

# Chapter 1

## An Overview of Queueing Network Modelling

### 1.1. Introduction

Today's computer systems are more complex, more rapidly evolving, and more essential to the conduct of business than those of even a few years ago. The result is an increasing need for tools and techniques that assist in understanding the behavior of these systems. Such an understanding is necessary to provide intelligent answers to the questions of cost and performance that arise throughout the life of a system:

- *during design and implementation*
  - An aerospace company is designing and building a computer-aided design system to allow several hundred aircraft designers simultaneous access to a distributed database through graphics workstations. Early in the design phase, fundamental decisions must be made on issues such as the database accessing mechanism and the process synchronization and communication mechanism. The relative merits of various mechanisms must be evaluated prior to implementation.
  - A computer manufacturer is considering various architectures and protocols for connecting terminals to mainframes using a packet-oriented broadcast communications network. Should terminals be clustered? Should packets contain multiple characters? Should characters from multiple terminals destined for the same mainframe be multiplexed in a single packet?
- *during sizing and acquisition*
  - The manufacturer of a turn-key medical information system needs an efficient way to size systems in preparing bids. Given estimates of the arrival rates of transactions of various types, this vendor must project the response times that the system will provide when running on various hardware configurations.

- A university has received twenty bids in response to a request for proposals to provide interactive computing for undergraduate instruction. Since the selection criterion is the “cost per port” among those systems meeting certain mandatory requirements, comparing the capacity of these twenty systems is essential to the procurement. Only one month is available in which to reach a decision.
- *during evolution of the configuration and workload*
  - A stock exchange intends to begin trading a new class of options. When this occurs, the exchange’s total volume of options transactions is expected to increase by a factor of seven. Adequate resources, both computer and personnel, must be in place when the change is implemented.
  - An energy utility must assess the longevity of its current configuration, given forecasts of workload growth. It is desirable to know what the system bottleneck will be, and the relative cost-effectiveness of various alternatives for alleviating it. In particular, since this is a virtual memory system, tradeoffs among memory size, CPU power, and paging device speed must be evaluated.

These questions are of great significance to the organizations involved, with potentially serious repercussions from incorrect answers. Unfortunately, these questions are also complex; correct answers are not easily obtained.

In considering questions such as these, one must begin with a thorough grasp of the system, the application, and the objectives of the study. With this as a basis, several approaches are available.

One is the use of *intuition and trend extrapolation*. To be sure, there are few substitutes for the degree of experience and insight that yields reliable intuition. Unfortunately, those who possess these qualities in sufficient quantity are rare.

Another is the *experimental evaluation of alternatives*. Experimentation is always valuable, often required, and sometimes the approach of choice. It also is expensive — often prohibitively so. A further drawback is that an experiment is likely to yield accurate knowledge of system behavior under one set of assumptions, but not any insight that would allow generalization.

These two approaches are in some sense at opposite extremes of a spectrum. Intuition is rapid and flexible, but its accuracy is suspect because it relies on experience and insight that are difficult to acquire and verify. Experimentation yields excellent accuracy, but is laborious and inflexible. Between these extremes lies a third approach, the general subject of this book: *modelling*.

A model is an abstraction of a system: an attempt to distill, from the mass of details that is the system itself, exactly those aspects that are essential to the system's behavior. Once a model has been *defined* through this abstraction process, it can be *parameterized* to reflect any of the alternatives under study, and then *evaluated* to determine its behavior under this alternative. Using a model to investigate system behavior is less laborious and more flexible than experimentation, because the model is an abstraction that avoids unnecessary detail. It is more reliable than intuition, because it is more methodical: each particular approach to modelling provides a framework for the definition, parameterization, and evaluation of models. Of equal importance, using a model enhances both intuition and experimentation. Intuition is enhanced because a model makes it possible to "pursue hunches" — to investigate the behavior of a system under a wide range of alternatives. (In fact, although our objective in this book is to devise *quantitative* models, which accurately reflect the performance measures of a system, an equally effective guide to intuition can be provided by less detailed *qualitative* models, which accurately reflect the general behavior of a system but not necessarily specific values of its performance measures.) Experimentation is enhanced because the framework provided by each particular approach to modelling gives guidance as to which experiments are necessary in order to define and parameterize the model.

