be improved to support an acceptable dependability level for the provided service. Dependability evaluation consists in estimating quantitative measures allowing to analyze how hardware, software or human related failures affect the system dependability. It can be carried out by dependability modeling with the goal of analyzing the various design alternatives in order to choose the final solution that better satisfies the requirements.

The rest of this chapter is structured as follows. Section 1.2 presents the main concepts of dependability. Section 1.3 outlines the main approaches that can be used for dependability evaluation. Section 1.4 reports some formalisms and tools used in the analytic modeling process. Section 1.5 describes some of the main problems related to large models and the existing techniques to deal with large models. Section 1.6 presents the related work on the evaluation of web-based services. Finally, section 1.7 summarizes the chapter.

1.2 Dependability concepts

Dependability is defined as the trustworthiness of a computer system such that reliance can justifiably be placed on the service it delivers [Laprie 1995, Laprie et al. 1996, Avizienis et al. 2004]. It is a global concept which includes various notions that can be grouped into three classes: *threats, means and attributes*. Dependability goal is to specify, conceive and investigate systems in which a fault is natural, predictable and tolerable.

The *threats* to dependability are: *faults, errors and failures*; they are undesired - but not in principle unexpected - circumstances causing or resulting from undependability.

The *means* for dependability are: *fault prevention, fault tolerance, fault removal and fault forecasting or prediction*; these are the methods and techniques that enable one

a) to provide the ability to deliver a service on which reliance can be placed, and b) to reach confidence in this ability.

The *attributes* of dependability are: *availability, reliability, safety, confidentiality, integrity, maintainability and security*.

- **Reliability**: continuity of correct service;
- **Availability**: readiness for correct service;
- **Safety**: absence of catastrophic consequences on the user(s) and the environment;
- **Confidentiality**: absence of unauthorized disclosure of information;
- **Integrity**: absence of improper system alterations;
- **Maintainability**: ability to undergo modifications and repairs;

Security is a composition of the attributes: confidentiality, integrity and availability requiring the concurrent existence of a) availability for authorized actions only, b) confidentiality, and c) integrity with 'improper' meaning 'unauthorized'.

In this study, we concentrate our attention on the *fault forecasting* domain taking into account only accidental faults. Fault forecasting is conducted by performing an *evaluation* of the system behavior with respect to fault occurrence or activation. Evaluation aims to estimate the presence, the creation and the consequences of faults or errors on dependability.

1.3 Dependability evaluation

Fault forecasting is performed using evaluations of the behavior of a system relative to the occurrence of faults and their activation. By adopting a structural view of a system, evaluation consists in reviewing and analyzing the failures of its components and their consequences on the system's dependability. Evaluation can be conducted in two ways [Laprie et al. 1996]:

- *Ordinal evaluation*: aims to identify, list and rank failures, or the methods and techniques implemented to avoid them;
- *Probabilistic evaluation*: aims to evaluate in terms of probabilities the degree of satisfaction of certain attributes of dependability;

The probabilistic evaluation is basically carried out using two complementary approaches: i) *measurement based evaluation* and ii) *modeling based evaluation*.

In the area of measurement, it is possible to distinguish two techniques:

- Fault injection: consists in injecting faults in a target system which is usually not operational, aiming to speed up its behavior in the presence of faults [Arlat et al. 1993]. This technique is based on controlled experiments which are subjected to a given workload;
- Operational observation: consists in collecting data characterizing the behavior of a target system during its operational execution. The estimation of the dependability measures is based on the statistical processing of failure and recovery events extracted from the data. Such analysis can be carried out based on the error logs maintained by the operating system [Simache & Kaâniche 2002, Simache 2004];

Measurements provide the most accurate information on already existing systems, although sometimes it is difficult to ensure by measurements that a system meets the dependability requirements. For example, in the case of highly reliable systems waiting for the system to fail enough times to obtain statistically significant samples would take years. Nevertheless, measurements data can be used for model validation, for instance to calibrate the input of the model parameters. In the case of a system that does not exist yet, the parameters could be estimated from measurements on similar systems. Consequently, the better is to combine both measurement and modeling as much as possible.

Modeling is useful to guide the development of the target system during the design phase by providing quantitative measures characterizing its dependability. Various design alternatives can be analyzed in order to choose the final solution that better satisfies the requirements. In particular, models can enable the designers i) to understand the impact of various design parameters from the dependability point of view, ii) to identify potential bottlenecks, iii) to aid in upgrade studies. These reasons make the modeling attractive in the probabilistic evaluation.

We can distinguish essentially two main modeling techniques: *simulation* and *analytical* modeling. Discrete-event simulation can be applied to almost all problems of interest, but the main drawback is that it can take a long time to run, mainly for large and realistic systems with high accuracy. In particular, it is not trivial to simulate with high confidence scenarios mainly in the presence of rare events [Heidelberger 1995, Heidelberger et al. 1992], even if rare events can be speeded up in a controlled manner using for example importance sampling techniques [Nicola et al. 1990].

A designer faces many issues that should be considered at the same time when choosing an adequate method. Simplicity, accuracy, computational cost and availability of information about the system under study are some factors that can be used to determine the well-suited method. Analytic-based approach favors simplicity allowing to cover essential aspects of the target system. Indeed, analytical models can be a cost-effective alternative to provide relatively quick answers to ”what-if” questions giving more insights for the system being studied.

Clearly, analytical models often use simplifying assumptions in order to make them tractable. However, considerable advances have extended the capabilities of analytic models. Our work explores how analytic-based models can be used for dependability evaluation of web-based services.

1.4 Modeling process

In the modeling process, it is possible to identify the major phases as well as the interdependencies among the phases. A modeling process can be structured basically into four phases as shown in the right side of Figure 1.1: choice of measures, model construction, model solution and model validation.

- The **choice of measures** consists in determining the measures of predominant importance reflecting the evaluation objective according to system characteristics and application context. The application context determines the type of information that should be represented. Also, there are certain **requirements** that are directly related to the choice of measures. System requirements can be defined as the desired qualities explicitly or implicitly specified that need to be taken into account.
- The second step consists in selecting the most adapted modeling technique allowing the evaluation of the desirable measures and the building of an adequate model. This step is referred to as **model construction**.
- Once a suitable model has been developed, we need to solve the model. **Model solution** corresponds to the computation of the model in order to evaluate the measures, e.g. in terms of probabilities.
- **Model validation** is the task of checking model assumptions and its correctness. It is executed in an iterative process involving experts trying to reach consensus into the validation of the model assumptions and results.

Whenever possible, the *computed measures* should be compared against *operational measures* obtained from measurement based evaluation executed in a *real environment* as indicated in Figure 1.1. The measurement results are used not only for model validation but also for providing numerical values for the model parameters.

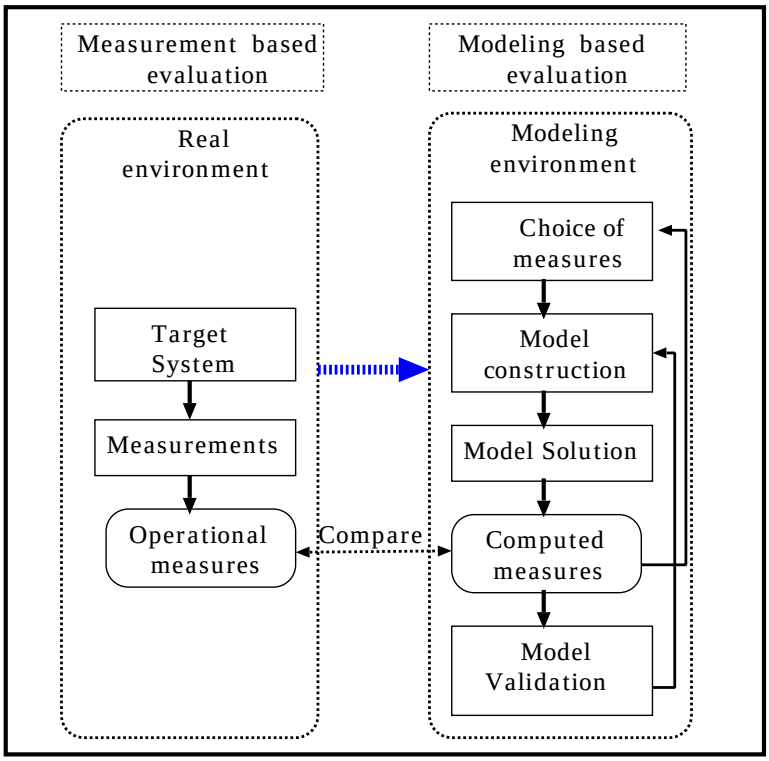


Figure 1.1: A simplified view of the modeling process and its interactions with measurement based evaluation

1.4.1 Dependability measures

The life of a system is perceived by its user(s) as an alternation between two states of the delivered service relative to the accomplishment of the system function:

- Correct service, where the delivered service accomplishes the system function;
- Incorrect service, where the delivered service does not accomplish the system function;

A service failure is an event that occurs when the delivered service deviates from correct service. A service failure is thus a transition from a state of correct service to a state of incorrect service. In contrast, the transition from incorrect to a correct service is a *restoration*. The period of delivery of incorrect service is an *outage*. Quantifying the alternation of correct-incorrect service enables some measures of dependability to be defined [Avizienis et al. 2004]:

- **Reliability**: a measure of the continuous delivery of correct service or, equivalently the time to failure;
- **Availability**: a measure of the delivery of correct service with respect to the alternation of correct and incorrect service. In other words, the system alternates through periods in which it is operational - the up periods - and periods in which it is down - the down periods;
- **Maintainability**: a measure of the time to restoration since the last failure, or equivalently, the continuous delivery of an incorrect service;

However, a system might have various *modes of service delivery*. These modes can be distinguished, ranging from full capacity to complete interruption going through different states of degradation. The modes can be classified as correct or incorrect according to the service requirement of the applications. From the dependability evaluation point of view, it is possible to consider two classes of systems:

- *Degradable* systems for which there are several modes of service delivery;
- *Non-degradable* systems that have only one mode of service delivery;

Fault-tolerant systems supporting recovery strategies with service restoration are examples of degradable systems. Such systems are usually capable of providing multi modes of service delivery achieving acceptable levels of performance (multi-performing systems). Dependability evaluation of those systems can be carried out considering different classes of service degradation.

It is important to notice that modeling any system with either a pure performance model or a pure dependability model can lead to incomplete or even misleading results. Analysis from pure performance viewpoint tends to be optimistic since it ignores

the failure/repair behavior of the system. On the other hand, pure dependability analysis taking into account system component failures tends to be too conservative if it does not consider the performance levels related to different modes of service delivery.

The combination of performance related measures and dependability measures is usually called **performability** [Meyer 1980, Meyer 1982], which is well suited to evaluate the impact of service degradation on system dependability. It allows to consider the performance of a given system based on its different modes of service delivery. Performability measures characterize the ability of fault-tolerant systems to perform certain tasks in the presence of failures, taking into account component failures and performance degradation.

In the following sections, we discuss in more details some of the main problems and methods used in model construction, solution and validation phases.

1.4.2 Model construction

Model construction is usually based on analytical or graphical description of the system behavior. Either information about a real computer system is used to build the model, or experiences gained in earlier modeling studies are implicitly used. Certainly, this process is rather complicated and it needs both modeling and system expertise.

Analytical modeling has proved to be an attractive technique. A model is an abstraction of a system that includes sufficient details to facilitate the understanding of the system behavior. Analytical models can be classified into *state space models* (e.g. Markov chains and extensions) and *non-state space models* (e.g. fault trees, reliability block diagrams and reliability graphs).

Non-state space models are commonly used to study system dependability [Vesely et al. 1981, Trivedi 2002, Dutuit & Rauzy 2000, Dutuit & Rauzy 1998]. They are concise, easy to understand, and have efficient solution methods. However, features such as non-independent behavior of components, imperfect coverage, and combination with performance can not be captured easily by these models. In the performance modeling, examples of non-state space models are precedence graphs (see the book [Roche & Schabes 1997] for a formal definition) and product form queueing networks [Jackson 1963]. A precedence graph is a directed and acyclic graph that can be used to model concurrency for the case of unlimited resources. On the other hand, contention for resources can be represented by a class of queueing networks known as product form queueing networks, for which there are efficient solution methods [Buzen 1973, Baskett et al. 1975, Reiser & Lavenberg 1980] allowing to derive steady state performance measures. However, they cannot model concurrency, synchronization, or server failures, since these violate the product form assumptions [Popstojanova & Trivedi 2000].

State space models enable us to overcome the limitations of the non-state space models by modeling complicated interactions among the components taking into

account the stochastic dependencies [Rabah & Kanoun 2003, Fota et al. 1999]. Also, they allow evaluation of different measures of interest. This evaluation can provide useful tradeoffs. Markov chains are the state space models most commonly used for modeling purposes. They provide great flexibility for modeling dependability, performance, and combined dependability and performance measures.

Markov chains can also have rewards assigned to states or to transitions between states of the model. Reward structures refine the possibilities of evaluation. Markov reward models have been used in Markov decision theory to assign cost and reward structures to states [Howard 1971a]. Meyer [Meyer 1980] adopted Markov reward models to provide a framework for an integrated approach to allow combined performance and dependability evaluation. By extension, it is equally well-suited to the evaluation of systems with several modes of service delivery, notably through the computation of performability type measures. In practice, modeling tools have been developed to deal with Markov reward models. For instance, TANGRAM-II [Carmo et al. 1998] supports performability-oriented analysis using object-oriented language [Berson et al. 1991].

Petri nets and their extensions (e.g. stochastic Petri nets, reward Petri nets) represent another formalism extensively used for model construction [Florin & Natkin 1985]. They provide a higher-level model specification from which a Markov chain is generated. When model specification becomes too complex (i.e., models become huge to handle manually), it is necessary to use a modeling tool. In the case of Petri nets and their extensions, there are a number of tools suitable for modeling and evaluation. Among these tools are the following: SURF2 [C.Béounes et al. 1993], SPNP [Ciardo et al. 1989], ULTRASAN [Couvillion et al. 1991] and DEEM [Bondavalli et al. 2000].

In addition to Petri nets, recent research studies have focused on the generation of dependability models from high level specification formalisms and languages such as UML [Bondavalli et al. 2001a, Pataricza 2002, Majzik & Huszerl 2002], ALTARICA [Signoret et al. 1998] and AADL [Rugina et al. 2005, Rugina 2005]. The objective is to integrate dependability modeling into the industrial development processes. It is achieved by i) the enrichment of the functional specification and design models with dependability related informations and assumptions and ii) the definition of transformation rules allowing to generate dependability evaluation models (fault trees, GSPNs, Markov chains) from the enriched model.

In this dissertation, we rely on Markov chains and Markov reward models as an abstraction model representing web-based services. Nevertheless, combined approaches using state space and non-state space based models are used for taking advantage of the different methods (e.g., the mixed reliability block diagrams with Markov chains structured in a multi-level hierarchical way).

1.4.3 Model solution

The algorithms for computation the state probabilities can be divided into those applicable for computing the steady-state probabilities and those applicable for computing the time-dependent state probabilities. The analyses presented in all chapters of this dissertation assume that the systems being studied are in *operational equilibrium* or at *steady-state* [Kleinrock 1975]. We compute the *steady-state probabilities* of the underlying model. The goal is to evaluate measures of interest, such as availability, reliability or performability.

Most techniques commonly used for model solution at steady-state fall into the following categories: *direct* or *iterative* numerical methods.

While direct methods yield exact results, iterative methods are generally more efficient, both in time and space. Disadvantages of iterative methods are that for some of these methods no guarantee of convergence can be given in general and the determination of suitable error bounds for termination is not always easy. Since iterative methods are considerably more efficient in solving Markov chains, they are widely used for larger models. For smaller models with less than a few thousand states, direct methods are reliable and accurate. Although closed-form results are highly desirable, they can be obtained for only a class of models. A comprehensive source on algorithms for steady-state solution is discussed in [Bolch et al. 1998].

There are many direct methods for solving the model at steady-state. Among the methods most applied are the well known Gaussian elimination and Grassmann algorithm or GHT algorithm[Grassmann et al. 1985]. In the case of iterative methods, among the most used are: SOR, Jacobi, Gauss-Seidel and Power methods. We do not discuss these issues because they go beyond the scope of this work. In fact, our focus is on smaller models with a few hundred states for which direct methods are reliable and accurate. We restrict ourselves to explore special structures of Markov chains investigating the potential existing closed-form equations.

1.4.4 Model validation

Model validation is the task of checking model correctness. Some properties of the model can be checked by answering the following questions:

- Is the model logically correct, complete and detailed?
- Are the assumptions justified?
- Are the features represented in the model significant for the application context?

Model validation can be split into two subproblems [Laprie 1983]:

- *Syntactic validation*: consists in checking the model properties in order to verify if the model is syntactically correct;

- *Semantic validation*: consists in checking that the model represents the behavior of the system under consideration;

The syntactic validation can be carried out before and after model processing. For instance, the evaluated measures must satisfy general properties, independently of the model such as: reliability is a non-increasing function of time that asymptotically tends to 0. This verification allows to validate the syntax after the model processing.

The semantic validation aims to check the representation of a system behavior with respect to the events acting on it. Usually, for implementing a semantic validation, it is required to perform a sensitivity study [1]. This study depends on the particular system under evaluation. However, it is possible to distinguish two classes:

- Sensitivity on the model parameters: values and consistency of the rates;
- Sensitivity on the model structure: events that should be taken into account;

These sensitivity studies may lead to model simplification, e.g., if large variations on the model parameters do not imply in a significant impact on the evaluated measures. If a real environment exists as in the left side of Figure 1.1, the errors can be found either in the measurement process or in the modeling environment. A model can also be considered valid if the measures calculated by the model match the measurements of the real system within a certain acceptable margin of error [Menascé & Almeida 2000] .

The validation results can be used for modification of the model or for improving our confidence on its validity. Certainly, many interactions are required for this important task. Note that a valid model is one that includes the most significant effects in relation to the system requirements [Bolch et al. 1998].

1.5 Dealing with large models

One of the main problems when building models for complex systems is the well-known *state explosion* problem. In fact, the number of states in a given model representing a complex system can quickly become very large. Many techniques have been suggested to deal with large models. Techniques addressing this problem fall into the following categories: *largeness avoidance* and *largeness tolerance* [Trivedi et al. 1994].

Largeness avoidance technique consists in separating the original large problem into smaller problems and to combine submodel results into an overall solution. The results of the sub-models are integrated into a single model that is small enough to be processed. Among these techniques, there are e.g. state lumping [Nicola 1990], state truncation [Muppala et al. 1992], model composition

[1] Also called sensitivity analysis.

[Bobbio & Trivedi 1986], behavioural decomposition [Balbo et al. 1988], hybrid hierarchical [Balakrishnan & Trivedi 1995], etc .

Largeness tolerance techniques deal with large models mastering the generation of the global system model through the use of concise specification methods and software tools that automatically generate the underlying Markov chain. The specification consists in defining a set of rules allowing an easy construction of the Markov chain. These rules may be either i) specifically defined for model construction (see e.g., [Goyal et al. 1986], [Berson et al. 1991]) or ii) formal rules based on Generalized Stochastic Petri Nets (GSPNs) and their extensions [Ciardo et al. 1989]. Several specification and construction methods based on model composition techniques have been developed. In particular for GSPNs based models, we can report the following modeling approaches [Betous-Almeida & Kanoun 2004, Kanoun & Borrel 2000, Bondavalli et al. 1999].

It is worth to note that both techniques are complementary and can be combined. Largeness avoidance can be used to complement efficiently largeness tolerance techniques using states truncation (i.e., states with very small probabilities) as in [Muppala et al. 1992]. Multilevel modeling is another interesting approach to combine both techniques. In this case, the target system is described at different abstraction levels with a respective model associated to each level. The flow of information needed among the submodels is organized in a *hierarchical model*. In [Kaâniche et al. 2003b] largeness avoidance is applied to the levels where independence or weak dependency assumptions hold, while largeness tolerance is used for constructing sub-models that exhibit strong dependencies. Other examples of hierarchical modeling approaches are presented in [Bondavalli et al. 2001b].

Another difficulty when processing large Markov models is the *stiffness* problem. It is due to the different orders of magnitude (sometimes 10^6 times) between the rates of performance-related events and the rates of the rare, failure-related events. Stiffness leads to difficulty in the solution of the model and numerical instability. Research in the transient solution of stiff Markov models follows two main lines: stiffness-tolerance and stiffness-avoidance. The first one is aimed at employing solution methods that remain stable for stiff models (see the survey [Bobbio & Trivedi 1986]). In the second approach, stiffness is eliminated from a model solving a set of non-stiff submodels using aggregation and disaggregation techniques [Bolch et al. 1998].

Largeness and stiffness problems can be avoided by using hierarchical model composition [Popstojanova & Trivedi 2000]. Occurrence rates of failure/repair events are several orders of magnitude smaller than the request arrival/service rates. Consequently, we can assume that the system reaches a (quasi-) steady state with respect to performance related events between successive occurrences of failure/repair events. That is, the performance measures would reach a stationary condition between changes in the system structure. This leads to a natural hierarchy of models. The structure state model is the higher level dependability model representing the failure/repair processes. For each state in the dependability model, there is a reward model, which is a performance model having a stationary structural state.

Several authors have used this concept for combined performance and dependability analysis [Haverkort et al. 2001, de Souza e Silva & Gail 1996, Trivedi et al. 1992, Smith et al. 1988].

This thesis explores the concept of composite performance and dependability approach [Meyer 1980, Meyer 1982] for modeling the dependability of web based services including hardware and software failures as well as performance degradation related failures. The performance model takes into account the request arrival and service processes relying on queueing theory. It allows to evaluate performance related measures conditioned on the states determined from the dependability model. The dependability model is used to evaluate the steady state probability associated to the states that result from the occurrence of failures and recoveries. The assumption of quasi steady state is acceptable in our context, since the failure/recovery rates are much lower than the request arrival/service rates.

1.6 Related work on web evaluation

The focus of our research is on the availability evaluation of web-based services. In this section, we review the studies that are basis for our investigation. We begin introducing some studies related to web measurements based evaluation. Web measurements provide important insights for the development of analytic models. Further, modeling approaches are briefly reported presenting prior studies with recent theoretical developments related to the web modeling based evaluation.

1.6.1 Measurements based evaluation

From a performance viewpoint, a significant body of work has focused on various aspects related to web performance measurements. An interesting aspect is the web traffic characterization which has received much attention. It has been evaluated in order to capture its most relevant statistical properties. Some of the properties considered in the literature deal with file size distributions [Arlitt & Williamson 1997], self-similarity [Crovella & Bestavros 1997] and reference locality [Almeida et al. 1996].

[Arlitt & Williamson 1997] identified some properties called invariants, for instance file sizes follow the Pareto distribution. [Almeida et al. 1996] showed that the popularity of documents served by web sites dedicated to information dissemination follows Zipf's law. [Crovella & Bestavros 1997] pointed to the self-similar nature of the web traffic. Intuitively, a self-similar process looks bursty across several time scales. In contrast, there are other works suggesting that the aggregate web traffic tends to smooth out as Poisson traffic [Iyengar et al. 1999, Morris & Lin 2000].

Using measurement from traces of real web traffic, [Morris & Lin 2000] presented evidences that traffic streams do not exhibit correlation, that is, the aggregation of sources leads the web traffic to smooth out as Poisson traffic [2]. Two explanations are

[2]These results are useful mainly within the context of a busy hour.

given. First, there appears to be little correlation among users in their consumption of bandwidth. Second, individual web users consume differing amounts of bandwidth mostly by pausing longer between transfers.

Modulated Markov Poisson Process (MMPP) has been shown to be well suited to capture some characteristics of the input traffic. Based on traffic traces, [Muscariello et al. 2004] showed how MMPP approximates the long-range dependency (LRD) characteristics. Also, MMPP based models have been often used for modeling the arrival rate. In [Chen et al. 2001], a multi-stage MMPP is used to describe the input traffic, capturing the arrival rate of requests varying according to the period of the day (daily cycles). In [Heyman & Lucantoni 2003], an MMPP is developed to study the multiple IP traffic streams.

From a dependability viewpoint, many commercial providers today advertise four-9's or five-9's for server availability (i.e., an availability of 99.99 or 99.999, respectively). In fact, high available servers are not sufficient for providing a highly available service since many types of failures can prevent users to access services. Practical experiences [Merzbacher & Patterson 2002] have shown that advertised numbers reflect performance under optimal operating conditions, rather than real-world environment. For example, [Paxson 1997] has shown that "significant routing pathologies" prevent certain pairs of hosts from communicating about 1.5% to 3.3% of the time. [Kalyanakrishnan et al. 1999] have suggested that average availability of a typical web service is two-9's. Therefore, in contrast with the 5 minutes per year of unavailability for a five-9's system, a typical two-9's web service will be unavailable for nearly 15 minutes per day from end users perspective.

A *failure* affecting the web service might be due to problems with the remote host (e.g., the site is overloaded), problems with the underlying network (e.g., a proper route to the site does not exist) or problems with the user host (e.g., a failure in the user's subnet preventing the access to the Internet). Recent studies has been devoted to measure the web service availability and reliability in order to characterize the failure behavior of web based systems. For instance, [Oppenheimer et al. 2003] have studied the causes of failures and the potential effectiveness of various techniques for preventing service failure using data from three large-scale web services. They suggest that the operator error is the largest cause of failures in two of the three services. Configuration errors are the largest category of operator errors and failures in front-end software are significant. They point out the fact that improving the maintenance tools used by service operators would decrease the time to diagnose and repair problems.

[Chandra et al. 2001] proposed a failure classification based on location in which we can distinguish three types of failures: i) "near-user" ii) "in-middle" and iii) "near-host". Near-user failures represent failures that disconnect a client machine or a client subnet from the rest of the Internet. Analogously, "near-host" failures make the web host unreachable from the outside world. "In-middle" failures usually refer to the Internet backbone connection malfunctions that separate the user and the specific service host, but the user may still visit a significant fraction of the remaining nodes on the Internet. Most notably, these failures represent an interruption of connectivity

between a single pair of nodes that does not affect any other pairs' to communicate. In reality failures in the middle of the Internet infrastructure will typically affect more than one pair of nodes [Labovitz et al. 1999].

[Labovitz et al. 1999] have explored route availability by studying routing table update logs. They found that only 25% to 35% of routes had an availability higher than 99.99% and that 10% of routes were available less than 95% of the time. They show that 60% of failures are repaired in thirty minutes or less and that the remaining failures exhibit a heavy-tailed distribution. There results are qualitatively consistent with the end-to-end analysis presented in [Dahlin et al. 2003] which provides additional evidence that connectivity failures may significantly reduce WAN service availability.

[Dahlin et al. 2003] have analyzed connectivity traces in order to develop a model suitable for evaluating techniques dealing with unavailability. They focus on WAN connectivity model that includes average unavailability, the distribution of unavailability durations and the location of network failures. They provided a synthetic unavailability model in which average unavailability is 1.25% with a request-average unavailability varying from 0.4% to 7.4% and unavailability duration distributions appear heavy-tailed, indicating that long failures account for a significant fraction of failure durations.

Finally, there are web sites today such as Netcraft (*www.netcraft.com*) that provide statistics based on remote measurements showing the quality of service supported by a wide variety of web sites in terms of availability and performance related issues. In practice, Netcraft is unable to detect the 'back end' computers that are hosting a web service. In fact, Netcraft measure consists in determining how long the responding computer hosting a web site has been running, i.e., uptime based on server availability ("time since last reboot"). However, the availability of the responding computer does not necessarily reflect the service availability delivered by the web site, given the fact that the service is usually hosted on several computers that are locally or geographically distributed.

Netcraft measurements are collected at intervals of fifteen minutes from four separate points around the Internet. The averages of the samples are calculated using an arithmetic mean over a time window period (default 90 days). The statistics are provided ordered by e.g. outage periods presenting a ranking of the fifty sites per month. We made a preliminary study using such statistics in which we found that 54% of the sites had an outage period in the order of 25 seconds per month or lower on average. Approximatively 18% of the sites had on average an outage of 4 minutes per month and 21% around 43 minutes per month. Finally, 6% of the sites had an outage in the order of 7 hours per month and 1% had an outage around 70 hours per month. The average unavailability was about 1 hour and 17 minutes per month. The data was obtained from the Netcraft's site in the period of June to December 2003, using the ranking reports available in the site.

1.6.2 Modeling based evaluation

While measurement based studies are essential for characterizing the current and past behavior of an existing system, it is also important to anticipate future behavior with different architecture configurations or under different workloads. In the context of web-based services, modeling has been mainly used for performance evaluation to support design decisions (capacity planning, scalability analysis, comparison of load balancing strategies, etc.). The proposed performance models are usually based on queuing theory and queuing networks, although recent theoretical developments have provided elegant theories such as network calculus [LeBoudec 1998], effective bandwidth [Elwalid & Mitra 1993] which have been used for analyzing the quality of service supported by the Internet [Firoiu et al. 2002]. The quantitative measures evaluated from the models include response time, throughput, queue length, blocking probability, etc.

Queuing network (QN) models are basically a network of interconnected queues that represent a computer system. A queue in a QN stands for a resource (e.g. CPU, disk) and the queue itself stands for requests waiting to use the resource. A QN model may be classified as *open QN* or *closed QN* depending on whether the number of requests in the QN is unbounded or fixed, respectively. Open QNs allow requests to arrive, go through the various resources, and leave the system. Closed QNs are characterized by having a fixed number of requests in the QN.

Various queuing models have been investigated for the performance evaluation of web servers, proxy servers, etc. The main differences between the proposed models concern the assumptions characterizing the arrival and service time distributions (e.g., M/M/1, M/M/c/k, M/G/1 etc.) as well as the level of detail of the models. In particular, two types of web server models can be distinguished: i) *system level* models where the web server is viewed as a black box or as a resource with a queue [Slothouber 1996, Andersson et al. 2003, Cao et al. 2003], and ii) *component level* models which take into account the different resources of the system (CPUs, memory, disks, threads, etc.) and the way they are used by distinct requests [Dilley et al. 2001, Menascé et al. 2001].

Recent efforts focused on the performance modeling of multi-tier web based applications and e-commerce sites [Urgaonkar et al. 2005, Reeser & Hariharan 2000, Menascé et al. 2001]. Such studies are aimed at taking into account various interactions among the different tiers involved in the processing of user requests (e.g. Web, Java, and database servers), considering various execution scenarios describing the users behaviors. The proposed models are generally based on a network of queues, where the queues represent different tiers of the applications.

As regards dependability, to our knowledge, only a few studies have been devoted to the dependability modeling and evaluation of web-based services and applications.

[Tang et al. 2004] presented an availability evaluation approach for a fault tolerant application server combining measurement and modeling analysis. The study deals with fault tolerant Sun Java System Application server, used for deploying web services

in Java 2 Enterprise Edition 7 three-tier architectures. Under conservative assumptions used for building the model and estimating model parameters from experimental tests[3], they showed that the average availability of the target system was evaluated to be at the 99.999% level in the sun server environment. This availability level does not necessarily reflect the quality of service perceived by the users, given that the behavior of users was not taken into account in the model.

[Xie et al. 2003] proposed a modeling approach for analyzing user-perceived availability based on Markov regenerative process models. Two different scenarios are considered: single-user/single-host and single-user/multiple-hosts. This study analyzes the dependency of the user perceived unavailability on parameters including the service platform failure rate, repair rate and user retry rate. The results suggest that the user perceived unavailability depends not only on the characteristics of the underlying infrastructure but also on the user's behavior.

In [Nagaraja et al. 2003], a two-phased methodology combining fault injection and analytical modeling is proposed to evaluate the performability of Internet services using cluster based architectures. A performability metric combining the average throughput and availability is defined for comparing alternative design configurations. The methodology is illustrated through the analysis of 4 versions of the PRESS cluster based web server [Carrera & Bianchini 2005, Carrera & Bianchini 2001] under 4 categories of faults (Network, disk, Node and Application), quantifying the effects of different design decisions on performance and availability. The models proposed in this study are based on the assumption that faults in different components are not correlated. The effects of multiple faults are combined into an average performance and availability metric based on the estimation of the average fraction of time spent in the degraded mode caused by each fault. An application of the methodology to analyze various state maintenance strategies in an online bookstore and an auction multi-tier Internet service is presented in [Gama et al. 2004, Nagaraja et al. 2005].

1.7 Conclusion

Dependability is a key issue in web-based services and applications. The web is a large evolving infrastructure incorporating new components and services at a very fast rate. Recently many high-tech companies providing service on the web have experienced operational failures. Such operational problems have resulted in degraded performance, unsatisfied users and heavy financial losses. Quantitative methods are needed to understand, analyze, design and operate such a large infrastructure. Quantitative measures need to be evaluated in order to estimate the quality of service and the reliance that can be placed on the provided service.

This chapter introduced the concepts of dependability as well as the required background including some approaches and techniques for web dependability evaluation. Our work relies on modeling based methodologies in order to provide a quantitative

[3]Using fault injections.

approach for evaluating the availability of web-based services. The models presented along the chapters are based on probability fundamentals, queueing theory and performability modeling approach.

An overview of the modeling process indicating its phases was briefly discussed. The phases are described presenting some of the main problems and methods related to each phase. We reviewed the related studies that have driven most of the work presented along the chapters. First, we focused on web evaluation based on measurements results. After this discussion, some investigations based on web modeling evaluation were presented.

All the studies outlined above are recent, illustrating the fact that this topic is not fully explored in the literature. While the proposed modeling approaches are useful to support dependability tradeoffs at the design stage, they are not sufficient to model the user perceived dependability of Internet based applications and services, taking into account the multiple interactions between the users and the various resources involved in the processing of user requests. Indeed, the user perceived dependability of web-based applications is affected by a variety of factors (e.g., user behaviors and workload characteristics, hardware and software distributed execution platforms, quality of service provided by the interconnection networks, fault tolerance and maintenance strategies, etc.). Also, both hardware and software failures as well as performance related failures have to be taken into account in the analysis. Due to the complexity of the target system and the difficulty to combine various types of information, a systematic and pragmatic modeling approach is needed to support the construction and processing of dependability models that can be used by the designers to further understand the dependability capabilities and limitations of their systems, under different workload scenarios and architecture configurations. The contributions presented in this dissertation are aimed at fulfilling these objectives. Our research builds on the multi-level modeling proposed in [Kaâniche et al. 2001, Kaâniche et al. 2003b] and presented in detail in Chapter 2.

In the following chapters, we are particularly interested in the service availability and user perceived availability evaluation and modeling. The directive lines of this work are structured as follows. In chapter 2, an availability modeling framework is presented using a web-based travel agency as example illustrating the main concepts of the framework. Chapter 3 provides a modeling based approach of web service availability supported by web clusters architectures. We address especially recovery strategies and traffic burstiness effects on web service availability. Finally, chapter 4 introduces a flexible analytic modeling approach for computing service unavailability due to long response times.

Chapter 2

Availability modeling framework

> Prediction is fine as long as it is not about the future.
>
> Mark Twain

THE objective of this chapter is twofold i) define a multi-level modeling framework for the dependability evaluation of web based applications, ii) illustrate the main concepts and the feasibility of the proposed framework using a web-based travel agency as an example. This framework allows us to evaluate the user perceived availability. Modeling is carried out considering four levels, namely: user, functions, service and resource. The first level describes how the users invoke the application and the three remaining levels detail how user requests are handled by the application at distinct abstraction levels.

The user perceived availability measure takes into account the combined impact of performance related failures and traditional software and hardware failures. For illustration purposes, sensitivity analysis results are presented to show how user perceived availability performs considering various situations, e.g. the users operational profile, the travel agency architecture and the fault coverage.

2.1 Problem statement

Web based services are implemented on largely distributed infrastructures, with multiple interconnected layers of software and hardware components, involving various types of servers such as web, application, and database. A typical web based service usually involves three key players shown in Figure 2.1: 1) the users, 2) the web application provider, and 3) external suppliers.

- The *users* interact through a web browser with the web application providing the service;
- The *web application provider* implements a set of e-business functions that are invoked by the users. These functions are based on a set of services and resources that are under the direct control of the application provider or are provided by external suppliers;
- The *external suppliers* contribute to the implementation of some of the functions and services provided by the web application;

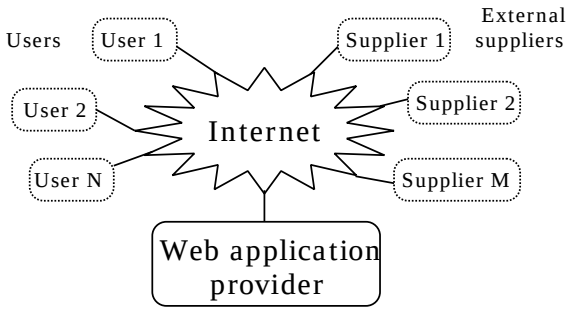


Figure 2.1: Three key players of a web based system

Every request initiated by a user is processed in several steps. It starts in the users browsers, flows through the Internet being executed in the web application provider (or shortly provider), and it can also be processed by the application using the external suppliers. For example, a provider can offer a book selling electronic service by outsourcing shipping, payment and billing to other external service providers (suppliers). At the provider level, the user requests and the interactions with the external suppliers are supported by a set of complex distributed applications and middleware such as web servers, application servers and database servers. Similar infrastructures are used at the external supplier sites.

Users usually have different behaviors and they may invoke the various functions provided in the application site in different ways with variable frequencies.

Considering a travel agency, the classes of users (i.e. managers, operators, sellers, etc.) have distinct objectives requiring a variety of different features of the system. Indeed, even in the same class of users, the behavior may be completely dissimilar. For example, there are clients who buy very often while others may do extensive searching and browsing without buying anything. Definitely the types of functions invoked and the resources involved to support such functions are not necessarily the same. As a consequence, the user perceived availability is affected not only by the user operational profile (i.e. workload) but also by the state of the components supporting the functions.

Generally, the service provider architecture is under direct control of web designers. Therefore, a detailed analysis of this architecture can be carried out to support design architectural decisions. However, only limited information is usually available about the infrastructure supporting the external suppliers. For the external suppliers, remote measurements should be performed in order to characterize the dependability of such services. These measures can then be incorporated into the models describing the impact of component failures and repairs on web service availability and also on user perceived availability.

The discussion above shows that several issues should be taken into account for modeling the availability of web based applications as perceived by users. From the designers point of view, it is critical to understand how the different components of the distributed infrastructure supporting the provided service might affect the user perceived availability. Hierarchical modeling has proven to be well suited to support such analysis by describing progressively the target system at different abstraction levels, with a sub-model associated to each level. The availability measure can be computed based on the hierarchical composition of the sub-models. The multi-level modeling framework proposed in [Kaâniche et al. 2001, Kaâniche et al. 2003b] for analyzing the user perceived availability of web based applications has been developed following such hierarchical composition approach. The modeling is carried out in two steps:

- hierarchical description of the system and its interactions with the users, from the functional and structural viewpoints;
- hierarchical construction and solution of the availability models based on the information obtained from the first step;

The rest of this chapter is structured as follows. Section 2.2 defines the main concepts of the modeling framework. Sections 2.3 and 2.4 present the travel agency example and its hierarchical description and modeling. Section 2.5 presents some sensitivity evaluation results either at web service availability level or at user perceived availability level. Finally, section 2.6 summarizes the chapter.

2.2 Dependability framework

The information needed for dependability modeling and analysis can be structured into four levels as shown in Figure 2.2, where the dependability measure considered is availability.

- The **user level** describes the user operational profile in terms of the types of functions invoked and their probability of activation.
- The **function level** describes the set of functions available to the user level.
- The **service level** describes the main services needed to implement each function and the interactions among them. Two categories of services are distinguished: those delivered by the web application provider (internal services) and those provided by external suppliers (external services).
- The **resource level** describes the architecture on which the services identified at the service level are implemented. At this level, the architecture, fault tolerance and maintenance strategies implemented at the provider site are detailed. However, each service provided by an external supplier is represented by a single resource that is considered as a black box.

The function and service levels describe according to a top-down approach, how the application software implementing the e-business logic is structured and decomposed, whereas the resource level describes the corresponding execution environment (software, hardware components, etc).

The hierarchical decomposition proposed above builds on some concepts defined in [Menascé & Almeida 2000] to describe e-business applications and analyze their performance. However, since our framework is oriented to dependability modeling and evaluation from the user perspective, we have refined these concepts and extended them to fulfill the objectives of our study.

In the following subsections, we present each of these levels describing how dependability modeling is carried out based on the hierarchical description, where the dependability measure considered is the availability.

2.2.1 User level

This level describes the user operational profile. It models the execution scenarios performed by the user when visiting the web site(s). Each scenario is defined by the set of functions invoked and the probability of activation of each function in the corresponding scenario. As illustrated in Figure 2.3, the user operational profile can be described as a graph with a set of nodes and transitions. A compact description of this graph is given by the matrix representation. The “Start” and “Exit” nodes correspond to the beginning and end of an user scenario when visiting the provider site(s). Each

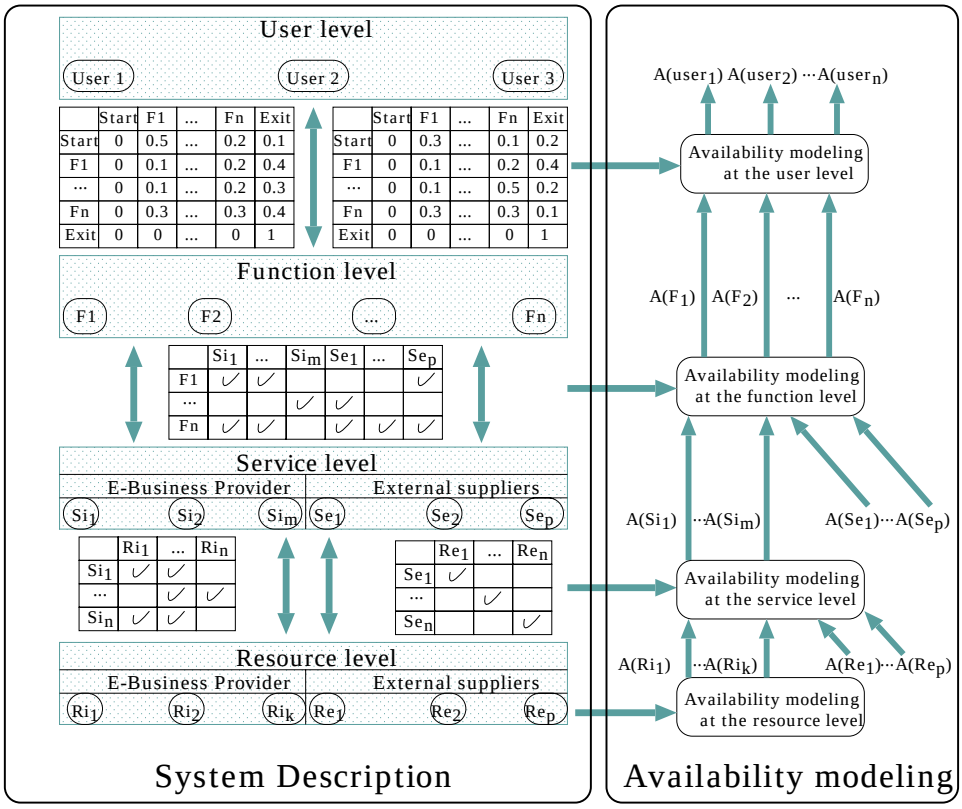
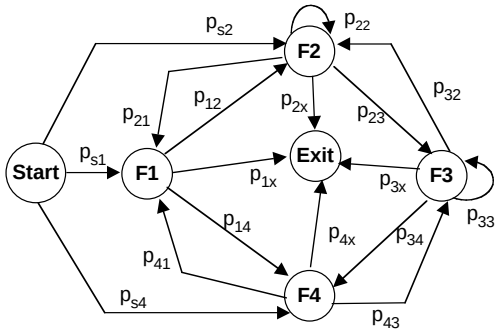


Figure 2.2: Hierarchical availability modeling framework



	Start	F1	F2	F3	F4	Exit
Start	0	p_{s1}	p_{s2}	0	p_{s4}	0
F1	0	0	p_{12}	0	p_{14}	p_{1x}
F2	0	p_{21}	p_{22}	p_{23}	0	p_{2x}
F3	0	0	p_{32}	p_{33}	p_{34}	p_{3x}
F4	0	p_{41}	0	p_{43}	0	p_{4x}
Exit	0	0	0	0	0	1

Figure 2.3: User's operational profile

node F_i means that function F_i is invoked by the user. A transition from node F_i to node F_j means that function F_j is executed after execution of F_i.

The transitions among the nodes and the associated probabilities p_{ij} describe how users interact with the web site. This means that a transition from state i to state j is said to occur when the request going to state j arrives at the server. The output transitions from the start node, and the corresponding probabilities p_{si}, specify the first function executed by the user when entering the web site(s). Finally, parameters p_{ix} specify the probability of leaving the web site after executing function F_i. The probabilities of activation with respect to the user scenarios can be derived using traditional methods such as flow graph reduction [Howard 1971b].

It is possible to define a class of users consisting of a specific set of p_{ij}. These probabilities can be obtained by collecting data on the web site (see e.g., [Menascé & Almeida 2000]). In addition, it should be noted that other techniques such as data mining have been used to analyze web logs in which a similar matrix exists [Zaiane et al. 1998]. The authors discuss the problem of relying on server side information in the context of data mining on http logs. They conclude that although server side information is not 100% complete, much useful information can be discovered from it, e.g. identify population of potential customers for electronic commerce.

The availability as perceived by the users can be evaluated by considering a particular scenario or by taking into account all the scenarios from the start node to the exit node. The availability measure will be affected by the probability of the corresponding scenario(s) and the availabilities of the functions involved in these scenarios. It is worth noting that different operational profiles with different probability matrices can be defined to analyze different classes of users: heavy buyers, occasional buyers, etc.

2.2.2 Function level

This level identifies the set of functions offered to the users at the web site(s). Table 2.1 presents some examples of such functions. Some of these functions (e.g., Search, Login) may be found in most web sites, whereas others are characteristic of certain web sites or of specific types of web sites. The identification of all functions offered by the web site and the classification of these functions according to their criticality require a thorough analysis of the web specifications and the users expectations in terms of quality of service. Different levels of degradation on the quality of service delivered to the users can be defined based on the evaluation of the impact of temporary loss or degradation of a function and its cost (e.g., loss of revenue). Such a classification should also take into account the impact of the loss or degradation of several functions.

Category	**Function**	**Description**
Common	Login	Login to the site
	Register	Register as a new user
	Search	Search site database
	Select	Show one of the results of a search
	Browse	Follow links within the site
Retail	Add Item	Add item to shopping cart
	Remove Item	Remove item from shopping cart
	See Shopping Cart	Check contents and value of shopping cart
	Pay	Pay for items in shopping cart
Trading	Open Account	Open account for trading
	Trade	Buy/sell/exchange stocks or mutual funds
	Create Portfolio	Create stock/funds portfolio
	Add to Portfolio	Add stock/funds to portfolio

Table 2.1: Examples of functions provided by e-Business web sites

2.2.3 Service Level

This level describes the mapping between the functions and the services required to implement them. Each function identified at the function level is decomposed and refined into a set of services implemented by various software entities (i.e., servers). Examples of servers include Web, Application, Authentication, Name, File, Database and Communication servers. Generally, the execution of one function may involve more than one server. Based on the analysis of client-server interactions, we can define

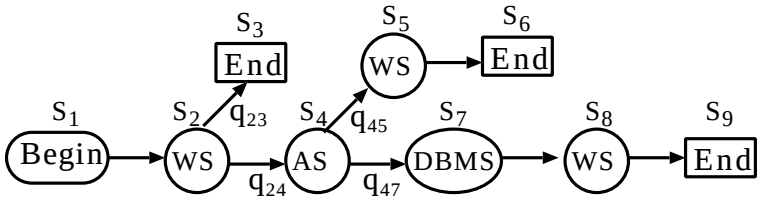


Figure 2.4: An example of Interaction diagram associated to a function

a matrix specifying the mapping between the functions identified at the function level and the servers identified at the service level.

Several graphical notations and formalisms (e.g. Unified Modeling Language-UML, Message Sequence Charts-MSC, etc.) are used not only to describe, for each function, the dynamic interactions among the servers involved in its accomplishment, but also to identify all possible execution scenarios of the function. The graphical representation given in Figure 2.4 is based on the concept of the *Interaction Diagram* defined in [Menascé & Almeida 2000]. The interaction starts and ends with the client node ("Begin" and "End" nodes). Each path between a pair of client nodes identifies the set of servers involved in the interaction.

Figure 2.4 presents three possible scenarios for the execution of a given function ($S_1 \rightarrow S_2 \rightarrow S_3$, $S_1 \rightarrow S_2 \rightarrow S_4 \rightarrow S_5 \rightarrow S_6$, and $S_1 \rightarrow S_2 \rightarrow S_4 \rightarrow S_7 \rightarrow S_8 \rightarrow S_9$). The nodes are numbered for the sake of clarity and each arc between two nodes i and j is labeled with the probability of occurrence of the corresponding transition (denoted q_{ij}).

All paths in the interaction diagram, from a "Begin" to an "End" node, should be accounted for in the evaluation of the availability of the corresponding function.

2.2.4 Resource level

This level describes the mapping between the services defined at the server level and the resources involved in the achievement of these services. Also, it provides information on the replication of each service as well as the fault tolerance and maintenance strategies implemented at the web site(s). A resource is a component of the system or an element of a component (computer host, software and hardware components, communication link) that contributes to the implementation of services. Indeed, one service may be partitioned and replicated among several resources or clusters of resources and one resource may host many services. At this level, we distinguish between internal and external services.

As the architecture whose the *external services* are implemented is not known, we associate to each of them a single resource. For example, an Internet service provider can be represented by a single resource providing service connectivity.

As regards *internal services*, a detailed analysis of the web site(s) architecture can be performed. We must define the mapping between the resources and the services, as well as the interactions among these resources, since the availability of each service will depend on the availability of the corresponding resources. Several alternative architectural solutions may be considered for implementing the internal services. These solutions may be defined based on:

- various organizations of the services on the hardware support (e.g., dedicated hosts for each server, vs. multiple servers on the same host);
- various fault tolerance strategies (non-redundant servers vs. replicated servers);
- various maintenance strategies adopted by the web (e.g., immediate maintenance vs. delayed maintenance, dedicated vs. shared repair resources).

Alternative architectures may be compared to help the designers in the selection of the most appropriate solution from the availability point of view. The analysis of the architectures should lead to the definition of the mapping between the resources and the services implemented on these resources. For each service, different accomplishment levels can be defined depending on the types of failures affecting the corresponding resources. In particular, when the service is distributed on several resources, the service accomplishment levels can be defined according to the number of resources that are still available to run the service (graceful degradation concept).

Knowledge of the system architecture is required for modeling purposes. Figure 2.5 presents examples of configurations of a Web server: a) a non-redundant configuration with a single server, b) a redundant configuration with geographically distributed replicas, and c) a cluster-based configuration with several Web servers interconnected through a LAN and centralized at a single site with a load balancer directing incoming requests to one of the servers. Configuration (b) requires the replica states to be kept mutually consistent to ensure that clients do not get out-of-date information. This is not easy to achieve on a large-scale system [Ingham et al. 2000]. Alternative solutions are proposed for instance in [Bowen et al. 2000] to ensure a weak coupling between functions implemented on geographically distributed servers. The cluster-based configuration is widely used for the implementation of internet-based applications. This configuration is able to handle heavy traffic loads, however it has a single point of failure, the load balancer. Therefore, fault tolerant solutions with error detection and recovery capabilities should be considered for the load balancer. Popular web sites generally combine the clustering approach with geographically distributed servers in order to provide high performance and available web service (see for instance, the architecture deployed by IBM for the Nagano Olympic games [Iyengar et al. 2000]).

Fault tolerance solutions should also be considered to ensure reliable communication among the Web servers as well as the availability of the data accessed by Web server processes. Different configurations can be considered for accessing data from the Web servers [Ingham et al. 2000]. For example, data can be partitioned among the servers and accessed through a shared bus, or a shared master copy of the data is

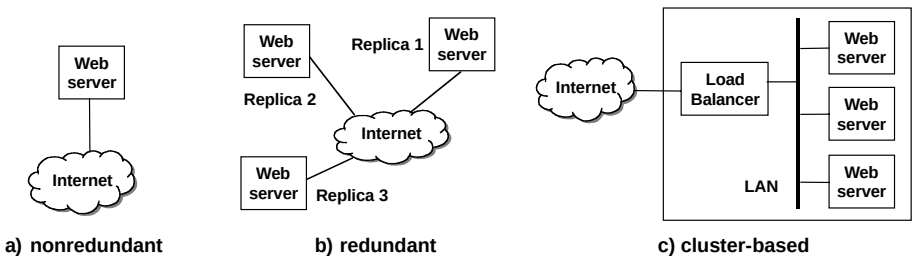


Figure 2.5: Example of configurations for a Web server

accessed by all the servers through a distributed file system. Each web server mounts and processes the same data set from the distributed file system.

2.2.5 Availability modeling

As shown in Figure 2.2, the availability modeling and evaluation step is directly related to the system hierarchical description. The outputs of a given level are used in the next immediately upper level to compute the availability measures associated to this level (denoted by $A(x)$ where x is a user, a function, a service or a resource). Accordingly, at the service level, the availability of each service is derived from the availability of the resources involved in its accomplishment. Similarly, at the function level, the availability of each function is obtained from the availability of the services implementing it. Finally, at the user level, the availability measures are obtained from the availability of the functions invoked by the users.

At the *service/resource level*, one or several availability models are built based on the knowledge of the infrastructure and the *resources* implementing the required services. This level includes also the fault tolerance and recovery mechanisms as well as the maintenance policies at the service provider site(s). Various techniques can be used to build and solve these models including non state-based techniques also called combinatorial techniques (e.g. fault trees, reliability block diagrams), and state-based techniques (e.g. Markov chains, Generalized Stochastic Petri Nets - GSPNs).

The selection of the right technique depends mainly on the kinds of dependencies among the elements of the underlying level and on the quantitative measures to be evaluated. Markov chains and GSPNs are well suited for strong dependencies. In particular, there are modeling approaches [Rabah & Kanoun 2003, Kanoun & Borrel 2000, Fota et al. 1999] that take into account the stochastic dependencies that might exist among the various components of the service/resource availability model. Examples of models at the service/resource level are given in the following sections. The outputs of this modeling step are the availability of internal services denoted $A(S_{ij})$.

The availability model at the *function level* relies on the knowledge of the availability of all services involved in function accomplishment and all possible execution

scenarios associated to each function. The outputs of this level are the availability of the functions denoted $A(F_i)$ that can be defined as follows:

$$A(F_i) = \sum_{j=1}^{N} \omega_j A(\sigma_j(F_i)) \tag{2.1}$$

- N is the number of execution scenarios for function F_i;
- ω_j is the probability of activation of execution scenario j;
- $\sigma_j(F_i)$ denotes the set of services involved in execution scenario j;
- $A(\sigma_j(F_i))$ is the availability of the services involved in execution scenario j.

This formula is general and can be applied whether the services are independent or not. When the services are independent, $A(F_i)$ can be expressed as:

$$A(F_i) = \sum_{j=1}^{N} \omega_j \prod_{x \in \sigma_j(F_i)} A(S_x) \tag{2.2}$$

$A(S_x)$ is the availability of a service S_x involved in execution scenario j. S_x could be an internal or external service.

At the *user level*, the availability model for a given user class is based on the execution scenarios activated by the user when visiting the web site. The outputs of this level are the availability as seen by the various classes of users denoted $A(user_k)$.

Similarly the function level, $A(user_k)$ can be obtained by the following formula:

$$A(user_k) = \sum_{j=1}^{M} \beta_j A(L_j) \tag{2.3}$$

- M is the number of user execution scenarios derived from the user operational profile;
- β_j is the probability of activation of scenario j;
- L_j is the set of functions involved in scenario j;
- $A(L_j)$ is the availability of the functions involved in scenario j.

Once again, if the functions are independent:

$$A(user_k) = \sum_{j=1}^{M} \beta_j \prod_{q \in L_j} A(F_q) \tag{2.4}$$

$A(F_q)$ is the availability of function F_q executed in scenario j.

The following section illustrates the main concepts of hierarchical framework using a travel agency application as an example. The objectives are : 1) to show how to apply the proposed framework based on the decomposition of the target application according to the four levels and 2) to present typical availability modeling and analysis results that can be used for supporting design decisions.

2.3 Travel Agency (TA) description

The TA is designed to allow the users to plan and book trips over the web. TA interacts through dedicated interfaces with several flight reservation systems (AF, KLM, Varig, ...), hotel reservation systems (Sofitel, Holiday Inn, ...), and car rental systems (Hertz, ...).

The TA application can be seen as composed of two basic components [Periorellis & Dobson 2001]: the travel agent front end-client side, denoted as TAFE-CS, and the travel agent front end-server side, denoted as TAFE-SS (Figure 2.6).

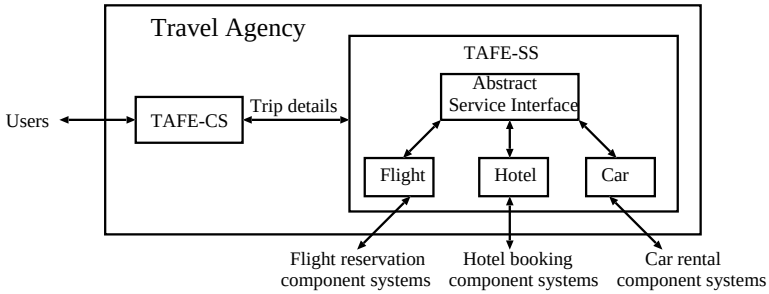


Figure 2.6: TA high-level structure

The client side handles user's inputs, performs necessary checks and forwards the data to the server side component. TAFE-SS is the main component of the TA. It is designed to respond to a number of calls from the client side concerning e.g., availability checking, booking, payment and cancellation of each item of a trip. It handles all transactions to, and from, the booking systems, composes items into full

trips, converts incoming data into a common data structure and finally handles all exceptions.

Starting from this very high-level description, we will further detail it according to the various aspects required for the hierarchical description. We will first focus on the function and user levels together, then the service and function levels before addressing the resource level.

2.3.1 Function and user levels

The behavior of the users accessing the TA web site is characterized by the operational profile example presented in Figure 2.7. This graph is the basis for capturing the navigational pattern of a group of clients, as viewed from the server side. The nodes "Start" and "Exit" represent the start and the end of a user visit to the TA web site, and the other nodes identify the requested functions during their visit. For illustration purposes, we have considered five functions for the TA example:

- Home: invoked when a user accesses the TA home page.
- Browse: the customer navigates through the links available at the TA site to view any page of the site. These links include weekly promotions, help pages, frequent queries, etc.
- Search: the TA checks the availability of trip offers according to the user specification. A user request can be composed of a flight, a hotel and a car reservation. Based on the information provided by the user, the TA converts the user requests into transactions to hotel, flight and car reservation systems and returning the results of the search to the user.
- Book: the customer chooses the trip that suits his request confirming his reservation.
- Pay: the customer is ready to pay for the trips booked on the TA site.

The operational profile defines all user execution scenarios (or shortly, user scenarios) when visiting the TA web site. Recall that the probabilities of activation with respect to the user scenarios can be derived using traditional methods such as flow graph reduction [Howard 1971b].

Table 2.2 shows the user scenarios obtained from Figure 2.7 and the corresponding probabilities of activation. The notations {Home - Browse}* and {Search-Book}* mean that these functions are activated more than once in the corresponding scenarios, due to the presence of cycles in the graph.

The identification of the most frequently activated scenarios provides useful insights into the most significant scenarios to be considered when evaluating the user perceived availability. Indeed, the higher the activation probability of a given scenario,

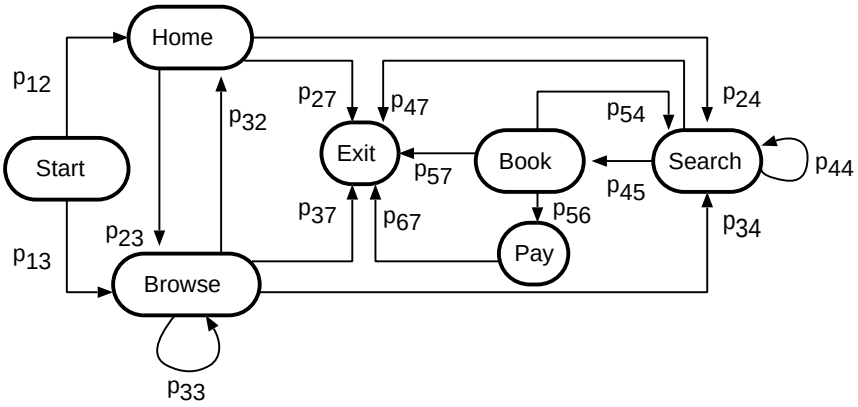


Figure 2.7: User operational profile graph

the higher its impact on the availability perceived at the user level. Such measure is affected by the availability of the functions, services and resources involved in the underlying scenario.

The scenarios listed in Table 2.2 can be grouped into four categories, denoted as SC1, SC2, SC3 and SC4 according to the activated functions:

- SC1 gathers all scenarios that lead to the execution of functions "Home" or "Browse" without invoking the other functions (scenarios 1-3 of Table 2.2).
- SC2 gathers all scenarios that include the invocation of the "Search" function, without going through the "Book" or "Pay" functions (scenarios 4-6 of Table 2.2). These scenarios may require several interactions between the TA and the flight, hotel and car reservation component systems. However, they do not end up with a booking or payment.
- SC3 gathers all scenarios that include the invocation of the "Book" function (scenarios 7-9 of Table 2.2). These scenarios involve several interactions between the TA and the booking systems.
- SC4 gathers all scenarios that reach the "Pay" function (scenarios 10-12 of Table 2.2). These scenarios end up with a payment.

Let us denote by $\Phi(SC1)$, $\Phi(SC2)$, $\Phi(SC3)$ and $\Phi(SC4)$ the activation probability of SC1, SC2, SC3 and SC4. These probabilities can be obtained from Table 2.2 by summing the probabilities associated to the corresponding scenarios.

Example of two user classes

Two user profiles (denoted as user class A and user class B) are defined with different values of transition probabilities p_{ij}. In particular, the class A is characterized

User scenario	Scenario activation probability (ϕ_i)
1: Start-Home-Exit	$p_{12}p_{27}$
2: Start-Browse-Exit	$\frac{p_{13}p_{37}}{1-p_{33}}$
3: Start-{Home-Browse}* -Exit	$\frac{p_{13}p_{32}p_{27}+p_{12}p_{23}p_{37}+p_{32}p_{23}[p_{12}p_{27}+\frac{p_{13}p_{37}}{1-p_{33}}]}{1-p_{33}-p_{32}p_{23}}$
4: Start-Home-Search-Exit	$\frac{p_{12}p_{24}p_{47}}{1-p_{44}}$
5: Start-Browse-Search-Exit	$\frac{p_{13}p_{34}p_{47}}{(1-p_{44})(1-p_{33})}$
6: Start-{Home-Browse}* -Search-Exit	$\frac{p_{47}(p_{13}p_{32}p_{24}+p_{12}p_{23}p_{34}+p_{32}p_{23}(p_{12}p_{24}+\frac{p_{13}p_{34}}{1-p_{33}}))}{(1-p_{33}-p_{32}p_{23})(1-p_{44})}$
7: Start-Home-{Search-Book}* -Exit	$\frac{p_{12}p_{24}(p_{45}p_{57}+\frac{p_{45}p_{54}p_{47}}{1-p_{44}})}{1-p_{44}-p_{45}p_{54}}$
8: Start-Browse-{Search-Book}* -Exit	$\frac{p_{13}p_{34}(p_{45}p_{57}+\frac{p_{45}p_{54}p_{47}}{1-p_{44}})}{(1-p_{44}-p_{45}p_{54})(1-p_{33})}$
9: Start-{Home-Browse}* -{Search-Book}*-Exit	$\frac{[p_{13}p_{32}p_{24}+p_{12}p_{23}p_{34}+p_{32}p_{23}(p_{12}p_{24}+\frac{p_{13}p_{34}}{1-p_{33}})][p_{45}p_{57}\frac{p_{45}p_{54}p_{47}}{1-p_{44}}]}{(1-p_{33}-p_{32}p_{23})(1-p_{44}-p_{45}p_{54})}$
10: Start-Home-{Search-Book}* -Pay-Exit	$\frac{p_{12}p_{24}p_{45}p_{56}p_{67}}{1-p_{44}-p_{45}p_{54}}$
11: Start-Browse-{Search-Book}* -Pay-Exit	$\frac{p_{13}p_{34}p_{45}p_{56}p_{67}}{(1-p_{44}-p_{45}p_{54})(1-p_{33})}$
12: Start-{Home-Browse}* -{Search-Book}*-Pay-Exit	$\frac{p_{45}p_{56}p_{67}(p_{13}p_{32}p_{24}+p_{12}p_{23}p_{34}+p_{32}p_{23}(p_{12}p_{24}+\frac{p_{13}p_{34}}{1-p_{33}}))}{(1-p_{44}-p_{45}p_{54})(1-p_{33}-p_{32}p_{23})}$

Table 2.2: TA user scenarios with associated probabilities ϕ_i

by a high proportion of users who are mainly seeking for information without a buying intention, whereas the class B is characterized by a higher proportion of users really seeking for booking a trip.

Tables 2.3 and 2.4 present the probability transition matrices associated to the class A and class B, respectively. Table 2.5 lists the user scenarios derived from Figure 2.7 and shows the probabilities of these scenarios (in terms of percentage) associated to the two user profiles. Table 2.6 shows the probabilities $\Phi(SC1)$, $\Phi(SC2)$, $\Phi(SC3)$ and $\Phi(SC4)$ associated with the scenario categories SC1 to SC4, involving the following functions: Browse, Search, Book and Pay respectively.