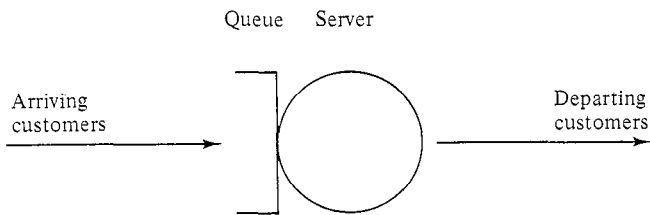
Modelling, then, provides a framework for gathering, organizing, evaluating, and understanding information about a computer system.

## 1.2. What Is a Queueing Network Model?

*Queueing network modelling*, the specific subject of this book, is a particular approach to computer system modelling in which the computer system is represented as a *network of queues* which is evaluated *analytically*. A network of queues is a collection of *service centers*, which represent system resources, and *customers*, which represent users or transactions. Analytic evaluation involves using *software* to solve efficiently a set of equations induced by the network of queues and its parameters. (These definitions, and the informal overview that follows, take certain liberties that will be noted in Section 1.5.)

### 1.2.1. Single Service Centers

Figure 1.1 illustrates a single service center. Customers arrive at the service center, wait in the *queue* if necessary, receive service from the *server*, and depart. In fact, this service center and its arriving customers constitute a (somewhat degenerate) queueing network model.



**Figure 1.1 – A Single Service Center**

This model has two parameters. First, we must specify the *workload intensity*, which in this case is the rate at which customers arrive (e.g., one customer every two seconds, or 0.5 customers/second). Second, we must specify the *service demand*, which is the average service requirement of a customer (e.g., 1.25 seconds). For specific parameter values, it is possible to evaluate this model by solving some simple equations, yielding *performance measures* such as *utilization* (the proportion of time the server is busy), *residence time* (the average time spent at the service center by a customer, both queuing and receiving service), *queue length* (the average number of customers at the service center, both waiting and receiving service), and *throughput* (the rate at which customers pass through the service center). For our example parameter values (under certain assumptions that will be stated later) these performance measures are:

utilization: .625  
 residence time: 3.33 seconds  
 queue length: 1.67 customers  
 throughput: 0.5 customers/second

Figures 1.2a and 1.2b graph each of these performance measures as the workload intensity varies from 0.0 to 0.8 arrivals/second. This is the interesting range of values for this parameter. On the low end, it makes no sense for the arrival rate to be less than zero. On the high end, given that the average service requirement of a customer is 1.25 seconds, the greatest possible rate at which the service center can handle customers is one every 1.25 seconds, or 0.8 customers/second; if the arrival rate is greater than this, then the service center will be saturated.

The principal thing to observe about Figure 1.2 is that the evaluation of the model yields performance measures that are qualitatively consistent with intuition and experience. Consider residence time. When the workload intensity is low, we expect that an arriving customer seldom will encounter competition for the service center, so will enter service immediately and will have a residence time roughly equal to its service requirement. As the workload intensity rises, congestion increases, and residence time along with it. Initially, this increase is gradual. As the