We note that for class B, 80% of user transactions lead to the invocation of the functions Search, Book or Pay. Such scenarios involve not only the TA system but also the external reservation systems. Therefore, the quality of the service supported by these reservation systems has a significant impact on the user perceived availability. The percentage is lower (50%) when considering the class A. It can be seen from Table 2.6 that the user class B exhibits a higher probability of activation for scenarios SC2, SC3 and SC4, compared to the user class A. The percentage of transactions that end up with a payment of a trip is around 20% for user class B while it is almost 3 times lower for user class A.

These two examples of user classes will be used in Section 2.4 to evaluate the user perceived availability.

	Start	Home	Browse	Search	Book	Pay	Exit
Start	0	0.50	0.50	0	0	0	0
Home	0	0	0.30	0.50	0	0	0.20
Browse	0	0.10	0.25	0.25	0	0	0.40
Search	0	0	0	0.25	0.20	0	0.55
Book	0	0	0	0.30	0	0.50	0.20
Pay	0	0	0	0	0	0	1
Exit	0	0	0	0	0	0	1

Table 2.3: Profile of user class A

2.3.2 Service and function levels

The service level identifies the set of servers involved in the execution of each function and describes their interactions. This analysis requires a deep understanding of the business logic and the technical solutions implemented in the TA system.

	Start	Home	Browse	Search	Book	Pay	Exit
Start	0	0.50	0.50	0	0	0	0
Home	0	0	0.30	0.50	0	0	0.20
Browse	0	0.10	0.25	0.55	0	0	0.10
Search	0	0	0	0.55	0.20	0	0.25
Book	0	0	0	0.30	0	0.50	0.20
Pay	0	0	0	0	0	0	1
Exit	0	0	0	0	0	0	1

Table 2.4: Profile of user class B

User scenario	Class A	Class B
1:Start-Home-Exit	10.0	10.0
2:Start-Browse-Exit	26.7	6.6
3:Start-{Home-Browse}*-Exit	11.3	4.2
4:Start-Home-Search-Exit	18.4	13.9
5:Start-Browse-Search-Exit	12.2	20.4
6:Start-{Home-Browse}*-Search-Exit	7.6	9.7
7:Start-Home-{Search-Book}*-Exit	3.0	4.7
8:Start-Browse-{Search-Book}*-Exit	2.0	6.9
9:Start-{Home-Browse}*-{Search-Book}*-Exit	1.3	3.3
10:Start-Home-{Search-Book}*-Pay-Exit	3.6	6.4
11:Start-Browse-{Search-Book}*-Pay-Exit	2.4	9.4
12:Start-{Home-Browse}*-{Search-Book}*-Pay-Exit	1.5	4.5

Table 2.5: User scenario probabilities (in %)

	$\Phi(SC1)$	$\Phi(SC2)$	$\Phi(SC3)$	$\Phi(SC4)$
Class A	47.9%	38.2%	6.4%	7.5%
Class B	20.8%	44%	14.9%	20.3%

Table 2.6: Scenario categories for user classes A and B

	Internal services			External services			
	Web	Application	Database	Flight	Hotel	Car	Payment
Home	◇						
Browse	◇	◇	◇				
Search	◇	◇	◇	◇	◇	◇	
Book	◇	◇	◇	◇	◇	◇	
Pay	◇	◇	◇				◇

Table 2.7: Mapping between functions and services

For the sake of illustration, Table 2.7 presents a simplified example of mapping between the functions provided at the TA site, the internal servers and the external servers managed by external suppliers.

The external suppliers correspond to the flight, hotel, and car reservation systems that provide information about a potential trip. Also, we assume that the TA provider uses the services of an external payment system for handling card-based transactions.

The internal services are supported by three types of servers: 1) Web servers that receive user requests and reply the requested data, 2) Application servers that implement the main operations needed to process user requests, and Database servers handling data related operations (for storing and retrieving information about flight, hotel and car reservation companies, as well as information about users orders).

The "Home" function execution involves only the web server. However, for the other functions several servers are required. In this case, it is necessary to analyze for each function the interactions among the servers and all possible execution scenarios (referred to as function scenarios). In the following, we present the interaction diagrams representing each function including Browse, Search, Book and Pay.

Browse

Figure 2.8 describes the interactions among the servers involved in the accomplishment of the Browse function. The "Begin" and "End" nodes identify the beginning and the end of a function execution. Each path from the "Begin" node to the "End" node identifies one possible function scenario. The probability of activation of each scenario can be evaluated taking into account the probabilities q_{ij} associated to the transitions of the underlying scenario. Note that the probability of activation of non-labeled transitions is one.

We can identify three scenarios described as follows:

- $1 \rightarrow 2 \rightarrow 3$: The user sends a request to the web server (node 2). The data requested is available and is returned back to the user (node 3). This marks the end of this interaction.

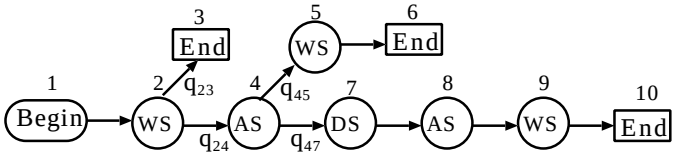


Figure 2.8: Interaction diagram of the Browse function

- $1 \rightarrow 2 \rightarrow 4 \rightarrow 5 \rightarrow 6$: The web server accepts the user request and sends it to the application server (node 4). In this case, the requested data is not available. The application server processes the request and returns a dynamically generated page to the web server (node 5). The latter is then forwarded to the user (node 6). The database is not involved in this case.
- $1 \rightarrow 2 \rightarrow 4 \rightarrow 7 \rightarrow 8 \rightarrow 9 \rightarrow 10$: The application server requires some specific information that is on the TA database server (node 7). After the database server has answered the application server, the latter processes the user request (node 8) and sends the results to the web server (node 9). The latter generates an HTML page incorporating the corresponding outputs (node 10).

Search

The interaction diagram describing the execution of the Search function is decomposed into 9 stages (Figure 2.9). The input data provided in the search request issued by the user (node 1) are first processed by the web server WS (node 2). WS performs necessary checks, and then breaks down the user request into three individual requests corresponding to each aspect of the trip. If data is correct and in the right format, it is forwarded to the application server AS (node 4), otherwise an exception is sent to the user (node 3). AS uses the request information to formulate a query and asks the database server (node 5) for the list of booking systems to be contacted. Based on the answer received, AS sends a query (node 6) to the selected systems (identified by the Flight, Hotel and Car nodes). The AND operator means that the request is submitted to the three types of booking systems (nodes 7.a, 7.b, 7.c). The answers returned to AS are formatted (node 8) and sent to WS (node 9) that forwards them to the user (node 10).

The number of Flight, Hotel and Car reservation systems is not indicated in this figure. We assume that the TA always interacts with the same systems. A transaction is successful when, for each service (Flight, Hotel and Car reservation), at least one system responds.

Book

An example of interaction diagram of the Book function is given in Figure 2.10. The trip booking order received from the user through WS is processed by AS. Using the parameters embedded in the book order, AS interacts with the corresponding flight, hotel and car booking systems to book the selected trip. The booking references

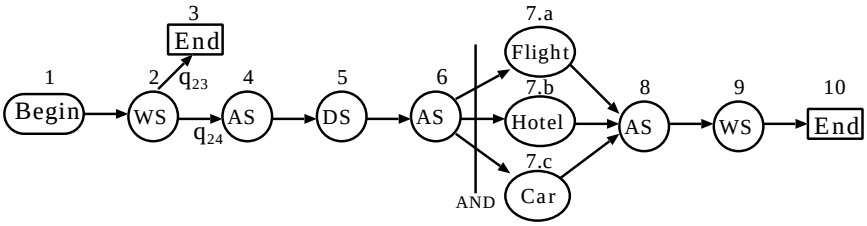


Figure 2.9: Interaction diagram of the Search function

returned to the application server are then stored in the database. After that, a confirmation is sent to the user.

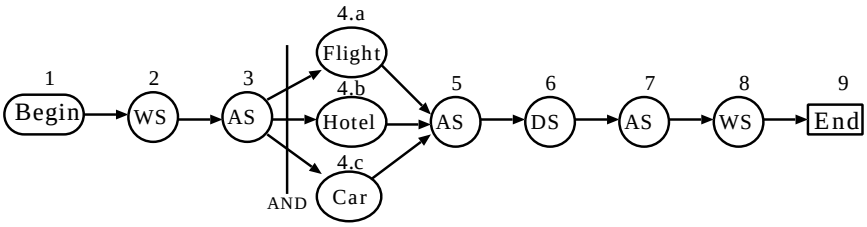


Figure 2.10: Interaction diagram of the Book function

Pay

The interaction diagram for the Pay function is presented in Figure 2.11. When a payment call is received through the web server, the booking data is first checked by the application server, then a call is sent to the payment server, for authentication and verification purposes, and definitely to accomplish the payment. Finally, the application server updates the information in the database concerning client orders, before sending a confirmation to the user.

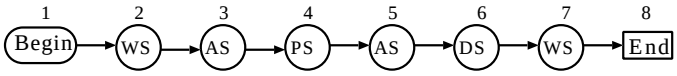


Figure 2.11: Interaction diagram of the Pay function

2.3.3 Resource level

The various services are mapped into the resources involved in their accomplishment. Therefore, we need to consider the real hardware and software organization of the system. With respect to external services, since the architecture is not known, we associate to each external service a single resource that is considered as a black box. For internal services, it is possible to detail the organization of internal resources for which the architecture is known.

For illustration purposes, we consider two simple architectures presented in figures 2.12 and 2.13. We assume that the external resources are identical for both architectures. They correspond to Flight reservation, Hotel reservation, Car reservation and Payment. Flight, hotel and car reservation services are provided by respectively N_F, N_H and N_C components each.

The basic architecture (Figure 2.12) consists of allocating a dedicated host to each server and interconnecting these hosts through a LAN. The LAN is viewed as a single resource providing communication among the servers.

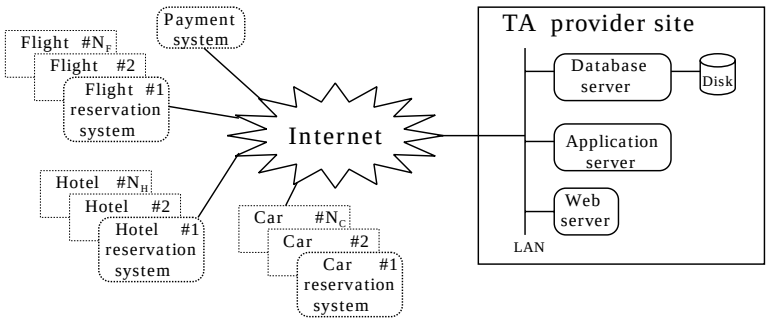


Figure 2.12: Basic architecture

The architecture described in Figure 2.13 employs redundancy in several points to improve the availability and scalability of the basic architecture. It is based on a server farm architecture with load balancing, including N web servers, two application servers and two database servers with two mirrored disks. Servers are connected through a LAN. Indeed, several LANs are usually used to interconnect these servers, nevertheless we will assume that all of them are represented as a single LAN. Also, to simplify the modeling, the load balancers are not explicitly described in this architecture.

In the following section, we will model the availability of both architectures.

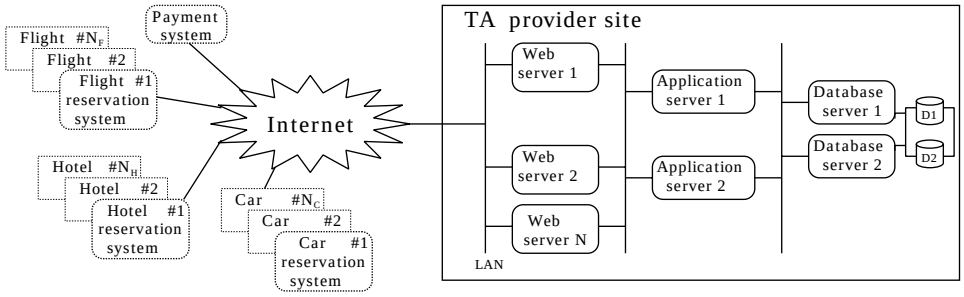


Figure 2.13: Redundant architecture

2.4 TA availability modeling

TA availability modeling will be carried out according to the hierarchical description of the system (see Figure 2.2), starting at the service level based on the description of the resource level for which two architectures are considered (Figures 2.12 and 2.13) .

2.4.1 Service level availability

In this section, we are interested in the evaluation of external and internal service availabilities.

2.4.1.1 External services

Each external system is modeled as a black box that is assumed to fail independently of all the others.

Let us consider the following notations: $A_F(i)$, $A_H(j)$ and $A_C(k)$: Availabilities of a flight, hotel and car reservation system, (i = 1, ..., N_F; j = 1, ..., N_H; k = 1,..., N_C).

- A_{PS} : Availability of the payment system.
- A_{net}: Availability of the TA connectivity to the Internet.

Using the failure independence assumption and considering that the service is provided as long as at least one reservation system for each item of a trip (flight, hotel and car reservation) is available, the availability of the external services can be derived as in Table 2.8. It is worth mentioning that if the TA connectivity to the Internet is unavailable, none of these services is provided. Thus, the availability of the TA connectivity to the Internet will be taken into account by multiplying the user perceived availability equation by A_{net}.

$A(Flight) = 1 - \prod_{i=1}^{N_F}[1 - A_F(i)]$	$A(Hotel) = 1 - \prod_{i=1}^{N_H}[1 - A_H(i)]$
$A(Car) = 1 - \prod_{i=1}^{N_C}[1 - A_C(i)]$	$A(Payment) = A_{PS}$

Table 2.8: External service availability

2.4.1.2 Internal services

The internal services are related to web, application and database services. For both architectures, the communication between servers is achieved by a local area network (LAN). Although an LAN can be fault tolerant supporting high availability, the LAN is assumed to be a single point of failure, i.e., when the LAN is unavailable, all internal services are unavailable. As a consequence, the LAN availability, denoted by A_{LAN}, is a multiplying factor for all equations in the following sections. A_{LAN} can be evaluated using for example the models discussed in [Kanoun & Powell 1991, Hariri & Mutlu 1991].

Since the first goal of the TA example is to show the applicability of the framework illustrating the main concepts, simplistic assumptions are stated for application and database services.

Application and database service availability

Let us denote by A_{CAS} and A_{CDS} the availabilities of the computer hosts associated with the application and database servers, respectively. The disk availability is denoted by A_{Disk}. To simplify the presentation, we assume that the computer hosts and the disks fail independently of each other. The application and database service availability (denoted $A(AS)$ and $A(DS)$) are given in Table 2.9.

	Basic architecture	Redundant architecture
$A(AS)$	A_{CAS}	$1 - [1 - A_{CAS}]^2$
$A(DS)$	$A_{CDS}\ A_{Disk}$	$(1 - [1 - A_{CDS}]^2)(1 - [1 - A_{Disk}]^2)$

Table 2.9: Application and database service availability

In the following, we focus on the evaluation of web service availability considering basic and redundant architectures, respectively.

Web service availability

To evaluate the availability of the web service, we distinguish two sources of failure:

1. hardware and software failures that affect the computer host and lead to web server failure.
2. performance-related failures that occur when the incoming requests are not served due to the limited capacity of the web servers.

The web service is assumed to be available when no failures of these types occurs. The impact of both types of failures on the web service availability can be accounted for by adopting a composite performance and availability evaluation approach (see section 1.4.2). The main idea consists of combining the results obtained from two models: a pure performance model and a pure availability model. The performance model takes into account the request arrival and service processes and evaluates performance related measures conditioned on the state of the system as determined from the availability model. The availability model is used to evaluate the steady state probability associated to the system states that result from the occurrence of failures and recoveries.

This approach is based on the assumption that the system reaches a quasi steady state with respect to the performance-related events, between successive occurrences of failure-recovery events. This assumption is valid when the failure/recovery rates are much lower than the request arrival/service rates, which is typically true in our context.

Basic architecture

The web service relies on a unique computer host. Let us denote by p_b the probability that the web server input buffer (whose size is b) is full upon a request arrival. The evaluation of p_b is derived from the performance model and depends on the assumptions made about the request arrival process and the request service process. Let us assume that the request arrivals are modeled by a Poisson process with rate λ and the request service times are exponentially distributed with rate μ. Then the web server behavior can be modeled by an M/M/1/b queue. In this classical queueing system, the probability that an arriving request is lost due to buffer overflow is well-known (see e.g., [Bolch et al. 1998]):

$$p_b = \begin{cases} \rho^b \frac{1-\rho}{1-\rho^{b+1}} & \text{, if } \lambda \neq \mu \\ \frac{1}{b+1} & \text{, if } \lambda = \mu \end{cases} \tag{2.5}$$

where $\rho = \frac{\lambda}{\mu}$ is the server load.

The traditional availability model consists of two states: up and down states. The steady state probability of the up state corresponds to the system steady-state availability denoted A_{CWS}. Thus, the availability of the web service is:

$$A(WS) = (1 - p_b) A_{CWS} \tag{2.6}$$

Redundant Architecture

The redundant architecture is composed of N identical web servers. We assume that all component failures are independent and that the web service is provided as long as at least one of the redundant component systems is available. The performance model representing this architecture is assumed to be a M/M/c/b queue, where c is the number of operational servers and b is the size of the buffer. For a system with c operational servers, the probability that web requests are lost due to buffer overflow, denoted $L_b(c)$, is given by (see, e.g. [Gross & Harris 1985]):

$$L_b(c) = \begin{cases} \frac{\rho^b}{c^{b-c}c!}[\sum_{j=0}^{c-1}\frac{\rho^j}{j!} + \sum_{j=c}^{b}\frac{\rho^j}{c^{j-c}c!}]^{-1} & \text{, if } b \geq c \\ \frac{\rho^b}{b!}[\sum_{j=0}^{b}\frac{\rho^j}{j!}]^{-1} & \text{, if } b < c \end{cases} \tag{2.7}$$

With respect to the availability model, the aim is to evaluate the redundant architecture behavior resulting from the occurrence of failures/repairs, in order to calculate the steady state probabilities associated to system states c (c is the number of operational servers, as denoted above). Two assumptions are made with regards to the coverage of web server failures: 1) perfect coverage, and 2) imperfect coverage.

Perfect coverage

In Figure 2.14, it is assumed that each web server runs on a dedicated computer host. Web server failures occur with rate γ. The model assumes shared repair facilities with repair rate τ. When a server fails, it is automatically disconnected and the system is reconfigured (with probability 1) with the web servers that are still operational.

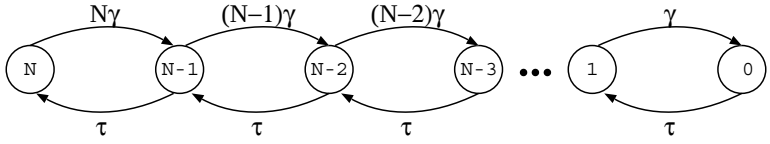


Figure 2.14: Perfect coverage model

Let us denote by π_c the steady-state probability of state c, $c = 0, 1, ..., N$. In state c, there are c web servers available to process the incoming requests (π_0 corresponds to the state in which all web servers are down). The availability of the web service is as follows:

$$\pi_c = \frac{1}{c!}\left(\frac{\tau}{\gamma}\right)^c \pi_0, \qquad c = 1 \cdots N \tag{2.8}$$

with

$$\pi_0 = \left[\sum_{c=0}^{N} \frac{1}{c!} \left(\frac{\tau}{\gamma}\right)^c \right]^{-1} \tag{2.9}$$

The web service availability is:

$$A(WS) = 1 - [\sum_{c=1}^{N} \pi_c L_b(c) + \pi_0] \tag{2.10}$$

in which $L_b(c)$ is computed using equation 2.7.

This equation corresponds to the probability that a web request is not served either due a) to buffer overflow or b) to the fact that all servers are unavailable.

Imperfect coverage

This model is described in Figure 2.15. From each state c, two transitions are considered:

1. After a covered failure (transition with rate $cv\gamma$) the system is automatically reconfigured into an operational state with $(c-1)$ web servers, where v is the coverage factor.
2. Upon the occurrence of an uncovered failure (transition with rate $c(1-v)\gamma$) the system moves to a down state Y_c, where a manual reconfiguration is required before moving the system to an operational state $(c-1)$. The reconfiguration times are exponentially distributed with mean $1/\beta$.

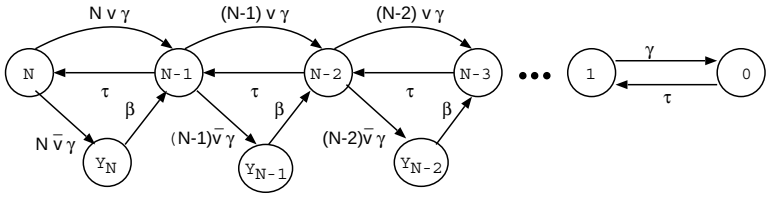


Figure 2.15: Imperfect coverage model

Solving this model at steady-state, we obtain:

$$\pi_c = \frac{1}{c!} \left(\frac{\tau}{\gamma}\right)^c \pi_0, \qquad c = 1 \cdots N \tag{2.11}$$

$$\pi_{Y_c} = \frac{\tau(1-v)}{\beta(c-1)!} \left(\frac{\tau}{\gamma}\right)^{c-1} \pi_0, \qquad c = 2 \cdots N \tag{2.12}$$

$$\pi_0 = \left[\sum_{j=0}^{N} \frac{1}{j!}(\frac{\tau}{\gamma})^j + \sum_{j=2}^{N}(\frac{\tau}{\gamma})^{j-1}\frac{\tau(1-v)}{\beta(j-1)!} \right]^{-1} \quad (2.13)$$

In this example, we assume that states Y_c correspond to down states. [1] Accordingly, the availability of the web service is computed as follows:

$$A(WS) = 1 - [\sum_{c=1}^{N} \pi_c L_b(c) + \sum_{c=2}^{N} \pi_{Y_c} + \pi_0] \quad (2.14)$$

Summary

Table 2.10 recalls all equations of the web service availability for basic and redundant architecture, assuming perfect and imperfect coverage.

2.4.2 Function level availability

The availability evaluation of each function is based on the availabilities of the services involved in its accomplishment. When many scenarios of execution are possible, the availability of each function relies on the activation probability of each scenario. Table 2.11 gives the availability for Home, Browse, Search, Book and Pay functions. $A(WS)$, $A(AS)$ and $A(DS)$ correspond to A(Web service), A(Application service) and A(Database service) (Tables 2.9 and 2.10). A_{PS} corresponds to A(Payment service) as well as A(Flight), A (Hotel) and A(Car) are given in Table 2.8. Parameters q_{ij} involved in the availability of the Browse function are associated to the three execution scenarios of this function discussed in Section 2.3.2.

Note that all equations include the product $A_{net}\ A_{LAN}$, therefore if the TA connectivity to the Internet or the internal communication among the servers is not available, none of the TA functions can be invoked by the users. Also, the Book function has the same availability equation as the Search function. This is due to the assumption that the former uses a subset of the resources used by the latter. Indeed, in the Book example, this function is achieved only if the Search function has succeeded.

[1] For illustration purposes, we deliberately consider simple assumptions in this chapter. More realistic assumptions, leading to more complex analytic models are considered in the following chapters.

Basic architecture

$$A(WS) = (1 - p_b)A_{CWS} \qquad p_b = \begin{cases} \rho^b \frac{1-\rho}{1-\rho^{b+1}} & , \rho \neq 1 \\ \frac{1}{b+1} & , \rho = 1 \end{cases}$$

Redundant architecture (perfect coverage)

$$A(WS) = 1 - [\sum_{c=1}^{N} \pi_c L_b(c) + \pi_0] \qquad \pi_c = \frac{1}{c!}\left(\frac{\tau}{\gamma}\right)^c \pi_0$$

$$L_b(c) = \begin{cases} \frac{\rho^b}{c^{b-c}c!}[\sum_{j=0}^{c-1}\frac{\rho^j}{j!} + \sum_{j=c}^{b}\frac{\rho^j}{c^{j-c}c!}]^{-1} & , \text{if } b \geq c \\ \frac{\rho^b}{b!}[\sum_{j=0}^{b}\frac{\rho^j}{j!}]^{-1} & , \text{if } b < c \end{cases} \qquad \pi_0 = \left[\sum_{c=0}^{N} \frac{1}{c!}\left(\frac{\tau}{\gamma}\right)^c \right]^{-1}$$

Redundant architecture (imperfect coverage)

$$A(WS) = 1 - [\sum_{c=1}^{N} \pi_c L_b(c) + \sum_{c=2}^{N} \pi_{Y_c} + \pi_0] \qquad \pi_c = \frac{1}{c!}\left(\frac{\tau}{\gamma}\right)^c \pi_0$$

$$L_b(c) = \begin{cases} \frac{\rho^b}{c^{b-c}c!}[\sum_{j=0}^{c-1}\frac{\rho^j}{j!} + \sum_{j=c}^{b}\frac{\rho^j}{c^{j-c}c!}]^{-1} & , \text{if } b \geq c \\ \frac{\rho^b}{b!}[\sum_{j=0}^{b}\frac{\rho^j}{j!}]^{-1} & , \text{if } b < c \end{cases} \qquad \pi_{Y_c} = \frac{\tau(1-v)}{\beta(c-1)!}\left(\frac{\tau}{\gamma}\right)^{c-1} \pi_0$$

$$\pi_0 = \left[\sum_{j=0}^{N} \frac{1}{j!}(\frac{\tau}{\gamma})^j + \sum_{j=2}^{N}(\frac{\tau}{\gamma})^{j-1}\frac{\tau(1-v)}{\beta(j-1)!} \right]^{-1}$$

Table 2.10: Web service availability

$$A(Home) = A_{net}A_{LAN}A(WS)$$

$$A(Browse) = A_{net}A_{LAN}A(WS)[q_{23} + A(AS)(q_{24}q_{45} + q_{24}q_{47}A(DS))]$$

$$A(Search) = A(Book) = A_{net}A_{LAN}A(WS)A(AS)A(DS)A(Flight)A(Hotel)A(Car)$$

$$A(Pay) = A_{net}A_{LAN}A(WS)A(AS)A(DS)A_{PS}$$

Table 2.11: Function level availabilities

2.4.3 User level availability

The user perceived availability is obtained by evaluating the availability of each user execution scenario derived from the operational profile. If many functions are invoked in a given scenario, there may be several dependencies among the functions due to shared services or resources. A careful analysis of the dependencies is needed in order to evaluate the availability associated to the scenario of the corresponding functions.

Based on the activation probabilities of all user scenarios denoted ϕ_i (Table 2.2) with the respective availability of the functions involved in each scenario, we have the following equation:

$$\begin{aligned} A(user) =& A_{net}A_{LAN}A(WS)[\phi_1 + (\phi_2 + \phi_3)\{q_{23} + A(AS)(q_{24}q_{45} + q_{24}q_{47}A(DS))\}+ \\ & A(AS)A(DS)A(Flight)A(Hotel)A(Car)\{(\phi_4 + \phi_5 + \phi_6 + \phi_7 + \phi_8 + \phi_9)+ \\ & (\phi_{10} + \phi_{11} + \phi_{12})A_{PS}\}] \end{aligned} \tag{2.15}$$

This equation integrates in a combinatorial way all the scenarios with the availability of the functions involved in each scenario. It can be seen that A_{LAN}, A_{net} and $A(WS)$ are the most relevant availabilities since their impact is of the first order, while the others are at least of the second order. This is explained by the fact that all requests (i.e., user scenarios) use these three services.

2.5 Evaluation results

We first evaluate the web service availability of the two architecture described in section 2.3.3. Then, based on the various equations derived so far, the user availability as perceived by user classes A and B will be evaluated.

2.5.1 Web service availability results

Figures 2.16 and 2.17 show the web service availability for perfect and imperfect fault coverage, with the number of web servers N varying from 1 to 10. When only one web server is used ($N = 1$), the results correspond to the basic architecture. The parameters used to obtain these curves are indicated on the figures. Sensitivity analyses are made considering different values of web server failure rates (10^{-2}, 10^{-3} and 10^{-4} per hour) and request arrival rates (50, 100 and 150 req/s). It is assumed that each web server has a service rate of $\mu = 100$ req/s and a repair rate $\tau = 1$ per hour. The mean reconfiguration rate of the web server architecture β is 12 per hour (i.e., $1/\beta = 5$ min) and the buffer size b is assumed to be 10.

Both figures show that increasing the number of web servers N (depending on the failure and request arrival rates) reduces the web service unavailability. However, when the coverage is imperfect for N higher than 4, the trend is reversed (Figure 2.17). This is due to the fact that when the coverage is imperfect, increasing the number of servers also increases the probability of the system being in states Y_c. In these states, the web service is unavailable and a manual reconfiguration action is required. In fact, the request loss probability plays a significant role until a certain value of N. When the number of servers is higher than a threshold value, the total service rate and the buffer capacity are sufficient to handle the flow of arrivals without rejecting requests. In this case, the unavailability of the web service is caused by hardware and software failures leading the web server architecture to a down state. Compared to the imperfect coverage model, it can be noticed that the model with perfect coverage is more sensitive to the variation of N. Indeed, the unavailability decreases exponentially with N, and this trend is not reversed for values higher than 4. Also, the web servers failure rate has a significant impact on availability only when the system load ($\rho = \lambda/\mu$) is lower than 1.

Design decisions can be made based on these results. In particular, we can determine the number of servers needed to achieve a given availability requirement, or evaluate the maximum availability that can be obtained when the number of servers is set to a given value. For instance, considering the model with imperfect coverage, the number of servers needed to satisfy an unavailability lower than 5 min/year (unavailability $< 10^{-5}$), with a failure rate equal to 10^{-3} per hour will be at least $N = 2$ if the request arrival rate is 50 req/s and $N = 4$ if the request arrival rate is 100 req/s. We obtain the same result with a failure rate of 10^{-4} per hour, however such a requirement cannot be satisfied with a failure rate of 10^{-2} per hour.

Similar sensitivity analyses can be done to study the level of availability that can be achieved when the number of web servers is set to a given value. For example, if we decide to employ three servers to support the web service, we would have an unavailability lower than 1 hour per year, when the failure rate varies from 10^{-2} per hour to 10^{-4} per hour and $\rho < 1$.

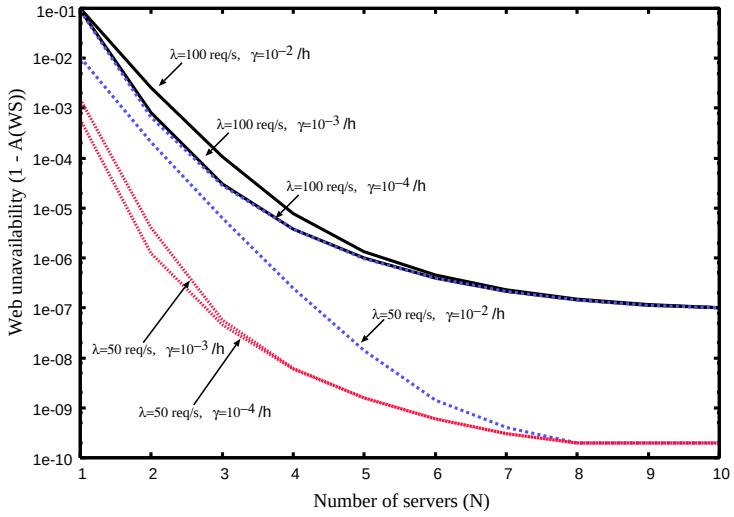


Figure 2.16: Web service unavailability with perfect coverage

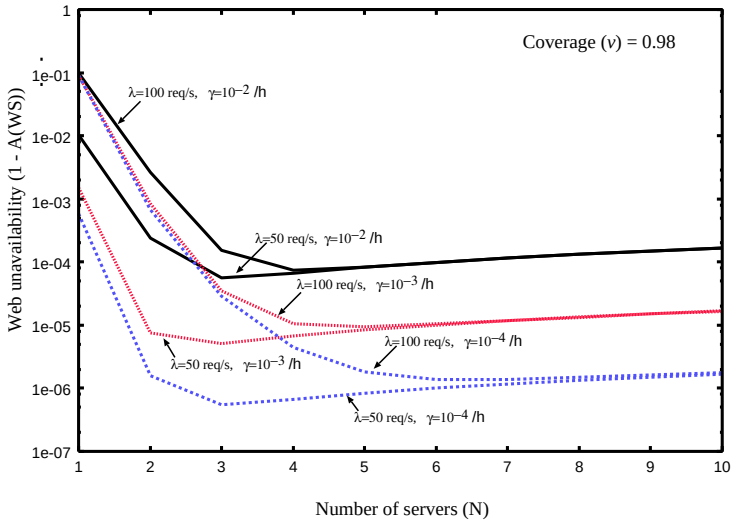


Figure 2.17: Web service unavailability with imperfect coverage

2.5.2 User level availability results

Considering equation 2.15, we will evaluate the availability as perceived by user classes A and B. The values of the parameters involved in this equation are given in Table 2.12. The probabilities characterizing user execution scenarios for classes A and B have been presented in Table 2.5. It is assumed that the web service is implemented on four servers, with imperfect coverage ($N = 4$, $v = 0.98$, $\lambda = 100$ req/s and $\gamma = 10^{-4}$/hour). For this study, A_{net} is assumed to be in the order of two nines according to the work of [Kalyanakrishnan et al. 1999]. $A(WS)$ is derived from the results presented in section 2.5.1.

$A_{net} = A_{LAN} = 0.9966$	$A_{CAS} = A_{CDS} = 0.996$	$A_{Disk} = 0.9$
$A_{PS} = A_F(i) = A_H(i) = A_C(i) = 0.9$	$A(WS) = 0.999995587$	$q_{23} = 0.2$
$q_{24} = 0.8$	$q_{45} = 0.4$	$q_{47} = 0.6$

Table 2.12: Numerical values of the model parameters

Table 2.13 presents the user perceived availability for user classes A and B, considering different values of flight, car and hotel reservation systems (N_F, N_H, N_C) that are external to the web site and interact with the travel agency system. Let us consider the same number of systems for N_F, N_H and N_C.

$N_F = N_H = N_C$	$A(users_A)$	$A(users_B)$
1	0.84235	0.76875
2	0.96509	0.95529
3	0.97867	0.97593
4	0.98004	0.97802
5	0.98018	0.97822
10	0.98020	0.97825

Table 2.13: User availabilities for classes A and B

The results show that for both user classes, user perceived availability increases significantly when the number of reservation systems increases from 1 to 4, and then increases slowly. The availability variation rate is directly related to the availability assigned to each reservation system.

A comparison between class A and B shows that different operational profiles might lead to significant differences on user perceived availability. For instance, considering the case $N_F = N_H = N_C \geq 5$, the user perceived unavailability is about 173 hours

per year for class A and 190 hours for class B. Such unavailability takes into account all the scenarios that might be invoked by the users.

The user perceived availability can be analyzed from another perspective by grouping user scenarios listed in Table 2.6 into four categories, denoted as SC1, SC2, SC3 and SC4, and evaluating the contribution of each category into the user perceived availability. Figure 2.18 shows users class A and class B, assuming that the web service is implemented with four servers with imperfect coverage. UA (users A) denotes the unavailability perceived by users class A, and $UA(SCi)$ denotes the contribution of scenarios SCi into the user perceived unavailability.

It can be seen that the unavailability caused by scenarios SC4 that end up with a trip payment is higher for class B compared to class A (43 hours downtime per year for class B compared to 16 hours for class A). Therefore, the impact in terms of loss of revenue for the TA provider will be higher. Indeed, if we assume that the users transaction rate is 100 per second, the total number of transactions ending up with a payment that are lost is 5.7 millions for class A and 15.5 millions for class B. Assuming that the average revenue generated by each transaction is 10 euros, then the loss of revenue becomes 57 millions of euros and 155 millions of euros, respectively. This result clearly shows that it is important to have a faithful estimation of the user operational profile to obtain realistic predictions of the impact of failures from the economic and business viewpoints.

2.6 Conclusion

The evaluation of quantitative measures characterizing user-perceived availability for web-based applications is widely recognized as highly important to faithfully reflect the impact of failures from the business point of view. However, there is still a lack of modeling framework and examples illustrating how to address this issue. This chapter has been intended to fill this gap by illustrating on a simplified example how models can be built and what kinds of practical results can be derived.

A hierarchical modeling framework for dependability evaluation of web-based applications is presented in this chapter. The framework relies on hierarchical decomposition using some concepts defined in [Menascé & Almeida 2000] to describe e-business applications and analyze their performance. However, since our framework is oriented to dependability modeling and evaluation from the user perspective, we have refined these concepts and extended them to fulfill the objectives of our study.

We have illustrated the main concepts of the framework proposed in [Kaâniche et al. 2001, Kaâniche et al. 2003b] through a modeling example of web based travel agency [Kaâniche et al. 2002, Kaâniche et al. 2003a]. Our objectives in this study were:

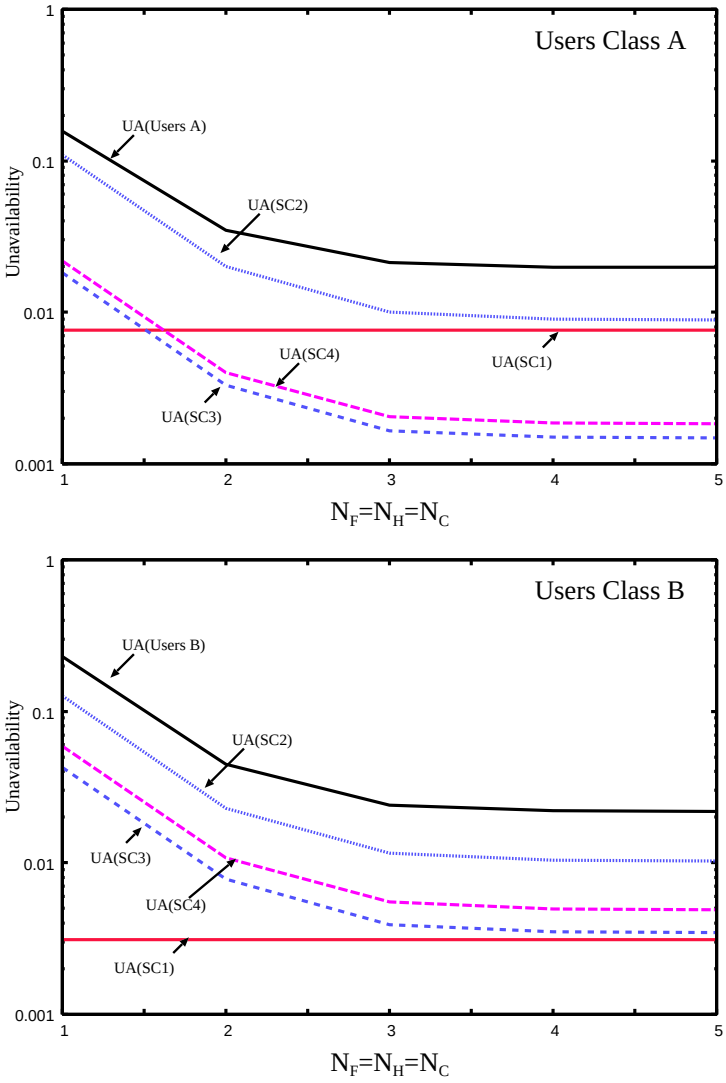


Figure 2.18: User perceived unavailability with $UA(SCi)$

1. to show how to apply our framework considering the decomposition of the target system according to four levels in a top-down approach: user, function, service, and resource levels.

2. to present typical availability analysis and evaluation results. The sensitivity analysis results show how web based applications are affected from different point of views: web service availability and user perceived availability. In practice, the obtained results provide useful guidelines orienting web designers in design decisions.

For the sake of illustration, we have deliberately considered simplified assumptions, dealing with users operational profile and TA architecture models. The effects of these aspects have been evaluated on the user perceived availability. The availability measure considered takes into account the impact of performance related failures as well as traditional software and hardware failures. Also, we have shown that the proposed hierarchical framework provides a systematic and pragmatic modeling approach, that is necessary to evaluate availability characteristics of the target application at different levels of abstractions.

In the following chapters, we develop more detailed analytic models for the web service availability focusing on the resource level. In chapter 3, we address especially recovery strategies issues and traffic burstiness effects on web service availability. In chapter 4, we deal with service unavailability due to long response time.

Chapter 3

Web service availability: impact of recovery strategies and traffic models

> Whenever you find you are on the side of the majority, it is time to pause and reflect.
>
> Mark Twain

In this chapter, the web service availability is analyzed focusing on the service and resource level of the modeling framework (see Figure 2.2) introduced in chapter 2. At the resource level, a web cluster architecture consisting of various web servers is used for supporting the service. We address especially recovery strategies in web cluster architectures. At the service level, we evaluate the traffic burstiness effects and the impact of recovery strategies on web service availability supported by web clusters.

Web systems with multiple nodes are leading architectures for building popular web sites that have to guarantee scalable, highly available and reliable services. Web clusters consisting of multiple nodes have proved to be a promising and cost-effective approach to support such services [Brewer 2001, V. Cardellini & Yu 2002]. Many giant service providers run on a cluster of nodes (e.g. Yahoo, eBay, Alta Vista, Netscape) supporting millions of requests per day.

This chapter presents an analytic modeling approach that is aimed at evaluating the web service availability taking into account explicitly:

1. the cluster architecture characteristics, considering the number of nodes in the cluster, the recovery strategy after a node failure, as well as the reliability of cluster nodes;
2. the traffic model, describing the web traffic characteristics (i.e. the access patterns);
3. various causes of request loss due to buffer overflow, or node failures as well as during recovery time.

Analytical performability models are developed to analyze the impact of the above aspects on the service availability based on web clusters. We obtain closed form equations for web service availability, for which sensitivity analyses are conducted with respect to cluster and traffic characteristics.

3.1 Introduction

Most popular sites (e.g. Altavista, Yahoo) support millions of requests per day using mainly web cluster architectures [Oppenheimer & Patterson 2002, V. Cardellini & Yu 2002, Brewer 2001]. Such architectures are composed of multiple web server nodes and one or several frontend dispatchers for distributing client requests to the servers.

One of the main problems faced by the web systems designers is to find an adequate sizing of the architecture to ensure high availability and performance for the delivered services. Modeling techniques have shown to be well suited to address this problem and to find the right tradeoffs for achieving availability while providing acceptable levels of performance.

A significant body of work has focused on various aspects of web cluster performance evaluation. Particular attention has been devoted to the performance analysis considering different algorithms for load balancing among the servers [V. Cardellini & Yu 2002]. Although many efforts have been dedicated to analyze the availability of web hosts using measurement based techniques [Oppenheimer & Patterson 2002, Kalyanakrishnan et al. 1999], less emphasis was devoted to the modeling of the web service availability taking into account the impact of server node failures and performance degradations. In this chapter, models are developed considering explicitly i) the cluster characteristics especially related to the recovery strategies and ii) the essential characteristics of the web traffic.

Regarding the *cluster characteristics*, we concentrate our attention on two recovery strategies, referred to as non client transparent and client transparent. In the first strategy, all the submitted requests are lost as long as a node failure has not been detected. In the second strategy, the submitted requests are smoothly migrated to the remaining nodes in a user transparent way.

Concerning *traffic characteristics*, to accurately represent the access patterns of real web based environments, a traffic model must capture the relevant properties of the request patterns for the system of interest. Some of the properties considered in the literature deal with file size distributions, self-similarity [Crovella & Bestavros 1997], reference locality [Almeida et al. 1996]. As discussed in section 1.6, the results presented in [Iyengar et al. 1999, Morris & Lin 2000] suggested that the aggregate web traffic tends to smooth out as Poisson traffic. In addition, other works [Chen et al. 2001, Muscariello et al. 2004] have shown that Modulated Markov Poisson Process (MMPP) is enable to capture relevant characteristics of the input traffic.

Based on the above observations, we consider two traffic models representing the request arrivals. First, a pure Poisson process capturing the average arrival rate of requests to the cluster, and secondly, a multi-stage Modulated Markov Poisson Process (MMPP) are used for modeling the arrival rate of requests varying according to the period of the day (daily cycles).

The rest of this chapter is organized as follows. Section 3.2 presents an overview of web cluster architectures focusing on fault tolerance strategies. Section 3.3 introduces the modeling assumptions. Section 3.4 describes the modeling approach providing closed-form equations for web service availability. Section 3.5 presents sensitivity analysis results and section 3.6 concludes the chapter. The results presented along this chapter were recently published in [Martinello et al. 2003, Martinello et al. 2005].

3.2 Fault tolerance strategies in web clusters

A cluster-based web system (briefly, *web cluster*) refers to a collection of machines that are housed together in a single location, interconnected through a high-speed network, and present a single system image to the outside. In literature, some alternative terminology is used to refer to a web cluster architecture for example *web farm*. We will use the term web cluster as proposed in [V. Cardellini & Yu 2002].

A typical web cluster consists of a front-end dispatcher (sometimes called load balancer) and many back-end servers. Over the past few years, a number of web cluster architectures have been proposed. These architectures implement a large variety of strategies for routing new requests (load balancing) and for fault tolerance. [V. Cardellini & Yu 2002] presents a recent overview of web cluster architectures.

3.2.1 Non Client Transparent (NCT) recovery strategy

Non client transparent (NCT) recovery strategy corresponds essentially to traditional web cluster solutions. These solutions do not provide transparent handling of requests at the time of node failure.

Round robin DNS [Brisco 1995] and DNS aliasing are examples of architectures in which the new requests are routed to available servers. If server replicas are running

on multiple hosts, these schemes can be used to provide fault tolerance for web service by changing the host name to IP address mapping depending on the state of the system. When a server failure is detected, the host name is no longer mapped to the IP address of the failed server host. As a result requests arriving after a failure will not be routed to failed servers. This scheme requires that clients re-issue the request if they do not receive a reply. In practice, clients may continue to see the old mapping due to DNS caching at the clients and DNS servers. This reduces the effectiveness of this scheme since after a failure, the client may need to re-issue the request several times before it is routed to a new server. Also, there is no support for recovering requests under processing at failure time.

Centralized schemes, such as Magic Router [E. Anderson & Brewer 1996] and Cisco Local Router [Inc 2000], require requests to travel through a central router where they are routed to the desired server. Typically, the router detects server failures and does not route packets to servers that have failed. The central router is a single point of failure and a performance bottleneck since all packets must travel through it. Distributed Packet Rewriting [Aversa & Bertavros 2000] avoids single entry point by allowing the servers to send messages directly to clients and by implementing some of the router logic in the servers, so that they can forward the requests to different servers. However, these architectures are only capable to provide high availability via redundancy. Actually, a failed component can be replaced with the available redundant component. None of these schemes support recovering requests that were being processed when the failure occurred, in other words all ongoing requests on the failed node will be lost (i.e., these requests must be resent by the users).

3.2.2 Client Transparent (CT) recovery strategy

Client transparent (CT) recovery strategy supports requests migration. It enables web requests to be smoothly migrated and recovered on other working node(s) in the presence of server failure, in a user transparent way. Recently, there have been cluster implementations providing client transparent recovery strategies, e.g. [Aghdaie & Tamir 2001, Luo & Yang 2002, Zhang et al. 2004]. These architectures support requests migration for static and dynamic objects.

In [Luo & Yang 2002], if a node fails while processing a request related to static objects, then the dispatcher will select a new server node with an idle pre-forked connection connected with the target server re-binding the client-side connection to the new selected server connection. After the new connection binding is determined, the dispatcher issues a "range request" on the new server connection. The "range request" is defined in the http 1.1 protocol allowing a client to request portions of a resource. Using this property, we can enable a request to continue downloading a file from another node. The web requests for dynamic content, for which responses are created on demand (cgi, scripts, asp) are mostly based on client-provided arguments. The size and content of such response is variable. A request related to a dynamic object is not "idempotent", i.e. sometimes the result of two successive requests may be different due for example to updates of the database. Consequently, they solve

this problem using "store and forward" of the response of a dynamic request. In other words, the dispatcher will not relay the response to the client until it receives the complete result.

On the other hand, the basic idea in the design of [Aghdaie & Tamir 2001] is to use the error handling mechanisms of TCP to ensure that the backup has a copy of each request before it is available to the primary. Once a reply is generated by the primary, a complete copy is sent to the backup before any reply has been sent to the client. If the primary fails before starting to transmit the reply, the backup can transmit its copy. If it fails while sending the reply, the error handling mechanism of TCP is used.

Additionally, a transparent TCP connection failover mechanism is presented in the architecture proposed by [Zhang et al. 2004]. The architecture consists of four components namely 1) failure detection, 2) ring maintenance, 3) connection state tracking and 4) connection failover. Periodically, each back-end server sends out a heartbeat message to the dispatcher. If the dispatcher does not heard from a server for some duration, this server is considered to be dead. In this architecture, the servers are organized into a ring. Consequently, each server can play the role of backup server for a predecessor and a successor in the ring. After a server failure detection, one of its backups is notified to take over its connections and the ring maintenance is invoked to repair the ring structure. The connection state tracking and failover are supported by a protocol named backup tcp (btcp). This protocol basically processes each incoming request in a similar way as tcp does, but it never sends back any reply to the client.

3.3 Modeling assumptions

Figure 3.1 shows a simple example of a clustered architecture composed of multiple server nodes with a dispatcher that distributes the incoming requests among the nodes. For this study, we assume that the arriving traffic is dispatched to the nodes according to a round robin strategy. It is also assumed that each node has an associated buffer with limited capacity. Thus, all the requests sent to the node are lost, if its buffer is full. In addition, the dispatcher runs a monitoring process, e.g., based on heartbeat messages in order to detect node failures. The objective is to early detect the failed nodes and disconnect them from the cluster. So, the cluster supports a failure-detection mechanism that is able to provide a *fail-stop* behavior, i.e. a node crashes and stops working in a way detectable from its neighbors. This is known as the *fail-stop* assumption.

In order to evaluate the web service availability, we focus on the server side that is under the direct control of the web designer. We do not include in our analysis the service unavailability caused by failures in the path from the client to the server, or by the failure of the dispatcher. Thus, we take the web designer perspective.

The web service availability is defined as the probability at steady state that requests submitted to the cluster are successfully processed. In other words, they are not lost due to server failures or overloads. We distinguish two main causes of failure:

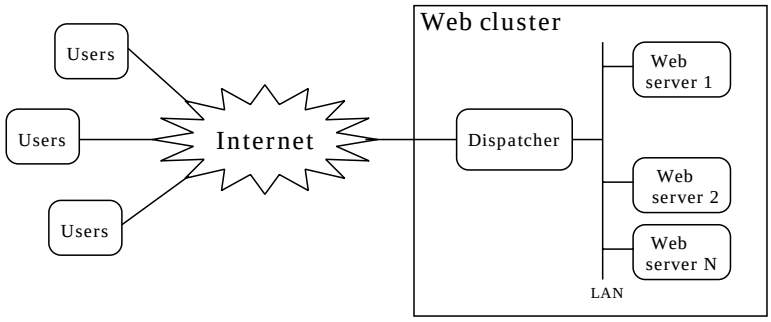


Figure 3.1: Basic web cluster architecture

1. Hardware and software failures that affect the computer hosts running server nodes;
2. Performance-related failures that occur when the incoming requests are not served due to limited capacity of the server buffers.

Failures of nodes have a direct impact on the performance capabilities of the web cluster. Indeed, when nodes fail and are disconnected from the cluster, the remaining healthy nodes have to handle all the original traffic, including the requests previously served by the failed nodes. This leads to an increase of the workload redirected to the remaining nodes, with a potential degradation of the corresponding quality of service. For example, a failure of two nodes in a five node cluster increases by 66% the load to be processed by three remaining nodes. Clearly, an accurate estimation of the web service availability should take into account the performance degradation of the web cluster.

The selection of the appropriate architecture supporting the web service requires knowledge about the recovery strategy following a node failure as well as the consequences of node failures. For this purpose, two recovery strategies are compared, namely: Non Client-Transparent (NCT), and Client-Transparent (CT) (see section 3.2 for more details).

From the modeling viewpoint, NCT and CT are defined as follows:

- NCT recovery strategy: all requests in progress as well as the input requests directed to the failed node before the failure is detected are lost;
- CT recovery strategy: all requests in progress and the input requests directed to the failed node during failover latency time [1] are not lost; they are redirected to the non failed nodes.

[1] *The failover latency* corresponds to the *detection and recovery time* i.e., the time taken by the dispatcher to detect a failure and remove the failed node from the list of alive nodes in the cluster.

In the following section, we present analytical models addressing these strategies in a unified approach. For CT recovery strategy, we assume that the architecture is implemented as in [Luo & Yang 2002].

3.4 Cluster Modeling

The evaluation of the web service availability taking into account the modeling assumptions presented in section 3.3 is carried out adopting the composite performance and availability approach (presented in section 1.5 and chapter 2). It is worth to mention that our approach builds on the traditional performability modeling approach by providing closed-form equations for requests loss probability. The requests loss probability can be caused by i) buffer overflow, ii) server failures or, iii) latency of failure detection. The evaluated availability measure allows a distinction among these causes and includes such requests loss probability as a source of web service unavailability.

In the following, we consider a web cluster system composed of N server nodes and a dispatcher balancing the requests among the servers in a round robin way, and capable of detecting the failure of the servers connected to the cluster.

3.4.1 Availability model

Let us assume that the times between node failures are exponentially distributed with rate γ, and that failure detection occurs with rate α. After detection of a node failure, the cluster is reconfigured by disconnecting the failed node. The latter is reintegrated into the cluster after restoration. The restoration times are assumed to be exponentially distributed with rate τ.

Figure 3.2 shows the availability model describing the behavior of the cluster governed by server failures, detection, recovery and restoration processes. In states $k = 0, ..., N$, the system has k available nodes capable of processing the input traffic. However, requests could be rejected in these states due to overload conditions. The failure of a server node in state k, leads the cluster to state D_k with a transition rate $k\gamma$. In states D_k, although the server has failed, this failure is not yet perceived by the dispatcher. Accordingly, client requests could still be directed by the dispatcher to the failed node during the failover latency time. Upon detection, the system moves to state $k-1$ indicating that the number of operational servers has been reduced by one and the restoration of the failed server is initiated. In this model, it is assumed that no other failure can occur when the system is in state D_k. This assumption is acceptable because the failover latency times are generally very small compared to the times to failure.

We have to solve this model to obtain the steady-state probabilities for states k and D_k, denoted as π_k and π_{D_k}, respectively. Processing is straightforward and the analytical expressions for ($k = 1, ..., N$) are given by equations (3.1) and (3.2)

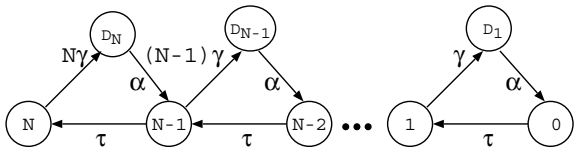


Figure 3.2: Availability model of the web cluster

$$\pi_k = \frac{N!}{k!}\left(\frac{\gamma}{\tau}\right)^{N-k} \pi_N, \qquad k = 0 \cdots N \tag{3.1}$$

$$\pi_{D_k} = \frac{N!}{k!}\left(\frac{\gamma}{\tau}\right)^{N-k} \frac{k\gamma}{\alpha} \pi_N, \quad k = 1 \cdots N \tag{3.2}$$

where

$$\pi_N = \left[\sum_{j=0}^{N} (\frac{\gamma}{\tau})^j \frac{N!}{(N-j)!} + \sum_{j=0}^{N-1} (\frac{\gamma}{\tau})^j \frac{\gamma(N-j)N!}{\alpha(N-j)!} \right]^{-1}$$

3.4.2 Performance model

In order to model the cluster input traffic, we consider two different traffic assumptions. First, the web traffic is characterized by a Poisson process. Then, in the second assumption, the traffic is assumed to be described by a Markov Modulated Poisson Process (MMPP).

3.4.2.1 Poisson Process Traffic

The first assumption, illustrated in Figure 3.3, is that the traffic arriving to the web cluster is modeled as a Poisson process with rate λ req/s (requests per second) and is independent of the failure process. Assuming that there are k available servers in the cluster with the input traffic being distributed among the servers, then this system has k independent Poisson arrival processes each one with rate $\lambda' = \frac{\lambda}{k}$.

Also, each server supports a maximum number of requests b (buffer size) and has a service rate of μ req/s. The requests arriving at the server when the buffer is full are rejected.

The performance behavior of each server can be modeled by an M/M/1/b queueing system[2]. It is important to remark that for this analytical model, each server has its

[2]Since there are no more than b requests in the queueing system, this system is stable for all values of arrival rate λ' and service rate μ (see [Bolch et al. 1998]).

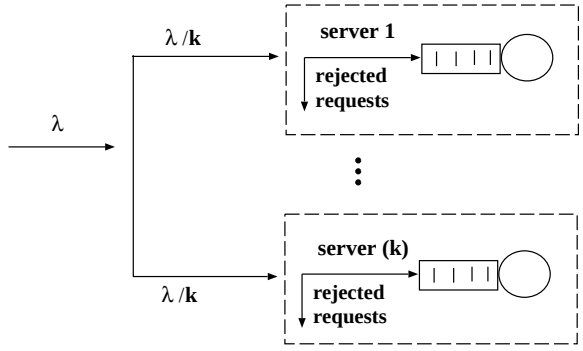


Figure 3.3: A web cluster with k servers available and load balancing

own independent queue, while the model of the section 2.4 of chapter 2, there is only a queue that is served by c servers (M/M/c/b queueing model). Let us denote by p_i the steady state probability of having i requests in this queue[3]. Thus, p_i is calculated as follows (see e.g., [Bolch et al. 1998]):

$$p_i = \begin{cases} \rho^i \frac{1-\rho}{1-\rho^{b+1}} & \text{, if } \lambda' \neq \mu \text{ and } 0 \leq i \leq b \\ \frac{1}{b+1} & \text{, if } \lambda' = \mu \text{ and } 0 \leq i \leq b \end{cases} \tag{3.3}$$

where ρ is the server load given by:

$$\rho = \frac{\lambda}{k\mu} \tag{3.4}$$

3.4.2.2 Markov Modulated Poisson Process Traffic

Based on the published results [Chen et al. 2001, Morris & Lin 2000, Iyengar et al. 1999] previously discussed in section 1.6, it is reasonable to employ an Markov Modulated Poisson Process (MMPP) as described in [Frost & Melamed 1994] to capture the seasonal behavior of the aggregate traffic in the web cluster, while preserving the tractability of modulated Poisson process. Thus, we assume that the request arrival rate is described as a Markov process $M(i)$ with state space $1, 2, ..., i, ..., S$. State i has arrivals with Poisson process at rate λ_i. To follow the seasonal behavior, especially the day/night cyclic behavior of the web cluster load, the traffic is divided into phases on a daily basis. Each phase corresponds to a state in the MMPP process $M(i)$.

[3]Note that the request loss probability due to buffer overflow is given by p_b based on the PASTA (Poisson Arrivals See Time Averages) theorem [Wolff 1982]. Recall that this theorem shows that the probability that a request entering in a queue finds such a queue in a given state is equal to the steady-state probability for this state.

Figure 3.4 shows an example of a MMPP model where the arrival traffic to the web cluster is divided into 5 phases along the day. Average arrival rate per day is also plotted in this figure.

For each phase of the MMPP, the performance model describing the arrival traffic to each server can be modeled by M/M/1/b queue system, where the input traffic is described by the Poisson process associated with the corresponding phase (rate λ_i). Accordingly, the request loss probability will be calculated conditioned on the phases of the MMPP process with its corresponding arrival rate using equation (3.3).

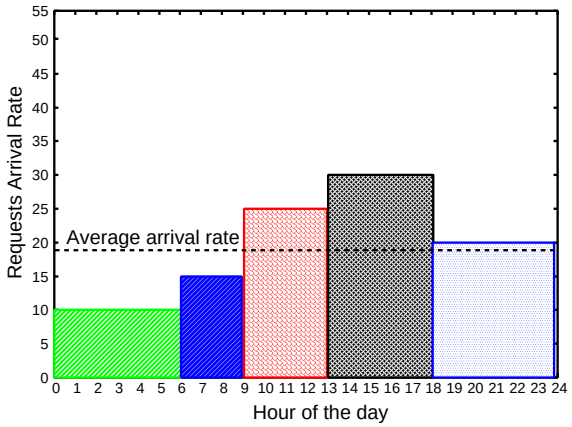


Figure 3.4: Markov Modulated Poisson Process modeling the request arrival process

3.4.3 Composite Availability - Performance model

The web service availability is evaluated based on a composite availability-performance model. Let us denote by UA, the web service unavailability defined as the probability in steady state that a request is not processed successfully. This means that requests can be lost: i) upon arrival to the cluster, or ii) while they are being processed or waiting for service, due to a server failure, or iii) during failure detection.

In order to evaluate UA, we need to compute the request loss probability in states k and D_k of the availability model (Figure 3.2), also during the transition between these states taking into account the traffic model.

For $k = 1, \cdots, N$, let us denote by:

- $L(k)$: the request loss probability due to buffer overflow in state k

- $L(k\gamma)$: the request loss probability caused by a transition from state k to state D_k, due to a server failure
- $L(D_k)$: the request loss probability in state D_k, during the node failure detection time.

For the Poisson traffic model, the web service unavailability UA is computed as follows:

$$UA = \sum_{k=1}^{N} \pi_k \{L(k) + L(k\gamma)\} + \sum_{k=1}^{N} \pi_{D_k} L(D_k) + \pi_0 \tag{3.5}$$

Recall that π_{D_k}, π_k, and π_0 are the steady state probabilities evaluated from the availability model, given by equations (3.1) and (3.2).

In the case of **MMPP traffic model**, it is important to note that the web service availability depends on the phase duration as well as on the arrival rate of each phase.

Thus, let us denote by:

- S: the number of phases;
- Δ_i: the steady-state probability of a given phase i;
- UA_i: the web service unavailability at phase i.

Therefore,

$$UA = \sum_{i=1}^{S} \Delta_i UA_i \tag{3.6}$$

where UA_i is computed by equation (3.5) using the corresponding request arrival rate λ_i of phase i (instead of λ).

The closed analytical equations for $L(k)$, $L(k\gamma)$ and $L(D_k)$, considering both recovery strategies (i.e., NCT and CT) are evaluated in the following sections.

3.4.3.1 Loss probability due to buffer overflow $L(k)$

Given that k servers are available in the cluster, let us assume that all servers have the same loss probability at steady-state. This assumption holds with a system containing equally utilized servers. Thus, when at least one of the available servers has the buffer full, the probability of loss can be computed as follows:

$$L(k) = [1 - (1 - p_b)^k] \tag{3.7}$$

Note that p_b is the request loss probability due to buffer overflow. It is obtained from equation (3.3) with a request arrival rate to each server given by $\lambda' = \frac{\lambda}{k}$. Also, it is worth to mention that $L(k)$ is the same for both recovery strategies.

3.4.3.2 Loss probability due to server node failure $L(k\gamma)$

This loss scenario occurs only with the NCT recovery strategy (see Section 3.4). The request loss probability caused by a transition from state k to state D_k corresponds to the loss of all requests (queued or in service), when a web server node fails. $L(k\gamma)$ is computed by the following closed-form equation (see appendix I for details)

$$L(k\gamma) = \left[\frac{\rho(1-\rho^b)}{1-\rho^{b+1}}\right] - \left[\frac{(1-\rho)v^2(1-v^b)}{\rho(1-\rho^{b+1})(1-v)}\right] \tag{3.8}$$

Recall that a M/M/1/b queue system is stable for all values of load $\rho \neq 1$ (see [Bolch et al. 1998]). Therefore, for $\rho = 1$, $L(k\gamma)$ is computed using

$$L(k\gamma) = \left[\frac{b}{b+1}\right] - \left[\frac{(\frac{\mu}{\mu+k\gamma})^2(1-\frac{\mu}{\mu+k\gamma})^b}{(b+1)(1-\frac{\mu}{\mu+k\gamma})}\right] \tag{3.9}$$

3.4.3.3 Loss probability during the node failure detection time $L(D_k)$

The computation of request loss probability when the system is in state D_k of the availability model, depends on the recovery strategy. In these states, a node has failed but the failure was not yet detected by the dispatcher. Therefore, before the system exits from states D_k, i.e. before detection occurs, the dispatcher continues to send requests to the failed node. For both strategies, we need to evaluate the loss probability caused by the failed server and the loss probability caused by the $k-1$ operational servers in the cluster due to their limited buffer capacity. In state D_k, we need to take into account two competing processes: the request arrival process to a server node with associated rate λ', and the failure detection process with associated rate α.

For **NCT recovery strategy**, all the requests sent to the failed node before the failure is detected are lost. The probability of loss in state D_k is given by the probability that one or more arrivals occur in the failed node before failure detection (system exits from state D_k). When the cluster enters in state D_k, only one of two events can occur: 1) detection, which takes the system to state $k-1$, or 2) an arrival, which increases the number of requests in the queue by one. Since these events are assumed to be independent and exponentially distributed with respective means $1/\lambda'$ and $1/\alpha$, then the probability that an arrival happens before detection is given by $\frac{\lambda'}{\lambda'+\alpha}$. As the probability of loss in state D_k corresponds to the probability that at least one arrival occurs, by little law there are λ'/α arrivals on average before detection, which leads to the following equation $\left(\frac{\lambda'}{\lambda'+\alpha}\right)^{\lambda'/\alpha}$.

In addition, the loss probability in state D_k caused by the $k-1$ operational servers due to their finite buffer is $[1-(1-p_b)^{k-1}]$, following arguments explained earlier in the previous section, where p_b is the loss probability for one operational server.

Therefore $L(D_k)$ is computed by the sum of the loss probability of requests sent to the failed node, and the loss probability caused by buffer overflow of the $k-1$ operational nodes in the cluster:

$$L(D_k) = \left(\frac{\lambda'}{\lambda'+\alpha}\right)^{\lambda'/\alpha} + [1-(1-p_b)^{k-1}] \tag{3.10}$$

For **CT recovery strategy**, the system provides request migration for all user submitted requests, even when a server node fails. Thus, the only loss scenario is due to buffer overflow.