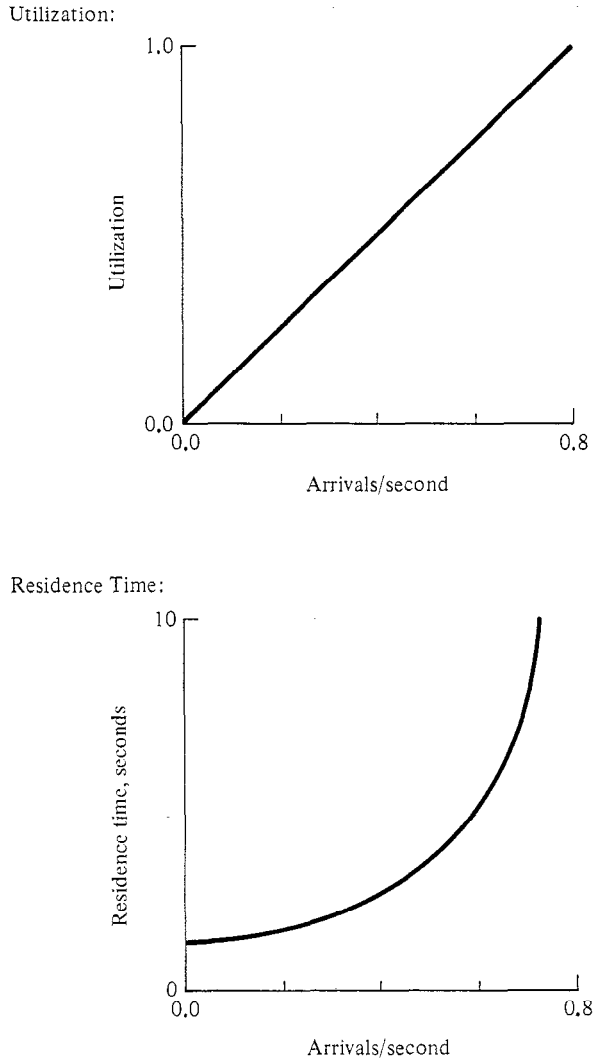
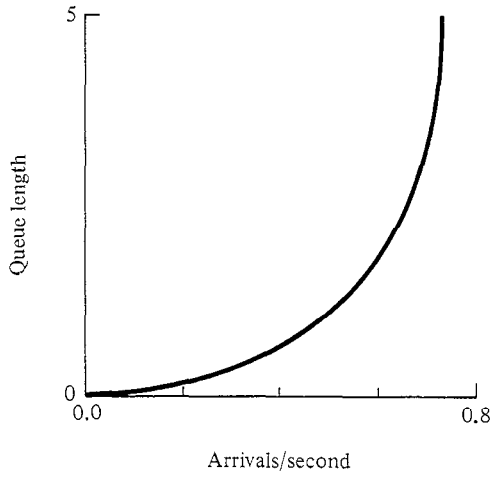


Figure 1.2a – Performance Measures for the Single Service Center

Queue Length:



Throughput:

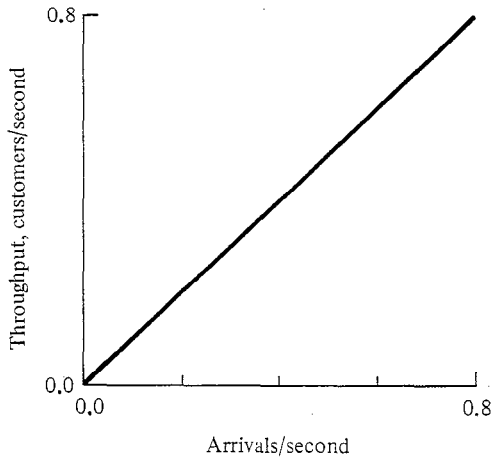


Figure 1.2b – Performance Measures for the Single Service Center

load grows, however, residence time increases at a faster and faster rate, until, as the service center approaches saturation, small increases in arrival rate result in dramatic increases in residence time.

### 1.2.2. Multiple Service Centers

It is hard to imagine characterizing a contemporary computer system by two parameters, as would be required in order to use the model of Figure 1.1. (In fact, however, this was done with success several times in the simpler days of the 1960's.) Figure 1.3 shows a more realistic model in which each system resource (in this case a CPU and three disks) is represented by a separate service center.