Let us denote by $l(\alpha)$ the probability of loss due to buffer overflow for a *failed node*. This probability can be derived following the approach proposed in [Garg et al. 1999] using probability fundamentals. When the cluster enters in state D_k, let us assume that there were i requests in the queue of size b. The loss probability is equal to the probability that $b-i$ arrivals occur before the system exits from state D_k (i.e., transition α takes place). This conditional probability is given by $(\frac{\lambda'}{\lambda'+\alpha})^{b-i}$. By PASTA theorem, the probability that there are i requests in the queue when a detection occurs is equal to the steady state probability that there are i requests in the queue denoted p_i. Thus

$$\begin{aligned}
l(\alpha) &= \sum_{i=0}^{b} p_i (\frac{\lambda'}{\lambda'+\alpha})^{b-i} \\
l(\alpha) &= \sum_{i=0}^{b} \left[\frac{1-\rho}{1-\rho^{b+1}}\rho^i\right]\left[(\frac{\lambda'}{\lambda'+\alpha})^{b-i}\right] \\
l(\alpha) &= \left[\frac{1-\rho}{1-\rho^{b+1}}\right]\left[\frac{\lambda'}{\lambda'+\alpha}\right]^b \left[\sum_{i=0}^{b} (\rho\frac{\lambda'+\alpha}{\lambda'})^i\right] \\
l(\alpha) &= \left[\frac{1-\rho}{1-\rho^{b+1}}\right]\left[\frac{\lambda'}{\lambda'+\alpha}\right]^b \left[\sum_{i=0}^{b} (\frac{\lambda'+\alpha}{\mu})^i\right]
\end{aligned}$$

Finally, the following closed-form equation is obtained

$$l(\alpha) = \left[\frac{1-\rho}{1-\rho^{b+1}}\right]\left[\frac{\lambda'}{\lambda'+\alpha}\right]^b \left[\frac{1-[\frac{\lambda'+\alpha}{\mu}]^{b+1}}{1-[\frac{\lambda'+\alpha}{\mu}]}\right] \tag{3.11}$$

When $\rho = 1$, $l(\alpha)$ is computed as follows

$$l(\alpha) = \left[\frac{1}{b+1}\right]\left[\frac{\lambda'}{\lambda'+\alpha}\right]^b \left[\frac{1-[\frac{\lambda'+\alpha}{\mu}]^{b+1}}{1-[\frac{\lambda'+\alpha}{\mu}]}\right] \tag{3.12}$$

We need also to include the loss probability in state D_k caused by the $k-1$ operational servers due to their finite buffer. Note that the operational servers do not stop working during the detection time and we assume that the migrated requests do not increase the loss of requests in the operational servers.

Hence, the loss probability for the CT strategy can be computed by equation:

$$L(D_k) = l(\alpha) + [1 - (1 - p_b)^{k-1}] \tag{3.13}$$

3.4.3.4 Summary

The web service unavailability UA **for the Poisson traffic** is computed by equation (3.5). Tables 3.1 and 3.2 summarize the components of equation (3.5) for both recovery strategies. In the case of **MMPP traffic model,** UA is calculated using equation (3.6).

3.5 Evaluation Results

This section presents some results in order to study the sensitivity of the web service availability to different design decisions. For both recovery strategies, we investigate the effect of the cluster size, the impact of the failure detection duration, the reliability of the cluster nodes as well as the overload effects under different traffic models (Poisson and MMPP). We quantify the different design decisions analyzing the performance and availability tradeoffs using the web service performability model.

The parameters used in the model may be obtained either via measurements or based on results published in the literature. In particular, the server failure and recovery rates as well as the mean time between request arrivals may be estimated for example through the analysis of the logs maintained by the web server systems.

We first assume that request arrivals to the web cluster follow an exponential distribution [Willinger & Paxson 1998], where the arrival rate is $\lambda = 20$ req/sec. Table 3.3 summarizes the nominal values used for performability evaluation, where:

- MTTF: mean time to a node failure;
- MTTD: mean time to detect a node failure;
- MTTR: mean time to node restoration.

The numerical value of $MTTR = 200$ sec. represents essentially failures recovered by reboot and restart of the node.

$$UA = \sum_{k=1}^{N} \pi_k \{L(k) + L(k\gamma)\} + \sum_{k=1}^{N} \pi_{D_k} L(D_k) + \pi_0$$

$$\pi_k = \frac{N!}{k!}(\frac{\gamma}{\tau})^{N-k}\pi_N, \qquad k = 0 \cdots N$$

$$\pi_{D_k} = \frac{N!}{k!}\frac{k\gamma}{\alpha}(\frac{\gamma}{\tau})^{N-k}\pi_N, \quad k = 1 \cdots N$$

$$\pi_N = \left[\sum_{j=0}^{N}(\frac{\gamma}{\tau})^j \frac{N!}{(N-j)!} + \sum_{j=0}^{N-1}(\frac{\gamma}{\tau})^j \frac{\gamma(N-j)N!}{\alpha(N-j)!} \right]^{-1}$$

$$\pi_0 = N!(\frac{\gamma}{\tau})^N \pi_N$$

$$L(k) = [1 - (1 - p_b)^k]$$

$$p_b = \begin{cases} \rho^b \dfrac{1-\rho}{1-\rho^{b+1}} & \text{, for } \rho \neq 1 \\ \dfrac{1}{b+1} & \text{, for } \rho = 1 \end{cases}$$

$$L(k\gamma) = \begin{cases} \left[\dfrac{\rho(1-\rho^b)}{1-\rho^{b+1}}\right] - \left[\dfrac{(1-\rho)v^2(1-v^b)}{\rho(1-\rho^{b+1})(1-v)}\right] & \text{, for } \rho \neq 1 \\ \left[\dfrac{b}{b+1}\right] - \left[\dfrac{(\frac{\mu}{\mu+k\gamma})^2[1-(\frac{\mu}{\mu+k\gamma})^b]}{(b+1)(1-\frac{\mu}{\mu+k\gamma})}\right] & \text{, for } \rho = 1 \end{cases}$$

$$L(D_k) = \left(\frac{\lambda'}{\lambda'+\alpha}\right)^{\lambda'/\alpha} + [1 - (1 - p_b)^{k-1}] \ , \quad \lambda' = \lambda/k$$

Table 3.1: Closed-form equations for NCT recovery strategy

$$UA = \sum_{k=1}^{N} \pi_k \{L(k) + L(k\gamma)\} + \sum_{k=1}^{N} \pi_{D_k} L(D_k) + \pi_0$$

$$\pi_k = \frac{N!}{k!}(\frac{\gamma}{\tau})^{N-k}\pi_N, \qquad k = 0 \cdots N$$

$$\pi_{D_k} = \frac{N!}{k!}\frac{k\gamma}{\alpha}(\frac{\gamma}{\tau})^{N-k}\pi_N, \quad k = 1 \cdots N$$

$$\pi_N = \left[\sum_{j=0}^{N} (\frac{\gamma}{\tau})^j \frac{N!}{(N-j)!} + \sum_{j=0}^{N-1} (\frac{\gamma}{\tau})^j \frac{\gamma(N-j)N!}{\alpha(N-j)!} \right]^{-1}$$

$$\pi_0 = N!(\frac{\gamma}{\tau})^N \pi_N$$

$$L(k) = [1 - (1 - p_b)^k]$$

$$p_b = \begin{cases} \rho^b \dfrac{1-\rho}{1-\rho^{b+1}} & \text{, for } \rho \neq 1 \\ \dfrac{1}{b+1} & \text{, for } \rho = 1 \end{cases}$$

$$L(k\gamma) = 0$$

$$L(D_k) = \begin{cases} \left[\frac{1-\rho}{1-\rho^{b+1}}\right]\left[\frac{\lambda'}{\lambda'+\alpha}\right]^b \left[\frac{1-\left[\frac{\lambda'+\alpha}{\mu}\right]^{b+1}}{1-\left[\frac{\lambda'+\alpha}{\mu}\right]}\right] + \left[1-(1-p_b)^{k-1}\right] & \text{, for } \rho \neq 1 \\ \left[\frac{1}{b+1}\right]\left[\frac{\lambda'}{\lambda'+\alpha}\right]^b \left[\frac{1-\left[\frac{\lambda'+\alpha}{\mu}\right]^{b+1}}{1-\left[\frac{\lambda'+\alpha}{\mu}\right]}\right] + \left[1-(1-p_b)^{k-1}\right] & \text{, for } \rho = 1 \end{cases}$$

Table 3.2: Closed-form equations for CT recovery strategy

Service rate	MTTF	MTTD	MTTR	Buffer size
μ	$1/\gamma$	$1/\alpha$	$1/\tau$	b
5 req/sec	10 days	20 sec.	200 sec.	20

Table 3.3: Numerical values of the model parameters

3.5.1 Sensitivity to MTTF

Figure 3.5 shows the web service unavailability UA as a function of the number of servers (cluster size) for two different values of MTTF: 10 days and 4 days. The other values of model parameters are those of Table 3.3. We analyze clusters with $N = 150$ nodes that is large enough for most of cluster architectures [4].

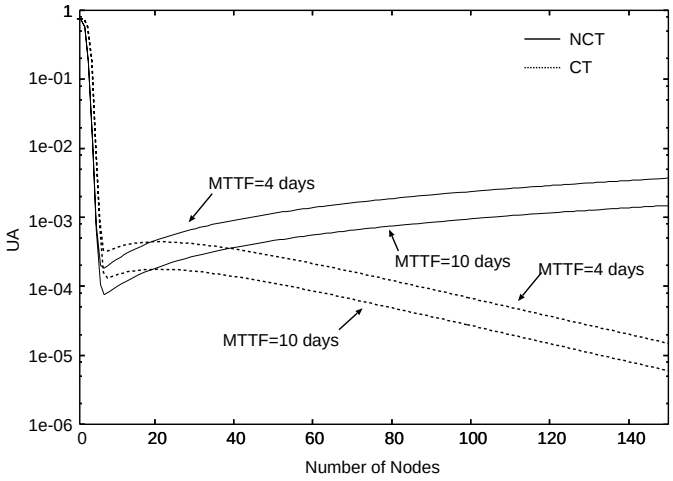


Figure 3.5: Impact of MTTF on UA

For both strategies, increasing the number of nodes N in the cluster from 1 to 8 leads to a significant availability improvement. However, for a higher number of nodes, we observe different trends:

- **For NCT strategy**, the trend is reversed for $N > 8$. This is explained by the fact that the higher is the number of nodes, the higher is the probability of the system being in states D_k, which increases the probability of requests loss during failure detection time. This fact is more effective when the sejourn time in states D_k is long, i.e. for longer MTTD[5].

[4]Although for the sensitivity analysis $N = 150$ is enough to illustrate the main trends of UA, we emphasize that the underlying model allows to evaluate clusters with a larger number of nodes ($N > 150$). Special care has to be taken for computing the factorial for large N in the equations of Tables 3.1 and 3.2. In this case, we can use the *Stirling approximation*: $N! = \sqrt{2\pi}N^{N+1/2}e^{-N}$.

[5]The time taken to detect a failed node relies on the frequency of heartbeat messages sent to the nodes. Clearly, the higher the frequency, the faster failed nodes will be detected.

- **For CT strategy**, UA increases for a cluster with 9 up to 40 nodes. In this case, the loss probability in states D_k is only due to buffer overflow. This loss probability decreases as the load submitted to each node decreases, which explains why UA restarts to decrease substantially for N beyond 40. In fact, another important reason is that the mean time to detect a failure (MTTD) is not fast enough to avoid the buffer overflow at a failure node, notably for $9 \leq N \leq 40$.

3.5.2 Sensitivity to MTTD

Figure 3.6 plots UA as a function of the number of servers. The goal is to analyze the impact of the mean time to detect a failure MTTD, i.e. the mean time spent in states D_k. We consider two different MTTDs: 20 and 2 sec., and two different values of MTTF:10 days and 4 days. The other parameter values are those of Table 3.3.

The two highest curves of Figure 3.6 correspond to those of Figure 3.5. According to these figures, it is clear that there is a notable difference between NCT and CT recovery strategies. In the case of NCT, as the number of nodes increases, UA reaches 10^{-4} for $MTTD = 20$, against $UA \approx 10^{-5}$ for $MTTD = 2$ and $8 \leq N \leq 20$. On the other hand for CT, UA is in the order of 10^{-4} for $MTTD = 20$, and for $MTTD = 2$, UA decreases substantially reaching 10^{-16} as the number of nodes increases to $N = 150$.

This is explained by the fact that when the number of servers increases, the loss probability due to buffer overflow decreases significantly especially for small values of MTTD. However, for NCT, even for small values of MTTD, all the requests that were queued when the server fails as well as those that are directed to this server before failure detection, are lost as well.

3.5.3 Sensitivity to service rate

Figure 3.7 shows the effect of the service rate on UA, considering two values of μ (5 and 10 req/sec). MTTD is set to 2 sec., while the other parameters are set to their nominal values of Table 3.3.

In fact, we note that UA decreases faster for a higher service rate as expected, for both strategies. However, the service rate plays a significant role until a certain threshold. According to the figure, the service rate variation does not have effect on UA in a cluster containing more than 10 nodes for both strategies. In contrast, the service rate reduces substantially UA from $N = 1$ to 5 with $\mu = 10$, while UA has been affected from $N = 1$ to 8 with $\mu = 5$.

In this study, the best evaluated UA, **for NCT**, would be obtained for a cluster with $N = 5$ nodes and $\mu = 10$ or with $N = 8$ nodes and $\mu = 5$. However, **for CT**, UA keeps decreasing as the number of nodes increases.

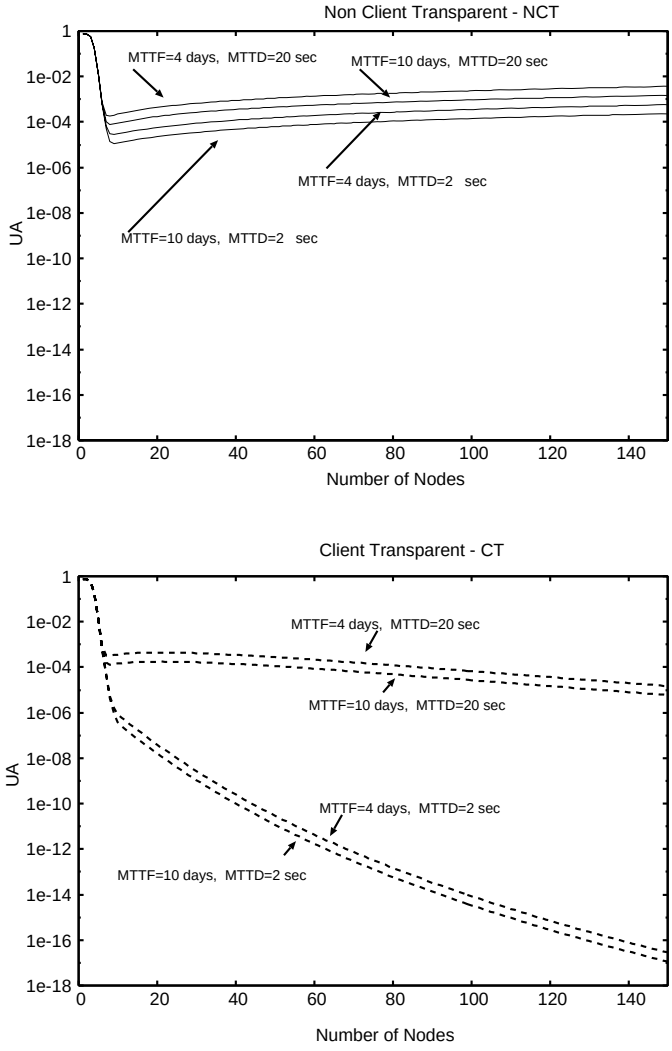


Figure 3.6: Impact of failure detection duration on UA for both recovery strategies

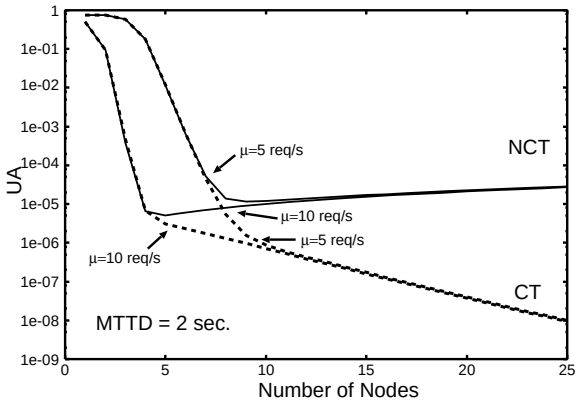


Figure 3.7: Impact of service rate μ on UA

3.5.4 Impact of traffic model

Preliminary analysis of the access logs from the web server of our laboratory suggests that the daily cycles can be divided into five phases during which we assume the arrival process being stationary. Figure 3.8 provides examples of MMPP traffic model, for which the first phase (from 1h to 6h) consists of the lowest period of the day, then the traffic increases (from 6h to 9h) and continues to increase from 9h to 13h. From 13h to 18h, it reaches the high intensity area and then decreases from 18h until 24h.

We have considered three traffic models referred to as MMPP1, MMPP2, MMPP3 illustrated in figure 3.8. The values of the request rates were chosen for comparison purposes with the basic Poisson traffic with average arrival rate of 20 req/sec.

The **traffic burstiness** is an important characteristic extensively used for traffic characterization [Menascé & Almeida 2002]. It can be defined by a pair of descriptors **(θ,β)** .

Let λ be the average arrival rate per day:

$$\lambda = \frac{1}{S}\sum_{i=1}^{S}\lambda_i \tag{3.14}$$

and let Λ be the sum of arrival rates that exceed λ. Accordingly,

$$\Lambda = \sum_{\forall i \cdots \lambda_i > \lambda} \lambda_i \tag{3.15}$$

Descriptor θ is defined as follows:

$$\theta = \frac{\Lambda}{\lambda} \tag{3.16}$$

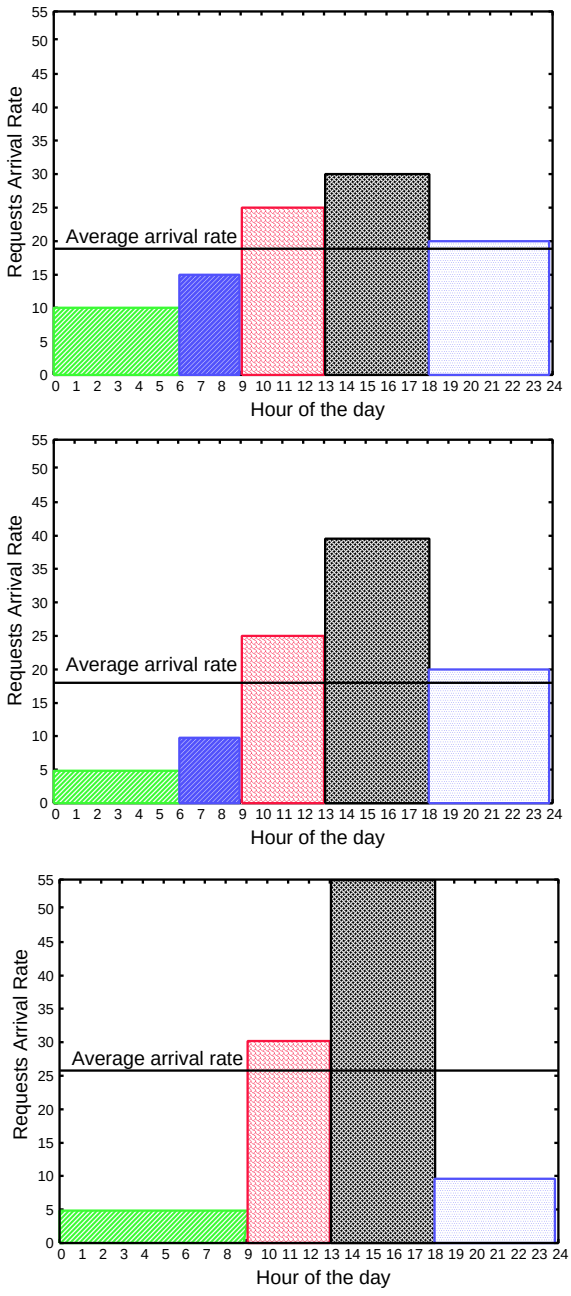


Figure 3.8: MMPP traffic models representing the traffic distribution along the day

β is the fraction of time during which the arrival rate exceeds the average rate.

Table 3.4 computes the traffic burstiness for MMPP models of figure 3.8. MMPP1 has the lowest burstiness value while MMPP3 has the highest one. If traffic is not bursty, which is the case of Poisson traffic model, then $\theta = \beta = 0$.

Day time	MMPP 1	MMPP 2	MMPP 3
0h-6h	10	5	5
6h-9h	15	10	5
9h-13h	25	25	30
13h-18h	30	40	55
18h-24h	20	20	10
Average	19.78	18.32	25.59
(θ,β)	(6.31,0.6)	(7.73,0.6)	(10.62,0.4)

Table 3.4: The MMPP models and traffic burstiness

3.5.4.1 Sensitivity to traffic burstiness

Figure 3.9 shows UA as a function of the number of servers, for the three different MMPP models using both recovery strategies. The parameter values used for this evaluation are given in Table 3.3 where MTTD is set to 2 sec.

Clearly, UA is directly affected by traffic burstiness for both recovery strategies. We can note that the higher is the traffic burstiness, the slower UA decreases. That is, service availability can be degraded due to high variability or "burstiness" on web traffic. In particular, the effect of descriptor θ on UA is considerably more dominant than β. For example, taking $N = 15$ in NCT and CT cases, we can see that UA is higher for MMPP3 than for the other traffic models, since this traffic model is characterized by the highest θ, even if β is lower than for MMPP1 and MMPP2.

Comparing NCT to CT, assuming that the cluster contains at least 20 nodes, all traffic models lead to the same UA for NCT, while in the case of CT, the traffic models still have a relevant impact on UA. Moreover, beyond of $N = 20$ nodes, UA is lower than 10^{-5} for all models in the case of CT, while for NCT, UA is in the order of 10^{-4}.

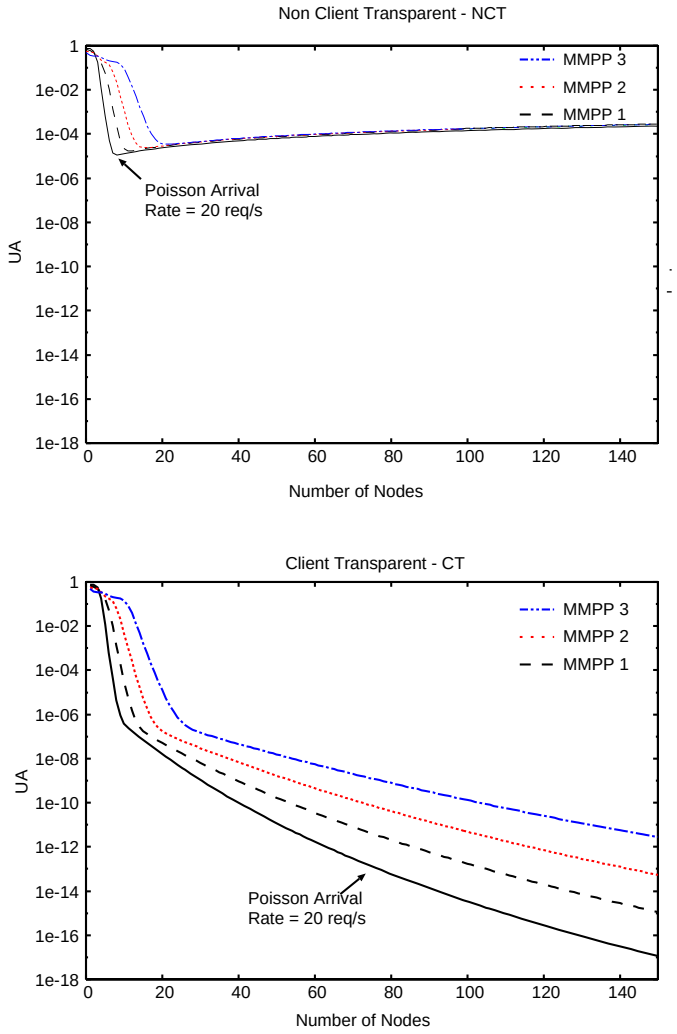


Figure 3.9: Effects of the traffic burstiness on UA for both recovery strategies

3.5.4.2 Load effects on UA

Figure 3.10 shows the results of UA variation with respect to the service load with a cluster containing $10, 20, 40$ and 80 nodes considering both recovery strategies with traditional Poisson arrival traffic compared to MMPP2 traffic model. In this figure, ρ denotes the **initial load** where all the nodes are available $k = N$ ($\rho = \frac{\lambda}{N\mu}$). The obtained results provide additional insights about the load effects on UA. MTTD is set to 2 seconds.

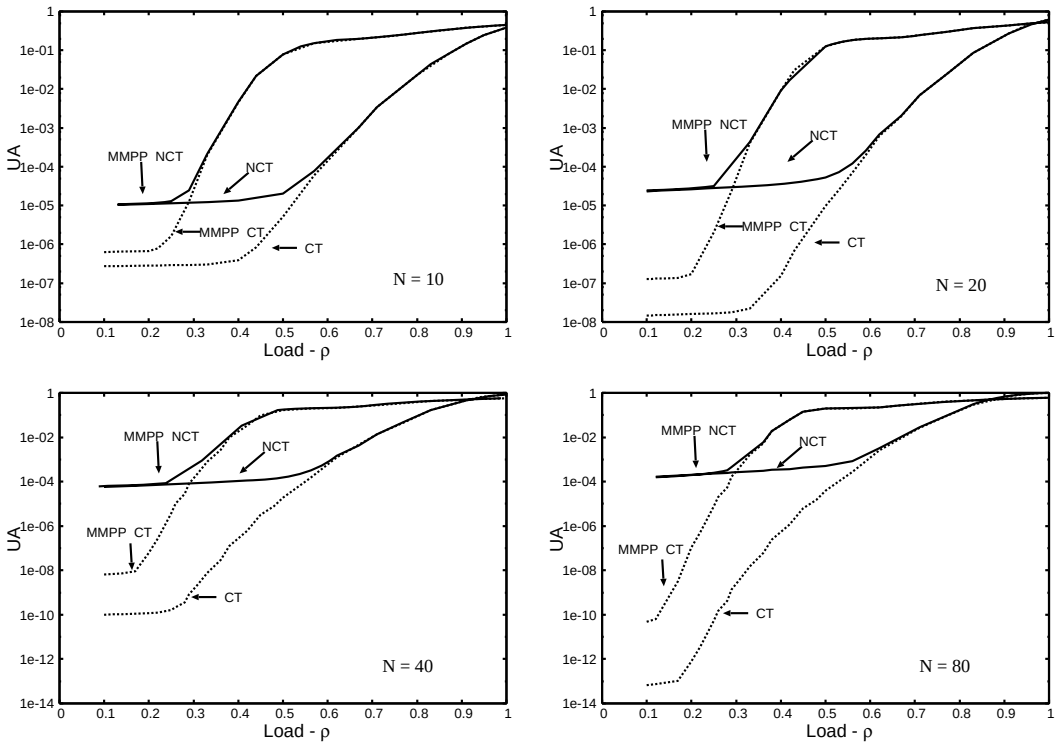


Figure 3.10: Effects of service load ρ on UA

Considering **NCT**, a first observation is that UA increases as the number of node increases for both traffic models subjected to light load ($\rho < 0.3$). For instance, for N=10, UA is in the order of 10^{-5} against $UA = 10^{-4}$ for N=80. Recall that the greater is the number of nodes (N), the higher is the probability of the system being in states D_k, which increases the loss of requests during failure detection latency. Also, it is

important to note that the difference between MMPP and Poisson becomes significant for heavier load[6].

However, for **CT**, MMPP and Poisson may differ even for light loads ($\rho < 0.3$). In particular, it can be noticed that the difference between the traffic models becomes significant as the number of nodes increases. For instance, when $N = 20$, UA is in the order of 10^{-8} for CT against 10^{-7} for MMPP-CT, and when $N = 40$, UA is in the order of 10^{-10} for CT against 10^{-8} for MMPP-CT.

The results presented in Figure 3.10 can be analysed from another perspective concerning the impact of the recovery strategies on UA. For the MMPP model, significant differences between NCT and CT results are observed only for light load values ($\rho < 0.3$). In the case of Poisson model, there is an impact of the recovery strategies on UA for higher load values ($\rho < 0.65$). We should emphasize the fact that UA takes into account the service unavailabity due to requests loss that is very sensitive to system overloads and traffic burstiness.

3.6 Conclusion

In this chapter, we presented performability models for analyzing the availability of web based services implemented in cluster architectures. In particular, we provided closed form equations for web cluster availability taking into account server failures and loss of requests due to system overload. The developed models consider explicitly the cluster recovery support for requests migration and the arrival traffic characteristics.

Sensitivity analyses with respect to cluster characteristics have shown the impact on cluster availability of i) the node failure rate, ii) the failure detection rate, iii) the service rate, iv) the traffic burstiness and v) the load effects. These analyses provide useful guidelines for the design of web-based services, since they can be used for dimensioning the web architecture and making the right tradeoffs for achieving high availability with acceptable performance levels. Also, such results allowed us to understand the impact of the arrival traffic burstiness and the load effects on the service availability supported by web clusters.

For instance, supposing that the traffic follows a Poisson process, we have found that when the number of servers in the cluster is lower than a given threshold (in this study lower than 8), the server processing capacity (service rate plus buffer size) have the highest impact on service availability. In this case, the availability related to the two recovery strategies (i.e., traditional non client transparent strategy and client transparent strategy) is of the same order of magnitude. When the number of servers is higher than this threshold, a significant difference is observed between the two recovery strategies.

[6]According to the model, recall that $\rho = 1$ does not imply that $UA = 1$.

The results confirm that the client transparent strategy is better from an availability point of view quantifying this impact. However, to develop a client transparent strategy requires more complex and expensive recovery mechanisms (that allow on-going and arriving requests to be migrated to other servers). Therefore, a tradeoff between the cost of the recovery strategy and its impact on service availability is necessary. Nevertheless, it is worth mentioning that the two recovery strategies are extreme cases. Indeed even in the client transparent strategy some requests may be lost due to imperfect coverage. Considering a coverage factor lower than 100% (i.e. not all the requests are migrated correctly to the non-failed servers), will certainly reduce the difference between the two strategies.

The sensitivity analysis results allowed to understand the impact of the traffic model on the web service availability. In particular, the higher the traffic burstiness, the slower unavailability decreases as the nodes increase. We have shown that Poisson and MMPP traffic models provide similar results only for light load (less than 0.3 in this study) in the non-client transparent strategy. On the other hand, for heavier load, Poisson traffic model tends to overestimate the web service availability. As expected, this result suggests that Poisson traffic model leads to a more optimistic availability than MMPP traffic models.

In our evaluation study, the support for requests migration has been efficient only for light load (load being less than 0.3 for MMPP and 0.65 for Poisson). For clusters relatively overloaded, the web traffic burstiness has been a more relevant cause of unavailability affecting significantly the web service availability supported by web clusters. We should emphasize the fact that UA takes into account the service unavailability due to requests loss that is very sensitive to system overloads and traffic burstiness.

Finally in this study, the detection rate was assumed to be constant independently of the number of nodes. However, in practice this rate could be variable as a function of the cluster size. Also, to simplify cluster modeling, we assumed that the dispatcher does not fail. However, the proposed model can be easily extended to include dispatcher failure and recovery in the analysis.

Chapter 4

Service unavailability due to long response time

> Don't say you don't have enough time. You have exactly the same number of hours per day that were given to Pasteur, Michaelangelo, Euler, Leonardo da Vinci, ...
>
> H. Jackson Brown

FROM the user point of view, the service is perceived as degraded or even unavailable if the response time is too long compared to what he or she is expecting. A long response time may discourage some users who will visit other service providers. For instance, if a request takes 30 seconds to complete, web users may consider the request failed.

This chapter is devoted to the evaluation of service unavailability due to long response time. We concentrate on the service and resource levels defined in our modeling framework (see Figure 2.2) presented in chapter 2.

The goal of this chapter is to provide a simple analytic modeling approach for computing service unavailability due to long response time relying on Markov reward models and queueing theory. To the best of our knowledge, the quantitative modeling and evaluation of the impact of long response time on web service unavailability has been seldom addressed in previous research. In order to provide a practical analytic approach, we introduce a flexible mathematical abstraction that is general enough to capture the essence of unavailability behavior due to long response time. We analyze this measure at steady state, starting with a single-server queueing system, then multi-server queueing systems are considered.

A long response time may be due to long delay in communication or to an overloaded server. We do not distinguish between these two causes in this chapter. The aim of our work is twofold:

- Evaluate the effects of long response times on service unavailability.
- Provide practical quantitative results that can help web server designers in design decisions.

Although the derivation of the response-time distribution is widely recognized to be not trivial [Trivedi et al. 2003], some interesting approaches have been proposed to define measures combining performance and dependability issues [Shin & Krishna 1986, Muppala et al. 1991, Mainkar 1997]. [Shin & Krishna 1986, Muppala et al. 1991] introduced a model for hard and soft real-time systems, while [Mainkar 1997] has considered transactional systems in which failures may be due to frequent violation of response time constraints.