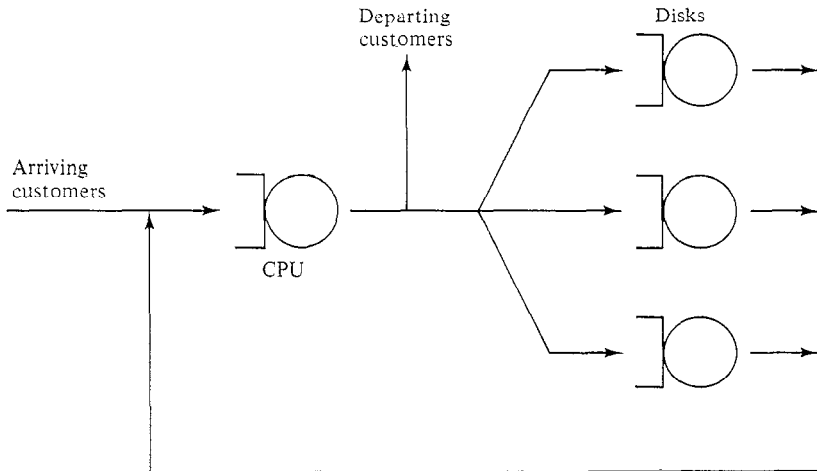


Figure 1.3 — A Network of Queues

The parameters of this model are analogous to those of the previous one. We must specify the workload intensity, which once again is the rate at which customers arrive. We also must specify the service demand, but this time we provide a separate service demand for each service center. If we view customers in the model as corresponding to transactions in the system, then the workload intensity corresponds to the rate at which users submit transactions to the system, and the service demand at each service center corresponds to the total service requirement per transaction at the corresponding resource in the system. (As indicated by the lines in the figure, we can think of customers as arriving, circulating among the service centers, and then departing. The pattern of circulation among the centers is not important, however; only the total service demand at each center matters.) For example, we might specify that

transactions arrive at a rate of one every five seconds, and that each such transaction requires an average of 3 seconds of service at the CPU and 1, 2, and 4 seconds of service, respectively, at the three disks. As in the case of the single service center, for specific parameter values it is possible to evaluate this model by solving some simple equations. For our example parameter values (under certain assumptions that will be stated later) performance measures include:

CPU utilization: .60  
 average system response time perceived by users: 32.1 seconds  
 average number of concurrently active transactions: 6.4  
 system throughput: 0.2 transactions/second

(We consistently will use *residence time* to mean the time spent at a service center by a customer, and *response time* to correspond to the intuitive notion of perceived system response time. Most performance measures obtained from queueing network models are average values (e.g., average response time) rather than distributional information (e.g., the 90th percentile of response times). Thus the word “average” should be understood even if it is omitted.)

## 1.3. Defining, Parameterizing, and Evaluating Queueing Network Models

### 1.3.1. Definition

Defining a queueing network model of a particular system is made relatively straightforward by the close correspondence between the attributes of queueing network models and the attributes of computer systems. For example, service centers in queueing network models naturally correspond to hardware resources and their software queues in computer systems, and customers in queueing network models naturally correspond to users or transactions in computer systems.

Queueing network models have a richer set of attributes than we have illustrated thus far, extending the correspondence with computer systems. As an example of this richness, specifying the rate at which customers arrive (an approach that is well suited to representing certain transaction processing workloads) is only one of three available means to describe workload intensity. A second approach is to state the number of customers in the model. (This alternative is well suited to representing batch workloads.) A third approach is to specify the number of customers and the average time that each customer “spends thinking” (i.e., uses a terminal) between interactions. (This alternative is well suited to representing interactive workloads.) In Figure 1.4 we have modified the model of



Figure 1.3 so that the workload intensity is described using this last approach. Figure 1.5 graphs system response time and CPU utilization for this model with the original service demands (3 seconds of service at the CPU and 1, 2, and 4 seconds of service, respectively, at the three disks) when the workload consists of from 1 to 50 interactive users, each with an average think time of 30 seconds. Once again we note that the behavior of the model is qualitatively consistent with intuition and experience.