The proposed modeling approach builds and extends the work introduced in [Mainkar 1997]. The latter uses i) a Markov model to evaluate *system availability*, and ii) a tagged job approach to compute the response time distribution. We take a step further by providing closed-form equations for response-time distribution and for service unavailability due to long response time in single and multi-server systems. The closed-form equations are derived using the well-known gamma function. We present several sensitivity analyses to illustrate how the designers can use the models to guide the system design.

The rest of the chapter is organized as follows. Section 4.1 defines the availability measure based on response time. In section 4.2, we introduce the modeling approach using single server queueing systems. This is followed by sensitivity analysis results illustrating the measure behavior. Section 4.3 provides a modeling approach using multi-server queueing systems with sensitivity analysis. Section 4.4 concludes the chapter.

4.1 Availability measure definition

In the context of this chapter, steady *service availability* is defined as the long-term fraction of time of service delivery within an acceptable delay. We assume that the service states as perceived by the users are partitioned into two sets: i) a set of states in which the service is perceived as available and ii) the complementary set in which the service is perceived as unavailable.

Let

- Ω: denote the set of all service states;
- p_i: be the probability that the service is in state i at steady-state;

- r_i: be a reward rate associated to the state i, defined as follows

$$r_i = \begin{cases} 1 & \text{, if the service is available} \\ 0 & \text{, otherwise} \end{cases}$$

Thus, the service availability *A* is given by

$$A = \sum_{i \in \Omega} r_i p_i \Leftrightarrow UA = 1 - \sum_{i \in \Omega} r_i p_i \tag{4.1}$$

UA denotes the service unavailability.

In order to define the service availability based on the response time, let us introduce

- $R(i)$: the random variable denoting the response time given that the system is in i at steady-state;
- d: the maximum acceptable response time (i.e., if the response time is longer, the service is considered as unavailable), it is also referred to as the maximum response time requirement;
- ϕ: the quality of service requirement (or the accepted quality of service) representing the minimum fraction of requests that satisfy the maximum response time requirement;
- $P[R(i) \leq d]$: the conditional response-time distribution (i.e., the probability that the response time of a request is lower than or equal to d, given that the system is in state i at steady-state).

Using the definitions above, the service is said to be available if the following condition is satisfied

$$P[R(i) \leq d] > \phi \tag{4.2}$$

Let us denote by K the states in which the service is available (i.e., equation (4.2) is satisfied for all states i, $i = 0$ to K). Then, the reward rate r_i is defined as follows

$$r_i = \begin{cases} 1 & \text{, if } P[R(i) \leq d] > \phi \text{ (i.e., for } i = 0 \text{ to } K) \\ 0 & \text{, otherwise (i.e., for } i > K) \end{cases} \tag{4.3}$$

With this notation, equation (4.1) becomes

$$A = \sum_{i=0}^{K} p_i \Leftrightarrow UA = 1 - \sum_{i=0}^{K} p_i \tag{4.4}$$

To summarize, the evaluation of the unavailability measure based on response time is carried out according to the following steps. First, one needs to specify the service model describing in particular the distribution of request arrivals and processing times as well as the servers capacity. Based on this specification, $P[R(i) \leq d]$ and p_i can be obtained for a given d in two steps as follows:

- For a given ϕ, using $P[R(i) \leq d] > \phi$, K is derived;
- Then the availability is computed by equation (4.4).

It is worth to mention that the two parameters, d and ϕ characterize the quality of service and should be specified a priori. For example, one can specify ϕ to be equal to $\phi = 0.9$ and $d = 5$ seconds. This means that the response time of the web-based service should be less than 5 seconds for at least 90% of all requests.

In the following sections, relying on queuing theory, we i) build analytical models for single-server systems and for multi-server systems, and ii) derive closed-form equations for the conditional response time probability and service unavailability.

4.2 Single server queueing systems

In this section, we assume that the web server is modeled as a single queueing system with exponential arrival and service times. The modeling approach using single server queueing systems is carried out in two steps. First, we define the availability measure itself based on the response time distribution at steady-state and then some numerical sensitivity analysis results are presented.

4.2.1 Modeling unavailability due to long response time

4.2.1.1 Conditional response time distribution

In this simplest example, we assume that there is only a single process serving the incoming requests at a constant rate μ requests/sec. The system is assumed to be in state i when there are i requests in the system ($i-1$ waiting for service in the queue and one being served). By definition, if a request arrives given that there are already i requests in the system, then the total time spent in the system by the request, denoted as $R(i)$, is given by a sum of $i+1$ random variables. Since the random variables are independent and identically distributed with mean $1/\mu$, it can be shown that $R(i)$ is described by an Erlang distribution [Kleinrock 1975] as follows

$$P[R(i) \leq d] = 1 - \sum_{j=0}^{i} \frac{(\mu d)^j}{j!} e^{-\mu d} \tag{4.5}$$

Let us consider the incomplete gamma function[1] defined as

$$\Gamma(i+1, \mu d) = \int_{-\mu d}^{\infty} e^{-t} t^i dt$$

Using the fact that

$$\left[1 + z + \frac{z^2}{2!} + \cdots + \frac{z^j}{j!}\right] e^{-z} = \frac{\Gamma(j+1, z)}{\Gamma(j+1)}$$

$P[R(i) \leq d]$ can be expressed as follows

$$P[R(i) \leq d] = 1 - \sum_{j=0}^{i} \frac{(\mu d)^j}{j!} e^{-\mu d} = 1 - \frac{\Gamma(i+1, \mu d)}{\Gamma(i+1)} \tag{4.6}$$

4.2.1.2 Service availability modeling

Consider a web-based system accessible to a very large population. The arrival process is characterized by requests arriving at an average arrival rate of λ requests/sec.

We first assume that all requests arriving are queued for service. This assumption is known as infinite buffer. Then we will consider the finite buffer case. All the analyses presented assume that the system being analyzed is in operational equilibrium.

Infinite buffer

Requests arrive at the web based system at a rate of λ requests/sec, queue for service, get served at rate μ requests/sec and depart. Such a system is a traditional M/M/1 queue system [Kleinrock 1975], in which the probability (p_i) that there are i requests at steady-state is well-known

$$p_i = (1 - \rho)\rho^i \tag{4.7}$$

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

Therefore, equation (4.4) becomes $A = \sum_{i=0}^{K} (1 - \rho)\rho^i$.

Using the fact that $\sum_{i=0}^{K} \rho^i = \frac{1 - \rho^{K+1}}{1 - \rho}$, we obtain

$$A = 1 - \rho^{K+1} \Leftrightarrow UA = \rho^{K+1} \tag{4.8}$$

[1]Note that $\Gamma(i+1)$ is defined by the following integral $\Gamma(i+1) = \int_0^{\infty} e^{-t} t^i dt$. If i is a positive integer, then $\Gamma(i+1) = i!$. It is also important to note that i is not a complex number.

Finite buffer

For a system supporting at most b requests including the request being processed (finite buffer) denoted by M/M/1/b queue system, we have

$$A = \frac{1-\rho^{K+1}}{1-\rho^{b+1}} \Leftrightarrow UA = 1 - \left[\frac{1-\rho^{K+1}}{1-\rho^{b+1}}\right] \tag{4.9}$$

where $K < b$.

Table 4.1 summarizes the equations for service unavailability due to long response time in single server queueing systems. Recall that the computation of UA requires to calculate K which corresponds to the maximum value of i satisfying equation (4.2).

Conditional response time probability

$$P[R(i) \leq d] = 1 - \sum_{j=0}^{i} \frac{(\mu d)^j}{j!} e^{-\mu d} = 1 - \frac{\Gamma(i+1, \mu d)}{\Gamma(i+1)}$$

Unavailability due to long response time for an M/M/1 queue system

$$UA = \rho^{K+1}$$

Unavailability due to long response time for an M/M/1/b queue system

$$UA = 1 - \left[\frac{1-\rho^{K+1}}{1-\rho^{b+1}}\right]$$

Table 4.1: Closed-form equations for single server queueing systems

4.2.2 Sensitivity analysis

In this section, some numerical results are presented in order to illustrate the behavior of UA using the equations derived in Table 4.1.

4.2.2.1 Variation of response time

An important consideration for design purposes is to define when the service is "too slow". In fact, one has to specify the threshold d for the acceptable response time. Practical experiences have suggested that *ten seconds* is well above the normal response time for all the sites studied in [Merzbacher & Patterson 2002]. The latter

divides timing problems affecting sites availability into "medium" (ten seconds) and "severe" (thirty seconds) problems.

Figure 4.1 shows the conditional response time distribution ($P[R(i) \leq d]$), given in Table 4.1, as a function of the number of requests, considering different values for μd. In fact, the evaluation of this distribution allows us to determine the K states for which ($P[R(i) \leq d] > \phi$) according to equation (4.3).

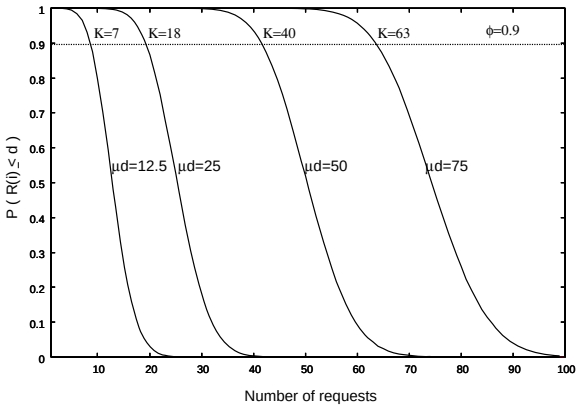


Figure 4.1: $P[R(i) \leq d]$ variation for single server queueing system

As it can be seen, the response time probability is directly affected by the product μd. It is noteworthy that μd corresponds to the average number of requests processed by the server during a period of time d. Another observation is that K increases with μd. For example, setting the quality of service parameter ϕ to 0.9, $K = 7$ for $\mu d = 12.5$ and $K = 63$ for $\mu d = 75$. In fact, the greater is K, the higher is the probability that the response time is lower than d. Figure 4.1 also shows that lower values for the quality of service parameter ϕ (e.g., $\phi = 0.8$) clearly lead to greater values for K. In other words[2], the greater is K, the more requests arriving at the web server are likely satisfied within the acceptable response time.

Such analyses are useful for design decisions, since the expected level of degradation of response time probability as a function of the number of requests queued in the system can be evaluated.

Once K is evaluated, one can compute the service unavailability using equation (4.8). Table 4.2 shows how UA varies as μd increases for different loads ρ and for $\phi = 0.9$. We clearly observe that UA decays faster as ρ decreases.

[2]It can be noticed that for a given ϕ, the longer is the response time requirement, the greater is K.

μd	K	$\rho = 0.9$	$\rho = 0.8$	$\rho = 0.7$	$\rho = 0.6$
12.5	7	4.3e-01	1.6e-01	5.7e-02	1.6e-02
25	18	1.3e-01	1.4e-02	1.1e-03	6.0e-05
50	40	1.3e-02	1.0e-04	4.4e-07	8.0e-10
75	63	1.2e-03	6.2e-07	1.2e-10	6.3e-15

Table 4.2: Effects on UA as μd increases for $\phi = 0.9$

4.2.2.2 Effects of K and ρ on UA

UA provides a useful indicator to analyze the impact of response time on service unavailability. By definition, parameter K represents the set of states for which the response time is acceptable for a given quality of service requirement. Figure 4.2 shows UA as a function of K for different loads ρ. The system is assumed to be composed by one server with infinite buffer (M/M/1). Note that by definition, $\rho = 1$ implies $UA = 1$ and $\lim_{K \to \infty} UA = 0$.

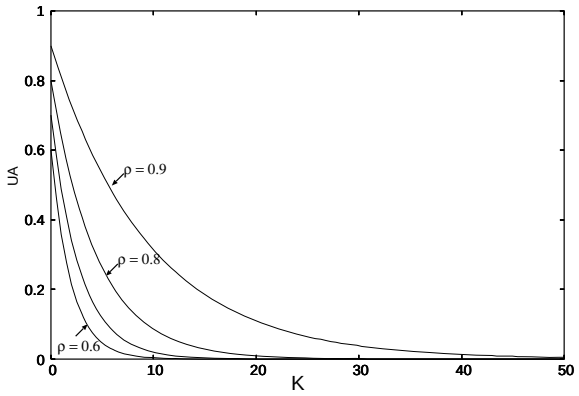


Figure 4.2: UA for an M/M/1 queue system model as a function of K

From the figure, we can see that UA is very sensitive to the load ρ. UA decays slowly for heavy loads ρ. In contrast, for "light" loads $\rho < 0.6$, the unavailability due to long response time is negligible. On the other hand, the greater is K, the lower is UA. It is better illustrated on Figure 4.3 that plots UA as a function of the load ($\rho > 0.6$) for different values of K. In particular, for systems in which $K >> 50$, there is a small probability that the service is perceived as unavailable due to long response time. Thus, according to the figure, it is likely that service unavailability due to long response time will be very low.

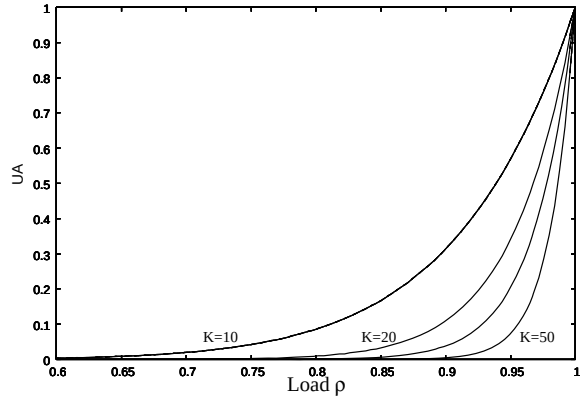


Figure 4.3: UA for an M/M/1 queue system model as a function of ρ

4.2.2.3 Finite buffer effects on UA

All the evaluations presented along the previous section can also be applied to a system with a finite buffer (i.e., considering an M/M/1/b queue).

Figure 4.4 shows a comparison between a system with a finite buffer (M/M/1/b) and a system with an infinite buffer (M/M/1). The results for M/M/1/b (dotted lines) are obtained using equation (4.9). For this example, it can be seen that for $\rho < 0.7$, there is no difference between M/M/1 and M/M/1/b for $b = 20$ and $b = 40$. For higher loads (e.g., $\rho = 0.9$), the difference becomes more visible especially for $b = 20$.

For a given request, the greater is its position in the buffer, the lower is the probability that it is served within the maximum response time requirement. In finite queueing systems, the requests arriving when the buffer capacity b is full are rejected, and therefore, they are not considered in service unavailability due to long response time. This explains the fact that UA for M/M/1/b is lower than UA for M/M/1.

4.2.2.4 Approximation for UA

The evaluation of UA requires to compute the parameter K which represents the set of states after which all arriving requests probably perceive the service as unavailable. K is not known a priori. We have investigated a more direct approach for computing UA based on an approximation for K. The goal is to obtain an analytical equation of UA as a function of only well-known parameters, such as service rate (μ) and maximum response time requirement (d), without needing to compute K in an intermediate step based on the conditional response time distribution.

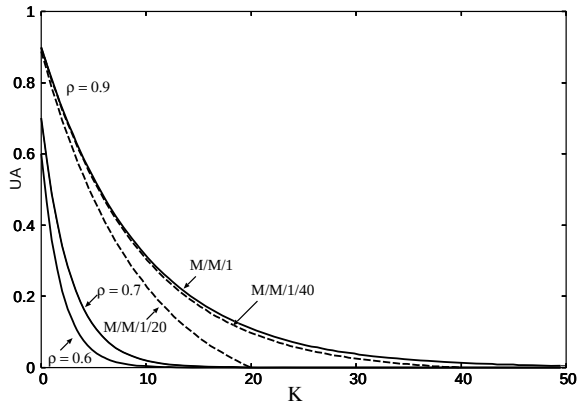


Figure 4.4: The effect of finite buffer size b on UA

By analyzing the properties of the finite series $f(n) = \sum_{j=0}^{n} \frac{(\mu d)^j}{j!}$, we found the following approximation[3] for K

$$K \approx \left\lceil \mu d - \alpha\sqrt{\mu d} \right\rceil \tag{4.10}$$

where α is a constant that can be set to support a given quality of service (e.g., $\alpha = 1.35$ for $\phi = 0.9$).

Accordingly, UA can be obtained as follows

$$UA \approx \rho^{(\lceil \mu d - \alpha\sqrt{\mu d} \rceil)+1} \tag{4.11}$$

Thus, UA can be evaluated directly as a function of μ and d only, for a given ϕ.

Figure 4.5 shows a comparison between the unavailability computed by equation (4.8) (where K is an integer value obtained from equation (4.6)) and the approximation given by equation (4.11) ($\alpha = 1.35$). UA is plotted as a function of μd, where dotted lines represent the approximation. As it can be seen, the approximation is very accurate. It differs from the exact value only for $\mu d < 5$. In other words, for any $\mu d > 5$ (which is the case of most web based systems capable of handling much more than 5 requests per second), there is no difference between the exact and the approximated value of UA.

Equation (4.11) and the results of figure 4.5 show that UA approaches 0 as μd approaches infinity, (i.e., $\lim_{\mu d \to \infty} UA = 0$). It means that for a given $d > 0$ with a server having infinite service rate $\mu = \infty$, there is no service unavailability due to long response time. At the same time, for a given service rate $\mu > 0$, there is no service unavailability for a user with an infinite patience $d = \infty$.

[3] In fact, K is obtained through a sub-linear approximation to the inflexion point of f(n) around μd.

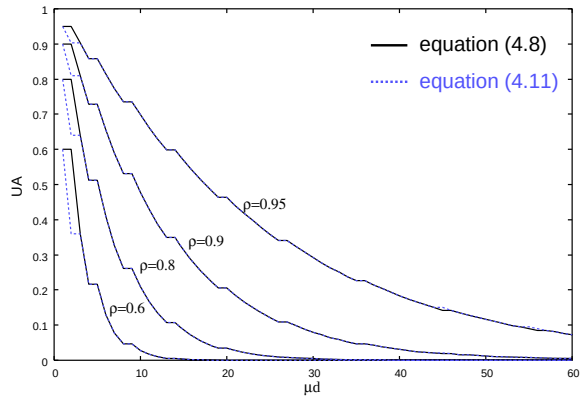


Figure 4.5: UA as a function of μd using equation (4.8) and equation (4.11)

4.3 Multi-server queueing systems

Let us consider a multi-server queueing system consisting of a queueing buffer of finite or infinite size, with multiple identical servers. Such an elementary queueing system is also referred to as a multi-server system. The modeling approach is carried out in two parts. First, a conditional response time probability is obtained providing a closed-form equation. Then, we extend UA to multi servers.

4.3.1 Modeling unavailability due to long response time

4.3.1.1 Conditional response time distribution

Let us suppose that the web-based system is composed of c identical servers, where each server is capable of handling μ requests/sec. Let $R_c(i)$ be the random variable denoting the response time in steady-state of an arriving request at a system with c servers and i requests. If a request arrives when there are already i requests in the system, two different cases can be distinguished to model the corresponding conditional response time distribution:

- If $i < c$, the new arrival can be processed immediately by one of the free servers. Thus, the response time $R_c(i)$ is an exponential random variable with parameter μ.
- If $i \geq c$, the new arrival must wait for $i - c + 1$ service completions before receiving service (If $i = c$, the new request must wait for one service completion. If $i = c+1$, two service completions are required, etc.). In this case, the response time distribution $R_c(i)$ is the sum of an Erlang random variable X corresponding to the request waiting time and an exponential random variable Y denoting the

service time. Therefore, by convolution
$P[R_c(i) = X + Y \leq d] = \int_0^d F(d-y)g(y)dy$, where
$F(x, i-c+1) = 1 - \sum_{j=0}^{i-c} \frac{(\mu c x)^j}{j!} e^{-\mu c x}$ and $g(y) = \mu e^{-\mu y}$.

Accordingly, we have

$$P[R_c(i) \leq d] = \begin{cases} 1 - e^{-\mu d} & \text{, if } i < c \\ \int_0^d \left[1 - \sum_{j=0}^{i-c} \frac{(\mu c(d-y))^j}{j!} e^{-\mu c(d-y)}\right] \mu e^{-\mu y} dy & \text{, otherwise} \end{cases}$$

After a set of transformations (see appendix II for all details), we obtain equation (4.12)

$$P[R_c(i) \leq d] = \begin{cases} 1 - e^{-\mu d} & \text{, if } i < c \\ 1 - (\frac{c}{c-1})^{i-c+1} e^{-\mu d} \left[1 - \frac{\Gamma(i-c+1,(c-1)\mu d)}{\Gamma(i-c+1)}\right] - \frac{\Gamma(i-c+1,c\mu d)}{\Gamma(i-c+1)} & \text{, if } i \geq c \end{cases} \quad (4.12)$$

4.3.1.2 Service availability modeling

Let us take the same web based system consisting of c identical servers, where each server is capable of handling μ requests/sec. We need to compute the probability that the system with c servers has i requests at steady-state denoted $p_i(c)$. Assuming that the sequence of interarrival times is described by a set of independent and identical exponential random variables of rate λ (a traditional M/M/c in which $p_i(c)$ is well-known [Kleinrock 1975] with $\rho = \frac{\lambda}{c\mu}$), we obtain for service unavailability the following closed-form equation (see appendix II for all details)

$$UA = \begin{cases} 1 - p_0 \frac{\Gamma(K+1, c\rho)e^{c\rho}}{\Gamma(K+1)} & \text{, if } K \leq c-1 \\ 1 - p_0 \left[\frac{\Gamma(c, c\rho)e^{c\rho}}{\Gamma(c)} + \frac{(c\rho)^c}{c!} \frac{(1-\rho^{K-c+1})}{1-\rho}\right] & \text{, if } K \geq c \end{cases} \quad (4.13)$$

where p_0 denotes the probability that there is no request in the system.

To summarize, for $i = 0, 1, 2, ...$ requests, $P[R_c(i) \leq d]$ can be computed using equation (4.12). Based on this distribution, K is obtained as the maximum value of i satisfying $P[R_c(i) \leq d] > \phi$, according to equation (4.3). Then, we can proceed to calculate UA for multi-servers in an infinite and finite queueing systems using equation (4.13). Table 4.3 summarizes the equations for service unavailability due to long response time in multi-servers queueing systems.

Conditional response time probability

$$P[R_c(i) \leq d] = \begin{cases} 1 - e^{-\mu d} & \text{, if } i < c \\ 1 - (\frac{c}{c-1})^{i-c+1} e^{-\mu d} \left[1 - \frac{\Gamma(i-c+1,(c-1)\mu d)}{\Gamma(i-c+1)}\right] - \frac{\Gamma(i-c+1,c\mu d)}{\Gamma(i-c+1)} & \text{, if } i \geq c \end{cases}$$

Unavailability due to long response time for multi servers queueing system

$$UA = \begin{cases} 1 - p_0 \dfrac{\Gamma(K+1, c\rho)e^{c\rho}}{\Gamma(K+1)} & \text{, if } K \leq c-1 \\ 1 - p_0 \left[\dfrac{\Gamma(c, c\rho)e^{c\rho}}{\Gamma(c)} + \dfrac{(c\rho)^c}{c!}\dfrac{(1-\rho^{K-c+1})}{1-\rho}\right] & \text{, if } K \geq c \end{cases}$$

Infinite buffer

$$p_0 = \left[\sum_{n=0}^{c-1} \frac{(c\rho)^n}{n!} + (\frac{1}{1-\rho})(\frac{(c\rho)^c}{c!}) \right]^{-1}$$

Finite buffer

$$p_0 = \left[\sum_{n=0}^{c} \frac{(c\rho)^n}{n!} + \frac{(c\rho)^c}{c!} \sum_{n=1}^{b-c} \rho^n \right]^{-1}$$

Table 4.3: Closed-form equations for multi-servers queueing systems

4.3.2 Sensitivity analysis

In this section, we study the effect of the number of servers on UA using the modeling approach developed in the previous section. The analysis is divided in two parts. First, in 4.3.2.1 the response time distribution is studied in order to quantify how the response time is affected by the number of servers. A simple example of system is presented illustrating some possible configurations. Then, we evaluate UA itself taking into account the response time variation, the load effects in 4.3.2.2, the impact of aggregated service rate in 4.3.2.3 and the number of servers in 4.3.2.4.

4.3.2.1 Variation of response time distribution

Figure 4.6 shows the response time distribution ($P[R_c(i) \leq d]$) as a function of the number of requests computed by equation (4.12). This function is evaluated varying the number of servers c and the product μd. As it can be seen, as c or μd increases the response time probability is improved. This is illustrated by the increase of K. Clearly, the greater is K, the lower is UA. Such a trend is expected and it shows that the effect of K on UA is similar for multi-servers (it was discussed in section 4.2.2.2 for single-server).

These results can be used for supporting design decisions. For instance, let us define by μc the aggregated service rate provided by c servers. Assume that a system is designed to support an aggregated service rate of $\mu c = 150$ requests/sec. Assuming the service rates of Figure 4.6 and setting $d = 1$, there are four possibilities of system configuration using only multi-servers (configurations i) to iv) of Table4.4), and configuration v) with a single server given in Table4.4.

Configuration	c	μ
i)	12	12.5
ii)	6	25
iii)	3	50
iv)	2	75
v)	1	150

Table 4.4: Configurations for an aggregated service rate of $\mu c = 150$ requests/sec.

The values[4] of K are $K = [116, 126, 130, 131, 133]$ corresponding to configurations i), ii), iii), iv) and v) respectively. This result shows that a configuration with only 1 server provides the greatest K. Clearly, the response time is longer as the aggregated

[4] K for each configuration can be computed using the procedure *Compute_K* implemented in maple and available in Appendix II.

service rate is shared among the servers. This fact explains why K decreases for configurations that employ various servers with low service rates (e.g., $K = 116$ for $c = 12$ and $\mu = 12.5$, compared to $K = 131$ for $c = 2$ and $\mu = 75$).

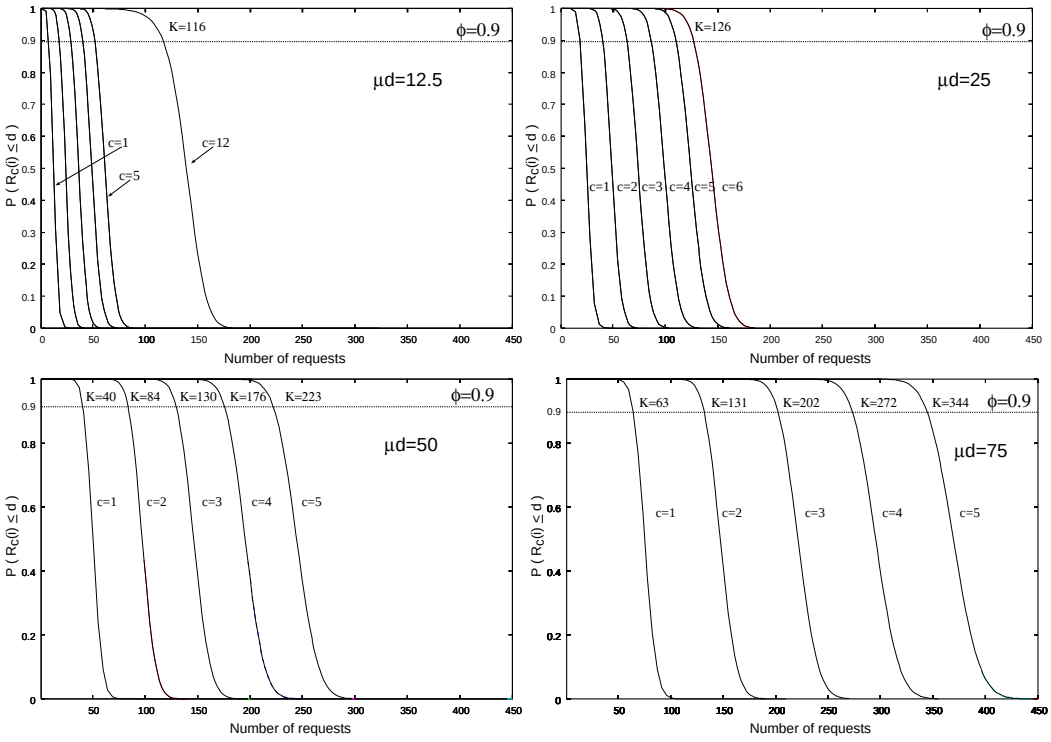


Figure 4.6: $P[R_c(i) \leq d]$ variation for multi-server queuing systems

4.3.2.2 Load effects on UA

Table 4.5 shows the impact of c and μd on UA for different loads ρ (setting d to 1 to simplify the analysis) assuming the same aggregated service rate of $\mu c = 150$ requests/sec. The service unavailability is given in minutes per year for $c = [12, 6, 3, 2, 1]$ servers subject to various loads $\rho = [0.7, 0.8, 0.9, 0.95]$. UA is computed using equation (4.13) based on the value of K obtained from equation (4.12). Note that for $\rho \leq 0.8$, there is a very small unavailability due to long response time.

According to the Table 4.5, it can be seen that the lowest UA is obtained for a single powerful server (configuration v). For all configurations, UA is less than 5 min 30 sec per year, which is not significant. Therefore, in this design study, all configurations

Configuration	c	μ	UA in minutes per year or days:hours:minutes $\rho = 0.8$	$\rho = 0.9$	$\rho = 0.95$
i)	12	12.5	0	5.27	1948.03 = 00:32:28
ii)	6	25	0	1.13	917.32 = 00:15:17
iii)	3	50	0	0.53	667.95 = 00:11:07
iv)	2	75	0	0.50	616.85 = 00:10:16
v)	1	150	0	0.38	544.02 = 00:09:04

Table 4.5: UA for an aggregated service rate of $\mu c = 150$ requests/sec.

support basically the same UA even for heavy loads $\rho \leq 0.9$. However, UA is significantly higher for $\rho = 0.95$. This fact suggests that UA decreases significantly as the load decreases for all configurations.

4.3.2.3 Impact of aggregated service rate on UA

In order to compare the effect of the aggregated service rate on UA, let us take another simple example. Assume that the system is able to provide an aggregated service rate of $\mu c = 50$ requests/sec. Table 4.6 illustrates the configurations with the evaluated UAs. If we compare the first example ($\mu c = 150$) to the second ($\mu c = 50$), it can be seen that the difference on UA among the configurations becomes negligible as the aggregate service rate increases. Actually, when $\rho \leq 0.9$, UA is in the order of hours per year for $\mu c = 50$ and minutes per year for $\mu c = 150$. This fact is better illustrated comparing Tables 4.5 and 4.6. Thus, for the same load $\rho = 0.8$, there is a difference on UA varying from 4 hours to 54 minutes per day (Table 4.6[5]), but such a difference does not exist when the aggregated service rate is 150 requests/sec (see Table 4.5). This trend suggests that the effect of loads relatively heavies on UA is less substantial as the aggregate service rate increases.

Configuration	c	μ	UA in days:hours:minutes per year $\rho = 0.8$	$\rho = 0.9$	$\rho = 0.95$
i)	4	12.5	00:04:08	09:20:56	63:06:09
ii)	2	25	00:01:34	06:07:24	50:15:22
iii)	1	50	00:00:54	05:01:31	44:13:26

Table 4.6: UA for an aggregated service rate of $\mu c = 50$ requests/sec.

[5]For $\rho = 0.8$, the obtained value of UA is acceptable ($UA \approx 4$ hours per year in the worst case)

4.3.2.4 Impact of the number of servers c on UA

Table 4.7 shows the impact of c for three values of service rates $\mu = [25, 50, 75]$, when the load is set to $\rho = 0.9$. It is important to note that increasing c is efficient for reducing UA especially when the load is not heavy $\rho < 0.9$. For instance, if $\mu = 25$ and $\rho = 0.9$, then UA is 49 days per year for $c = 1$ compared to ($UA \approx 16$ minutes per year) for $c = 3$. It becomes more efficient as μ increases (e.g., for $\mu = 50$, $UA \approx 4$ days per year for $c = 1$ compared to $UA \approx 1$ minute per year for $c = 2$).

c	$\mu = 25$	$\mu = 50$	$\mu = 75$
1	49:07:22	04:20:29	00:10:16
2	06:07:24	00:01:09	0
3	00:15:51	0	0
4	00:01:38	0	0
5	00:00:07	0	0

Table 4.7: UA in days:hours:minutes per year for $\rho = 0.9$.

4.4 Conclusion

It is widely recognized that long response time has an impact on the overall service availability. To our knowledge, however, previous research dealing with availability modeling of web-based services has not taken into account the long response time effects on service availability especially from the web user perspective.

In this chapter, we have introduced a simple analytic modeling approach for computing unavailability due to long response time based on queueing systems theory and Markov reward models. In order to provide a practical approach, closed-form equations have been obtained introducing a flexible mathematical abstraction that is general enough to capture the essence of service unavailability due to long response time. One of the main advantages of closed-form equations is that they show explicitly the relationships between the measures and the various parameters characterizing the service. The developed approach can be applied to a wide range of systems needing to provide service under time constraints.