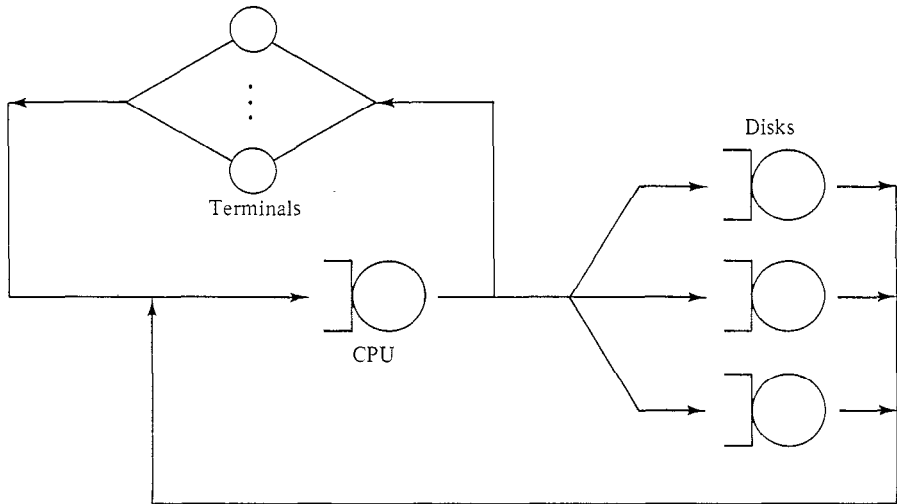
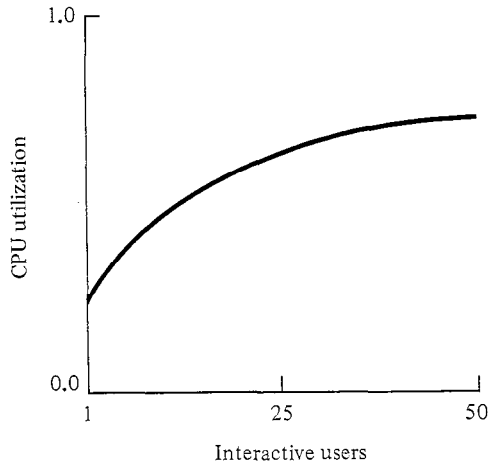


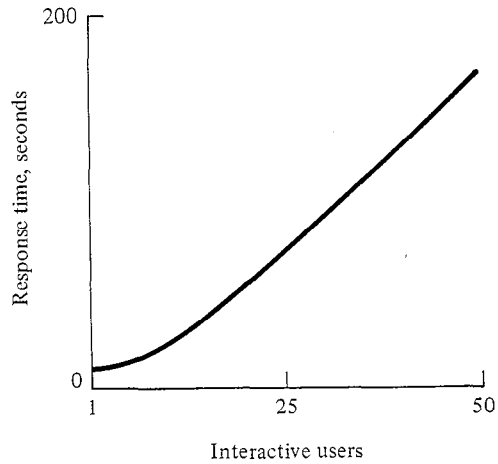
Figure 1.4 – A Model with a Terminal-Driven Workload

As another example of this richness, most computer systems have several identifiable workload components, and although the queueing network models that we have considered thus far have had a single *customer class* (all customers exhibit essentially the same behavior), it is possible to distinguish between a system's workload components in a queueing network model by making use of multiple customer classes, each of which has its own workload intensity (specified in any of the ways we have described) and service demands. For example, it is possible to model directly a computer system in which there are four workload components: transaction processing, background batch, interactive database inquiry, and interactive program development. In defining the model, we would specify four customer classes and the relevant service centers. In parameterizing the model, we would provide workload intensities for each class (for example, an arrival rate of 10 requests/minute for transaction processing, a multiprogramming level of 2 for background batch, 25 interactive database users each of whom thinks for an average of two minutes between interactions, and 10 interactive program development

CPU Utilization:



System Response Time:



**Figure 1.5 – Performance Measures for the Terminal-Driven Model**

users each of whom thinks for an average of 15 seconds between interactions). We also would provide service demands for each class at each service center. In evaluating the model, we would obtain performance measures in the aggregate (e.g., total CPU utilization), and also on a per-class basis (e.g., CPU utilization due to background batch jobs, response time for interactive database queries).

### 1.3.2. Parameterization

The parameterization of queueing network models, like their definition, is relatively straightforward. Imagine calculating the CPU service demand for a customer in a queueing network model of an existing system. We would observe the system in operation and would measure two quantities: the number of seconds that the CPU was busy, and the number of user requests that were processed (these requests might be transactions, or jobs, or interactions). We then would divide the busy time by the number of request completions to determine the average number of seconds of CPU service attributable to each request, the required parameter.