In the context of this thesis, the sensitivity analysis results allow the web server designers to draw some practical conclusions concerning the impact of various parameters on the service unavailability. For example, the results have shown that for "light" loads (i.e., $\rho \leq 0.7$), the unavailability due to long response time is negligible. From the web server designer perspective, the evaluations suggest that systems with low service rate subject to a heavy load $\rho \geq 0.9$ tend to exhibit the highest unavailability

due to long response time. The effect of heavy loads on UA is less substantial as the aggregate service rate increases. Also, the obtained results have suggested that the difference on UA among the configurations becomes negligible as the aggregate service rate increases. Finally, increasing the number of servers (c) has been efficient for reducing UA especially for not heavy loads $\rho < 0.9$.

It has been shown that it is possible to provide a service satisfying a response time requirement using only servers with low service rate, although this is not the optimal configuration. In fact, we should employ either a powerful single server or various servers preventing as much as possible the overloaded periods ($\rho \geq 0.9$). For multi-servers systems, the response time is longer as the aggregated service rate is shared among the servers. This fact explains why configurations that employ various servers with low service rates are not the optimal configuration.

All the analyzes presented have focused on the unavailability due to long response time, assuming that all the servers are available. We emphasize the fact that if we take into account the failures of one or more servers, the impact of long response time on service unavailability should be more significant. Although the optimal configuration consists of a powerful single server, it represents a single point of failure under the availability viewpoint. Therefore, an alternative configuration employing more than a single server should provide a better tradeoff supporting degradable service under the presence of failures.

Finally, it is recognized that the service unavailability in the context of web based systems might be due to problems with the host (e.g., the remote host is too busy handling other requests), problems with the underlying network (e.g., a proper route to the site does not exist) or problems in the user host. In this chapter, our attention was devoted to the service unavailability due to long response time concentrated at the server side. In order to analyze the impact of the response time on the service availability as perceived by users, it is necessary to include other components affecting the time spent by a user request, e.g. the network delay (latency and transmission time), etc.

Conclusion

L'utopie n'est pas ce qui est irréalisable mais ce qui reste à réaliser.

THIS thesis has addressed the problem of modeling and evaluating the availability of web-based service using a pragmatic approach. Much research effort has been devoted to the analysis of the causes of service unavailability. These studies are based on measurements and monitoring of web sites. However, there is still a need for analytical modeling approaches in order to provide a support to the web designers for the availability evaluation of the web-based services. This thesis is aimed at contributing to fill this gap, introducing a pragmatic analytical modeling approach for analyzing the availability of such services. The developed models included multiple sources of service unavailability taking into account, in particular i) hardware and software failures affecting the servers, and ii) performance related failures that are due to e.g. the overload of the servers, or that lead to very long response times unacceptable from the user perspective.

Contributions

The main contributions of the work presented in this thesis are summarized in the following :

1) We have presented a hierarchical modeling framework that is aimed to provide a pragmatic approach to the designers of web-based applications and systems in order to analyze and quantify the availability of the service delivered to the users. The proposed framework distinguishes four different abstraction levels, namely, user, function, service and resources levels. Such decomposition enables the designers to better understand how the various components of the web based application and infrastructure might impact the quality of service delivered to the users from the availability point of view. A performability modeling approach combining Markov chains and queuing models is proposed to evaluate the quantitative availability measures

and to carry out various sensitivity analyses considering different assumptions on the architecture, the faults types, the recovery strategies, the users profile and the traffic characteristics. It is noteworthy that although our research is specially focused on the availability evaluation, the multi-level modeling framework presented in this thesis can be applied to evaluate other dependability measures, e.g., reliability.

The main concepts and the feasibility of the proposed framework have been illustrated using the example of a travel agency implemented on the web. We have shown that the proposed hierarchical framework provides a systematic and pragmatic modeling approach, that is necessary to be able to evaluate the dependability characteristics of the target application at different levels of abstractions. Several sensitivity analyses were presented to show the impact of user operational profile, the fault coverage and the travel agency architecture. In particular, this example showed that different user operational profiles might affect significantly the user perceived availability. Thus, it is important to have an accurate characterization of the behavior of the users interacting with the web site in order to have realistic estimation of the impact of failures from the user perspective.

2) With respect to cluster-based architectures that are the leading architectures for implementing large commercial web sites, we have developed detailed analytical models presented in Chapter 3. Our approach builds on the traditional performability modeling approach by providing closed-form equations for requests loss probability. The requests loss probability can be caused by i) buffer overflow, ii) servers failures or, iii) latency of failure detection. The evaluated availability measure includes such requests loss probability as a source of web service unavailability extending the classical notion of availability.

The developed models are well suited to analyze the impact on the web service availability of two different recovery strategies, namely i) Non Client Transparent (NCT) recovery strategy, for which all requests in progress as well as the arriving requests directed to the failed server node before failure detection are lost; and ii) Client Transparent (CT) recovery strategy, that in contrast ensures the migration of all those requests to the non-failed nodes of the cluster, in a transparent way. Such analysis is carried out using two simple traffic models (Markov Modulated Poisson Process (MMPP) and Poisson) addressing the impact of traffic burstiness on web service availability for both recovery strategies.

We have carried out several sensitivity analyses to study the impact on the service availability of i) the number of server nodes in the cluster, ii) the failure rates of the server nodes, iii) the failure detection rate, iv) the service rate, v) the traffic burstiness and vi) the load. These analyses provide useful guidelines for the design of web-based clusters, since they can be used for dimensioning the web architecture and making the right tradeoffs for achieving high availability with acceptable performance levels. An interesting result suggests that the support for requests migration (CT) has been significantly efficient only for light loads (load being less than 0.3 for MMPP and 0.65 for Poisson). Concerning the impact of the traffic model on the results, we have observed that the Poisson and MMPP models provide similar results only for light load (less than 0.3 in our study). For higher load values significant differences between the

availability results associated to these models are observed. Also, we have shown that for clusters that are relatively overloaded, the service availability supported by web clusters is highly sensitive to web traffic burstiness.

3) Considering the service availability from the user perspective, the service is generally perceived as degraded or even unavailable if the response time is too long compared to what the users are expecting. Thus, it is important to take into account the response time when evaluating the service availability. Unfortunately, this is not usually the case. The analytical modeling approach for computing service unavailability due to long response time and the sensitivity analysis results presented in Chapter 4 are aimed to fill this gap. The proposed approach relies on Markov reward models and queuing theory. In particular, a new quantitative measure is defined to characterize the service unavailability taking into account the maximum acceptable response time and the required quality of service. The computation of the service unavailability measure is based on the evaluation of the response time distribution. Closed-form equations are derived for conditional response-time distribution and for the service unavailability due to long response time, considering single and multi-server queueing systems.

In addition, sensitivity analysis results are presented to illustrate how designers can draw some practical conclusions from the models when considering the service unavailability due to long response time. For example, for the parameters evaluated in our study, the results have shown that for "light" loads (i.e., $\rho < 0.7$), the unavailability due to long response time is negligible. From the web designer perspective, the evaluations suggest that systems with low service rate subject to a heavy load $\rho \geq 0.9$ tend to exhibit the highest unavailability due to long response time. It has been shown that it is possible to provide a service satisfying a response time requirement using only servers with low service rate, although this is not the optimal configuration. In fact, we should employ either a powerful single server or various servers preventing as much as possible the overloaded periods.

Future research

Several directions can be explored for future research to extend the contributions presented in this thesis towards the availability modeling and evaluation of web-based services. First, it is important that we apply our hierarchical multi-level modeling to a more complex case study with detailed information describing the web based application. Indeed, although the travel agency example presented in this thesis illustrated the main concepts and the feasibility of our approach, a more realistic example is needed to study the scalability and suitability of our approach in other application contexts. In particular, with the emergence and the increasing use of the web service standard developed by the web consortium, we should analyze how to adapt our modeling framework in such context. In addition, it would be useful to complement the models presented in this thesis with measurement-based analyses in order to validate some of the modeling assumptions and sensitivity analysis results.

Also, it is very important to identify the factors that will shape the web traffic in the future. Further developments are needed to analyze how the web traffic characteristics might affect the user perceived availability. The representativity of the considered web traffic models (Poisson and MMPP) has been the subject of several debates [Muscariello et al. 2004, Chen et al. 2001, Iyengar et al. 1999, Morris & Lin 2000]. Extensions of our modeling results are needed to analyze other traffic models that better reflect self-similarity and long range dependence (LRD) properties observed in e.g. in the traces analyzed in [Paxson 1997, Crovella & Bestavros 1997, Arlitt & Williamson 1997].

Finally, in this thesis, we have analyzed service availability taking into account unavailability causes that might occur at the service provider sites due to accidental faults. Extensions are needed to include failures due to intentional malicious faults. For example, denial of service attacks constitute a serious problem preventing users to access the provider sites. Failures occurring at the network side and at the client side should also be included to analyze the end-to-end service availability. Such analysis is likely to lead to complex models that may not be tractable if we use analytical models. Simulation based approaches or approximate solutions should be more appropriate to carry out such analyzes.

Appendix I

L'imagination est plus importante que le savoir.

Albert Einstein

The objective of this appendix is twofold: i) to show the proofs of the obtained equations related to the chapter 3; ii) to provide an implementation of the developed models in chapter 3.

Loss probability due to server node failure $L(k\gamma)$

Claim 1 *Using the notations introduced in chapter 3, we obtain*

$$L(k\gamma) = \left[\frac{\rho(1-\rho^b)}{1-\rho^{b+1}}\right] - \left[\frac{(1-\rho)v^2(1-v^b)}{(1-\rho^{b+1})\rho(1-v)}\right]$$

Proof:

As shown in chapter 3, this loss scenario occurs only with the NCT recovery strategy. The request loss probability caused by a transition from state k to state D_k corresponds to the loss of all requests (queued or in service), when a web server node fails.

In order to determine the probability that a node failure affects the requests, let $L(k|i)$ be defined as the conditional probability that i requests are lost when a failure occurs. By PASTA theorem [Wolff 1982], the probability of loss denoted $L(k\gamma)$ due to server failure is equal to the steady state probability that there are i requests in the queue p_i when a failure occurs, given by

$$L(k\gamma) = \sum_{i=1}^{b} p_i L(k|i)$$

where p_i denotes the steady state probability for an M/M/1/b queue defined as follows

$$p_i = \frac{1-\rho}{1-\rho^{b+1}}\rho^i$$

In order to determine $L(k|i)$, let us define the following random variables:

i) F denoting a node failure event with probability distribution $(1 - e^{-k\gamma t})$;

ii) S denoting the total time spent in an M/M/1/b queue with probability density function $s(t) = \frac{\mu^{i+1}t^i e^{-\mu t}}{i!}$ [Kleinrock 1975], given that there are i requests in the system[6].

Assuming that the random variables F and S are independent, we need to calculate the probability that one variable is smaller than another $P[F < S]$, which captures the probability that a node failure affects the queued (or in service) request(s). Thus, given that there are i requests (at least 1 request $1 \leq i \leq b$) in the system before the server failure, the conditional probability of loss denoted $L(k|i)$ can be calculated by conditioning on S as follows

$$\begin{aligned}
L(k|i) &= \int_0^\infty P[F < S|S = t]s(t)dt \\
&= \int_0^\infty P[F < t]\frac{\mu^{i+1}t^i e^{-\mu t}}{i!}dt \\
&= \int_0^\infty (1 - e^{-k\gamma t})\frac{\mu^{i+1}t^i e^{-\mu t}}{i!}dt \\
&= \int_0^\infty \frac{\mu^{i+1}t^i e^{-\mu t}}{i!}dt - \int_0^\infty \frac{\mu^{i+1}t^i e^{-(\mu+k\gamma)t}}{i!}dt \\
&= \mu^{i+1}\int_0^\infty \frac{t^i e^{-\mu t}}{i!}dt - \mu^{i+1}\int_0^\infty \frac{t^i e^{-(\mu+k\gamma)t}}{i!}dt \\
&= 1 - \left[\frac{\mu}{\mu+k\gamma}\right]^{i+1}
\end{aligned}$$

Therefore, we have

$$L(k\gamma) = \sum_{i=1}^{b}\left[\frac{1-\rho}{1-\rho^{b+1}}\rho^i\right]\left[1 - (\frac{\mu}{\mu+k\gamma})^{i+1}\right]$$

[6]The total time spends in the system is the sum of $i+1$ independent exponential random variables, each with mean $1/\mu$

Let v be defined as $v = \dfrac{\rho\mu}{\mu + k\gamma}$,

$$\begin{aligned} L(k\gamma) &= \left[\frac{1-\rho}{1-\rho^{b+1}}\right]\left[\frac{1}{\rho}\right]\left[\sum_{i=1}^{b}\rho^{i+1} - \sum_{i=1}^{b} v^{i+1}\right] \\ L(k\gamma) &= \left[\frac{1-\rho}{1-\rho^{b+1}}\right]\left[\frac{\rho(1-\rho^b)}{(1-\rho)} - \frac{v^2(1-v^b)}{\rho(1-v)}\right] \end{aligned}$$

Finally we obtain the following closed-form equation

$$L(k\gamma) = \left[\frac{\rho(1-\rho^b)}{1-\rho^{b+1}}\right] - \left[\frac{(1-\rho)v^2(1-v^b)}{(1-\rho^{b+1})\rho(1-v)}\right]$$

Recall that an M/M/1/b queue system is stable for all values of load $\rho \neq 1$ (see [Bolch et al. 1998]). In the case of $\rho = 1$, $L(k\gamma)$ should be computed using

$$L(k\gamma) = \left[\frac{b}{b+1}\right] - \left[\frac{(\frac{\mu}{\mu+k\gamma})^2(1-\frac{\mu}{\mu+k\gamma})^b}{(b+1)(1-\frac{\mu}{\mu+k\gamma})}\right]$$

□

Implementation using Matlab version 6.5

NCT recovery strategy

This section presents an implementation of the model developed in chapter 3 corresponding to the **NCT recovery strategy**. The function *NCT* computes the unavailability measure UA introduced by equation (3.5) as a function of the number of nodes N in the cluster (e.g., $N = 1...25$ can be executed doing NCT([1:25]) in the matlab environment). In fact, the model specification can be made by assigning numerical values to the required parameters (see the example below). In this case, the unavailability measure UA is computed for N varying from 1 to 25. The results are written to the file *UA.plot* as well as the model specification is saved into the file *Model_Parameters*. Also, if the matlab support for graphical generation is available, a graphic is plotted to the standard output.

```
function y=NCT(x)
  [m,n] = size(x);
  if (~((m == 1) | (n == 1)) | (m == 1 & n == 1))
     error('Input must be a vector')
  end
```

```
fd = fopen('UA.plot','w');

% Model specification

   % Arrival Rate
   lambda=20;     % requests per second

   % Node Service Rate
   mu=5;     % requests per second

   % Buffer Size
   B=20;

   % Failure Rate
   gamma=1/864000;  % Mean time to failure = 10 days = 864000 seconds

   % Repair Rate
   tau=1/200;       % Mean time to repair = 200 seconds

   % Detection Rate
   alpha=1/2;       % Mean time to detection = 2 seconds

% N = Number of nodes
for N=1:(length(x))

% Computing steady state probabilities of the availability model
facN = factorial(N);
frac1=0
frac3=0

% Computing probability on N available nodes
 for f=0:N
   Rhof = (gamma/tau)^f;
   facNf = factorial(N-f);
   aux1 = Rhof * (facN/facNf) ;
   frac1 = frac1 + aux1;

   if ( f <= (N-1) )
     aux3 = gamma * (N-f) * facN;
     aux3 = aux3/ ( alpha * facNf );
     aux3 = Rhof * aux3;
```

```
     frac3 =  frac3 + aux3;
   end
 end
 aux1 = frac1 + frac3;
 ProbN = 1 / aux1;

 ProbN_f0= (gamma/tau)^N * ProbN * facN;
 for f=0:(N-1)
   Rhof = (gamma/tau)^f;
   facNf = factorial(N-f);
   aux1 = Rhof * (facN/facNf) * ProbN;
   ProbN_f(N-f) = aux1;

   aux1 = aux1 * ( ( gamma * (N-f) ) / alpha) ;
   ProbD_f(N-f) = aux1;
 end

 % Computing probability of loss conditioned on states of the
 % availability model

for f=0:(N-1)
    lambda_aux = lambda / (N-f);
    % Utilisation factor
    rho=lambda_aux/mu;
    if (lambda_aux == mu)
         Pb = 1/ (B+1);
         P0 = Pb;
    else
         Pb = (1-rho) * rho^(B);
         Pb = Pb/ (1 - (rho^(B+1)) );
         P0 = (1-rho) / (1 - (rho^(B+1)) );
    end

    Loss(N-f)= 1 - ( (1 - Pb )^(N-f) ); % N-f = k in the model

    % Probability of loss for requests in processing
    if (lambda_aux == mu)
      aux3 = (gamma * (N-f)) + mu;
      aux4 = mu/aux3;
      aux5 = B/ (B+1);
      aux6 = 1 - aux4^B;
      aux7 = 1 - aux4;
```

```
      aux8 = aux6/ (aux7 * (B + 1));
      aux10 = aux4^2 * aux8 ;
      Lk  = aux5 - aux10;
    else
      aux3 = (gamma * (N-f)) + mu;
      aux4 = 1 - rho^(B+1);
      aux5 = mu*rho/aux3;
      aux6 = 1 - rho^B;
      aux7 = 1 - aux5^B;
      aux8 = (aux6 / aux4) * rho;
      aux9 = (1 - rho)  * aux5^2 * aux7;
      aux10 = rho *  aux4 * (1 - aux5) ;
      aux10 = aux9/aux10;
      Lk   = aux8 - aux10;
    end
    Loss_in_proc(N-f)= Lk;

    % request arrival during detection time
    temp =  (lambda_aux /  alpha ) ;
    aux =  (lambda_aux / (lambda_aux + alpha) ) ;
    aux = aux^temp;

    aux2 = ( 1 - ( (1 - Pb )^(N-f-1) ) );
    Loss_Detection(N-f) = aux + aux2 ;

  end

  U = ProbN_f0; % probability that all nodes are unavailable
  for f=0:(N-1)
    aux =  (ProbN_f(N-f) * Loss(N-f)) +  (ProbN_f(N-f)* Loss_in_proc(N-f));
    aux2 = ProbD_f(N-f) * Loss_Detection(N-f);
    U = U + aux + aux2;
  end

  Un(N)=U;
  fprintf(fd,'%d %.20f\n',N,U);

  end  % end of the for N

  fclose(fd);

y = 0 ;
```

```
% end of NCT function

% Plotting the results
xlabel('Number of nodes in the cluster ')
ylabel('UA')
title(['NCT assumption '])

hold on;
plot((1:N), (Un(1:N)));
```

CT recovery strategy

This section presents an implementation of the model developed in chapter 3 corresponding to the **CT recovery strategy**. As in the previous section, the function *CT* computes the unavailability measure UA defined by equation (3.5) as a function of the number of nodes N in the cluster. The function implements the same features as in *NCT* (i.e., results written to the file *UA.plot*, model specification saved in *Model_Parameters*, graphic generation if matlab support is available).

```
function y=CT(x)

  [m,n] = size(x);
  if (~((m == 1) | (n == 1)) | (m == 1 & n == 1))
    error('Input must be a vector')
  end
  fd = fopen('UA.plot','w');

  % Model specification

       % Arrival Rate
       lambda=20;     % requests per second

       % Node Service Rate
       mu=5;     % requests per second

       % Buffer Size
       B=20;

       % Failure Rate
       gamma=1/864000;  % Mean time to failure = 10 days = 864000 seconds

       % Repair Rate
```

```
     tau=1/200;       % Mean time to repair =  200 seconds

     % Detection Rate
     alpha=1/2;       % Mean time to detect = 2 seconds

% N = Number of nodes
for N=1:length(x)

% Computing steady state probabilities of availability model
facN = factorial(N);
frac1=0
frac3=0

% Computing probability on N available nodes
for f=0:N
  Rhof = (gamma/tau)^f;
  facNf = factorial(N-f);
  aux1 = Rhof * (facN/facNf) ;
  frac1 = frac1 + aux1;

  if ( f <= (N-1) )
    aux3 = gamma * (N-f) * facN;
    aux3 = aux3/ ( alpha * facNf );
    aux3 = Rhof * aux3;
    frac3 =  frac3 + aux3;
  end
end
aux1 = frac1 + frac3;
ProbN = 1 / aux1;

ProbN_f0= (gamma/tau)^N * ProbN * facN;
for f=0:(N-1)
  Rhof = (gamma/tau)^f;
  facNf = factorial(N-f);
  aux1 = Rhof * (facN/facNf) * ProbN;
  ProbN_f(N-f) = aux1;

  aux1 = aux1 * ( ( gamma * (N-f) ) / alpha) ;
  ProbD_f(N-f) = aux1;
end

% Computing probability of loss conditioned on the states of the
```

```
% availability model
for f=0:(N-1)
   lambda_aux = lambda / (N-f);

   % Utilisation factor
   rho=lambda_aux/mu;
   if (lambda_aux == mu)
        Pb = 1/ (B+1);
        P0 = Pb;
   else
        Pb = (1-rho) *rho^(B);
        Pb = Pb/ (1 - (rho^(B+1)) );
        P0 = (1-rho) / (1 - (rho^(B+1)) );
   end

   aux =(lambda_aux /  (lambda_aux + alpha))^(B) ;
   sum = P0 * aux;
   for i = 1:B
     if (lambda_aux == mu)
        P(i) = 1/ (B+1);
     else
        Pi = (1-rho) * rho^(i);
        Pi = Pi/ (1 - (rho^(B+1)) );
        P(i) = Pi;
     end
     aux =(lambda_aux /  (lambda_aux + alpha))^(B-i) ;
     aux = aux * P(i);
     sum = sum + aux;
   end

   Loss(N-f)= 1 - ( (1 - Pb )^(N-f) ) ;
   aux2 = 1 - ( (1 - Pb )^(N-f-1) )  ;
   Loss_Detection(N-f) = sum + aux2 ;
end

U = ProbN_f0; % probability that all nodes are unavailable
for f=0:(N-1)
   aux =  ProbN_f(N-f) * Loss(N-f);
   aux2 = ProbD_f(N-f) * Loss_Detection(N-f);
   U = U + aux + aux2;
end
```

```
  Un(N) = U;
  fprintf(fd,'%d %.20f\n',N,U);

  end % end of for N

  fclose(fd);

y = 0 ;
% end of CT function

% Plotting the results
xlabel('Number of nodes in the cluster ')
ylabel('UA')
title(['CT assumption '])

hold on;
plot((1:N), (Un(1:N)));
```

Appendix II

That I have been able to accomplish anything in mathematics is really due to the fact that I have always found it so difficult.

David Hilbert

The objective of this appendix is twofold: i) to show the proofs of the obtained equations related to the chapter 4; ii) to provide an implementation of the approach introduced in chapter 4.

Conditional Response Time Distribution

Claim 2 *Using the notations introduced in section 4.1, we have*

$$P[R_c(i) \leq d] = \begin{cases} 1 - e^{-\mu d} & \text{, if } i < c \\ 1 - (\frac{c}{c-1})^{i-c+1} e^{-\mu d} \left[1 - \frac{\Gamma(i-c+1,(c-1)\mu d)}{\Gamma(i-c+1)}\right] - \frac{\Gamma(i-c+1,c\mu d)}{\Gamma(i-c+1)} & \text{, if } i \geq c \end{cases}$$

Proof:

As shown in section 4.3

$$P[R_c(i) \leq d] = \begin{cases} 1 - e^{-\mu d} & \text{, if } i < c \\ \int_0^d \left[1 - \sum_{j=0}^{i-c} \frac{(\mu c(d-y))^j}{j!} e^{-\mu c(d-y)}\right] \mu e^{-\mu y} dy & \text{, otherwise} \end{cases}$$

For $i \geq c$, this equation can be written as follows

$$P[R_c(i) \leq d] \quad = \quad 1 - e^{-\mu d} - \mu e^{-c\mu d} \int_0^d \sum_{j=0}^{i-c} \frac{(\mu c(d-y))^j}{j!} e^{+\mu y(c-1)} dy \quad (4.14)$$

Let $I(k)$ be defined as

$$I(k) \quad = \quad \int_0^d \sum_{j=0}^{k} \frac{(\mu c(d-y))^j}{j!} e^{+\mu y(c-1)} dy$$

The integration by parts of $I(k)$ leads to an induction formula from which we can easily obtain a simple expression for $I(k)$

$$\begin{aligned}
I(k) &= \frac{1}{\mu(c-1)} \left[e^{+\mu d(c-1)} - \sum_{j=0}^{k} \frac{(\mu cd)^j}{j!} \right] + \frac{c}{c-1} \int_0^d \sum_{j=0}^{k-1} \frac{(\mu c(d-y))^j}{j!} e^{+\mu y(c-1)} dy \\
&= V(k) + \frac{c}{c-1} I(k-1) \\
&= V(k) + \frac{c}{c-1} V(k-1) + (\frac{c}{c-1})^2 V(k-2) + \cdots + (\frac{c}{c-1})^{k-1} V(1) + (\frac{c}{c-1})^k V(0) \\
&= \frac{1}{\mu(c-1)} \sum_{l=0}^{k} (\frac{c}{c-1})^{k-l} \left[e^{+\mu d(c-1)} - \sum_{j=0}^{l} \frac{(c\mu d)^j}{j!} \right]
\end{aligned}$$

Replacing $I(k)$ in (4.14) with $k = i - c$ and $i \geq c$, we obtain

$$\begin{aligned}
P[R_c(i) \leq d] &= 1 - e^{-\mu d} - \frac{e^{-c\mu d}}{c-1} \sum_{l=0}^{i-c} (\frac{c}{c-1})^{i-c-l} \left[e^{+\mu d(c-1)} - \sum_{j=0}^{l} \frac{(c\mu d)^j}{j!} \right] \\
&= 1 - e^{-\mu d} - (\frac{c}{c-1})^{i-c} \frac{e^{-c\mu d}}{c-1} \sum_{l=0}^{i-c} (\frac{c-1}{c})^l \left[e^{+\mu d(c-1)} - \sum_{j=0}^{l} \frac{(c\mu d)^j}{j!} \right]
\end{aligned}$$

Using the fact that $\sum_{l=0}^{i-c} (\frac{c-1}{c})^l = c \left[1 - (\frac{c-1}{c})^{i-c+1} \right]$, the expansion of this equation results in

$$
\begin{aligned}
P[R_c(i) \leq d] &= 1 - e^{-\mu d} - (\frac{c}{c-1})^{i-c} \frac{e^{-c\mu d}}{c-1} \left(c \left[1 - (\frac{c-1}{c})^{i-c+1} \right] \right) e^{+\mu d(c-1)} \\
&\quad + \left[(\frac{c}{c-1})^{i-c} \frac{e^{-c\mu d}}{c-1} \sum_{l=0}^{i-c} (\frac{c-1}{c})^{l} \sum_{j=0}^{l} \frac{(c\mu d)^j}{j!} \right] \\
&= 1 - e^{-\mu d} + e^{-\mu d} - (\frac{c}{c-1})^{i-c+1} e^{-\mu d} \\
&\quad + \left[(\frac{c}{c-1})^{i-c} \frac{e^{-c\mu d}}{c-1} \sum_{l=0}^{i-c} (\frac{c-1}{c})^{l} \sum_{j=0}^{l} \frac{(c\mu d)^j}{j!} \right] \\
&= 1 - (\frac{c}{c-1})^{i-c+1} e^{-\mu d} + \frac{e^{-c\mu d}}{c-1} \sum_{l=0}^{i-c} (\frac{c}{c-1})^{i-c-l} \sum_{j=0}^{l} \frac{(c\mu d)^j}{j!} \quad (4.15)
\end{aligned}
$$

Note that the last term of the equation can be simplified through the inversion of the two sums

$$
\begin{aligned}
\sum_{l=0}^{i-c} (\frac{c}{c-1})^{i-c-l} \sum_{j=0}^{l} \frac{(c\mu d)^j}{j!} &= (\frac{c}{c-1})^{i-c} \sum_{j=0}^{i-c} \frac{(c\mu d)^j}{j!} \sum_{l=0}^{i-c-j} (\frac{c-1}{c})^{j+l} \\
&= (\frac{c}{c-1})^{i-c} \sum_{j=0}^{i-c} \frac{((c-1)\mu d)^j}{j!} \sum_{l=0}^{i-c-j} (\frac{c-1}{c})^{l}
\end{aligned}
$$

Simplifying $\sum_{l=0}^{i-c-j} (\frac{c-1}{c})^{l}$ as done in the previous step, we will have

$$
\sum_{l=0}^{i-c} (\frac{c}{c-1})^{i-c-l} \sum_{j=0}^{l} \frac{(c\mu d)^j}{j!} = c(\frac{c}{c-1})^{i-c} \sum_{j=0}^{i-c} \frac{((c-1)\mu d)^j}{j!} \left[1 - (\frac{c-1}{c})^{i-c-j+1} \right]
$$

Applying this result in the equation (4.15) and rearranging the terms, we obtain the following equation

$$
P[R_c(i) \leq d] = 1 - (\frac{c}{c-1})^{i-c+1} e^{-\mu d} \left[1 - e^{-(c-1)\mu d} \sum_{j=0}^{i-c} \frac{((c-1)\mu d)^j}{j!} \right] - e^{-c\mu d} \sum_{j=0}^{i-c} \frac{(c\mu d)^j}{j!}
$$

Finally, using the incomplete gamma function, this equation can be rewritten as follows

$$P[R_c(i) \leq d] = \begin{cases} 1 - e^{-\mu d} & \text{, if } i < c \\ 1 - (\frac{c}{c-1})^{i-c+1} e^{-\mu d} \left[1 - \frac{\Gamma(i-c+1,(c-1)\mu d)}{\Gamma(i-c+1)}\right] - \frac{\Gamma(i-c+1,c\mu d)}{\Gamma(i-c+1)} & \text{, if } i \geq c \end{cases}$$

□

Unavailability due to long response time for multi-server queueing system

Claim 3 *Using the notations introduced in section 4.1, we have*

$$UA = \begin{cases} 1 - p_0 \dfrac{\Gamma(K+1, c\rho)e^{c\rho}}{\Gamma(K+1)} & \text{, if } K \leq c-1 \\ 1 - p_0 \left[\dfrac{\Gamma(c, c\rho)e^{c\rho}}{\Gamma(c)} + \dfrac{(c\rho)^c}{c!}\dfrac{(1-\rho^{K-c+1})}{1-\rho}\right] & \text{, if } K \geq c \end{cases}$$

Proof:

Assuming that the sequence of interarrival times $C_1, C_2, \cdots$ and the sequence of service times $B_1, B_2, \cdots$ are described by a set of independent and identical exponential random variables, this system is a traditional M/M/c in which $p_i(c)$ the probability that the system has i requests at steady-state is well-known [Kleinrock 1975]

$$p_i(c) = \begin{cases} p_0 \dfrac{(c\rho)^i}{i!} & \text{, if } i \leq c-1 \\ p_0 [\dfrac{\rho^i c^c}{c!}] & \text{, if } i \geq c \end{cases}$$

where $\rho = \frac{\lambda}{c\mu}$, with

$$p_0 = \left[\sum_{n=0}^{c-1} \frac{(c\rho)^n}{n!} + (\frac{1}{1-\rho})(\frac{(c\rho)^c}{c!}) \right]^{-1}$$

By definition $A = \sum_{i=0}^{K} p_i(c)$, and using the fact that

$$\left[1 + z + \frac{z^2}{2!} + \cdots + \frac{z^j}{j!}\right] = \frac{\Gamma(j+1, z)e^z}{\Gamma(j+1)}$$

If $K \leq c-1$

$$A = p_0 \left[1 + c\rho + \frac{[c\rho]^2}{2!} + \cdots + \frac{[c\rho]^K}{K!}\right] = p_0 \frac{\Gamma(K+1, c\rho)e^{c\rho}}{\Gamma(K+1)}$$

If $K \geq c$ then

$$
\begin{aligned}
A &= p_0 \left[1 + c\rho + \frac{[c\rho]^2}{2!} + \cdots + \frac{[c\rho]^{c-1}}{(c-1)!} \right] + p_0 \left[\frac{c^c}{c!} \sum_{i=c}^{K} \rho^i \right] \\
A &= p_0 \left[\frac{\Gamma(c, c\rho) e^{c\rho}}{\Gamma(c)} \right] + p_0 \left[\frac{(c\rho)^c}{c!} \frac{1 - \rho^{K-c+1}}{1-\rho} \right]
\end{aligned}
$$

By definition $UA = 1 - A$ which yields in the following closed-form equation

$$
UA = \begin{cases}
1 - p_0 \dfrac{\Gamma(K+1, c\rho) e^{c\rho}}{\Gamma(K+1)} & \text{, if } K \leq c-1 \\
1 - p_0 \left[\dfrac{\Gamma(c, c\rho) e^{c\rho}}{\Gamma(c)} + \dfrac{(c\rho)^c}{c!} \dfrac{(1 - \rho^{K-c+1})}{1-\rho} \right] & \text{, if } K \geq c
\end{cases}
$$