A major strength of queueing network models is the relative ease with which parameters can be modified to obtain answers to “what if” questions. Returning to the example in Section 1.2.2:

- What if we balance the I/O activity among the disks? (We set the service demand at each disk to  $\frac{1+2+4}{3} = 2.33$  seconds and re-evaluate the model. Response time drops from 32.1 seconds to 20.6 seconds.)
- What if the workload subsequently increases by 20%? (We set the arrival rate to  $0.2 \times 1.2 = 0.24$  requests/second and re-evaluate the model. Response time increases from 20.6 seconds to 26.6 seconds.)
- What if we then upgrade to a CPU 30% faster? (We set the service demand at the CPU to  $3/1.3 = 2.31$  seconds and re-evaluate the model. Response time drops from 26.6 to 21.0 seconds.)

Considerable insight can be required to conduct such a *modification analysis*, because the performance measures obtained from evaluating the model can be only as accurate as the workload intensities and service demands that are provided as inputs, and it is not always easy to anticipate every effect on these parameters of a change to the configuration or workload. Consider the first “what if” question listed above. If we assume that the system’s disks are physically identical then the *primary effect* of balancing disk activity can be reflected in the parameter values of the model by setting the service demand at each disk to the average value. However, there may be *secondary effects* of the change. For example, the total amount of disk arm movement may decrease. The result in

the system would be that the total disk service requirement of each user would decrease somewhat. If this secondary effect is anticipated, then it is easy to reflect it in the parameter values of the model, and the model, when evaluated, will yield accurate values for performance measures. If not, then the model will yield somewhat pessimistic results. Fortunately, the primary effects of modifications, which dominate performance, tend to be relatively easy to anticipate.

Models with multiple customer classes are more common than models with single customer classes because they facilitate answering many “what if” questions. (How much will interactive response time improve if the volume of background batch is decreased by 50%?) Single class models, though, have the advantage that they are especially easy to parameterize, requiring few assumptions on the part of the analyst. Using contemporary computer system measurement tools, it is notoriously difficult to determine correctly resource consumption by workload component, especially in the areas of system overhead and I/O subsystem activity. Since single class models can be parameterized with greater ease and accuracy, they are quicker and more reliable than multiple class models for answering those questions to which they are suited.

### **1.3.3. Evaluation**

We distinguish two approaches to evaluating queueing network models. The first involves calculating bounds on performance measures, rather than specific values. For example, we might determine upper and lower bounds on response time for a particular set of parameter values (workload intensity and service demands). The virtue of this approach is that the calculations are simple enough to be carried out by hand, and the resulting bounds can contribute significantly to understanding the system under study.

The second approach involves calculating the values of the performance measures. While the algorithms for doing this are sufficiently complicated that the use of computer programs is necessary, it is important to emphasize that these algorithms are extremely efficient. Specifically, the running time of the most efficient general algorithm grows as the product of the number of service centers with the number of customer classes, and is largely independent of the number of customers in each class. A queueing network model with 100 service centers and 10 customer classes can be evaluated in only seconds of CPU time.

The algorithms for evaluating queueing network models constitute the lowest level of a queueing network modelling software package. Higher levels typically include transformation routines to map the characteristics of specific subsystems onto the general algorithms at the lowest level, a

user interface to translate the “jargon” of a particular computer system into the language of queueing network models, and high-level front ends that assist in obtaining model parameter values from system measurement data.

### **1.4. Why Are Queueing Network Models Appropriate Tools?**

Models in general, and queueing network models in particular, have become important tools in the design and analysis of computer systems. This is due to the fact that, for many applications, queueing network models achieve a favorable balance between accuracy and efficiency.

In terms of accuracy, a large body of experience indicates that queueing network models can be expected to be accurate to within 5 to 10% for utilizations and throughputs and to within 10 to 30% for response times. This level of accuracy is consistent with the requirements of a wide variety of design and analysis applications. Of equal importance, it is consistent with the accuracy achievable in other components of the computer system analysis process, such as workload characterization.

In terms of efficiency, we have indicated in the previous section that queueing network models can be defined, parameterized, and evaluated at relatively low cost. Definition is eased by the close correspondence between the attributes of queueing network models and the attributes of computer systems. Parameterization is eased by the relatively small number of relatively high-level parameters. Evaluation is eased by the recent development of algorithms whose running time grows as the product of the number of service centers with the number of customer classes.

Queueing network models achieve relatively high accuracy at relatively low cost. The incremental cost of achieving greater accuracy is high — significantly higher than the incremental benefit, for a wide variety of applications.

### **1.5. Related Techniques**

Our informal description of queueing network modelling has taken several liberties that should be acknowledged to avoid confusion. These liberties can be summarized as follows:

- We have not described networks of queues in their full generality, but rather a subset that can be evaluated efficiently.
- We have incorrectly implied that the only analytic technique for evaluating networks of queues is the use of software to solve a set of equations induced by the network of queues and its parameters.
- We have neglected the fact that simulation can be used to evaluate networks of queues.
- We have not explored the relationship of queueing network models to queueing theory.

The following subsections explore these issues.

### 1.5.1. Queueing Network Models and General Networks of Queues

This book is concerned with a subset of general networks of queues. This subset consists of the *separable* queueing networks (a name used for historical and mathematical reasons), extended where necessary for the accurate representation of particular computer system characteristics.

We restrict our attention to the members of this subset because of the efficiency with which they can be evaluated. This efficiency is mandatory in analyzing contemporary computer systems, which may have hundreds of resources and dozens of workload components, each consisting of many users or jobs.

Restriction to this subset implies certain assumptions about the computer system under study. We will discuss these assumptions in later chapters. On the one hand, these assumptions seldom are satisfied strictly. On the other hand, the inaccuracies resulting from violations of these assumptions typically are, at worst, comparable to those arising from other sources (e.g., inadequate measurement data).

General networks of queues, which obviate many of these assumptions, can be evaluated analytically, but the algorithms require time and space that grow prohibitively quickly with the size of the network. They are useful in certain specialized circumstances, but not for the direct analysis of realistic computer systems.

### 1.5.2. Queueing Network Models and Simulation

The principal strength of simulation is its flexibility. There are few restrictions on the behavior that can be simulated, so a computer system can be represented at an arbitrary level of detail. At the abstract end of this spectrum is the use of simulation to evaluate networks of queues. At the concrete extreme, running a benchmark experiment is in some sense using the system as a detailed simulation model of itself.

The principal weakness of simulation modelling is its relative expense. Simulation models generally are expensive to define, because this involves writing and debugging a complex computer program. (In the specific domain of computer system modelling, however, this process has been automated by packages that generate the simulation program from a model description.) They can be expensive to parameterize, because a highly detailed model implies a large number of parameters. (We will see that obtaining even the small number of parameters required by a queueing network model is a non-trivial undertaking.) Finally, they are expensive to evaluate, because running a simulation requires substantial computational resources, especially if narrow confidence intervals are desired.

A tenet of this book, for which there is much supporting evidence, is that queueing network models provide an appropriate level of accuracy for a wide variety of computer system design and analysis applications. For this reason, our primary interest in simulation is as a means to evaluate certain submodels in a study that is primarily analytic. This technique, known as *hybrid modelling*, is motivated by a desire to use analysis where possible, since the cost of evaluating a simple network of queues using simulation exceeds by orders of magnitude the cost of evaluating the same model using analysis.

### 1.5.3. Queueing Network Models and Queueing Theory

Queueing network modelling can be viewed as a small subset of the techniques of queueing theory, selected and specialized for modelling computer systems.

Much of queueing theory is oriented towards modelling a complex system using a single service center with complex characteristics. Sophisticated mathematical techniques are employed to analyze these models. Relatively detailed performance measures are obtained: distributions as opposed to averages, for example.

Rather than single service centers with complex characteristics, queueing network modelling employs networks of service centers with simple characteristics. Benefits arise from the fact that the application domain is restricted to computer systems. An appropriate subset of networks of queues can be selected, and evaluation algorithms can be designed to obtain meaningful performance measures with an appropriate balance between accuracy and efficiency. These algorithms can be packaged with interfaces based on the terminology of computer systems rather than the language of queueing theory, with the result that only a minimal understanding of the theory underlying these algorithms is required to apply them successfully.