Using the fact that $c\rho = \lambda/\mu$, the last equation can be written as follows

$$
UA = \begin{cases}
1 - p_0 \dfrac{\Gamma(K+1, \lambda/\mu) e^{\lambda/\mu}}{\Gamma(K+1)} & \text{, if } K \leq c-1 \\
1 - p_0 \left[\dfrac{\Gamma(c, \lambda/\mu) e^{\lambda/\mu}}{\Gamma(c)} + \dfrac{(\lambda/\mu)^c}{c!} \dfrac{(1 - \rho^{K-c+1})}{1-\rho} \right] & \text{, if } K \geq c
\end{cases}
$$

The development is similar for an M/M/c/b queueing system for $K < b$ with

$$
p_0 = \left[\sum_{n=0}^{c} \frac{(c\rho)^n}{n!} + \frac{(c\rho)^c}{c!} \sum_{n=1}^{b-c} \rho^n \right]^{-1}
$$

□

Implementation using Maple version 9.5

Conditional Response Time Distribution

Using the same notation introduced in section 4.1, the procedure *Resp_Time* implements the response time probability as a function of the number of requests i given by equation (4.12). In fact, it requires the specification of the parameters μd, c and max. The response time probability is computed for i varying from 1 until *max*. The results are written to the file *response-time.plot*. This computing is performed with a precision of 30 digits when calculating with floating-point numbers. The precision is fundamental especially when calculating the gamma function. In fact, maple allows to specify a precision which led us to implement this model in maple instead of matlab. The default value of digits in maple is 10.

```
Resp_Time := proc(ud,c,max)
 local i,P,fd;
 fd := fopen("response-time.plot", WRITE);
 for i from 1 to max
  do
  P := evalf[30](1 - (c/(c-1))^(i+1) * exp(-ud)
  *(1 -GAMMA(i+1,(c-1)*ud)/GAMMA(i+1) ) - GAMMA(i+1,c*ud)/GAMMA(i+1));
  fprintf(fd," %d %.30f\n",i,P);
 od;
 fclose(fd);
end;
```

The procedure *Compute_K* computes the parameter K based on the response time probability. It is required to specify μd, c and the quality of service requirement ϕ. The number of requests i increases, as long as the probability denoted by P is greater than ϕ. This procedure returns the value of K writing the response time probability results to the file *response-time.plot*.

```
Compute_K := proc(ud,c,phi)
 local i,P,fd;
 P := 1;
 fd := fopen("response-time.plot", WRITE);
 for i from 1 by 1 while P > phi
  do
  P := evalf[30](1 - (c/(c-1))^(i+1) * exp(-ud)
  *(1 -GAMMA(i+1,(c-1)*ud)/GAMMA(i+1) ) - GAMMA(i+1,c*ud)/GAMMA(i+1));
  fprintf(fd," %d %.30f\n",i,P);
 od;
 fclose(fd);
 return (i-2);
end;
```

Unavailability due to long response time

The procedure *UA_MMc* computes the service unavailability due to long response time UA for an M/M/c queueing system given by equation (4.13). All the parameters follow the notation introduced in section 4.1.

```
 UA_MMc := proc(c,rho,ud,phi)
  local K, UA, P0;
  K := Compute_K(ud,c,phi);
  P0 := P0_MMc(c,rho);
```

```
  if (K < c) then
    UA := 1 - (P0*((GAMMA(K+1,c*rho)/GAMMA(K+1))*exp(c*rho)));
  else
    temp := (GAMMA(c,c*rho)/GAMMA(c))*exp(c*rho);
    temp2 := (c*rho)^c/c! * ((1 - rho^(K-c+1))/(1 - rho));
    UA := 1 - P0*(temp + temp2);
  end;
  return(UA);
end;
```

The procedure *P0_MMc* computes p_0 for an M/M/c queueing system.

```
P0_MMc := proc(c,rho)
  local n, sum, p;
  sum := 0;
  for n from 0 to (c-1)
  do
    p :=  (c*rho)^n/n!;
    sum := sum + p;
  od;
  p := (c*rho)^c/c!;
  p := p * (1/(1-rho));
  p := p + sum;
  p := 1/p;
  return(p);
end;
```

The procedure *P0_MMcb* computes p_0 for an M/M/c/b queueing system, where b denotes the buffer size. In order to implement UA for an M/M/c/b queueing system, it is worth to note that we need to change the procedure *UA_MMc* adding the parameter b and replacing the call to *P0_MMc* by *P0_MMcb*.

```
 P0_MMcb := proc(c,rho,b)
  local n, sum, sum2, p;
  sum := 0;
  sum2 := 0;
  for n from 0 to c
  do
    sum := sum + (c*rho)^n/n!;
  od;
  for n from 1 to (b-c)
  do
```

```
    sum2 := sum2 + rho^n;
  od;
  p := (c*rho)^c/c!;
  p :=  sum + (p * sum2 );
  p := 1/p;
  return(p);
end;
```

Bibliography

[Aghdaie & Tamir 2001] Aghdaie, N. & Tamir, Y. (2001). Client-Transparent Fault-Tolerant Web Service. *IEEE International Performance, Computing, and Communications Conference*, pages 209–216.

[Almeida et al. 1996] Almeida, V., Bestavros, A., Crovella, M., & de Oliveira, A. (1996). Characterizing reference locality in the WWW. *Conference on Parallel and Distributed Information Systems (PDIS)*, pages 92–103.

[Andersson et al. 2003] Andersson, M., Cao, J., Kihl, M., & Nyberg, C. (2003). Performance modeling of an apache web server with bursty arrival traffic. *International Conference on Internet Computing*.

[Arlat et al. 1993] Arlat, J., Costes, A., Crouzet, Y., Laprie, J. C., & Powell, D. (1993). Fault injection and dependability evaluation of fault-tolerant systems. *IEEE Transactions on Computers*, 8(42):913–923.

[Arlitt & Williamson 1997] Arlitt, M. & Williamson, C. (1997). Internet Web servers: Workload characterization and implications. *IEEE/ACM Transations on Networking*, 5(5):631–644.

[Aversa & Bertavros 2000] Aversa, L. & Bertavros, A. (2000). Load balancing a cluster of web servers using distributed packet rewriting. *IEEE International Performance, Computing and Communication Conference*, pages 24–29.

[Avizienis et al. 2004] Avizienis, A., Laprie, J. C., & Randell, B. (2004). Dependability and its threats: A taxonomy. In *18th IFIP World Computer Congress*, pages 91–120.

[Balakrishnan & Trivedi 1995] Balakrishnan, M. & Trivedi, K. S. (1995). Component-wise decomposition for an efficent reliability solution of complex models of system with repairable components. *International Symposium on Fault-Tolerant Computing (FTCS)*, pages 259–268.

[Balbo et al. 1988] Balbo, G., Bruell, S., & Ghanta, S. (1988). Combining queuing networks and GSPNs for the solution of complex models of system behaviour. *IEEE Transaction on Computers*, 37:1251–1268.

[Baskett et al. 1975] Baskett, F., Chandy, K., Muntz, R., & Palacios, F. (1975). Open, Closed, and Mixed Networks of Queues with Different Classes of Customers. *Journal of ACM*, 22(2).

[Berson et al. 1991] Berson, S., de Souza e Silva, E., & Muntz, R. (1991). A Methodology for the Specification and Generation of Markov Models. *Numerical Solution of Markov Chains*, pages 11–36.

[Betous-Almeida & Kanoun 2004] Betous-Almeida, C. & Kanoun, K. (2004). Construction and stepwise refinement of dependability models. *Performance Evaluation*, 1(56):277–306.

[Bobbio & Trivedi 1986] Bobbio, A. & Trivedi, K. (1986). An aggregation technique for the transient analysis of stiff markov chains. *IEEE Transactions on Computers*, 9(35):803–814.

[Bolch et al. 1998] Bolch, G., Greiner, S., de Meer, H., & Trivedi, K. S. (1998). *Queueing Networks and Markov Chains*. John Willey and Sons, Inc.

[Bondavalli et al. 2001a] Bondavalli, A., Cin, M. D., Latella, D., Majzik, I., Pataricza, A., & Savoia, G. (2001a). Dependability analysis in the early phases of uml based system design. In *International Journal of Computer Systems - Science and Engineering*, volume 16, pages 265–275.

[Bondavalli et al. 2000] Bondavalli, A., Mura, I., Chiaradonna, S., Filippini, R., Poli, S., & Sandrini, F. (2000). DEEM: a Tool for the Dependability Modeling and Evaluation of Multiple Phased Systems. In *International Conference on Dependable Systems and Networks*, pages 231–236. IEEE Computer Society Press.

[Bondavalli et al. 1999] Bondavalli, A., Mura, I., & Trivedi, K. S. (1999). Dependability modeling and sensitivity analysis of scheduled maintenance systems. *European Dependable Computing Conference EDCC*, pages 7–23.

[Bondavalli et al. 2001b] Bondavalli, A., Nelli, M., Simoncini, L., & Mongardi, G. (2001b). Hierarchical modeling of complex control systems : dependability analysis of a railway interlocking. *Journal of Computer System Science and Engineering*, 4(16):249–261.

[Bowen et al. 2000] Bowen, N., Sturnam, D., & Liu, T. T. (2000). Towards continuous availability of internet services through availability domains. *International Conference on Dependable Systems and Networks (DSN)*, pages 559–566.

[Brewer 2001] Brewer, E. (2001). Lessons from Giant-Scale Service. *IEEE Internet Computing*, pages 46–55.

[Brisco 1995] Brisco, T. (1995). DNS support for load balancing . *IETF RFC 1794*.

[Buzen 1973] Buzen, J. P. (1973). Computational Algorithms for Closed Queuing Networks with Exponential Servers. *Communications ACM*, 16(9).

[Cao et al. 2003] Cao, J., Andersson, M., Nyberg, C., & Kihl, M. (2003). Web performance modeling using an m/g/1/k*ps queue. *International Conference on Telecommunications*.

[Carmo et al. 1998] Carmo, R., de Carvalho, L., de Souza e Silva, E., Diniz, M., & Muntz, R. (1998). Performance/Availability Modeling with the TANGRAM-II Modeling Environment. *Performance Evaluation*, 33:45–65.

[Carrera & Bianchini 2001] Carrera, E. V. & Bianchini, R. (2001). Efficiency vs. portability in cluster-based network services. *Symposium on Principles and Practice of Parallel Programming*.

[Carrera & Bianchini 2005] Carrera, E. V. & Bianchini, R. (2005). Press: a clustered server based on user-level communications. *IEEE Transactions on Parallel and Distributed Systems*, 16(5):385–395.

[C.Béounes et al. 1993] C.Béounes, M.Aguéra, J.Arlat, S.Bachmann, C.Bordeau, J.-E.Doucet, K.Kanoun, J.C.Laprie, S.Metge, de Souza, J., D.Powell, & P.Spiesser (1993). SURF-2: A Program for Dependability Evaluation of Complex Hardware and Software Systems. In *23th IEEE International Symposium on Fault-Tolerant Computing*, pages 668–673. IEEE Computer Society Press.

[Chandra et al. 2001] Chandra, B., Dahlin, M., Gao, L., & Nayate, A. (2001). End-to-end WAN Service Availability. *Third Usenix Symposium on Internet Technologies and Systems (USITS01)*.

[Chen et al. 2001] Chen, X., Mohapatra, P., & Chen, H. (2001). An Admission Control Scheme for Predictable Server Response Time Web Accesses. *IEEE World Web Conference*.

[Ciardo et al. 1989] Ciardo, G., Muppala, J., & Trivedi, K. (1989). SPNP: Stochastic Petri Net Package. In *International Workshop on Petri Nets and Performance Models*, pages 142–151. IEEE Computer Society Press.

[Couvillion et al. 1991] Couvillion, J., Freire, R., Johnson, R., Obal, W., Qureshi, M., Rai, M., Sanders, W., & Tvedt, J. (1991). Performability modeling with UltraSAN. *IEEE Software*, 5(8):69–80.

[Crovella & Bestavros 1997] Crovella, M. & Bestavros, A. (1997). Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes. *IEEE/ACM Transactions on Networking*, 6(5):835–846.

[Dahlin et al. 2003] Dahlin, M., Chandra, B., Gao, L., & Nayate, A. (2003). End-to-end wan service availability. *ACM/IEEE Transactions on Networking*, 11(2).

[de Souza e Silva & Gail 1996] de Souza e Silva, E. & Gail, H. R. (1996). The uniformization method in performability analysis. Technical Report RC20383, IBM.

[Dilley et al. 2001] Dilley, J., Friedrich, R., Jin, T., & Rolia, J. (2001). Web server performance measurements and modleing techniques. *Performance evaluation*, 33:5–26.

[Dutuit & Rauzy 1998] Dutuit, Y. & Rauzy, A. (1998). A guided tour of minimal cut-sets handling by means of binary decision diagrams. *Probabilistic Safety Assessment Conference (PSA)*, 2:55–62.

[Dutuit & Rauzy 2000] Dutuit, Y. & Rauzy, A. (2000). Efficient algorithms to assess components and gates importances in fault tree analysis. *Reliability Engineering and System Safety*, 2(72):213–222.

[E. Anderson & Brewer 1996] E. Anderson, D. P. & Brewer, E. (1996). The MagicRouter, an application of fast packet interposing. *http://www.cs.berkeley.edu*.

[Elwalid & Mitra 1993] Elwalid, A. & Mitra, D. (1993). Effective Bandwidth of General Markovian Traffic Sources an Amission Control of High-Speed Networks. *IEEE ACM Transactions on Networking*, 1(3):329–343.

[Firoiu et al. 2002] Firoiu, V., Boudec, J. L., Towsley, D., & Zhang, Z. (2002). Theories and Models for Internet Quality of Service. *IEEE Proceedings*.

[Florin & Natkin 1985] Florin, G. & Natkin, S. (1985). Les réseaux de Petri stochastic. *Technique et Science Informatiques*, 4(1):143–160.

[Fota et al. 1999] Fota, N., Kaâniche, M., & Kanoun, K. (1999). Dependability evaluation of an air traffic control computing system. *Performance Evaluation*, 4(34):553–573.

[Frost & Melamed 1994] Frost, V. S. & Melamed, B. (1994). Traffic modeling for telecommunications network. *IEEE Communications Magazine*, 32(3):70–81.

[Gama et al. 2004] Gama, G., Nagaraja, K., Bianchini, R., Martin, R. P., Meira, W., & Nguyen, T. D. (2004). State maintenance and its impact on the performability of multi-tiered internet services. *Symposium on Reliable Distributed Systems - SRDS*, pages 146–158.

[Garg et al. 1999] Garg, S., Huang, Y., Kintala, C., Trivedi, K., & Yajnik, S. (1999). Performance and Reliability Evaluation of Passive Replication Schemes in Application Level Fault Tolerance. *IEEE Dependable Systems and Networks*.

[Goyal et al. 1986] Goyal, A., Carter, W., de Souza e Silva, E., Lavenberg, S., & Trivedi, K. (1986). The system availability estimator. In *16th IEEE International Symposium on Fault-Tolerant Computing*, pages 84–89. IEEE Computer Society Press.

[Grassmann et al. 1985] Grassmann, W., Taksar, M., & Heyman, D. (1985). Regenerative Analyses and Steady-State Distribution for Markov Chains. *Operations Research*, (33):1107–1116.

[Gross & Harris 1985] Gross, D. & Harris, C. (1985). *Fundamentals of Queueing Theory*. John Wiley and Sons - New York.

[Hariri & Mutlu 1991] Hariri, S. & Mutlu, H. B. (1991). A hierarchical modeling of availability in distributed systems. *IEEE Conference on Distributed Computing Systems*, pages 190–197.

[Haverkort et al. 2001] Haverkort, B., Marie, R., Rubino, G., & Trivedi, K. (2001). *Performability Modelling - Techniques and Tools*, volume I. Wiley.

[Heidelberger 1995] Heidelberger, P. (1995). Fast simulation of rare events in queueing and reliability models. *ACM Transactions on Modeling and Computer Simulation*, 1(5):43–85.

[Heidelberger et al. 1992] Heidelberger, P., Nicola, V., & Shahabuddin, P. (1992). Simultaneous and efficient simulation of highly dependable systems with different underlying distributions. *Winter Simulation Conference*, (4):137–164.

[Heyman & Lucantoni 2003] Heyman, D. P. & Lucantoni, D. (2003). Modeling multiple ip traffic streams with rate limits. *IEEE/ACM Transactions on Networking*, 11(6):948–958.

[Howard 1971a] Howard, R. (1971a). Dynamic probabilistic systems: Semi-markov and decision processes. *Wiley, New York*.

[Howard 1971b] Howard, R. A. (1971b). *Dynamic Probabilistic Systems*, volume Volume I - Markov Models. Wiley.

[Inc 2000] Inc, C. S. (2000). Failover Configuration for LocalDirector. *http://www.cisco.com*.

[Ingham et al. 2000] Ingham, D. B., Shrivastava, S. K., & Panzieri, F. (2000). Constructing dependable web services. *IEEE Internet Computing*, pages 25–33.

[Iyengar et al. 2000] Iyengar, A., Challenger, J., Dias, D., & Dantzig, P. (2000). High-performance web site design techniques. *IEEE Internet Computing*, pages 17–26.

[Iyengar et al. 1999] Iyengar, A. K., Squillante, M. S., & Zhang, L. (1999). Analysis and Characterization of Large-Scale Web Server Access Patterns and Performance. *IEEE World Wide Web Conference*, pages 85–100.

[Jackson 1963] Jackson, J. (1963). Jobshop-Like Queuing Systems. *Management Science*, 1(10):131–142.

[Kalyanakrishnan et al. 1999] Kalyanakrishnan, M., Iyer, R. K., & Patel, J. U. (1999). Reliability of Internet Hosts: a Case Study from the End User's Perspective. *Computer Networks*, (31):47–57.

[Kaâniche et al. 2003a] Kaâniche, M., Kanoun, K., & Martinello, M. (2003a). User-Perceived Availability of a Web Based Travel Agency. *Proceeding of IEEE International Conference on Dependable Systems and Networks (DSN)*, pages 709–718.

[Kaâniche et al. 2001] Kaâniche, M., Kanoun, K., & Rabah, M. (2001). A framework for modeling the availability of e-business systems. *International Conference on Computer Communications and Networks (ICCCN)*, pages 40–45.

[Kaâniche et al. 2003b] Kaâniche, M., Kanoun, K., & Rabah, M. (2003b). Multi-level modeling approach for the availability assessment of e-business applications. *Software-Practice and Experience*, (33):1323–1341.

[Kaâniche et al. 2002] Kaâniche, M., Simanche, C., Kanoun, K., & Martinello, M. (2002). SoS Dependability Assessment: Modelling Example and Measurement-based experiments. *Deliverable CSAD3*.

[Kanoun & Borrel 2000] Kanoun, K. & Borrel, M. (2000). Fault-tolerant system dependability - explicit modeling of hadware and software component-interactions . *IEEE Transactions on Reliability*, 4(49):363–376.

[Kanoun & Powell 1991] Kanoun, K. & Powell, D. (1991). Dependability evaluation of bus and ring communication topologies for the Delta-4 distributed fault-tolerant architecture. *Symposium on Reliable Distributed Systems (SRDS)*, pages 130–141.

[Kleinrock 1975] Kleinrock, L. (1975). *Queueing Systems*, volume I - Theory. Wiley.

[Labovitz et al. 1999] Labovitz, C., Ahuja, & Jahanian, F. (1999). Experimental study of internet stability and backbone failures. *IEEE International Symposium on Fault-Torelant Computing Systems*.

[Laprie et al. 1996] Laprie, J., Arlat, J., Blanquart, J., A.Costes, Crouzet, Y., Deswarte, Y., Fabre, J., Guillhermain, H., Kaâniche, M., Kanoun, K., Mazet, C., Powell, D., Rabéjac, C., & Thévenod, P. (1996). *Guide de la sûrete de fonctionnement*. Cépaduès-Éditions, Toulouse.

[Laprie 1983] Laprie, J. C. (1983). Trustable evaluation of computer systems dependability. In *Applied Mathematics and Performance/Reliability Models of Computer Systems*. Invited contribution to the International Workshop - Pisa.

[Laprie 1995] Laprie, J. C. (1995). Dependable computing: Concepts, limits, challenges. In *25th IEEE International Symposium on Fault-Tolerant Computing - Special Issue*, pages 42–54. IEEE Computer Society Press.

[LeBoudec 1998] LeBoudec, Y. (1998). Application of network calculus to guarranteed services networks. *IEEE Transactions on Information Theory*, 44:1087–1096.

[Luo & Yang 2002] Luo, M. Y. & Yang, C. S. (2002). Enabling fault resilience for web services. *Computer Communications*, (25):198–209.

[Mainkar 1997] Mainkar, V. (1997). Availability Analysis of Transaction Processing Systems based on User-Perceived Performance. *Proceedings of 16th Symposium on Reliable Distributed Systems*, pages 10–17.

[Majzik & Huszerl 2002] Majzik, I. & Huszerl, G. (2002). Towards dependability modeling of ft-corba architectures. In *EDCC-4: Proceedings of the 4th European Dependable Computing Conference on Dependable Computing*, volume 2485, pages 121–139.

[Martinello et al. 2003] Martinello, M., Kaâniche, M., & K.Kanoun (2003). Web Service Availability : Impact of Error Recovery. *International Conference on Computer Safety, Reliability and Security (Safecomp)*, pages 165–178.

[Martinello et al. 2005] Martinello, M., Kaâniche, M., & K.Kanoun (2005). Web Service Availability : Impact of Error Recovery and Traffic Model. *Journal of Reliability Engineering and System Safety (RESS)*, 89(1):6–16.

[Meehan 2000] Meehan, M. (2000). Upate: System Outages Top Online Brokerage Exec's Concerns. *Computerworld*.

[Menascé et al. 2001] Menascé, D., Barbara, D., & Dodge, R. (2001). Preserving qos of e-commerce sites through self-tuning: a performance model approach. *ACM conference on e-commerce*, pages 224–234.

[Menascé & Almeida 2000] Menascé, D. A. & Almeida, V. A. F. (2000). *Scaling for E-Business: Technologies, Models, Performance and Capacity Planning*. Prentice Hall Inc.

[Menascé & Almeida 2002] Menascé, D. A. & Almeida, V. A. F. (2002). *Capacity Planning for Web Services*. Prentice Hall Inc.

[Merzbacher & Patterson 2002] Merzbacher, M. & Patterson, D. A. (2002). Measuring End-User Availability of the Web:Practical Experience. *Dependable Computing and Network*.

[Meyer 1980] Meyer, J. F. (1980). On evaluating the performability of degradable computing systems. *IEEE Journal on Selected Areas in Communications*, 29(8):720–731.

[Meyer 1982] Meyer, J. F. (1982). Closed-form solutions of performability. *IEEE Transactions on Computing*, 7(31):648–657.

[Morris & Lin 2000] Morris, R. & Lin, D. (2000). Variance of aggregated web traffic. *IEEE Infocom*.

[Muppala et al. 1992] Muppala, J., Trivedi, K., Woolet, S., & Haverkort, B. R. (1992). Composite performance and dependability analysis. *Performance Evaluation*, 14(3):197–216.

[Muppala et al. 1991] Muppala, J., Woolet, S., & Trivedi, K. (1991). Real-time systems performance in the presence of failures. *IEEE Computer*, pages 37–47.

[Muscariello et al. 2004] Muscariello, L., Mellia, M., Meo, M., Marsan, M. A., & Cigno, R. (2004). Markov models of internet traffic and a new hierarchical MMPP model. *IEEE International Conference on Communications*.

[Nagaraja et al. 2005] Nagaraja, K., Gama, G., Bianchini, R., Martin, R., Meira, W., & Nguyen, T. (2005). Quantifying the performability of cluster-based services. *IEEE Transactions on Parallel and Distributed Systems*, 16(5):456–467.

[Nagaraja et al. 2003] Nagaraja, K., Li, X., Zhang, B., Bianchini, R., Martin, R., & Nguyen, T. (2003). Using fault injection and modeling to evaluate the performability of cluster-based services. *In Proceedings of the Usenix Symposium on Internet Technologies and Systems*.

[Nicola 1990] Nicola, V. (1990). Lumpability of Markov Reward Models. *Technical report IBM T.J. Watson Research Center*.

[Nicola et al. 1990] Nicola, V., Nakajama, M., Heidelberger, P., & Goyal, A. (1990). Fast Simulation of Dependability Models with General Failure, Repair and Maintenace Processes. *IEEE International Symposium on Fault-Torelant Computing Systems*.

[Oppenheimer et al. 2003] Oppenheimer, D., Ganapathi, A., & Patterson, D. A. (2003). Why do Internet services fail, and what can be done about it? *Usenix Symposium on Internet Technologies and Systems (USITS)*.

[Oppenheimer & Patterson 2002] Oppenheimer, D. & Patterson, D. A. (2002). Architecture and Dependability of Large-Scale Internet Services. *IEEE Internet Computing*, pages 41–49.

[Pataricza 2002] Pataricza, A. (2002). From the general resource model to a general fault modeling paradigm. In Jürjens, J., Cengerale, M. V., Fernandez, E. B., Rumpe, B., & Sandner, R., editors, *Critical Systems Development with UML- Proceedings of the UML'02 Workshop*, volume TUM-I0208, pages 163–171. Technische Universität München.

[Patterson et al. 2002] Patterson, D. A., Brown, A., Broadwell, P., Candea, G., Chen, M., Cutler, J., Enriquez, P., Fox, A., Kiciman, E., Merzbacher, M., Oppenheimer, D., Sastry, N., Tetzlaff, W., Traupman, J., & Treuhaft, N. (2002). Recovery Oriented Computing (ROC): Motivation, Definition, Techniques, and Case Studies. *UC Berkeley Computer Science Technical Report*.

[Paxson 1997] Paxson, V. (1997). Measurements and Analysis of End-to-End Internet Dynamics. *PhD thesis, University of California*.

[Periorellis & Dobson 2001] Periorellis, P. & Dobson, J. (2001). The Travel Agency Case Study. *DSoS Project IST-1999-11585*.

[Popstojanova & Trivedi 2000] Popstojanova, K. G. & Trivedi, K. S. (2000). Stochastic Modleing Formalism for Dependability, Performance and Performability. In *Performance Evaluation: Origins and Directions*, pages 403–422. Springer-Verlag.

[Rabah & Kanoun 2003] Rabah, M. & Kanoun, K. (2003). Performability evaluation of multipurpose multiprocessor systems: the separation of concers. *IEEE Transactions on Computers*, 2(52):223–236.

[Reeser & Hariharan 2000] Reeser, P. & Hariharan, R. (2000). Analytic Model of Web Servers in Distributed Environments. *Proceedings of Second International Workshop on Software an Performance*, pages 158–167.

[Reiser & Lavenberg 1980] Reiser, M. & Lavenberg, S. (1980). Mean-value Analysis of Closed Multi-Chain Queuing Networks. *Journal ACM*, 27(2).

[Roche & Schabes 1997] Roche, E. & Schabes, Y., editors (1997). *Finite-State Language Processing*. Bradford Book. MIT Press, Cambridge, Massachusetts, USA.

[Rugina 2005] Rugina, A. E. (2005). System dependability evaluation using aadl (architecture analysis and design language). *Rencontres Jeunes Chercheurs en Informatique Temps Réel (RJCITR)*.

[Rugina et al. 2005] Rugina, A. E., Kanoun, K., Kaâniche, M., & Guiochet, J. (2005). Dependability modelling of a fault tolerant duplex system using aadl and gspns. *Rapport du LAAS*, (05315).

[Shin & Krishna 1986] Shin, K. & Krishna, C. M. (1986). New performance measures for design and evaluation of real-time multiprocessors. *Computer System Science and Engineering*, 4(1):179–192.

[Signoret et al. 1998] Signoret, J. P., Lajeunesse, S., Point, G., Thomas, P., Griffault, A., & Rauzy, A. (1998). The altarica language. *European Safety and Reliability Association Conference (ESREL)*.

[Simache 2004] Simache, C. (2004). Évaluation de la sûrete de fonctionnement de systèmes Unix et Windows à partir de données opérationnelles : méthode et application. *PhD Thesis*.

[Simache & Kaâniche 2002] Simache, C. & Kaâniche, M. (2002). Pacific Rim International Symposium on Dependable Computing (PRDC). *IEEE Computer Society*, pages 311–325.

[Slothouber 1996] Slothouber, L. P. (1996). A model of web server performance. *International World Wide Web Conference*.

[Smith et al. 1988] Smith, R., Trivedi, K., & Ramesh, A. (1988). Performability Analysis: Measures, An Algorithm and a Case Study. *IEEE Transactions on Computers*, C-37(4):406–417.

[Tang et al. 2004] Tang, D., Kumar, D., Duvur, S., & Torbjornsen, O. (2004). Availability measurement and modeling for an application server. *IEEE Dependable Systems and Networks*.

[Trivedi et al. 1994] Trivedi, K., Haverkort, B., Rindos, A., & Mainkar, V. (1994). Techniques and tools for reliability and performance evaluation: Problems and perspectives. *Lecture Notes in Computer Science*, pages 1–24.

[Trivedi et al. 1992] Trivedi, K., Muppala, J., Woolet, S., & Haverkort, B. (1992). Composite performance and dependability analysis. *Performance Evaluation*, 14:197–215.

[Trivedi 2002] Trivedi, K. S. (2002). *Probability and Statistics with Reliability, Queuing, and Computer Science Applications*. John Willey and Sons, Inc.

[Trivedi et al. 2003] Trivedi, K. S., Ramani, S., & Fricks, R. (2003). Recent advances in modeling response-time distributions in real-time systems. *Proceeding of the IEEE*, 91(7):1023–1037.

[Urgaonkar et al. 2005] Urgaonkar, B., Pacifini, G., Shenoy, P., Spreitzer, M., & Tantawi, A. (2005). An analytical model for multi-tier internet services and its applications. *International Conference on Measurement and Modeling of Computer Systems SIGMETRICS*, 33(1):291–302.

[V. Cardellini & Yu 2002] V. Cardellini, E. Casalicchio, M. C. & Yu, P. S. (2002). The state of the art in Locally Distributed Web-Server System. *ACM Computing Surveys*, 34(2):363–371.

[Vesely et al. 1981] Vesely, W. E., Goldberg, F. F., Robert, N. H., & Haasl, P. F. (1981). Fault tree handbook. *Technical Report WUREG, US Nuclear Regulatory Commision*.

[Willinger & Paxson 1998] Willinger, W. & Paxson, V. (1998). Where Mathematics meets the Internet. *Notices of the American Mathematical Society*, 45(8):961–970.

[Wolff 1982] Wolff, R. (1982). Poisson arrivals see time averages. *Operations Research*, (30):223–231.

[Xie et al. 2003] Xie, W., H.Sun, Cao, Y., & Trivedi, K. S. (2003). Modeling of user perceived webserver availability. *IEEE Internation Conference on Communications*.

[Zaiane et al. 1998] Zaiane, O. R., Xin, M., & Han, J. (1998). Discovering Web access patterns and trends by applying OLAP and data mining technology on web logs. *Advances in Digital Libraries conference*, pages 19–29.

[Zhang et al. 2004] Zhang, R., Abdelzaher, T. F., & Stankovic, J. A. (2004). Efficient TCP Connection Failover in Web Server Clusters. *IEEE Infocom*.

Modélisation et évaluation de la disponibilité de services mis en œuvre sur le web - Une approche pragmatique

Cette thèse porte sur le développement d'une approche de modélisation pragmatique permettant aux concepteurs d'applications et systèmes mis en œuvre sur le web d'évaluer la disponibilité du service fourni aux utilisateurs. Plusieurs sources d'indisponibilité du service sont prises en compte, en particulier i) les défaillances matérielles ou logicielles affectant les serveurs et ii) des dégradations de performance (surcharge des serveurs, temps de réponse trop long, etc.). Une approche hiérarchique multi-niveau basée sur une modélisation de type performabilité est proposée, combinant des chaînes de Markov et des modèles de files d'attente. Les principaux concepts et la faisabilité de cette approche sont illustrés à travers l'exemple d'une agence de voyage. Plusieurs modèles analytiques et études de sensibilité sont présentés en considérant différentes hypothèses concernant l'architecture, les stratégies de recouvrement, les fautes, les profils d'utilisateurs, et les caractéristiques du trafic.

Mots clefs: sûreté de fonctionnement, web, performabilité, disponibilité de service, modélisation hiérarchique, évaluation.

Availability modeling and evaluation of web-based services - A pragmatic approach

This thesis presents a pragmatic modeling approach allowing designers of web-based applications and systems to evaluate the service availability provided to the users. Multiple sources of service unavailability are taken into account, in particular i) hardware and software failures affecting the servers, and ii) performance degradation (overload of servers, very long response time, etc.). An hierarchical multi-level approach is proposed based on performability modeling, combining Markov chains and queueing models. The main concepts and the feasibility of this approach are illustrated using a web-based travel agency. Various analytical models and sensitivity studies are presented considering different assumptions with respect to the architectures, recovery strategies, faults, users profile and traffic characteristics.

Key words: dependability, web, performability, service availability, hierarchical modeling, evaluation.