## 1.6. Summary

This chapter has surveyed the questions of cost and performance that arise throughout the life of a computer system, the nature of queueing network models, and the role that queueing network models can play in answering these questions. We have argued that queueing network models, because they achieve a favorable balance between accuracy and cost, are the appropriate tool in a wide variety of computer system design and analysis applications.

## 1.7. References

This book is concerned exclusively with computer system analysis using queueing network models. Because of this relatively narrow focus, it is complemented by a number of existing books. These can be divided into three groups, distinguished by scope.

Books in the first group, such as Ferrari's [1978], discuss computer system performance evaluation in the large.

Books in the second group consider computer system modelling. Examples include books by Gelenbe and Mitrani [1980], Kobayashi [1978], Lavenberg [1983], and Sauer and Chandy [1981].

Books in the third group treat a particular aspect of computer system performance evaluation at a level of detail comparable to that of the present book: computer system measurement [Ferrari et al. 1983], the low-level analysis of system components using simple queueing formulae [Beizer 1978], the analysis of computer systems and computer communication networks using queueing theory [Kleinrock 1976], and the mathematical and statistical aspects of computer system analysis [Allen 1978; Trivedi 1982].

Queueing network modelling is a rapidly advancing discipline. With the present book as background, it should be possible to assimilate future developments in the field. Many of these will be found in the following sources:

*EDP Performance Review*, a digest of current information on tools for performance evaluation and capacity planning, published by Applied Computer Research.

*Computer Performance*, a journal published by Butterworths.

The *Journal of Capacity Management*, published by the Institute for Software Engineering.



The *Proceedings of the CMG International Conference*. The conference is sponsored annually by the Computer Measurement Group, which also publishes the proceedings.

The *Proceedings of the CPEUG Meeting*. The meeting is sponsored annually by the Computer Performance Evaluation Users Group, which also publishes the proceedings.

The *Proceedings of the ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*. The conference is sponsored annually by the ACM Special Interest Group on Measurement and Evaluation. The proceedings generally appear as a special issue of *Performance Evaluation Review*, the SIGMETRICS quarterly publication.

The *ACM Transactions on Computer Systems*, a journal published by the Association for Computing Machinery.

The *IEEE Transactions on Computers* and the *IEEE Transactions on Software Engineering*, two journals published by the Institute of Electrical and Electronics Engineers.

The *Proceedings of the International Symposium on Computer Performance Modelling, Measurement and Evaluation*. The symposium is sponsored at eighteen month intervals by IFIP Working Group 7.3 on Computer System Modelling.

*Performance Evaluation*, a journal published by North-Holland.

[Allen 1978]

Arnold O. Allen. *Probability, Statistics, and Queuing Theory with Computer Science Applications*. Academic Press, 1978.

[Beizer 1978]

Boris Beizer. *Micro-Analysis of Computer System Performance*. Van Nostrand Reinhold, 1978.

[Ferrari 1978]

Domenico Ferrari. *Computer Systems Performance Evaluation*. Prentice-Hall, 1978.

[Ferrari et al. 1983]

Domenico Ferrari, Giuseppe Serrazi, and Alessandro Zeigner. *Measurement and Tuning of Computer Systems*. Prentice-Hall, 1983.

[Gelenbe & Mitrani 1980]

Erol Gelenbe and Israel Mitrani. *Analysis and Synthesis of Computer Systems*. Academic Press, 1980.

[Kleinrock 1976]

Leonard Kleinrock. *Queueing Systems — Volume II: Computer Applications*. John Wiley & Sons, 1976.

[Kobayashi 1978]

Hisashi Kobayashi. *Modeling and Analysis — An Introduction to System Performance Evaluation Methodology*. Addison-Wesley, 1978.

[Lavenberg 1983]

Stephen S. Lavenberg (ed.). *Computer Performance Modeling Handbook*. Academic Press, 1983.

[Sauer & Chandy 1981]

Charles H. Sauer and K. Mani Chandy. *Computer Systems Performance Modeling*. Prentice-Hall, 1981.

[Trivedi 1982]

Kishor S. Trivedi. *Probability and Statistics with Reliability, Queuing, and Computer Science Applications*. Prentice-Hall, 1982.